

PHD. DISSERTATION

---

**Model for WCET Prediction, Scheduling  
and Task Allocation for Emergent  
Agent-behaviours in Real-time Scenarios**

---



**VNiVERSIDAD  
D SALAMANCA**

Department of Computer Science and Automation  
Faculty of Science  
University of Salamanca  
Spain

**Author:** Davinia Carolina Zato Domínguez

**Academic advisors:**

Dr. D. Juan Francisco de Paz Santana  
Dr. D. Javier Bajo Pérez

July 2014



La memoria titulada *“Model for WCET prediction, scheduling and task allocation for emergent agent-behaviours in real-time scenarios”* que presenta Dña. Davinia Carolina Zato Domínguez para optar al Grado de Doctor por la Universidad de Salamanca ha sido realizada bajo la dirección del profesor Dr. D. Juan Francisco de Paz Santana, Profesor Ayudante Doctor del Departamento de Informática y Automática de la Universidad de Salamanca, y por el profesor Dr. D. Javier Bajo Pérez, Profesor Titular de Universidad del Departamento de Inteligencia Artificial de la Universidad Politécnica de Madrid.

Salamanca, julio de 2014

El Doctorando:



Fdo. Dña. Davinia Carolina Zato Domínguez

Los Directores:



Fdo: Dr. D. Juan Francisco de Paz Santana  
Profesor Ayudante Doctor  
Informática y Automática  
Universidad de Salamanca



Fdo: Dr. D. Javier Bajo Pérez  
Profesor Titular de Universidad  
Inteligencia Artificial  
Universidad Politécnica de Madrid

Carolina Zato Domínguez  
*Model for WCET prediction, scheduling and task allocation for emergent  
agent-behaviours in real-time scenarios.*

PhD. in Intelligent Systems  
Advisors: Juan F. De Paz and Javier Bajo  
University of Salamanca

All Rights Reserved © July 2014



*A mi padre, por ser un hombre valiente, bueno y un gran ejemplo a seguir, con una cabeza y un corazón envidiables.*

*Gracias por tu apoyo incondicional.*

*Te quiero.*



## Agradecimientos

*Fran, no sólo has sido un buen director de tesis sino que también un excelente compañero. Gracias por tu paciencia y toda tu ayuda. Espero que sientas esta tesis tan tuya como mía. ¡He aprendido muchísimo trabajando contigo!*

*Javi, gracias por tus revisiones de la tesis, de artículos y demás. Gracias por tus palabras de ánimo y por supuesto, gracias por tus sabios consejos.*

*Sin las buenas ideas de mis directores no habría alcanzado el final así que, otra vez, ¡GRACIAS!*

*También le tengo que agradecer a Juan Manuel la oportunidad que me brindó, hace ya 5 años, para iniciarme en el mundo de la investigación.*

*Gabri, PANGEA no sería lo mismo sin ti, ¡su nacimiento fue gracias a ti!*

*Pablo, Parra y Alberto, muchas gracias por prestarme vuestros "roboticos" para el caso de estudio, ¡son vuestras grandes creaciones!*

*A Fer, mi compañero y amigo desde que nos iniciamos en este mundo de la informática, ¡no hubiese sido lo mismo sin ti! Todo un camino juntos y ¡ya hemos llegado!*

*A mi gran amiga Sara le tengo que agradecer tantas cosas... las risas, los llantos, los desayunos, los artículos a toda prisa, los consejos, la ayuda y sobre todo, ¡su amistad!*

*También tengo que dar las gracias a todo el grupo BISITE, a los que siguen compartiendo los días de trabajo, Anto, Jesús, Loza, María, Barri, Dani, Carlos y Santos. Y a los que se fueron dejando un gran recuerdo, Rober, Elena, Virginia, Bea, Álvaro, Marisol, Honti y Alejandro.*

*A mis padres, les agradezco profundamente sus esfuerzos para darme una buena educación que me permite estar aquí hoy. A "Erico", siempre serás mi hermanito pequeño pero ¡me has enseñado tantas cosas! ¡estoy muy orgullosa de ti!*

*A mis abuelos, Loli y Manolo, gracias por cuidarme, protegerme y quererme como si fuera una hija. Y sobre todo, gracias por saber mantener a la familia unida, sois un gran ejemplo para todos nosotros.*

*A mi primo André, por su sonrisa y su alegría, su bondad y nuestras "tardes de peli". A mis tíos, Titi, María, Javi y Yoli, y a mi primito Martín, gracias por los momentos felices que pasamos todos juntos y que me dan vida.*

*A mis suegros, María Jesús y Licesio, gracias por acogerme en vuestra casa con los brazos abiertos, por preocuparos, por hacerme feliz en los momentos difíciles y dejarme formar parte de vuestra familia.*

*Finalmente, a Jesús, a quien se lo debo todo, sin él no habría llegado a escribir esta página. Gracias por darme tu fuerza, tu alegría, tu motivación, tu optimismo, tu tiempo y tu amor. Gracias por tu paciencia y por ¡TODO!*



# Abstract

To date, there are no known real-time models specially developed for their use in open systems, such as Virtual Organization of Agents (VOs). Conventionally, real-time models are applied to closed systems where all the variables are known. This research includes new contributions and the innovative integration of real-time agents in VOs. From our knowledge, this is the first model specifically designed to be applied in VOs with time constraints.

This dissertation provides a new perspective that combines the required openness and dynamism of the VO with the real-time constraints. This is a difficult task because the first paradigm is not as strict as the term "open" indicated but the second paradigm must fulfill strict constraints. In summary, the model presented enables defining the actions that an organization of agents should carry out within a deadline, considering the changes that may occur during the execution of a particular plan. It is a real-time scheduling within an organization of agents.

One of the main contributions of this dissertation is a new estimation model for Worst Case Execution Time (WCET). The proposal is an effective model for calculating the execution time in the worst case scenario when an agent desires to form part of a VO and wants to include tasks or behaviours in real-time systems, i.e. to calculate the WCET in emergent behaviours. A local planning within each node and a global distribution of tasks among the available nodes of the system are also developed. For both models, advanced mathematical and statistical methods are used to create an adaptive, robust and efficient method which can be used within intelligent agents in different organizations.

Additionally, the lack of awareness of the existence of an execution platform for open systems that supports agents and can also perform tasks with time constraints, including the necessary mechanisms for the management and control VOs, provides us with the main motivation for the development of the PANGEA+RT agent-platform. PANGEA+RT is an innovative multi-agent platform that provides effective support to the execution of VOs in real-time environments.

Finally, a case study is presented based on the collaboration of heterogeneous agent-robots under the PANGEA+RT platform where the proposed model and the associated platform are tested in detail. The results and conclusions obtained are presented in the final part of this PhD dissertation.



# Resumen

Hasta el momento no se conocen modelos de tiempo real específicamente desarrollados para su uso en sistemas abiertos, como las Organizaciones Virtuales de Agentes (OVs). Convencionalmente, los modelos de tiempo real se aplican a sistemas cerrados donde todas las variables se conocen a priori. Esta tesis presenta nuevas contribuciones y la novedosa integración de agentes en tiempo real dentro de OVs. Hasta donde alcanza nuestro conocimiento, éste es el primer modelo específicamente diseñado para su aplicación en OVs con restricciones temporales estrictas.

Esta tesis proporciona una nueva perspectiva que combina la apertura y dinamicidad necesarias en una OV con las restricciones de tiempo real. Ésto es un aspecto complicado ya que el primer paradigma no es estricto, como el propio término de sistema abierto indica, sin embargo, el segundo paradigma debe cumplir estrictas restricciones. En resumen, el modelo que se presenta permite definir las acciones que una OV debe llevar a cabo con un plazo concreto, considerando los cambios que pueden ocurrir durante la ejecución de un plan particular. Es una planificación de tiempo real en una OV.

Otra de las principales contribuciones de esta tesis es un modelo para el cálculo del tiempo de ejecución en el peor caso (WCET). La propuesta es un modelo efectivo para calcular el peor escenario cuando un agente desea formar parte de una OV y para ello, debe incluir sus tareas o comportamientos dentro del sistema de tiempo real, es decir, se calcula el WCET de comportamientos emergentes en tiempo de ejecución. También se incluye una planificación local para cada nodo de ejecución basada en el algoritmo FPS y una distribución de tareas entre los nodos disponibles en el sistema. Para ambos modelos se usan modelos matemáticos y estadísticos avanzados para crear un mecanismo adaptable, robusto y eficiente para agentes inteligentes en OVs.

El desconocimiento, pese al estudio realizado, de una plataforma para sistemas abiertos que soporte agentes con restricciones de tiempo real y los mecanismos necesarios para el control y la gestión de OVs, es la principal motivación para el desarrollo de la plataforma de agentes PANGEA+RT. PANGEA+RT es una innovadora plataforma multi-agente que proporciona soporte para la ejecución de agentes en ambientes de tiempo real.

Finalmente, se presenta un caso de estudio donde robots heterogéneos colaboran para realizar tareas de vigilancia. El caso de estudio se ha desarrollado con la plataforma PANGEA+RT donde el modelo propuesto está integrado. Por tanto al final de la tesis, con este caso de estudio se obtienen los resultados y conclusiones que validan el modelo.



# CONTENTS

---

<b>List of Figures</b>	<b>v</b>
<b>List of Tables</b>	<b>vii</b>
<b>I Introduction</b>	<b>1</b>
<b>1 Introduction</b>	<b>3</b>
1.1 Introduction . . . . .	4
1.1.1 Motivation and hypothesis . . . . .	6
1.1.2 Objectives . . . . .	7
1.2 Methodology . . . . .	8
1.3 Structure of the document . . . . .	10
<b>II Basic Concepts and Related Works</b>	<b>13</b>
<b>2 Virtual Organizations of Agents</b>	<b>15</b>
2.1 The concept of agent . . . . .	16
2.2 Multi-agent systems . . . . .	20
2.3 Societies of agents . . . . .	23
2.3.1 Open societies . . . . .	25
2.3.2 Closed societies . . . . .	26
2.3.3 Semi-open societies . . . . .	26
2.3.4 Semi-closed societies . . . . .	26
2.4 Organizations of agents . . . . .	27
2.5 Conclusions . . . . .	33
<b>3 Real-Time Systems</b>	<b>35</b>
3.1 Introduction . . . . .	36
3.1.1 Classification of real-time systems . . . . .	36
3.2 Models of tasks in real-time systems . . . . .	37
3.3 Real-time scheduling . . . . .	39
3.3.1 Local scheduling within computational nodes . . . . .	40
3.3.2 Global scheduling in a real-time system . . . . .	43
3.4 Distributed real-time systems . . . . .	47
3.4.1 Multi-agent systems in real-time environments . . . . .	50
3.5 Conclusions . . . . .	54

<b>III</b>	<b>Proposed Model</b>	<b>57</b>
<b>4</b>	<b>Proposed Model</b>	<b>59</b>
4.1	Introduction . . . . .	60
4.2	Background and related works . . . . .	62
4.2.1	Related works of WCET analysis . . . . .	62
4.2.2	Real-time scheduling and task allocation . . . . .	64
4.3	WCET in emergent behaviours . . . . .	67
4.3.1	Node evaluation . . . . .	68
4.3.2	Code evaluation . . . . .	74
4.3.3	Statistical improvements of the WCET in execution . . . . .	82
4.4	Model of scheduling and task allocation . . . . .	83
4.5	Local scheduling . . . . .	84
4.6	Global scheduling . . . . .	85
4.7	Conclusions . . . . .	96
<b>IV</b>	<b>PANGEA+RT Platform</b>	<b>97</b>
<b>5</b>	<b>PANGEA</b>	<b>99</b>
5.1	Introduction . . . . .	100
5.2	Related works . . . . .	102
5.3	PANGEA overview . . . . .	106
5.3.1	Reorganization and task allocation model . . . . .	110
5.4	The PANGEA database . . . . .	112
5.5	The monitoring tool . . . . .	115
5.6	The norms in PANGEA . . . . .	117
5.7	The communication module . . . . .	119
5.7.1	Testing the communication . . . . .	122
5.8	Testing the SnifferAgent . . . . .	124
5.9	The subscription model . . . . .	126
5.10	The Gateway agent . . . . .	128
5.10.1	Request protocol . . . . .	129
5.10.2	Subscription protocol . . . . .	130
5.10.3	Contract-net protocol . . . . .	131
5.10.4	Inform protocol . . . . .	131
5.11	Conclusions . . . . .	132
<b>6</b>	<b>PANGEA+R</b>	<b>135</b>
6.1	Introduction . . . . .	136
6.2	Platforms and middlewares for robotic systems . . . . .	137
6.3	Multi-agent robotics systems . . . . .	141

6.4	PANGEA+R platform . . . . .	142
6.4.1	The growing need for cooperation in Robotics . . . . .	143
6.4.2	The contribution of VOs in Robotics . . . . .	144
6.4.3	Main characteristics of PANGEA+R . . . . .	146
6.5	Communication module . . . . .	148
6.5.1	Message format . . . . .	150
6.5.2	Command messages examples . . . . .	152
6.5.3	Servers and clients . . . . .	156
6.6	Conclusions . . . . .	158
<b>7</b>	<b>PANGEA+RT</b>	<b>161</b>
7.1	Introduction . . . . .	162
7.1.1	Problems of Java in real-time environments . . . . .	163
7.2	Real-time specification for Java . . . . .	166
7.3	Real-time Java virtual machines . . . . .	169
7.4	Annotation for bounded loops . . . . .	171
7.5	Agents of the platform . . . . .	176
7.6	Modification of the classes . . . . .	178
7.7	Conclusions . . . . .	181
<b>V</b>	<b>Case Study and Conclusions</b>	<b>183</b>
<b>8</b>	<b>Case Study</b>	<b>185</b>
8.1	Collaboration of heterogeneous robots for surveillance tasks . . . . .	186
8.2	Related works . . . . .	187
8.3	Presentation of the heterogeneous robots . . . . .	192
8.4	The problem and the VO solution . . . . .	197
8.4.1	Proposed VO of agents . . . . .	198
8.5	Collaboration description . . . . .	202
8.5.1	Calculations for the collaborative movement . . . . .	204
8.5.2	Common area calculation . . . . .	208
8.5.3	Calculation of horizontal and vertical movement for the HAWK waypoints . . . . .	209
8.6	Deployment of the involved agents . . . . .	211
8.7	Conclusions . . . . .	213
<b>9</b>	<b>Results of the Proposed Model</b>	<b>215</b>
9.1	General evaluation of the model . . . . .	216
9.1.1	WCET evaluation . . . . .	216
9.2	Case study results . . . . .	225
9.3	Results of the collaboration search . . . . .	229
9.4	Conclusions . . . . .	232

<b>10</b>	<b>Conclusions and Future Work</b>	<b>235</b>
10.1	Conclusions and Main Contributions . . . . .	236
10.2	Future Work . . . . .	238
10.2.1	Future lines related to the model . . . . .	238
10.2.2	Future lines related to PANGEA+RT . . . . .	240
10.2.3	Future lines related to the Case Study . . . . .	241
<b>VI</b>	<b>Resumen</b>	<b>243</b>
<b>11</b>	<b>Resumen</b>	<b>245</b>
11.1	Introducción . . . . .	246
11.2	Conceptos básicos y trabajos relacionados . . . . .	248
11.2.1	Análisis del WCET . . . . .	249
11.2.2	Planificación en tiempo real y distribución de tareas . . . . .	250
11.3	WCET en comportamientos emergentes . . . . .	253
11.3.1	Evaluación de los nodos . . . . .	254
11.3.2	Evaluación del código . . . . .	257
11.3.3	Adaptación estadística del WCET en tiempo de ejecución	261
11.4	Planificación y distribución de tareas . . . . .	262
11.5	La plataforma PANGEA+RT . . . . .	268
11.5.1	El protocolo de comunicación . . . . .	268
11.5.2	Agentes de PANGEA+RT . . . . .	272
11.5.3	Modificación de las clases . . . . .	274
11.6	Caso de estudio . . . . .	277
11.6.1	Despliegue de los agentes propuestos . . . . .	280
11.6.2	Resultados . . . . .	281
11.7	Conclusiones . . . . .	285
	<b>Bibliography</b>	<b>291</b>
<b>A</b>	<b>Appendix</b>	<b>315</b>
A	Related projects . . . . .	315
A.1	OVAMAH project . . . . .	315
A.2	AZTECA project . . . . .	316
A.3	iHAS project . . . . .	317
B	Related publications . . . . .	318
B.1	International journals . . . . .	318
B.2	Book chapters . . . . .	319
B.3	Conferences . . . . .	320



# LIST OF FIGURES

---

3.1	Dhall Effect Behaviour . . . . .	45
3.2	Distributed real-time system . . . . .	48
4.1	Steps of Java code execution process . . . . .	69
4.2	Steps of the node evaluation . . . . .	72
4.3	Steps of the WCET calculation . . . . .	75
4.4	ICFG Example . . . . .	81
4.5	Overview of the RTS . . . . .	84
4.6	Timeline for the task set $\mathcal{T}$ . . . . .	87
4.7	Example of the Branch and Bound method to approximate the values to integers . . . . .	95
5.1	Main classes of the system . . . . .	107
5.2	PANGEA agents . . . . .	108
5.3	OV topology . . . . .	110
5.4	Overview of the agents involved . . . . .	111
5.5	Database . . . . .	113
5.6	InformationAgent replication . . . . .	114
5.7	Communication scheme for representing events in the monitoring tools . . . . .	116
5.8	NormAgent scheme . . . . .	118
5.9	Addition of a new norm in the Monitoring tool . . . . .	119
5.10	Sequence of steps for an agent to enter an organization . . . . .	121
5.11	Test Case . . . . .	123
5.12	Messages through the server . . . . .	124
5.13	Diagram generated by the SnifferAgent (I) . . . . .	125
5.14	Diagram generated by the SnifferAgent (II) . . . . .	126
5.15	Communication lines among agents for the Subscription model . . . . .	128
5.16	Communication between FIPA and PANGEA agents . . . . .	129
5.17	Request protocol . . . . .	130
5.18	Subscription protocol . . . . .	130
5.19	Contrat-net protocol . . . . .	131
5.20	Inform protocol . . . . .	132
5.21	Comparison of the most used Virtual Organization of agentss (VOs) platforms . . . . .	134
6.1	Software tool of GECKO . . . . .	147
6.2	MQTT Message Format . . . . .	150
6.3	CONNECT Message Format . . . . .	153
6.4	SUBSCRIBE Message Format . . . . .	155

6.5	PUBLISH Message Format . . . . .	155
6.6	Communication with the MQTT protocol . . . . .	156
6.7	Comparison of the most used VOs and Robotics platforms . . . . .	160
7.1	Platform Overview . . . . .	177
7.2	Main classes of the platform . . . . .	179
8.1	Image of GECKO . . . . .	192
8.2	Image of HAWK . . . . .	193
8.3	Characteristics of HAWK and GECKO . . . . .	195
8.4	Software tool of HAWK . . . . .	196
8.5	Software tool of GECKO . . . . .	197
8.6	Proposed VO of agents . . . . .	201
8.7	Waypoints that must follow HAWK . . . . .	203
8.8	General steps of the coordinated movement . . . . .	204
8.9	Camera placement . . . . .	206
8.10	Corners of the field of view . . . . .	207
8.11	Corners of the field of view . . . . .	208
8.12	Dimensions of the CCD in relation to the displacement of HAWK . . . . .	209
9.1	Time execution of some bytecodes in node 1, 2 and 3 . . . . .	217
9.2	Screenshot of the bytecodes in the JBE tool . . . . .	220
9.3	<i>CFG</i> of the example . . . . .	221
9.4	Real WCET execution and WCET estimation . . . . .	223
9.5	Results of the utilization factor $\rho_i^j$ . . . . .	227
9.6	Changes of angle according to the path . . . . .	231
11.1	Pasos para la evaluación de los nodos . . . . .	254
11.2	Pasos para el cálculo del WCET . . . . .	257
11.3	Vista general del STR . . . . .	263
11.4	MQTT Message Format . . . . .	269
11.5	Arquitectura de comunicación con el protocolo MQTT . . . . .	271
11.6	Agentes de PANGEA+RT . . . . .	273
11.7	Clases principales de la plataforma . . . . .	275
11.8	OV propuesta para el despliegue del caso de estudio . . . . .	279
11.9	Resultados del factor de utilización $\rho_i^j$ . . . . .	283

# LIST OF TABLES

---

2.1	Possible organization topologies . . . . .	32
3.1	Example Dhall Effect . . . . .	44
3.2	Overview of the main global scheduling algorithms . . . . .	46
4.1	Example FPS . . . . .	87
5.1	Summary of middlewares or platforms for VOs . . . . .	105
5.2	IRC Primitives . . . . .	121
5.3	Test results . . . . .	124
6.1	Robotics-oriented middlewares and platforms . . . . .	140
6.2	MQTT message types . . . . .	151
7.1	Overview of Java Virtual Machines for real-time . . . . .	170
8.1	Relation between the nodes and their characteristics . . . . .	212
9.1	Bytecode execution in Nodes 1 and 2 . . . . .	216
9.2	Bytecode execution in Node 3 . . . . .	217
9.3	Example of some executions . . . . .	218
9.4	Bytecodes involved in the example . . . . .	219
9.5	Comparison between the real WCET execution and WCET estimation . . . . .	222
9.6	Values of the statistical adaptation of the WCET (I) . . . . .	224
9.7	Values of the statistical adaptation of the WCET (II) . . . . .	225
9.8	Results of the task allocation model . . . . .	228
9.9	Modification of the $\rho_{19}^j$ . . . . .	228
9.10	Results of the scheduling and task allocation model . . . . .	229
9.11	Results obtained by modifying the hight of HAWK and the camera . . . . .	230
9.12	Results obtained by modifying the hight and the focal lenght, $\lambda$ . . . . .	230
9.13	Results of the horizontal and vertical displacements, $(xDist, yDist)$ , for the calculations of the movement $SM$ . . . . .	231
9.14	Summary of flying results . . . . .	231
11.1	MQTT message types . . . . .	270
11.2	Relación entre los nodos y sus características . . . . .	281
11.3	Resultados del modelo de distribución de tareas . . . . .	284
11.4	Modificación del parámetro $\rho_{19}^j$ . . . . .	284
11.5	Resultados del modelo de planificación y distribución de tareas . . . . .	285



# Part I

INTRODUCTION



# 1

## INTRODUCTION

---

*This chapter describes the motivation and the main hypothesis of this PhD thesis and gives an overview of the methodology used during the research process of this dissertation. In addition, the main objectives to accomplish are presented.*

### Contents

---

<b>1.1</b>	<b>Introduction</b>	<b>4</b>
1.1.1	Motivation and hypothesis	6
1.1.2	Objectives	7
<b>1.2</b>	<b>Methodology</b>	<b>8</b>
<b>1.3</b>	<b>Structure of the document</b>	<b>10</b>

---

## 1.1 Introduction

---

The design of Real-time System (RTS) is an activity that involves meticulous planning and management of multiple resources. These resources need to be orchestrated predictably and in concert with one another to ensure that tasks executing on the system will meet stringent timing requirements and provide the desired performance to the application.

Resources need to be allocated with considerations to the functional and non-functional aspects (timing) of the system and the cost of the design. Online resource management of a RTS requires fast reaction to changing workload and efficient tests for schedulability, even when there are several constraints to deal with. The traditional bounds have been conservative and are such that as long as the task set does not exceed the utilization bound, all tasks will meet their deadlines. Alternatively, if a set of tasks violates the utilization bound, some tasks may miss their deadlines. RTS differ from most computing systems because of the timeliness properties that they need to satisfy. Tasks in a RTS are associated with explicit timing constraints which need to be met for correct behavior. Time becomes an important non-functional element in RTS because correctness of the system depends not only on the correct functional execution of tasks, but on the timely completion of tasks. We would like to distinguish between scheduling and resource management, even though they are closely related. Scheduling is the aspect of resource management concerned with ensuring that a set of tasks meets its timing requirements. Resource management, in its totality, is concerned with higher-level decisions that determine which resources will carry out a specific task.

Related to the mentioned problem of resource allocation and specifically, task scheduling, the algorithms are generally independent of resource management framework. In our case, we propose combining schedulability and resource allocation since once utilization bounds and the worst case execution time are estimated, resource management decisions could be made within the region of schedulability. This is considered as an NP-hard problem: solvable in theory, but nearly impossible to solve in practice. Nevertheless, we propose a new approach based on the combination of mathematical methods that lead to a good solution according to the time constraints.

In this study, we try to put together the required openness and dynamism of the VO and real-time constrains. The proposal is a real-time scheduling within an organization of agents. Current research focused on the design of Multi-agent System (MAS) from the organizational point of view is gaining



ground. The prevailing idea is that modelling the interactions of a MAS cannot be limited to the agent and its communication capabilities, but instead requires organizational engineering. The concepts of norms [386], institutions [105] and social structures [279] were born from the idea that we need a higher level of abstraction, independent of the agent, and the ability to explicitly define the organization in which the agents reside. The agents in a MAS based on organizational concepts work in coordination and exchange services and information; they need to be able to negotiate, collaborate and reach agreements, and can perform other more complex social actions. The term coined for these systems is Virtual Organization of Agents (VO) [111]. The dynamics of open environments is one of the reasons that have encouraged the use of VO. Nowadays, VOs are an open research issue in MAS due to the need of these systems to become more open and dynamic. In an open MAS [27] such a VO should allow the interaction between heterogeneous agents, which change over time, and architectures, and even different languages. Because of their inherent changing nature, we cannot rely on agents' behaviour when it is necessary to establish controls on the basis of norms or social rules. For this reason, and because of the characteristics of open environments, new approaches are needed to support evolutive systems and to facilitate their dynamic growth and runtime updates. A VO [111] [113] is an open system designed for grouping; it allows for the collaboration of heterogeneous entities and provides a separation between the form and function that define their behaviour.

In summary, the proposed model enables defining the actions that an organization of agents should carry out within a deadline, considering the changes that may occur during the execution of a particular plan. It is a real-time scheduling within an organization of agents. This is a difficult task because the VO paradigm is not as strict as the term "open" indicates but the RTS paradigm must fulfill strict constraints.

This thesis is part of the open research lines within the BISITE research group ([bisite.usal.es](http://bisite.usal.es)). The group has extensive experience in the application of intelligent techniques in multi-agent systems and in their evolution towards open MAS, where much pioneering research has been published. Moreover, the group is also working on providing formalism to the technique used in scheduling and adaptation. This thesis initiates the VO line framed in real-time with critical constraints.

### 1.1.1 Motivation and hypothesis

---

Research is mainly the recognition of the validity of previous studies and ideas, followed by their implementation, use and improvements in different contexts. The importance of what has been done, what has already been investigated, is evident in every new method, in every new model or each new theory conceived [303]. This PhD thesis provides a new idea and a new model along with its context, motivation and assumptions that made them arise.

Virtual Organizations [111] are a new way of understanding the modelling of agent systems from a sociological and organizational point of view. Within the development of organizations at the agent level, we find a set of requirements [85] that demands new planning models in which the use of open systems [389] and dynamic [90] are possible. Open systems are characterized by the heterogeneity of their participants and their need to adapt to the environment [134] [387] [372] [65] [301], but this is a difficult challenge when we take into account emerging entities with a priori unknown behaviors. Currently, it is intended to develop software systems able to respond to changes and act for themselves in reaction to changes that occur in their environment. To do this it is necessary to define theories, models, mechanisms, methods and tools to develop systems that can reorganize and thus adapt to future changes in their environment.

Moreover, if we add these characteristics to a RTS, the problem is more complex. To date, there are no known real-time models specially implemented for their use in open systems, such as VOs. Conventionally, real-time models are applied to closed systems where all the variables are known. Therefore, the initial hypothesis is that: *it is possible to develop a scheduling and task allocation model that successfully enables the agents integrated into a VO to fulfil their time constraints. This model will be theoretically formalized and then implemented to evaluate the results.*

Additionally, the lack of awareness of the existence of an execution platform for open systems that supports agents and can also perform tasks with time constraints, including the necessary mechanisms for the management and control of VOs, provides us with the main motivation for the development of the PANGEA+RT agent-platform.

---

**1.1.2** Objectives

---

Verifying the hypothesis led to other problems that are specified as specific objectives of this study, including:

- Study and evaluate mechanisms for temporal prediction in real-time execution.
- Develop an additional model previous to the scheduling model, since this must use a task estimation of execution time. In open MAS and emergent behaviors, task estimation should be calculated at the time the agent wants to join the system and once the agent indicates its role and the tasks that can carry out.
- Study useful formal mechanisms in solving optimization problems and operational research that can be applied to scheduling and allocation process.
- Develop a scheduling model that must be bounded temporarily. This is a compulsory requisite to be applied in real-time scenarios.

Once the scheduling and task allocation model is developed, the absence of a framework for the VO execution in real-time brought up additional objectives:

- Develop a complete platform that can manage the VOs that have strict real-time constraints.
- This platform should enable a dynamic adaptation and group formation in execution time with different topologies. It must also cover all the main issues that the organizations approach requires, such as the norms.
- The platform must allow the heterogeneous agents, which are developed in different languages and operating systems, to interact in a dynamic environment.
- The platform must ensure robust communication.

Finally, we must validate the model and the platform. This brings up the following objectives:

- Carry out a study on the specific problems where a practical application of the model can be used together to VOs. The application must take place in dynamic environments and in real-time. This study must include heterogeneous agents where the formation of groups can improve the results.

- Evaluate the results empirically in the real application environment.

## 1.2

## Methodology

---

The methodology for the development of this research follows a structure previously planned and based on the required scientific principles for this type of work. First, we review the existing literature in the two areas that are addressed in this research, the VOs and the RTS. In the absence of a platform to support VOs, we continue with the steps for the development of the PANGEA agent-platform and subsequently, we extend the requirements for its application in the field of robotics. To do this, the advantages and disadvantages of the existing platforms are studied. With the knowledge gained during the development of state of the art, we study the possible problems and solutions to develop the model for planning and distribution of tasks in real-time organizations. This model is developed and integrated into a new extension of PANGEA, called PANGEA+RT which includes time constraints and VO capabilities. Then, we design and carry out the necessary tests. Finally, a real case study is designed to validate the model results and the PANGEA+RT platform. This is an innovative and original work that meets the pre-established steps for a correct scientific research.

The research methodology followed in the development of this work is divided into six main activities:

### 1. Study of the basic concepts and related works

- a) Study VOs and their main characteristics.
- b) Study the characteristics of RTS.
- c) Evaluate existing planning models in VOs.
- d) Evaluate existing scheduling methods in RTS.
- e) Prepare a first draft that considers the issues and the problems of mixing the two areas (VOs and RTS).

### 2. Development of the PANGEA agent-platform

- a) Study and evaluation of the previous and/or existing agent-platforms.
- b) Analyze the desired characteristics of the new PANGEA platform.
- c) Design of the platform according to the VO requirements.

- d) Integrate a previous proposed scheduling method [391].
- e) Case Study: AZTECA project.

### 3. Development of the +R middleware

- a) Study and evaluate previous and/or existing robotics platforms.
- b) Analyze the desired characteristics of the middleware to adapt to it to the robotics field.
- c) Design the middleware according to the VOs and robotics requirements.
- d) Integrate a previous proposed scheduling method [303].
- e) Integrate the middleware +R in PANGEA.
- f) Case Study: Collaboration of heterogeneous robots (HAWK and GECKO).

### 4. Model for scheduling and task allocation in real-time

- a) Study and evaluation of previous models.
- b) Adapt the studied mechanisms in the activity 1 for scheduling and task allocation to the problem of VO and emergent behaviours.
- c) Evaluate possible mathematical approaches to solve the problem.
- d) Develop the proposed model.
- e) Test the proposed model and obtain evaluation results.

### 5. Development of the +RT middleware

- a) Analyze the desired characteristics of middleware to adapt the solution obtained in the activity 3 to the real-time field.
- b) Integrate the proposed model.
- c) Integrate the +RT middleware in PANGEA.
- d) Case Study: Collaboration of heterogeneous robots with real-time constraints.

### 6. Generation of the dissertation document

- a) Analyze all the generated material and publications.
- b) Write the dissertation document.

**1.3****Structure of the document**

---

This document has been divided into nine chapters that correspond closely to the time-line of the methodology.

**PART I**

The first part of this document corresponds to the present chapter 1, which includes an introduction to the research done and to the dissertation and describes the existing problem regarding the development of organizations based on multi-agent systems and the difficulty of integrating into hard RTS. Objectives, assumptions and the motivation that led us to the development of the model are also presented. Finally, the applied research methodology and a brief description of the structure of this report are both explained in detail.

**PART II**

An overview of the theories and research areas related to this dissertation are presented to gain a better understanding of the concepts and ideas presented later. The purpose is not to give a complete view of these issues, but show the foundations on which this research is based. This review will constitute the basis to detect needs, strengths and weakness of the related work, and to remark the innovations presented in this PhD thesis. This part is divided into two chapters.

In chapter 2, we explain the paradigm of agents and MAS, emphasizing organizational issues and their related concepts such as group formation, adaptation, coordination and scheduling.

In chapter 3, we show the most important concepts to understand the strict requirements of RTS. We focus on the scheduling techniques used in these systems and the issues that arise when joining real-time with distributing system and agents.

**PART III**

This part is devoted to the chapter 4, which contains two main sections holding the main theoretical content of the dissertation. Moreover, together with Chapter 7 constitutes the greatest innovation of this work since it is not known any previous model integrated in a platform that combines agents working in real-time VOs.

The first section includes the proposed Worst Case Execution Time (WCET) estimation model. This is an effective model for calculating the execution time in the worst case scenario when an agent desires to form part of a VO

and wants to include tasks or behaviours in real-time system, i.e. to calculate the WCET in emergent behaviours. This measure will be recalculated and adjusted in the system based on statistical measurements during subsequent executions.

The second section presents existing problems that arise when virtual organizations of agents and hard-real time systems are combined. Later, the model dedicated to the local planning within each node and the global distribution of tasks among the available nodes are explained.

## **PART IV**

This part presents the implementation issues of the proposed agent-platform and the model from part III. It is divided into 3 chapters.

Chapter 5 presents the new agent-platform called PANGEA. PANGEA is a new platform that can develop open MAS, specifically those including organizational aspects. In this chapter, we present the platform's architecture, the agents involved, and their functionality. As it will be shown, the platform offers many advantages and facilities to develop this kind of systems.

Chapter 6 focus on a new perspective to design +R architectures and explains the relationship and benefits of applying VOs to robotics. The +R middleware was designed for this purpose; it acts as a repository for developing robotics oriented services. Therefore, the adaptation of PANGEA, called PANGEA+R, is presented.

The last chapter in this section, chapter 7, presents the complete PANGEA+RT agent-platform. In this platform, the model proposed in chapter 4 is integrated and all the necessary mechanisms to ensure the time constrains are added. Moreover, the MQTT protocol and the Real-Time Specification for Java are introduced as main concepts of the platform extension. This platform is used in Chapter 8 for the case study.

## **PART V**

The evaluation and conclusions are presented in the fifth part. To begin, chapter 8 includes a case study that allows us to evaluate the PANGEA+RT platform described in chapter 7 and to, subsequently, obtain objective results and conclusions. This case study focuses on the collaboration of heterogeneous robots under real-time constraints. The mentioned results of the model proposed in part III of this document, with regard to scheduling, task allocation and WCET estimation, are explained in chapter 9 together with the results of the case study.

Finally, chapter 10 presents the conclusions and future work in the different lines that this dissertation can follow.

**PART VI**

The last part of the document, Part 6 is "Resumen" ("Summary"), which is composed of an abstract in Spanish of all the work done: initial studies, proposals, results and final conclusions.

Finally, a list of acronyms and a list of references used to carry out this work are provided, followed by an appendix with the related publications and projects.



# Part II

BASIC CONCEPTS AND  
RELATED WORKS



# 2

## VIRTUAL ORGANIZATIONS OF AGENTS

---

*In this chapter we discuss MAS from its organizational perspective and VOs. The concept of agent has already been studied in depth in the literature and hence for the part corresponding to agents and multi-agent systems we only offer a brief summary of such technology. However, we go further into the consideration of agent as an entity that operates within a society. Also, we explain issues that are of particular relevance for an agent within an organization, such as cooperation, grouping, norms etc. The main aim of this first section is to address current trends in the development of MAS from the organizational point of view, establishing all the aspects that should be taken into account when designing a virtual organization of agents operating in an open and dynamic environment. This chapter is the first step to establish the characteristics of the VOs and analyse them in order to adjust their functionality to the constraints of the scheduling model mentioned as one objectives of this dissertation.*

### Contents

---

<b>2.1</b>	<b>The concept of agent</b>	<b>16</b>
<b>2.2</b>	<b>Multi-agent systems</b>	<b>20</b>
<b>2.3</b>	<b>Societies of agents</b>	<b>23</b>
2.3.1	Open societies	25
2.3.2	Closed societies	26
2.3.3	Semi-open societies	26
2.3.4	Semi-closed societies	26
<b>2.4</b>	<b>Organizations of agents</b>	<b>27</b>
<b>2.5</b>	<b>Conclusions</b>	<b>33</b>

---

**2.1**The concept of agent

---

The evolution of software, and more to the point of the software that incorporates elements of artificial intelligence, tends towards the creation of entities with behaviours and conducts similar to those of human beings. The theory of agents rests on the concept of agent [309]. An agent is an autonomous entity endowed with certain capacities typical of human beings. It may be seen as a development of the concept of software object, perfected thanks to the influence of artificial intelligence, that allows characteristics such as rationality, intelligence, autonomy and learning to be incorporated. As in the case of human beings, agents must have social skills and be able to perform tasks or solve problems in a distributed fashion. One then speaks in terms of a multi-agent system, in which the agents cooperate and interact to achieve the final aims of the system.

Owing to the multidisciplinary setting in which the concept of agent has always found itself, it has been difficult to provide a uniform definition. One of the most widely accepted definitions is as follows:

**Definition 1** *An agent is an encapsulated computational system located in a given setting that is able to act autonomously, and flexibly in that setting to achieve the objectives for which it was designed [309].*

Some of the concepts introduced in this definition merit further explanation. Thus, with "encapsulated computational system" it should be understood that there is a clear distinction between the agent and its surroundings. Moreover, the definition implies that there is a well-defined frontier and a specific interface between the agent and its environment. The key aspect of the definition is autonomy, referring to the principle that agents are able to act by themselves without needing human guidance. An autonomous agent has control over its own actions and internal state; that is, an agent can decide whether it wants to perform an action that has been requested of it. The definition places an agent in a specific environment in which the agent can "feel" and "act". This leads us to response behaviour. The definition also implies that agents aim to find solutions to problems with well-defined limits in order to achieve a specific aim, in other words with specific goals to be attained, and that they exhibit an active, flexible and pro-active behaviour. For [377], the term agent has the following properties:

- **Autonomy:** Agents operate without direct intervention from humans or other sources and have some type of control over their actions and internal states.
- **Social skills:** Agents interact with other agents (and possibly with humans) through some language for communication among agents.
- **Reactivity:** Agents perceive their environment (which may be the physical world, a user through a graphic interface, a collection of other agents, the Internet, or perhaps a combination of all the above) and respond within a given period of time to the changes occurring during this time.
- **Pro-activeness:** Agents are not limited to acting in response to their surroundings but are able to manifest types of behaviour directed towards goals through their own initiative.

A more formal definition would be as follows:

**Definition 2** *An agent is a physical or abstract entity that is able to perceive its environment through sensors: it is able to assess such perceptions and make decisions through simple or complex mechanisms; it is able to communicate with other agents to obtain information, and it is able to act -through executors- on the milieu in which it is working. In particular, intelligent agents are considered a series of entities that attempt to mimic the process of human reasoning or behaviour. Agents can be used in different areas to facilitate tasks for users, such as data acquisition, supervision, the filtering of information, etc. [210]*

There are other definitions of the term agent. One of those most widely accepted can be found in [309]:

**Definition 3** *The notion of agent appears as a tool for analyzing systems, not an absolute characterization that divides the world into agents and non-agents.*

For Russel, an agent can be seen as “something” able to perceive its environment through sensors and act in that environment through actuators. Thus, Russel considers that a human agent has sensory organs such as eyes, ears, etc and that she/he has actuator organs such as hands, legs, a mouth etc. Likewise, a robotic agent has sensors to gauge pressure and temperature, cameras etc., and motors as actuators. A software agent receives keys, files, network packages as input data and acts on its environment by displaying things on screens, writing files or sending network packages.

Despite the differences found in the definitions that the various authors may give to the term agent, there is a series of characteristics that agents must necessarily have. These are:

- **Autonomy.** Agents can act without the need for external interventions, be they human or otherwise, having some control over their actions and internal states.
- **Situation.** Agents are located in an environment, whether real or virtual.
- **Reactivity.** Agents perceive their environment and act on this with the capacity to adapt to its needs.
- **Pro-activeness or Rationality.** Agents are able to take the initiative to define goals and plans that will allow them to achieve their aims.
- **Social skills.** Agents are able to interact with other agents and even with human beings.
- **Intelligence.** Agents are able to create knowledge (beliefs, desires, intentions and goals) from their environment.
- **Organization.** Agents are able to organize themselves in societies with structures similar to those defined in human or ecological societies.
- **Learning.** Agents have the ability to adapt themselves progressively to changes in dynamic environments through learning techniques.

The way in which agents are decomposed into a set of modules and the way in which such modules interact with one another to reach the desired functionality is driven by the architecture of the agent selected [236]. Several architectures have been proposed and these classify the agents living in them [370]:

- **Agents based on logic:** reasoning and decision making are performed through logic and deduction [132] [214] [112].
- **Reactive agents:** decision making is carried out through direct mapping of a situation to an action [57] [230].
- **Belief-desire-intention agent architecture (BDI) agents:** decision making depends on the manipulation of the representation of the beliefs, desires and intentions of the agent [300].
- **Agents based on layers:** decision making is performed through several software layers, each of them bearing the reasoning about the environment at different levels of abstraction [56].

Planning, or scheduling, in decision making is a key criterion when determining the functioning of agents. Planning systems use models of knowledge representation and symbolic reasoning and their modus operandi is defined by the need to satisfy certain basic goals for the elaboration of a schedule. An important disadvantage of these systems is in their application to real-time environments since these planning algorithms are sometimes unable to respond in the time

demanded by the system. Owing to this, new alternatives are being sought that will implement new models of representation and knowledge. The following classification attends to three types of architecture that differ with respect to the reasoning model:

- **Deliberative architecture:** Deliberative architectures use models of symbolic representation of knowledge and are usually based on the classic theory of planning, starting out from an initial state in which there are a series of schedules and a final state to be arrived at. These agents are endowed with a scheduling system that allows the determination of the steps to be made in order to achieve the desired goal. Intentional agents (BDI) can be implemented using a deliberative architecture. These agents are based on a series of beliefs and intentions that are used to generate plans [184]. Among these architectures, the one most widely used and studied is that based on the BDI model [300].
- **Reactive architecture:** This type lacks complex symbolic reasoning and knowledge or representation of the environment, such that the mechanisms for communication with other agents are very basic. The agents that use this type of architecture receive stimuli from their environment and react to them by modifying their behaviour and the environment itself [230].
- **Hybrid architecture:** These architectures are intermediate between the other two. Agents of this type include reactive and deliberative behaviours, generating a perception-decision-action cycle. The reactive behaviour is used to react to events that do not require complex decisions about certain actions.

Of the architectures described, special attention should be focused on that based on the BDI model. On one hand, this type of architecture has become de facto standard for models of agents and is accepted by the Foundation for Intelligent Physical Agents (FIPA) [336] and, on the other, it is sufficiently generic to allow the modeling of agents.

Along this thesis, we shall argue that the agent models of architectures cannot be based only on the internal specifications of the individual agents. A crucial aspect is the cooperation of the individual (agent) within a community (organization). Since it is a generic architecture, BDI offers the best approach for this requirement.

**2.2**Multi-agent systems

---

The usefulness of any technology, including MAS can be judged from two perspectives: (i) its ability to solve new types of problem and (ii) its ability to improve the efficiency of current solutions [300]. With this in mind, agents and multi-agent systems provide a natural way to characterize intelligent systems. Intelligence and interaction are ineluctably united concepts and the technology of agents reflects this condition very well. When speaking of MAS, we are extending the idea of a solitary agent, completing it with an infrastructure for interaction and communication. Ideally, MAS have the following characteristics:

- They are typically open and have a non-centralized design.
- They contain autonomous, heterogeneous and distributed agents, with different “personalities” (cooperative, selfish, honest, etc).
- They provide infrastructure for specifying communications and interaction protocols. The applications of the agent paradigm can be categorized in three classes [92]. Open systems, complex systems and ubiquitous systems.
- Open systems are systems in which the structure is able to change dynamically. Their components are not known a priori; they change with time and may be heterogeneous. An example of an open system is the Internet: any informatics system that is to work in the Internet must be able to operate with organizations of a very different type and, also, without constant guidance from users. This type of functionality requires techniques of negotiation and cooperation and we also find these in the domain of MAS.
- Complex systems are related to large, unpredictable domains – i.e., complex domains. The most powerful tools to tackle the complexity of these systems are modularity and abstraction. A problem to be resolved with agents can be divided into a number of sub-problems of lesser complexity, which are easier to handle. This decomposition allows agents to use the most appropriate solution of those possible to solve a given sub-problem.
- Ubiquitous systems aim at improving the use of an informatics system through the use of computers available in a physical environment, normally distributed, but doing everything in a way that is concealed from the user. These systems are more or less the opposite of virtual reality.



Where virtual reality places people within a computer-generated world, ubiquitous computation obliges the computer “to live” in the world of people [369]. The system must cooperate with the user to achieve its aims. The applications must behave like an intelligent agent.

Open MAS should allow the participation of heterogeneous agents, with different architectures and even different languages [389]. Accordingly, it is not possible to trust the behaviour of agents and it is necessary to establish controls based on social norms. To do so, developers have focused on the organizational aspects of the society of agents, guiding the process of systems development through the concepts of organization, norms, roles, etc.

Investigations focused on the design of MAS from the organizational point of view are currently in the timelight. The idea that modeling interactions with MAS cannot be related only to the agent itself and its capacity for communication is steadily gaining ground. Instead, it is necessary to have organizational engineering. The concepts of norms [386], institutions [105] and social structures [279], to be addressed below, stem from the idea that a greater level of abstraction, independent of the agent, is required: one which will explicitly define the organization in which the agents live.

The agents in a MAS based on organizational concepts coordinate with each other and exchange services and information. They are able to negotiate and arrive at consensus and they can perform other more complex social actions. Coordination and cooperation are very important in MAS. In a MAS the agents must find one another, announce their abilities and the task they are able to carry out, and request tasks from other agents.

In a MAS it is necessary to develop tasks involving communication, coordination and negotiation. For agents to be able to interact in a coherent way they must share information about their aims and tasks. Thanks to the exchange of this information agents coordinate the development of activities, and are able to negotiate in the event of conflicts arising and to plan their actions to meet a goal.

In the interactions of a MAS it is necessary to distinguish between four concepts; although these are intimately related, they refer to different characteristics of MAS. These are communication, coordination, cooperation and negotiation.

### **Communication.**

This is the skill of agents to communicate with one another; that is, exchange information and knowledge in an understandable fashion. It allows them to obtain the necessary information to decide on the sequence of actions they must carry out to meet their objectives.

Communication enables interaction among agents and for this the messages exchanged must use not only a common language but, also, the agents must be able to understand and interpret the information that is exchanged and be able to exchange it with other agents.

These languages, commonly known as Agent Communication Languages (ACLs), are inspired in the theory of speech acts [323] and have served as a basis of the repertory of basic communicative actions defined by current standards. Agents can exchange messages with one another through a normalized ACL. The FIPA [336] as the main organization that promotes the agent-based technology, has developed a current standard, called FIPA-ACL, based on the Knowledge Query and Manipulation Language (KQML). This language allows the development of a set of 22 communicative actions, grouped in 4 classes (information, the performance of actions, negotiation and intermediation).

### **Coordination.**

Coordination is a key characteristic in the development of MAS. Malone [234] [109] described the coordination of actions as a set of supplementary actions that can be carried out in a multi-agent environment to reach a goal that a single agent, with the same aims, would be unable to achieve. In [109], Ferber defines 4 main reasons for carrying out coordinated actions:

- Agents need information and results that can only be supplied by other agents
- Resources are limited. Agents can share resources to be able to carry out their actions.
- Coordination allows costs to be optimized, since it removes the development of unjustified or redundant actions.
- Agents have different but interdependent goals and hence they can achieve their objectives by benefiting from that interdependence.

### **Cooperation and negotiation.**

Cooperation is the mechanism through which agents, working together to attain a common goal, define a strategy to achieve that goal. By contrast, negotiation moderates the coordination among self-interested agents able to reach binding agreements. Negotiation allows joint coordination decisions to be made through explicit communication [250]. These mechanisms are inspired in models taken from the social sciences, and especially from economics, where special attention is paid to strategic negotiation and game theory.

MAS attend to the interactions of the agents comprising them. These agents form part of a collection and can coordinate their knowledge, aims, skills and plans together to perform an action or solve a global challenge. In all

systems there should be a process of rationalization for the coordination of the set of agents. In general, in these systems the agents -with their beliefs, desires and intentions- build up the problem and the schedule or sequence of actions required to solve it. Coordination is a key point in the development of this investigation. According to Ferber [109] agents share resources, demand optimization of the costs of the system, and are able to reach their objectives and become adapted in an independent way.

## 2.3

## Societies of agents

Up to this point we have been using the concept of agents from a cooperative point of view, but always on an individual basis. That is, we have reviewed their characteristics as software entities in the environment in which they may interact and become established, their architectures and current trends. Henceforth in this chapter and along this thesis we shall address the social characteristics of agents and make an exhaustive analysis of the organizational concepts. Prior to this, however, it would seem worth clarifying the key concept: society.

**Definition 4** *An artificial society is defined as a set of artificial interrelated and interacting entities that are governed by certain rules and conditions [9].*

The concept of society is widely used in contexts of human and ecological organization, and in the present case it is used in the context of agents. The main function of a society is to allow its members to coexist in a shared environment and attain their goals by cooperating or not with the other members. The main characteristics defining a society are, for example, the norms governing it and controlling the behaviour of its members. The structure is determined by the roles, norms of integration and communication language among its members. In general, norms can be said to describe the desirable behaviour of the members of a society and establish taboos and restrictions that ensure the safety of its members. The main advantage of the development of MAS from the social point of view is that this allows the creation of systems with very different languages, together with heterogeneous applications and characteristics.

Organizations can be understood as a set of entities regulated by mechanisms of social order that pursue common objectives. The architectures that help to model and build MAS based on organizations must support frames of

coordination among agents as well as be able to become dynamically adapted to changes in their structure, goals and interactions [92].

From the business point of view, the overall behaviour of the system and the organizational aspects of the domain (stability over time, consensus about the goals and strategies to be used) are very important. However, when speaking from the point of view of MAS based on organizations these same factors are vital for the system. This shows that MAS can be understood much better if they are inspired in human social behaviour [17] [356] [388].

When MAS are considered from the social point of view, the concept of desirable behaviour becomes of crucial importance. That is, the individual behaviour of agents in a society must be understood and described in relation to the social structure in which the agents are located and the global aims of the society. Until a short time ago, MAS were mainly seen from the individual perspective, i.e., as aggregations of agents that interacted with one another, only bearing in mind how the behaviour of individual agents affected the environment and viceversa [110]. In an individualistic view of MAS, agents are individual entities in a society located in a given environment. That is, their behaviour depends only on the reactions to the environment and on the behaviour of other agents [83]. It is not possible to impose demands and goals from the global aspects of the system, something essential in business environments. However, societies of agents oriented towards organization require a collectivist view of the relationship between the agent and the environment [303]. An agent in a society needs to consider not only its own behaviour but also the behaviour of the system as a whole and how agents affect one another.

In [84], the authors propose a classification for artificial societies based on the following characteristics:

- Openness, which describes the possibilities of any agent joining the society.
- Flexibility, which indicates the extent to which the behaviour of the agent is restricted by the norms of the society.
- Stability, which defines the predictability of the consequences of actions, and
- Trustfulness, which specifies the extent to which agents can trust in the society.

Depending on the aim for which it was created, a society needs to support all these characteristics to different extents. On one hand, we have open societies, which do not impose restrictions on the agents comprising it; that is, they support flexibility and openness very well but they are lacking in stability and trustworthiness.

---

**2.3.1** Open societies

---

Open societies assume that the agents participating are designed and developed outside the sphere and design of the society itself and hence the society cannot base itself on the incorporation of organizational and normative elements in the intentions, desires and beliefs of the participating agents; instead they must represent these elements explicitly. These considerations lead to the following requisites for methodologies in engineering for the construction of open societies of agents [91]:

- Societies of agents must include formalisms for the description, construction and control of the organizational and normative elements of a society (roles, norms and objectives) instead of only the states of the agents [17] [388].
- The methodology should provide mechanisms to describe the environment of the society and the interactions between the agents and the society, and also to formalize the expected result regarding functions in order to check the general behaviour of the society.
- The organizational and normative elements of a society should be specified explicitly from an open society; they cannot depend on their insertion into the intentions, desires and beliefs of each agent [87] [274].
- Models and tools are necessary to be able to understand whether the design of a society of agents satisfies its design needs and its objectives [187].
- The methodology should provide directives about the capacity of communication and the capacity to adjust to the behaviour expected of the agents participating in the society.

Agents can enter these societies with no restrictions. They can be generated creating minimum controls, and sometimes none at all, hence bolstering the characteristics such as flexibility and the level of openness. However, they are not beneficial for characteristics such as stability and trustworthiness because there is no control over which entities enter the society and much less over the work done or actions taken.

**2.3.2** Closed societies

---

Closed societies, at the other extreme, are characterized by having featuring trustworthiness and stability, owing to their policies stipulating that no external agents can gain access to the society. Moreover, the norms governing the system are well delimited as regards the attainment of specific goals. Agents in closed societies are expressly designed to cooperate in order to achieve a common objective and they are often highly interdependent with society [388]. Each of the agents can trust in the others because there will never be outsiders in the society. Also, this type of society is characterized because each of its members pursues a common objective. This means that all of them have local goals but that each goal will make its own contribution to the general objective. Until recently, most MAS were closed.

In [187] the authors introduce two new types of agent societies: semi-open and semi-closed. These combine the flexibility of open societies with the stability of closed societies. This balance between the flexibility and stability of the results is achieved through mechanisms designed to ensure that the behaviour among agents will be ethical.

**2.3.3** Semi-open societies

---

Semi-open societies are characterized by having a certain level of balance with respect to the above features because in order to access one of them it is first necessary to enter into contact with an admission control mechanism, which will evaluate the entity requesting entry into the society. Depending on the characteristics demanded by the society and what is inferred about the entity, access will be granted or denied. In this way, ingress is only allowed for entities that seem to be trustworthy and capable in the eyes of the decider. Semi-open societies to a certain extent limit the openness and flexibility of open societies but they are able to provide greater stability and trustworthiness.

**2.3.4** Semi-closed societies

---

Unlike semi-open societies, semi-closed societies and closed societies do not allow access to any new entity, although through a mechanism that communicates

with the outside it is possible to request to creation of a new entity inside the society, which would function in the name of the external agent. Although the new agent creates functions as a representative of the agent outside the society, this new agent is not strictly identical to the exterior one; instead, it is created based on the roles existing within the society. Accordingly, it will not have any characteristic other than those of the entities making up the society and, in the long run, control will be more effective [41]. This increases the flexibility and openness of the society with no detriment to stability and trustworthiness, since the agents participating are designed following the requirement of the society and the owner of the society still controls the general architecture of the system. Semi-closed societies are as open as semi-open societies but less flexible. For example, this is the focus adopted by the ISLANDER platform, where the external agents provide an API as an interface for the institution, which regulates and controls the whole interaction [43].

## 2.4

## Organizations of agents

---

Organizations of agents of the open type are increasingly in vogue in current research. In fact, the VOs can be considered to be open systems formed by the grouping and collaboration of heterogeneous agents, where there is a clear separation between the structure and function defining how the entities behave [113] [47]. Much of the research work carried out recently has focused not only on the use of organizational structures during the design process but also on regulation and adaptation in open MAS [265] [169].

Not so long ago, MAS were designed by individual teams or in collaborative work. In these systems, which were developed for a particular domain, there would be a group of agents that shared a set of objectives. These systems were scalable only in controlled or simulated environments. The languages of communication and protocols for interaction were custom-made. That is, they were designed by developers before the agent was executed and could interact with other agents. The development platforms, as well as the techniques for design and modeling, were tailor-made, inspired by the agent-oriented paradigm rather than by the use of standardized methodologies [227].

Currently, MAS allow the participation of heterogeneous agents, designed by different teams and organizations. All agents must be able to participate in these systems, developing public, standardized types of behaviour. However, these systems continue to be developed for domains of specific applications. The

systems developed are implemented with specific methodologies and organizations that include templates and patterns for different agents and organizations. The development of programming languages and specific tools is increasing, which allows the application of techniques with a formal specification. The semantic elements are of great importance, for example, for coordinated actions between heterogeneous agents to be performed.

In the not-too-distant future (current trends in research follows these lines), we shall see the development of MAS in open domains involving the participation of heterogeneous agents designed by different teams and organizations. The agents which participate in these systems will be able to learn suitable types of behaviour for participating in any interaction that might arise during execution. Moreover, the identification of protocols of interaction and communication will be selected automatically.

What but a short time ago was seen as a mid- to long-term future [227] is now almost a reality: agents that are able to form or belong to coalitions that are created dynamically and whose properties (positive and negative) are not defined a priori but emerge along the execution process of the agents.

Two definitions of VO are:

**Definition 5** *An organization will provide a working framework for the activity and interaction of agents through the definition of roles, expectations about behaviour and authority relations, such as control [129].*

**Definition 6** *An organization is a collection of roles that maintain certain relationships with one another and that take part in patterns of interaction with other roles in an institutionalized and systematic fashion [389].*

The structuralist theory [333] holds that systems are closed when they are isolated from external variables and are deterministic. Thus, the system requires that all the variables be known and that they be controllable and predictable. The organizational efficiency of this type of system will always prevail if the organizational variables are controlled within certain known limits. However, virtual organizations have all the characteristics of open systems. Some basic characteristics by which organizations are seen as open systems are [333]:

- In growth
- In the fact of becoming more complex as they grow
- In the fact that on becoming more complex their parts demand increasing independence
- Because their life is longer than the lives of their component units



- Because in both cases there is increasing integration accompanied by increasing heterogeneity

Change, independence and heterogeneity are concepts of open systems and are also manifest in societies. Schlein [315] proposes a list of aspects that should be taken into account in the creation of an organization and that jointly define the concept of “social model” from a general perspective:

- The organization should be considered an open system.
- The organization should be conceived as a system of multiple objectives or functions.
- The organization should be viewed as a constitution of many subsystems that interact dynamically with one another.
- Since they are mutually dependent subsystems, a change in one of them will affect the others.
- The organization exists in a dynamic environment that includes other systems
- The multiple links between the organization and its environment hinders the definition of the frontiers of any organization.
- Social models defend the design of MAS inspired in social theories and concepts such as norms, social conventions (or customs) or organizations. Accordingly, the organizational structure is appropriate for designing mechanisms of coordination in MAS [128] [280].

Now focusing on the computational paradigm, systems can be seen –and in a natural way- in terms of entities (commonly, agents) that provide and consume resources [226] and that have probably been designed by different development teams and that can enter or exit an organization at different times and for different reasons. Additionally, they may form coalitions or organizations with each other and have the same aims. Current trends clearly lead to the VO paradigm [111]

**Definition 7** *A Virtual Organization is a set of individuals and institutions that need to coordinate their resources within certain institutional limits [113] [47].*

Accordingly, a VO is an open system formed by the grouping and collaboration of heterogeneous entities, where there is a separation between the form and the function defining the behaviour of the agent.

Multi-agent technology, which allows the dynamic formation of organizations of agents, is particularly well-suited to the development of this type of system. The modeling of organizations based on open MAS not only makes it possible to describe the structural composition of the system (for example, roles, agents, groups, tasks, plans and services) but also the norms for the control of the behaviour of the agents, the dynamic entry/exit of the components, and the formation, also dynamic, of groups of agents. Research on organizations of agents varies, ranging from basic concepts such as groups, communities, roles, etc. [159] [111] [94], the Theory of Human Organization [14], structural topologies [159] [14] to normative research, including the representation of norms [224], deontological logics [93] and institutional approaches [103] [105]. The development of open MAS is currently a recent field of enquiry in the agent paradigm and will allow this technology to be applied in new and complex domains [29] [157] [185]. In this sense, it is necessary to investigate new methods for modeling virtual organizations based on open MAS and innovate in techniques that will endow the virtual organization with capacity.

Following on with the organizational perspective, a system is described by a social structure and a set of norms that govern interactions among agents. Such a description identifies the functional components of the system (agents), their responsibilities (the tasks they must perform) and their resources (knowledge, hardware, software, tools, etc) and the relationship between them (communication, assignation, etc). Below we describe some of them.

### **Social Entity**

Organizations are formed by components or social entities that in turn may be composed of a specific number of members or agents. According to [280], these entities are as follows:

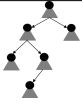
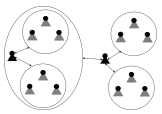
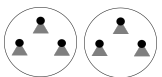
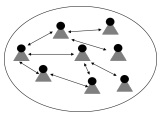
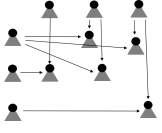
- They have responsibilities - that is, a set of sub-tasks that they must carry out- included within the aims of the organization.
- They have and they consume resources. The components have certain resources with which to perform their tasks. The resources required by a component will depend on the role it is playing at that moment in the organization.
- They are structured following given patterns of communication,
- They attempt to attain the global aims of the organization, and
- They are regulated by norms and restrictions.

### **Structure**

The entities of an organization are not independent of one another. They interact. Commands, information etc, are units that are passed between them.

In general, these relations are not given individually within an organization; instead, a conjunction of relations among groups of entities is required.

In this conjunction, different aspects must be considered: roles, topology, authority relations. All these will determine the structure of an organization. Structure can be defined as the distribution, order and interrelationship of the different parts comprising the organization. In this structure, agents are ordered and communicate, depending on the topology defining the system. There are different organization's topologies that Rodríguez presents in [303] and are shown in the table 2.1.

<p><b>Hierarchy:</b> Agents are ordered in a tree-type structure, in which the lower levels have the basic functionality and the upper ones are responsible for decision making and control.</p>	
<p><b>Oligarchy:</b> They are nested and hierarchical structures of holons. A holon is a part of a larger entity, the result of the grouping of subordinated entities. This type of topology tends to be applied in domains where the objectives are decomposed recursively into subtasks.</p>	
<p><b>Coalition:</b> These are transient organizations of agents formed to achieve a specific objective, which usually provides certain benefits and reduces costs. Coalitions are dissolved when the aim has been reached, since there is no longer any need for grouping or when a critical number of agents abandons the grouping. Internally, it is usually represented as a flat structure or with a leader (group representative) and externally as a single atomic unit.</p>	
<p><b>Groups:</b> These are groupings of cooperative agents, who work together to achieve a common goal. Thus, they maximize the usefulness of the team. The representation of aims, beliefs and schedules is carried out at team level. The groups are usually applied when solving problems can be achieved better with joint work. The groups imply greater redundancy and flexibility for uncertain environments, although also an increase in communication needs.</p>	
<p><b>Matrix-like organizations:</b> In this type of organizational topology, an agent can be controlled by more than one supervising agent. Accordingly, it is necessary to use commitment assessment mechanisms and the resolution of local conflicts. It is like a grid-type structure in which the Manager agents are located around agents.</p>	

<p><b>Federations:</b> These are grouping of agents with a representative. The members of the rest of the organization interact only with this representative and lose some of their autonomy. This “representative” agent also acts as an intermediary between the group and the outside world, carrying out functions such as (i) broker: this distributes tasks among the members of a group; (ii) monitor: this facilitates interactions among different agents (it establishes contacts); (iii) mediator: this controls the states of agents and reports events, and (iv) embassy: this controls the communication between external agents and those of the federation (it translates ontology).</p>	
<p><b>Congregations:</b> These are groupings of agents with similar or complementary characteristics. In this case, they do not involve the achievement of a specific goal but do facilitate the search for suitable collaborators for that goal to be attained. Accordingly, this type of topology is usually considered for long-term objectives.</p>	

**Table 2.1:** Possible organization topologies

### Functionality

The functionality of an organization is determined by its mission; that is, by global aims that describe the reason for its own existence. The mission defines the strategy, the functional requirements (done by the organization) and those of interaction (how it is done). The objectives can be classified as follows:

- Functional: of each organization group or unit
- Operative: of the agents, their plans (the tasks they will carry out)

### Norms

The social norms define the consequences of the actions of agents:

- Restrictions regarding the organization
- Obligations, sanctions to be applied.
- Control over external access
- Decomposition
  - Actions that elicit the activation of the norm.
  - Set of obligations acquired by the agent.

- Actions that should be performed to remove the obligation.

### Environment

The environment defines what exists around the system: resources, applications, objections, assumptions, restrictions, stakeholders (providers, clients, beneficiaries). By defining the environment, it is possible to establish a list of roles with respect to the elements of the environment: mode of access (reading, interaction, the extraction of information), access permits, etc.

### Dynamicity

The organizational dynamics is related to the entry/egress of agents, with the adoption of roles by these, the creation of groups, and the control of behaviour. In the definition of the dynamics of an organization, the following must be specified:

- Regarding the entry of agents: when agents are allowed to enter the organization; what their position in the organization will be; the processes involved for expelling agents displaying anomalous behaviour. For example, in the model defined in [104] there is an Institution Manager agent, which authorizes the entry of external agents into the institution.
- Regarding role adoption: how the agents adopt a given role; the association of agents with one or more roles. For example, in [104] a description is given of the transitions between scenes as a function of roles; exchange of roles.
- With respect to the dynamic creation of groups: the definition of federations, coalitions, congregations etc.
- Finally, with regard to behaviour control: how to control the conformity of the behaviour of the agents with the social norms. For example, in [104] there is a social layer that guarantees that interactions will occur in accordance with the norms.

In this chapter we have explored the VOs, which are considered to be the most suitable MAS development for open environments since they show flexibility, openness, heterogeneity, capacity to adapt and, above all, with their organization capacity they are able to emulate the functioning of human society. Accordingly, this specific technology was chosen.

Moreover, we have established the main characteristics that must be taken into account to apply an organization and open policy: the different topologies, the norms, the adaptation, the dynamicity, etc. These issues will be applied in the development of the model and the PANGEA agent-platform.

In the next chapter, we present the main issues and concepts of the RTS that must be also integrated in the proposal of this dissertation.

# 3

## REAL-TIME SYSTEMS

---

*The real-time computation plays an increasingly important role in computer systems, since the future directs the evolution of these systems towards a greater interaction with the real environment. This environment sends stimuli in real-time expected to be captured and processed by the systems within temporary constraints that are the most important issue to take into account in a RTS. From the analytical point of view, a RTS consists of a set of tasks and each task at the same time is composed by a set of activities or subtasks that run concurrently cyclic or according to a schedule. An ordinary real time task is characterized by three main parameters: the runtime, the period and the deadline. Moreover, the task execution must be controlled by any scheduling algorithm, which is responsible for organizing the execution time between periodic, aperiodic or sporadic tasks. In this chapter, we introduce all these basic concepts and present some related works of MAS in real-time environments. As mentioned in the previous chapter, this one is also essential to analyse the characteristics of a RTS in order to adjust their functionality to the constraints of the scheduling model mentioned as one objectives of this dissertation.*

### Contents

---

<b>3.1</b>	<b>Introduction</b>	<b>36</b>
3.1.1	Classification of real-time systems	36
<b>3.2</b>	<b>Models of tasks in real-time systems</b>	<b>37</b>
<b>3.3</b>	<b>Real-time scheduling</b>	<b>39</b>
3.3.1	Local scheduling within computational nodes	40
3.3.2	Global scheduling in a real-time system	43
<b>3.4</b>	<b>Distributed real-time systems</b>	<b>47</b>
3.4.1	Multi-agent systems in real-time environments	50
<b>3.5</b>	<b>Conclusions</b>	<b>54</b>

---

## 3.1 Introduction

---

In a RTS correction depends not only on the logical result of a computation but also on the moment at which that result is obtained [341]. According to this definition, apart from the necessary logical corrections to the system it is also necessary to take into account its temporal correction, which is expressed by a set of temporal restrictions imposed by the environment. Accordingly, an RTS must ensure the fulfillment of such restrictions within a deadline to be met.

Typically, an RTS can be seen as a set of events/responses, since the system normally performs a cyclic process in which, initially, data from the environment are read. Then, a response is obtained, based on those data, and finally the environment is acted on depending on the solution obtained [352]. Currently, most RTS are implemented with concurrent applications formed by a set of tasks, each of which must solve a given part of the problem. The tasks tend to be characterized by priority, a deadline and in the worst case a limited computing time.

### 3.1.1 Classification of real-time systems

---

The deadline defines the longest period of time since activation in which the tasks should have been performed and, therefore, a response obtained. If the answer is obtained after that time, it will probably not be of any use. In fact, on the basis of what happens if a solution is obtained after the deadline, it is possible to differentiate between two types of RTS [337].

#### **Soft real-time systems**

Soft RTS, also known as uncritical, are characterized by the fact that the completion of a task after its deadline may negatively affect the quality of the results of the task.

Normally, when a computer controls some type of external device there are certain temporal requirements. Embedded informatics systems often belong to this type. The required response time is typically shorter than that of interactive systems, and is often found in the 10-100 millisecond range. To maintain control over the external devices it is important that the system should meet the deadlines. In uncritical (soft) RTS, however, occasional failures to



meet deadlines can be tolerated. A typical example of a soft RTS described in [237] is the control of a telephone. To serve a client efficiently, this control must respond rapidly to the actions of the person making the call. When the person making the call lifts the handset the telephone must generate a dialing tone and must be prepared to receive a telephone number. If the telephone fails to meet these requirements the person calling might think that the service is unavailable at that particular moment. In fact, even though the system fails, the integrity of the system remains unaffected.

The dividing line between interactive systems and soft RTSs is often difficult to pinpoint. A system of audio reproduction, for example, could be considered a soft (non-strict) RTS owing to the temporal constraints of its response.

### Hard real-time systems

Hard RTSs, also known as critical systems, are characterized by the fact that the execution of a task after its deadline may negatively affect the quality of the results of the task. Some RTSs comprise processes with critical temporal requirements. The loss of a deadline in this type of system often leads to complete systems failure [55]. Many automatic control systems belong to this category. This type of system, for example, may be an aerial navigation system, the control of a robotic arm, or the control of an industrial plant. Such systems usually require very rapid response times, in the order of milliseconds. The loss of some deadline in this type of system may lead the control algorithms to become unstable, with catastrophic results.

As well as this classification of RTS, Bernat et al. [42] have proposed a third type, called weakly critical systems. In these, a diffusion of the temporal limits in which the deadlines may not be reached is proposed. These systems are critical in the sense that they must guarantee beforehand that their temporal specification will be fulfilled, although they do have some non-critical activities associated with average response times.

## 3.2

## Models of tasks in real-time systems

---

As mentioned above, an RTS is implemented by a set of tasks. If one considers the importance of the work performed by each task in the maintenance of the integrity of the system, the existing classification is:

- **Critical:** A failure in one of these tasks may lead the system to a catastrophic situation. The failure may be due to an error in the program code or to a delay in achieving the result.

- Acritical: These are tasks that collaborate in the functioning of the system without compromising its integrity in the event of failure or non-execution.

By contrast, attending to the temporal characteristics of each task, the tasks can be classified as [237]:

- Periodic: These are tasks that are repeated during their execution in the system. They are defined by a period that indicates how often the task will be activated again. Each activation must be completed within a set response time. There are three parameters that characterize these tasks,  $\mathcal{T}_i = (C_i, D_i, T_i)$  [237] where:  $C_i$  is the computing time, in the worst case, required in each activation;  $D_i$  is the relative deadline: the maximum time available for activation to complete task execution. If for some reason the result of the execution is not available in  $D_i$  units of time after activation has been started, a failure is said to have occurred and  $T_i$  is the activation period, the distance in time between two consecutive requests.
- Aperiodic: These tasks,  $J_i$ , must attend to events that arise unpredictably, such as an order given by an operator or the appearance of an obstacle. Depending on the urgency, they can be subdivided into the following groups:
  - With no deadline: Their execution is not critical. They have no definitive limit in which they must be completed. They can be used to improve the precision of the control actions or they can generate reports about the state of the system. Although they have no time limits for completion, it is desirable that the scheduler execute them as soon as possible in order to achieve a short response time. The only parameters that characterizes them is the computation time,  $J_i=C_i$ .
  - With a firm deadline: As in aperiodic tasks with no deadline, these are not critical either. The result of their execution is only of use to the system if the tasks end before its deadline. If the deadline cannot be met, the planner must cancel their execution as soon as possible in order not to waste computing time. This type of task is based on two parameters:  $J_i = (C_i, D_i)$ .
- Sporadic: These tasks have a maximum deadline and failure to meet this may be catastrophic for the system [248]. Accordingly, the planner must guarantee correct execution. The parameters of this type of task are the same as in the case of the periodic tasks except that the period represents the minimum distance between two consecutive activations instead of indicating the time between activations

In most of the existing works [221] [215] [248] [338] [382], in order to guarantee a correct execution the authors base themselves on explicit knowledge of the periods of the tasks or, at least, on knowledge of the minimum times between requests. Accordingly, the tasks considered to be critical must be transformed into periodic ones, if they are not so already. In particular, sporadic tasks should be considered as though they were periodic during the designs phase and in the analysis of the system [237].

Apart from the temporal characteristics of the tasks, in the first real-time studies the following series of restriction was assumed, although as the theory has evolved and matured, some restrictions have been eliminated:

- All periodic tasks are critical.
- All tasks can be eliminated from the processor at any moment.
- All tasks begin their execution the moment they are active.
- $C_i \leq D_i \leq P_i$
- The overload due to the change of context is considered to be negligible.
- There is no dependence between tasks; that is, the tasks do not share common resources and there is no precedence relationship between them. As set of tasks with precedence relations can be transformed into another set with no precedence problems by modification of the deadlines [70].
- No task is suspended voluntarily, with the exceptions of expulsions caused by the planning algorithm.

In the context of real-time applications, the actions are tasks (also called processes) and the organization of their execution by the processors of the computational architecture is called the real-time scheduling of the tasks. In the following section, we shall go further into scheduling.

### 3.3

## Real-time scheduling

---

Within real-time planning it is necessary to distinguish between local planning and global planning. Local planning refers to the order of execution of the tasks in a node and global planning refers to the distribution of tasks between the different nodes comprising the system. Since this thesis covers both aspects, below we review the most relevant concepts of both types of scheduling.

### 3.3.1 Local scheduling within computational nodes

Since one of the relevant points of this thesis is a scheduling model for VOs in real-time environments, it is important to establish the bases and the evolution of the concept.

We represent by  $\mathcal{T}$  the set of periodic (and sporadic) tasks that comprise the load of the processor. We shall assume that  $N$  is the cardinal of the set  $\mathcal{T}$ . The most important parameters of the periodic load is the processor use factor,  $\rho$ , which is defined as:

$$\rho_i = C_i/T_i \quad (3.1)$$

A set of tasks,  $\mathcal{T}$ , is schedulable if and only if there is some scheduling algorithm able to meet the deadlines of all the activations of  $T$ . It should be noted that the condition of schedulability is a property of the set of tasks and not of the scheduler. Starting out from these definitions, it is clear that there are sets of tasks that can be scheduled but that a given scheduling algorithm might be unable to plan correctly. A scheduler is said to be optimal if and only if it is able to plan all sets of schedulable tasks correctly.

Scheduling must fulfill the temporal restrictions of the application. The procedures governing the ordered execution of tasks are referred to as the scheduling policy.

In classic informatics systems, the main objective of the scheduler is usually to minimize the response times of the tasks; i.e., the feedback-driven scheduling [342], or sharing processing times among tasks in the most equitable way possible; Round Robin (RR), and, finally, avoiding context changes with First Come First Served (FCFS). None of these policies is appropriate in RTs [341] [340]. The main aim of any real-time scheduling policy should be to ensure that all the tasks fulfill their temporal restrictions. As well as this basic objective, it is desirable that the scheduling policy consider the following issues [237]:

- Management of the resources when they are shared among the different tasks of the system
- Recovery from failures
- Minimization of the response time of all the tasks.
- Execution of acritical tasks with no deadline.

- Replacement of tasks during the time of execution (mode change).
- Consideration of the temporal load of the scheduling algorithm itself, of the change in context, etc.

If possible, it should be easy and simple to use this in practical work. The only way to guarantee that a given scheduling algorithm will plan a specific set of tasks correctly is through analysis and through analytical or exhaustive methods. As stated in [382], simulation is not a valid method for guaranteeing correct scheduling, since "*simulation shows the presence of errors, but not their absence*" [95].

Accordingly, to determine a priori whether a RTS is temporally correct, i.e. whether it fulfills the temporal restrictions imposed in the specification of the system- schedulability tests are implemented. We say that a set of tasks in real time is schedulable if each task completes its execution before its maximum deadline is reached. Among the basic analytical algorithms for schedulability, we have Rate Monotonic (RM) [216], Deadline Monotonic Analysis (DMA) [366], and Earliest Deadline First Scheduling (EDF) [221].

Since we are interested in ensuring that the deadlines will be met in all cases to fulfil the constraints of the hard real-time systems, in these tests a pessimistic analysis is used to guarantee that there will be a schedule for the execution of tasks in which the execution times foreseen for the worst case are within the deadline demanded for each task [237]. It seems clear that the time of execution for the worst case, or WCET, of each task is very important for the construction and verification of the RTS. In most works addressing the issue of the scheduling of RTSs, it is assumed that the WCET of all the tasks is known, although as we shall see along this work this information is very hard to obtain and in many cases it is only possible to work with predictions. In sum, a scheduling model is defined by:

- A scheduling algorithm, which determines the order of access to the system's resources.
- A method of analysis, which allows the temporal behaviour of a system to be calculated. It checks whether the temporal requirements are guaranteed in all the cases possible (the worst case will be studied).

#### 3.3.1.1

#### Scheduling models

---

The simplest planning is static planning, also known as executive cyclic scheduling or time-driven scheduling [205] [390] [382]. It is the class of schedule most

widely used nowadays. Most of the planning work in this type of scheduling is done during the design of the system. A table is constructed (also called static plan or calendar) indicating the instants at which each task should start being executed and when it must end. The guarantee of correct execution of all the tasks (schedulability analysis) is done implicitly during the construction of the schedule itself. The main advances of this policy can be said to be predictability of efficiency in the time of execution. By contrast, among the disadvantages of this model are inefficiency in predicting aperiodic events, the reduced flexibility for modifying the load characteristics (a small change in one of the tasks implies having to modify the whole schedule) and a possibly excessive size of the table.

Despite the above, J. Xu and D.L. Parnas made a fiery defense of cyclic schedulers in [382], and championed the notion that some types of problem can only be solved with this type of algorithm.

### **Scheduling by priorities**

Unlike cyclic schedulers, there are planners controlled by priorities. In these, the need for a preset plant is eliminated, allowing decisions to be made by the scheduler.

The basic scheme of functioning is as shown in [237]. At each moment (typically when a task arrives or is completed) the scheduler chooses the task whose priority function is higher than that of all the other active tasks. Depending on what the priority function is we shall have the following: schedulers with fixed priorities if a fixed priority is initially assigned to each task, and this value is used as a priority function, and schedulers with dynamic priorities, in cases in which the priority function has as parameters data relating to the load present at the moment of its evaluation.

These methods require a previous test that will guarantee that the particular planning policy employed will be able to schedule all the critical tasks of the system correctly.

Within the system of fixed priorities, the two most widely used methods are Fixed Priority Scheduling (FPS) and EDF [213]. The FPS is also known as RM. Each task has a static and set priority that is known a priori. The tasks are executed in the order determined by such priority. The priority is determined by temporal restrictions (short deadline corresponds to higher priority), not by its importance for the correct functioning of the system. With the EDF the tasks are executed in an order stipulated by the task's deadline such that the next task to be executed will be the one that has a shorter deadline or one closer in time. In recent years, scheduling schemes based on fixed priorities have attracted much attention in the scientific community and have now reached a high degree of maturity and recognition [338] [23] [326] [214] [213] [59] [21].

In [23], it is possible to find a historical study of the advances made in scheduling by fixed priorities.

In [61], the following reasons are offered to explain why FPS is considered a better model:

- It is easier to implement. The parameter used for the scheduling, priority, is static at the moment when the scheduling is made (although it may vary in successive iterations), whereas with EDF more computation time is required and hence this may delay planning; also, its tendency to produce deviations is greater.
- It is easy to introduce tasks without a specific deadline simply by assigning them a priority. However, assigning an arbitrary deadline is more abstract and artificial.
- The notion of priority can include other aspects, such as an evaluation of the urgency or importance of the task. Nevertheless, the deadlines cannot be changed.
- In a situation of overload, FPS shows more reasonable behaviour since poor functioning will occur in the tasks with less priority. However, in EDF more critical tasks with longer deadlines may be affected, leading to a domino effect in the other tasks and causing much greater delays.

Despite the above, EDF does have one advantage over FPS. The node use factor is greater but in a hard RTS the main aim is to ensure that the deadlines of all the tasks will be met, and hence having a lower use factor is considered to be a reasonable option. Accordingly, with these considerations the FPS was chosen although the system of priority application was modified slightly in order to take into account these critical tasks when the system does not have more computation nodes to redistribute the agents and their services and hence control the situation of overload while the system is being re-scheduled or adapted. In chapter 5 we detail all the characteristics of this model since it was chosen as the starting point for the work carried out in the preparation of this thesis.

### 3.3.2

### Global scheduling in a real-time system

The global scheduling of a processing system has the main aim of obtaining a better yield in the processing capacity of the system. This can be achieved thanks to the existence of a greater temporal distribution of the load between the processors: unlike distributed systems, with global scheduling it is possible

to avoid having a very busy processor, with a long queue of pending tasks, while other processors are idling. In this way it is possible to advance in the work, thereby allowing the system to be more available to attend to new demands [31].

As in the case of monoproductors, the assignation of priorities may be static (FPS) or dynamic (Dynamic Priority Scheduling (DPS)). However, FPS loses a great advantage -determinism- since even though the priorities are fixed if some unexpected event occurs, such as could be the mere arrival of an aperiodic task, the whole assignation of periodic tasks to nodes may change, and even the schedulability of the system may become compromised. By contrast, dynamic assignation of priorities seeks greater flexibility, in accordance with the aim of global scheduling [31].

The assignation of fixed priorities, based on some criterion, orders all the tasks of the system and assigns them decreasing priority. This priority is global and persists throughout the execution of each of the activations of a task. In certain cases, these methods may encounter problems with load distribution. For example, when there are "large" tasks to be performed, where "large" depends on the use factor of each algorithm, they are assigned a wrong priority. Thus, most algorithms of this family will be variants of the RM with exceptions for the case of "large" tasks.

For these reasons, we have chosen to use the scheduling with static assignation of priorities. Next, we will present the most important schedulers of this type.

### Global Rate-Monotonic

The Global Rate-Monotonic (GRM) consists of ordering all the tasks of a system from smaller to greater periods and assigning them decreasing priority. During the time of execution of active tasks, they will join a single global queue ordered according to the priority assigned and the global planner will extract them as processors become freed-up. Unfortunately, this scheme has a use limit equal to zero. The table 3.1 and the figure 3.1 show an example of this effect, which is call the "Dhall Effect" [88] [89].

Task	Period	Deadline	WCET
$\mathcal{T}_i, i \leq m$	1	1	$2\varepsilon$
$\mathcal{T}_{m+1}$	$1 + \varepsilon$	$1 + \varepsilon$	1

**Table 3.1:** Example Dhall Effect



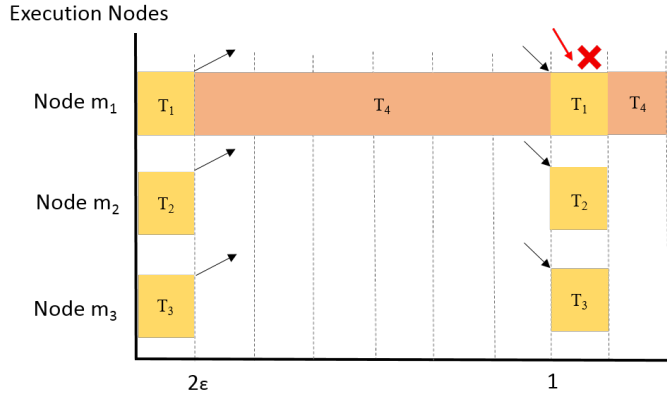


Figure 3.1: Dhall Effect Behaviour

A set of  $m + 1$  tasks in  $m$  processors is scheduled with GRM. The  $m$  first tasks have a period  $T_i = 1$  and a poorer computation  $C_i = 2\varepsilon$  ( $\varepsilon$  being small), while the last task has a period  $T_{m+1} = 1 + \varepsilon$  and a  $C_{m+1} = 1$ . If all the tasks are activated when  $t = 0$ , the first  $m$  are executed since they have a smaller period. The last task would end at  $t = 2\varepsilon + 1$ , losing its place,  $T_{m+1} = D_{m+1} = 1 + \varepsilon$ . If  $m \rightarrow \infty$  and  $\varepsilon \rightarrow 0$ , the use of the system tends to zero  $\rho(\mathcal{T}) = m2\varepsilon/1 + 1/(1 + \varepsilon)$  if  $\varepsilon \rightarrow 0$  and hence  $\rho(\mathcal{T}) = 1$  and  $\rho = 1/m$  and it is not schedulable.

$$\sum_{i=1}^{m+1} \rho_i = m \frac{2\varepsilon}{1} + \frac{1}{1 + \varepsilon} \rightarrow 1 \text{ if } \varepsilon \rightarrow 0 \quad (3.2)$$

It should be noted that this example is also applied to GRM and EDF. In general, all algorithms that fail to take into account the maximum load of the individual tasks may undergo the Dhall effect.

### Other algorithms for global scheduling

The Table 3.2 shows a summary of the work carried out in [31], where different global scheduling algorithms are evaluated.

As can be seen, the Dhall effect can only be avoided completely with the PriD [138] and Proportionate-Fair Planning (Pfair) [34] schedulers. However, with the PriD, the maximum load is unknown and neither does any method of evaluation exist. Accordingly, the best of these algorithms is the Pfair, discussed below.

### Proportionate-Fair planning

	Planner	Load limit	Dhall Effect
Static Priorities	GRM	0	Yes
	AdaptativeTkC	Variable	Sometimes avoided
	RM-U	1/3	Avoided because only small loads
Dynamic Priorities	EDF-US	1/2	Avoided because only small loads
	GFB	$1 - \lambda_{max}$	Avoided if small loads
	Baker	?	?
	EDF	?	Avoided depending on $k$
	PriD	?	Avoided
	M-TBS	$1 - \lambda_{max}$	Avoided if small loads
	M-CBS	$1 - \lambda_{max}$	Avoided if small loads
	Pfair	1	Avoided

**Table 3.2:** Overview of the main global scheduling algorithms

Pfair [34], is based on the idea of executing tasks proportionally and accurately by using fractions of time or quanta. The execution time devoted to a given task during a period of time is proportional to the weight of the task  $C_i/T_i$ . It is a scheduling with preemption but this can only occur at certain moments of time, at the end of the quantum. This means that the time devoted to a task in a period of time  $t$ , is approximately proportional to its weight  $t(C_i/T_i)$ , with a maximum error of the order of the quantum. The algorithm will minimize the maximum error committed  $lag$ , expressed in the following formula:

$$lag(\mathcal{T}_i, t) = y(C_i/T_i) - dedicated(\mathcal{T}_i, t) \quad (3.3)$$

If the task  $\mathcal{T}_i$  is executed, the  $lag(\mathcal{T}_i, t)$  will decrease by  $1 - (C_i/T_i)$  whereas if it is not executed it is increased by  $C_i/T_i$ . An algorithm is said to be *pfair* if and only if for all  $\mathcal{T}_i$  the following is satisfied:

$$-1 < lag(\mathcal{T}_i, t) < 1 \quad (3.4)$$

Pfair algorithms use dynamic priorities to minimize this error at all times, executing the  $m$  tasks with the highest priority. The following scheme describes the PF algorithm and is *pfair*:

The condition for a set of tasks to be schedulable with this algorithm is that the load should not surpass the processing capacity of the multiprocessor. In other words, it is an optimum algorithm. The greatest drawback is the high number of appropriations if the quantum is small, or greater error and poor use of the computational capacity when the quantum is large. Additionally, the

---

**Algorithm 1** Steps of the PF Algorithm

---

- 1: Execute all the urgent tasks; that is the tasks  $\mathcal{T}_i$  such that  $lag(\mathcal{T}_i, t) > 0$  and  $lag(\mathcal{T}_i, t + 1) \geq 0$  if the task were executed.
  - 2: Do not execute the tasks  $\mathcal{T}_i$  such that  $lag(\mathcal{T}_i, t) < 0$  and  $lag(\mathcal{T}_i, t+1) \leq 0$  if the task were not executed.
  - 3: For the other tasks, execute the task that has the smallest  $t' > t$  such that  $lag(\mathcal{T}_i, t') > 0$
- 

algorithm for recalculating the priorities, despite being executed in polynomial time, must be executed at each quantum.

The main drawbacks of such planners are [31]:

1. Require the quantum level synchronization.
2. Part of the quantum can be wasted because some tasks finish before.
3. Produce a high number of context switches; (iv) can produce a large number of migrations.
4. Planning decisions are grouped at the beginning of the quantum.

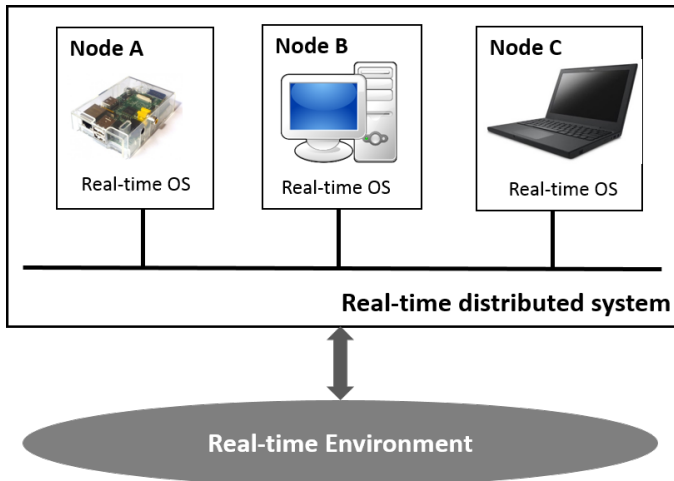
Moreover, in a planning test  $U \leq 1$  there is no possibility of introducing sporadic tasks that can arrive frequently in highly changeable systems. In the solution proposed in this thesis the Dhall effect is not produced because it takes into account the maximum load of the individual tasks. The replanning is also avoided in each quantum and so, the number of migrations is reduced.

**3.4**Distributed real-time systems

---

Many of the problems currently found may require complex solutions that are difficult to approach with a single computer, either owing to the time costs for the task to be performed or owing to the excessive complexity of the solution of objectives that are too complex if only one computer is used. In turn, some systems require the ability to be able to reset themselves in the case of a failure causing the whole system to fail, and even to prevent that failure from occurring. Accordingly, a solution via centralized systems in which the complete solution to the problem is obtained with a single computer may be a bad approach. Distributed systems have been developed to approach complex solutions by distributing the load of the solution among different nodes, such that the complex problem is divided into simple tasks that can be addressed individually at each node. We define a distributed system as

a collection of autonomous computers connected in a network and with the appropriate distributed software for the system to be viewed by users as a single entity able to provide computational facilities [79].



**Figure 3.2:** Distributed real-time system

Figure 3.2 shows the general structure of a distributed RTS. In a distributed RTS, the following characteristics are desirable [79]:

- **Resource sharing:** the resources in a distributed RTS are physically encapsulated in one of the computers and can only be accessed by other computers via communication (the network). For resource sharing to be effective it must be managed by a program that offers a communications interface, allowing the resource to be accessed, manipulated and updated reliably and consistently.
- **Openness:** An informatics system is open if the system can be extended in different ways. A system may be open or closed with respect to hardware extensions (adding peripherals, memory or communications interfaces, etc) or with respect to software extension (adding characteristics to the operating system, communications protocols and resource sharing services, etc). The openness of distributed systems is first determined by the degree to which the new resource-sharing services can be added without damaging or duplicating those already present.
- **Concurrency:** In a distributed system based on the resource-sharing model the possibility of parallel execution occurs for two reasons: (i) Many users interact simultaneously with application programs. (ii) Many server processes are executed concurrently, each of them responding to different

requests from the client processes. The first case is less conflictive since interaction applications are normally executed in an isolated fashion at the user's workstation and do not conflict with the applications executed at the workstations of other server users. The second case arises owing to the existence of one or two server processes for each type of resource.

- **Scalability:** Distributed systems operate effectively and efficiently at many different scales. The smallest scale consists of two workstations and a file server, while a distributed system built around a simple local area network could involve hundreds of workstations, several file servers, printing servers and other servers for specific aims. Neither the software of the system nor that of the application should change when the scale of the system is increased. The need for scalability is not only a problem of the network or hardware capability but is tightly linked to all the design aspects of distributed systems. The design of the system should explicitly recognize the need for scalability; otherwise there will be serious limitations.
- **Tolerance to failure:** Sometimes, informatics systems fail. When failures occur in the software or hardware the programs may provide incorrect results or could stop running before the computing they are performing has been completed. The design of systems tolerant to failure is based on two complementary issues: Hardware redundancy (the use of redundant components) and software recovery (the design of programs able to recover from failures). In distributed systems, redundancy can be approached at a lower level than the hardware; the individual servers that are essential for the continued operation of critical applications can be replicated. Software recovery is related to the design of software able to recover the states of the permanent data before the failure occurred.
- **Transparency:** Transparency is defined as the concealment from the user and applications programmer of the separation of the components of a distributed system such that the system will be perceived as a single entity instead of as a collection of independent components. Transparency exerts considerable influence on the design of the software of the system.

As seen, distributed RTSs are of great importance and cover a large number of real situations. Additionally, these systems are highly complex and many problems remain to be solved. In this sense, MAS can be seen as an alternative form for the development of RTSs. For this reason, in the next section, we discuss works addressing this area of enquiry.

**3.4.1**

## Multi-agent systems in real-time environments

Regarding improvements in the resolution of highly complex systems in distributed RTSs, recent years have seen the incorporation of the use of Artificial Intelligence (AI) techniques, and more specifically of the MAS paradigm, to offer distributed RTSs the capacity to carry out intelligent problem solving that will allow such problems to be addressed from other perspectives. The methods used in AI must be adapted for use in domains in which a real-time response is demanded. The Real-Time Artificial Intelligence Systems (RTAIS) area was created with the aim of satisfying these needs. These systems must be able to satisfy complex, critical goals in settings, probably dynamic, with temporal restrictions by means of the use of AI techniques. Further information about RTAIS can be found in [124] [255] [157].

Different approaches have been proposed to adapt classic techniques of artificial intelligence so that they will be able to cope with real-time requirements. The main work has revolved around the following aspects:

- The modification of classic algorithms of AI so that their executions will be temporally bounded, and –in this way- their execution time will be predictable.
- Software architectures of AI appropriate for operations in real-time, for example, the MAS paradigm.
- Modification of traditional planning policies in RTSs to adapt them to the capacity of using less predictable algorithms.

Recent years have seen the coining of the term Real-time Agent (RTA) to refer to agents that are able to manage temporal restrictions in some of its capacities [189]. In these agents it is necessary to take temporal correction into account; this is expressed through a set of temporal restrictions imposed by the environment. Accordingly, RTA must guarantee the fulfillment of these temporal restrictions. RTA tasks must therefore, on one hand, have their temporal behaviour clearly defined through the parameters associated with the tasks in a RTS and, on the other, it must be possible to analyze and plan such tasks via a scheduling policy that will allow us to obtain an order of task execution within the agent, bearing in mind that the tasks must be executed with their temporal restrictions taken into account.

Once an RTA has been defined, we can define a Real-time Multi-Agent System (RTMAS) as a MAS with at least one RTA integrated in it [189]. In these systems, a series of requirements must be met [332]:

- Management of a global time: In a distributed RTS it is crucial to have a global time for all the elements belonging to the system. The same is the case in an RTMAS. There must be a global time that will permit the synchronization of the different agents of the system.
- Real-time communication: In the communication process carried out between the agents of the system it may be necessary to transmit temporal concepts in the messages exchanged. Accordingly, temporal representations should be permitted in the messages exchanged in the system. Moreover, depending on the characteristics of the system, the use of efficient protocols that ensure a maximum time of message arrival and that will allow communication with a low, restricted lag (interval of time between the sending and arrival of a message) is necessary.
- Failure-tolerant execution: Real-time systems are considered to be predictable but at the same time they must be able to tolerate failure. Tolerance to failure in RTMAS should be more restrictive. The execution of the agents should be ensured after an internal failure and also after a failure in the communication process.

In the literature addressing RTAIS we find systems whose main aim is the development of mechanisms designed to support real-time agents. Several real-time MAS architectures have been proposed, and different authors have explored the ways how the scheduling of tasks offered by real-time agents within the architecture is addressed. A large part of the problem of task scheduling by real-time agents relies in the assumption that to complete a task an agent, or set of agents, may have several different options to solve the problem. Each method of solution requires a different execution time to calculate a valid result and the response will have a different level of quality from the results obtained with the other methods. It is logical to think that the longer the time available for solving the problem, the better the quality of the result obtained. This consideration is taken into account in most real-time MAS architectures because it allows a balance between the quality of the result and the amount of time required to overcome the temporal restrictions specified to be offered. Using these concepts, Garvey et al. [125] described their design-to-time scheduling algorithm for incremental decision making that provides a hierarchical abstraction of problem-solving processes. This algorithm is extended in [366] to develop a more general model, termed *design-to-criteria*, which takes into account other criteria apart from the time for task planning. These criteria range from adding costs and quality in the methods to including the concept of uncertainty as part of the decision process. An example of the use of the *design-to-criteria* model is the DECAF architecture [142], which incorporates the programming of algorithms based on this model.

The ObjectAgent architecture [349] is another example of a RTA. Created in 2001 by Princeton Satellites, this architecture is used for the control of the so-called cloud of satellites: small single-function satellites launched into space in a specific formation. All function as a single satellite with multiple functions. Each of the mini-satellites is identified with an agent with its own temporal restrictions. ObjectAgent supports real-time communication as long as the topology of the network is known and the environment is predictable. To accomplish this, ObjectAgent makes use of special mono-task agents called post-offices, whose only mission is to send messages from themselves to the agents. Thus, assuming a static emplacement of the post-offices it is possible to predict how long a message will take to arrive from a given point in the network. Unfortunately, this assumption is only true for certain environments (CAN networks, laser links between satellites, etc), such that on extrapolating this platform to common network media (Ethernet, Serial Port, Wifi, etc) these characteristics are lost.

DiPippo et al [96] [97] have proposed a MAS based on RT-Corba [317] for distributed RTSs. The functioning of these systems is based on that of CORBA, except that in this case the clients and servers have added real-time characteristics. For example, a server records a task in its Scheduling Service together with certain deadline and quality demands that it is able to meet. When a client later requests that task within a certain time the Scheduling Service and its real-time component select a server following algorithms of analysis of EDF schedulability and pass them to the client, which finally makes a remote procedure call to that server. Following this distributed architecture of clients and servers in real time, the RTMA establishes a basic platform for the functioning of agents. To do so, the architecture has the services required by the agents Agent Management System (AMS) and Directory Facilitator (DF). Above the middleware, the agents are implemented with real-time demands. The agents that offer services are announced in the DF using an extension of KQML, modified to contain real-time declarations. Via this message, an agent announces that it is able to execute a task with certain specific parameters of service quality that relate the response quality to the time taken to fulfill the service. The problem underlying this approach is that both the time of deliberation of the Scheduling Service and the communication time itself are not bounded temporally and hence are not reflected in the declaration made by the agent. Another problem is reliability. An agent may say that it is able to perform a task in a given time with the associated quality but if that agent is not executed on a real-time operating system, it is possible that it may not live up to expectations.

Another example of a RTA is that proposed by Prouskas et al. [292]. The authors define their agents, called time-aware agents, as agents with the capacity



to operate in two temporal dimensions, from agent to agent and from human to agent, combining the predictability and reliability of exchanges in real time at small scale, and of the temporal requirements in diffuse human interactions at large scale. Time-aware systems are joined in a set of hardware, software, human interactions and interactions in real time to reason about the temporal restrictions localized in the system for each type of interaction, carrying out the necessary transformations between them and duly coordinating the activities, regardless of their limitations. An architecture in which these agents can exist and work together to achieve their aims has been developed and tested by means of a prototype called TARA and it has been implemented using the Agent Process Interaction Language (APRIL) [292].

BarSystems [86] are a type of MAS of the so-called swarm intelligence. The individual agent is not endowed with intelligence; it is the system that behaves intelligently (emergent intelligence). These systems are modeled as a soft RTS. In this type of problem, some agents with limited capacities must cope with a complex, distributed problem, attending correctly and in a timely fashion to all the requests from each of the clients. Accordingly, each client will be considered a task to be met with real-time restrictions and, overall, the aim is to satisfy a function of global usefulness in some way. The important part of this process is to pinpoint at each moment which task is the most attractive, and this depends on each domain in particular. BarSystems is specified at formal level and there is a software simulator that offers the solution to a concrete problem (the emplacement of boxes in a warehouse) following this method. However, since there is no real implementation of the agents of this architecture, it is not clear what their real-time constraints are or what happens if these are not fulfilled.

ARTIS agents [55] are specifically designed to develop real-time agents. The ARTIS architecture is an extension of the blackboard model [263], adapted to work in critical real-time environments, using Real-time Artificial Intelligence (RTAI) techniques to achieve this and guaranteeing that the real-time agents can react to changes in the environment in a flexible and dynamic way. ARTIS agents incorporate a control module responsible for the execution of the tasks associated with the agent in real-time, controlling how and when the different components of the agent should be executed. To manage the communication processes between agents, the ARTIS agent has been extended with a communications module called CoMO [332]. Additionally, the execution of the reactive components (components with critical real-time restrictions) is controlled by a sub-module of the control module integrated in the real-time operative system. With these characteristics, the architecture of the ARTIS agent guarantees the response of the agent, fulfilling all the critical temporal restrictions of the system. Its capacity for problem solving, its adaptability

and pro-activeness help to provide a sufficiently good response for current real-time environments. Its requirements of critical time are 100% guaranteed by means of an off-line schedulability analysis. The main problems of this proposal are the inability of the ARTIS agents to perform complex reasoning and the complexity of the design and implementation of the processes used by them, complicating the use of this proposal by developers not familiar with their architecture. The SIMBA architecture [332] [190] allows the development of MAS in real-time domains and, more specifically, in domains where the temporal restrictions must be strictly fulfilled (hard real-time). SIMBA systems are mainly composed of ARTIS agents. This set of agents controls the sub-system of the real-time environment with critical temporal restrictions. Additionally, the system can integrate different types of agent that allow the temporally non-critical activities of the system to be controlled, using normalized processes of interaction between different agents. The main problem with this proposal is the integration of agents without temporal restrictions in their activities with real-time agents and the use of specific modeling and communications languages.

Finally, java Agent for Real-Time (jART) [174] is one of the most recent works. The platform provides to the agents of the system the necessary tools for their creation, control and communication with other agents in the platform. For this, a serie of components and auxiliary agents have been created to provide services to agents in the system. This platform complies with the FIPA specification, an AMS agent is responsible, among other things, of providing location services (white pages) and the communication is given by FIPA-ACL messages. However, no organizational concepts are taken into account and the communication protocol does not meet the requirements for our specific case.

---

**3.5**

## Conclusions

---

In this chapter we have presented the main classification of the RTS and shown the main characteristics of a hard-RTS, which is the desired type of system in this dissertation. We have also discussed the main issues of local scheduling and presented the reasons that lead us to choose the FPS algorithm for each node of our system.

We have also explained the main characteristics of global scheduling. The planning mechanisms discussed here reveal limitations (complexity with large number of tasks, Dhall Effect, etc.), as explained. These limitations will be resolved with the proposed model, which will allow an effective planning model

to be included in real-time environments and in the developed agent-platform. It should also be noted that the model encompasses both the overall planning and distribution of tasks between nodes and also local planning within each node.

Finally, we have establish the desirable characteristics of a distributed RTS that have to be taken into account for the design of the PANGEA+RT platform presented in chapter 7. In the next chapter, the proposal to cover the local and global scheduling in VOs and taking into account the real-time constraints is explained.



# Part III

PROPOSED MODEL



# 4

## MODEL OF WCET PREDICTION, SCHEDULING AND TASK ALLOCATION

---

*In this chapter we present the model that gives the title to this dissertation. The chapter is divided into two different parts. The first one concerns to the WCET estimation of each task or service when an emergent behaviour is detected or a new agent wants to enter in a VO. The WCET analysis is divided into two phases: the high-level WCET analysis that concerns the task's control flow. And the low-level WCET analysis that concerns the execution time of individual basic blocks, and specifically the individual machine code instructions they represent. This involves estimating the execution time of individual instructions on the target execution environment. The second part deals with the scheduling and task allocation taking into account the real-time constraints and the available nodes where the agents belonging to the VO are executing.*

### Contents

---

<b>4.1</b>	<b>Introduction</b>	<b>60</b>
<b>4.2</b>	<b>Background and related works</b>	<b>62</b>
4.2.1	Related works of WCET analysis	62
4.2.2	Real-time scheduling and task allocation	64
<b>4.3</b>	<b>WCET in emergent behaviours</b>	<b>67</b>
4.3.1	Node evaluation	68
4.3.2	Code evaluation	74
4.3.3	Statistical improvements of the WCET in execution	82
<b>4.4</b>	<b>Model of scheduling and task allocation</b>	<b>83</b>
<b>4.5</b>	<b>Local scheduling</b>	<b>84</b>
<b>4.6</b>	<b>Global scheduling</b>	<b>85</b>
<b>4.7</b>	<b>Conclusions</b>	<b>96</b>

---

---

## 4.1 Introduction

---

Real-time computation plays an increasingly important role in computer systems, since the future is directing the evolution of these systems towards greater interaction with the environment. This environment sends stimuli to be captured and processed by the systems within temporal constraints, which are the most important issue to take into account in a RTS [60]. The simplest example is found in a system with sensors and actuators. Once the information has been captured through a sensor, a real-time system is expected to operate through the actuator in accordance with a given deadline. Depending on whether the tasks are considered critical, we can talk about hard or soft real-time systems. A hard real-time system is a system whose responses must occur within specified deadlines. A soft real-time system is a system that functions correctly if the deadline is occasionally missed [60] [222]. The contribution of this work is with hard real-time systems. Despite the widespread nature of the concept of real-time, there are still some misunderstanding in its practical use [341].

Many RTS include heuristics and ad-hoc techniques [64]; however, if we deal with hard-real-time system, this must be treated with care: The use of heuristics, which can, most of the time, provide approximate results under large computational time, can result in the inability to ensure time constraints. If all critical constraints are not verified a priori and no specific mechanisms are provided to handle the tasks time, the system can be unpredictable. Apparently, it can function properly for a period of time but in certain situations it may collapse. Depending on the application case, this collapse can be serious or fatal. Therefore, in a hard real-time system all tasks must be strictly bounded, which is a problem in dynamic systems where the behaviours are unknown and the environment is unpredictable and highly changeable. One of the research lines in dynamic systems focuses on finding mechanisms to control this problem, such as [258] [156] [375] [394]. In our system, we propose a method that allows each agent to calculate the WCET of the tasks to be performed which provides a new perspective to face off the problem.

From the analytical point of view, a RTS consists of a set of tasks and each task at the same time is composed of a set of activities or subtasks that run concurrently, cyclical, or according to a schedule. An ordinary real-time task is characterized by three parameters: the execution time, including all activities or sub-tasks; the period, i.e., how often the task must be executed; and finally the deadline, which determines the time that the task must be completed [397]. The task execution must be controlled by any scheduling algorithm, which is responsible for organizing the execution time between periodic or aperiodic



tasks. There are many existing scheduling algorithms [60], often executed by a Scheduler, and one should always be included in an oriented real-time execution module. For our work, the Scheduler of the agents is the algorithm known as Fixed-Priority Scheduling, which is explained in detail in section 4.6.

In VO terms, each agent knows the tasks and services that it offers and should, as a result, be capable of determining certain values such as computation or execution time, resources needed, interaction among other tasks, etc. Therefore, to create a hard real-time model, the task cannot be considered an unknown process entity; rather, its parameter should be used by the scheduling mechanism to verify the viability (known as schedulability, in this case) of executing these tasks while the time constraints are also fulfilled [64].

Open MAS are those which have experienced a big evolution and dynamicity. In fact, a VO can be considered an open system formed by the grouping and collaboration of heterogeneous entities with a clear separation between the structure and functionality that defines how these entities behave [113] [47]. In these system, the agents should be deployed in different devices with different characteristics, and interact with other agents and the rest of the system elements. The concept of environment is very important in the definition of open MAS.

In this dissertation, we present a major challenge in providing a mechanism for planning tasks and task allocation to members of VOS in real-time. Scheduling and task allocation is the focal point of a large number of publications within the research community [20] [385] [276] [381]. The main objective of these publications is centered on searching mechanisms to achieve a viable distribution of tasks among the available resources.

The called "ad-hoc approach" is based on design-implementation-tests iterations; however, the model proposed in this study enables adding restrictions to the schedule and task allocation both systematically and in real time. In addition, our work is founded in mathematical methods, so that it is backed by more than analysis simulations. As explained, it is particularly difficult to develop a hard real-time system combined with a highly changing environment. We present the problem of real-time scheduling and task allocation as an optimization problem and show the mechanisms to solve. The feasibility test of the FPS algorithm is applied as a constraint in the optimization method.

Most real-time systems shown in the previous chapter need to know the execution cost of the tasks, which ensures their proper planning. However, in dynamic environments with emergent behaviors it is difficult to determine the time needed to complete the task. Therefore, these tasks cannot be handled by most real-time works previously outlined. The first part of our proposed model

aims to solve this problem and obtain temporal bounded tasks. This bounding will be refined in successive executions through a statistical model.

This guarantees better performance in systems that interact with an unpredictable and dynamic environment. In these open and uncertain environments, classical methodologies for real-time system designs are hardly applicable since the set of applications to be executed and their corresponding resource and timing requirements are changeable in runtime. As such, runtime adaptation is essential in order to achieve a desired level of performance [283].

Two main objectives arise with this work:

- Proposing an effective model for calculating the execution time in the worst case (WCET) when an agent desires to form part of a virtual organization and wants to include tasks or behavior in the real time system, i.e. to calculate the WCET in emergent behaviours. This measure will be recalculated and adjusted in the system based on statistical measurements during subsequent executions.
- Proposing a scheduling model to allocate tasks for the whole system, i.e. a global scheduling. We use optimization techniques to reduce the quantity of computational nodes needed and to find the best distribution of tasks among them. This is done under the real-time constraints.

## 4.2

## Background and related works

---

This section is divided in two different subsections since we face two problems. First, we deal with the WCET analysis and study some interesting publications. The second part is dedicated to task allocation and global scheduling.

### 4.2.1

### Related works of WCET analysis

---

In a general form, the WCET calculation problems has been studied from different points of views. In [327] Shaw presents timing schemas to calculate minimum and maximum execution times for common language constructs. The rules allow collapsing the abstract syntax tree of a program until a final single value represents the WCET. However, this approach does not allow for the straight forward incorporation of global low-level attributes, such as pipelines or caches. The resulting bounds are not tight enough to be practically

useful. Computing the WCET with an Integer Linear Programming (ILP) solver is proposed in [295] [218]. This approach is extended to model the instruction cache with cache conflict graphs [217]. The WCET analysis of object-oriented languages is presented by Gustafsson [147]. Gustafsson uses abstract interpretation for automatic flow analysis to find loop bounds and infeasible paths.

Moreover, automatic cost analysis has many interesting applications. For instance, the receiver of the code may want to infer cost information in order to decide if it should reject code with excessively large cost requirements in terms of computing resources (in time and/or space) and accept code which meets the established requirements [4] [80] [18]. In fact, this is the main motivation for the Mobile Resource Guarantees (MRG) research project [18], which establishes a Proof-Carrying Code [261] framework for guaranteeing resource consumption. Furthermore, the Mobility, Ubiquity and Security (MOBIUS) research project [32], also considers resource consumption as one of the central properties of interest. Also, in parallel systems, knowledge about the cost of different procedures can be used in order to guide the partitioning, allocation and scheduling in parallel.

Java has potential for real-time systems development but is inappropriate for this purpose in its traditional sense. For instance, it lacks the notion of a deadline and high-resolution real-time clocks. Therefore, recent research has focused on introducing these concepts in Java by a number of initiatives such as the Safety Critical Java (SCJ) specification [182] and Predictable Java (PJ) [49] specification targeted safety-critical systems development. The specification includes, among others, a new programming model that is more amenable to be proved temporally correct. Temporal correctness can be ensured by using the information from a WCET analysis to perform schedulability analysis. With respect to Java, the WCET analysis is complicated by the presence of the Java Virtual Machine (JVM), since the JVM introduces an additional layer between the application itself and the underlying hardware platform. To mitigate this complexity, research has been focused on eliminating this middle layer by implementing the JVM in hardware, thereby achieving native execution of Java Bytecode (JBC) [318]. From this, it has been possible to experiment with Java to explore its applicability for the development of real-time systems.

WCET analysis at the bytecode level was first considered in [43]. It is argued that the well-formed intermediate representation of a program in Java bytecode, which can also be generated from compilers for other languages (e.g. Ada), is well suited for a portable WCET analysis tool. In that paper, annotations for Java and Ada are presented to guide the WCET analysis at the bytecode level. The work is extended to address the machine-dependent low-level timing analysis [36]. Worst case execution frequencies of Java bytecodes are introduced for

a machine-independent timing information. In [293] a portable WCET analysis is proposed. The abstract WCET analysis is performed on the development site and generates abstract WCET information. The concrete WCET analysis is performed on the target machine by replacing abstract values within the WCET formulae by the machine-dependent concrete values. An approach on how the portable WCET information can be embedded in the Java class file is presented in [37], an extension of [43] [36]. It is suggested that the final WCET calculation is performed by the target JVM. The calculation is performed for each method and the static call tree is traversed by a recursive call of the WCET analyzer. JVM timing models for the portable WCET analysis can be derived by measurements [51]. However, measurements cannot guarantee safe upper bounds of the timing. In [228] the authors describe the design and the capabilities of the static timing analysis tool TetaSARTS, which assists in the temporal verification of Safety Critical Java (SCJ) systems. Finally, [48] presents a selection of tools that assist hard real-time application developers to verify that programs conform to a Java real-time profile and that platform-specific resource constraints are satisfied. The problem of these presented works is that they are closed frameworks that we cannot include in our VOS environment and adapt to the organizational constraints.

In short, of the ability to provide hard real-time guarantees a Java execution environment featuring common embedded processors and software implementations of the JVM, which will greatly increase the incentive for adopting Java in the domain of real-time systems development [119].

#### 4.2.2

#### Real-time scheduling and task allocation

In this section some basic concepts of real-time systems will be reviewed, followed by a list of the most relevant studies related to this field of application.

In these systems there are two basic types of planning schemes: static, where predictions are made before the execution and when all tasks are already in the system; and the dynamic, where the decisions are taken at runtime. However, this classification is very strict and, as proposed in this paper, a model considered as static (the FPS model) can be adapted and included in dynamic environments. This forces the appropriate control and upgrades to be carried out, taking into account the time of possible changes and bounding the computation time of replanning. For a hard real-time system where the tasks are considered critical and all the tasks have a strict response deadline, introducing a dynamic planning scheme can lead to malfunction. In the same

definition of dynamic planning established by [70] the following characteristics are mentioned:

- It is not possible to guarantee the response times.
- It is not suitable for critic systems.
- It is not a stable schedule.

In recent years, planning schemes based on fixed priorities have received great attention in the scientific community, achieving a high degree of maturity and recognition [339] [348] [213] [326] [59] [21]. A historical study of the progress in planning for fixed priorities can be found in [23] and [325]. In addition, a review of various works [133] [275] [51] [116] will show that any dynamic planning model is highly dependent on the environment and the execution context. Moreover, it is important to highlight that the proposed methods are based on iterative systems with great complexity of computation or heuristics, where the computation time cannot be ensured and therefore, the tasks cannot be strictly bounded.

A planning scheme is characterized by two aspects:

- The planning algorithm: determines the order of access to the available resources in the system.
- The analysis model: needed to calculate and predict the temporal behavior of the system, i.e., whether the temporal requirements are guaranteed in all possible cases. To be sure, the worst case scenario must be taken into account in the calculations.

The RM or FPS algorithm and its extensions are static scheduling algorithms and represent one major paradigm for real-time scheduling. The RM was created by C.L. Liu and J.W. Layland in the work [221]. It consists of assigning the highest priorities to the shortest periods; that is, the priority of each task is inversely proportional to its period. To implement this policy, the response time of each task matches with its period.

EDF is the second major paradigm for realtime scheduling. Under certain conditions, EDF [221] [341] is an optimal dynamic scheduling algorithm in resource sufficient environments. While real-time system designers try to design the system with sufficient resources, cost and unpredictable environments often make it impossible to guarantee that the system resources are sufficient. In this case, EDFs performance degrades rapidly in overload situations. The Spring scheduling algorithm [396] can dynamically guarantee incoming tasks via on-line admission control and planning, and thus is applicable in resource-insufficient environments. Many other algorithms [341] have also been developed to operate in this way. These admission control-based algorithms represent the

third major paradigm for real-time scheduling. However, despite the significant body of results in these three paradigms of real-time scheduling, many real world problems are not easily supported. While algorithms such as EDF, RM and the Spring scheduling algorithm can support sophisticated task set characteristics (such as deadlines, precedence constraints, shared resources, jitter, etc.), they are all "open loop" scheduling algorithms. Open loop refers to the fact that once schedules are created they are not "adjusted" based on continuous feedback. While open-loop scheduling algorithms can perform well in predictable environments in which the workloads can be accurately modeled (e.g., traditional process control systems), they can perform poorly in unpredictable environments, i.e., systems whose workloads cannot be accurately modeled.

In recent years, a new category of soft real-time applications executing in open and unpredictable environments is rapidly growing [343]. As a cost-effective approach to guarantee performance in unpredictable environments, several adaptive scheduling algorithms have been recently developed [1] [38] [251] [355] [202] [201]. The problem with these proposals is that they are proposed for soft tasks, above all, in the field of multimedia and communication.

While early research on real-time scheduling was concerned with guaranteeing complete avoidance of undesirable effects such as overload and deadline misses, adaptive real-time systems are designed to handle such effects dynamically. The problem is that the act itself of handling the effects implies a chance that they will occur, and in a hard real-time system this is a non-allowed and non-desired characteristic. From our point of view, the need to join more strict scheduling and adaptive systems is still in progress. To be strict, and find works related with critical tasks, we have to search in the control and automation field. Several works choose to apply the control theory [115] to real-time computing systems. For example, several papers [350] [314] [358] [257] presented flexible or adaptive real-time scheduling techniques to improve digital control system performance. These techniques are tailored to the specific characteristics of control systems instead of general adaptive real-time computing systems. Several other papers [343] [367] [199] [54] presented adaptive scheduling algorithms or QoS management architectures for computing systems such as multimedia and communication systems. After studying previous works, we can verify that providing a scheduling and task allocation model in real time when the environment is dynamic and adaptive (such as the VOS) is an unexplored area. Any methods that have been revised due to their inability to ensure computation time linked to an effective solution cannot be considered for application within a system of hard-real-time. To achieve this, the system must be based on deterministic models with a solid mathematical foundation to validate the operation.

To the best of our knowledge, no unified model exists to date for designing an adaptive system with a global and local scheduling where the computation nodes are minimized and have the ability to dynamically adapt.

**4.3**

## Estimation of the WCET in emergent behaviours

---

As mentioned, hard real-time systems are not allowed to exceed their deadlines. To ensure this, a schedulability analysis can be conducted. This determines if the set of tasks comprising the system can be scheduled such that no deadlines are exceeded. The WCET of specific tasks, which can be determined using WCET analysis, is an integral component for conducting schedulability.

WCET analysis is in general an undecidable problem. Program restrictions, as given in [294], make this problem decidable. Loops and recursion depths need to be bounded loops [33]. As the full application has to be available for the WCET analysis, we disallow dynamic class loading, although for embedded real-time systems this is not a severe restriction.

- *High-Level WCET Analysis:* the high-level WCET analysis concerns the task's control flow. The control flow can be represented using a Control Flow Graph (CFG) which is a directed graph consisting of vertices, denoted as basic blocks, and edges. Each vertex represents a set of sequentially executed instructions. Connecting the basic blocks with edges represents the control flow of task capturing e.g. branches. In this context, the high-level analysis concerns the WCET of the task based on the control flow, and thus assumes that the cost of the basic blocks are known, e.g. through low-level analysis. Generating the CFG from the compiled program is preferable since we can take into account the optimizations introduced by the compiler. This is important since compiler optimizations significantly affect the WCETs of a program [50].
- *Low-Level WCET Analysis:* the low-level WCET analysis concerns the execution time of individual basic blocks, and specifically the individual machine code instructions they represent. This involves estimating the execution time of individual instructions on the target execution environment, which raises a number of concerns. First of all, these are often not made available by the vendors. Furthermore, a variety of mechanisms

used for reducing the average execution time, such as caching and pipelining, must be taken into account since their absence in the analyses yield very pessimistic results.

These analyses are further complicated by the presence of an OS and even a virtual machine, since they add additional layers of logic between the program itself and the hardware [119].

---

**4.3.1** Node evaluation

---

The node evaluation refers to the Low-Level WCET Analysis where the hardware specification has to be taken into account. This involves estimating the execution time of individual instructions on the target execution environment, which raises a number of concerns. First of all, these are often not made available by the vendors.

---

**4.3.1.1** Java virtual machine and Java bytecodes

---

Knowledge about the Java Bytecode (JBC) and the Java Virtual Machine (JVM) is imperative for understanding the mechanisms that affect the WCET of Java programs.

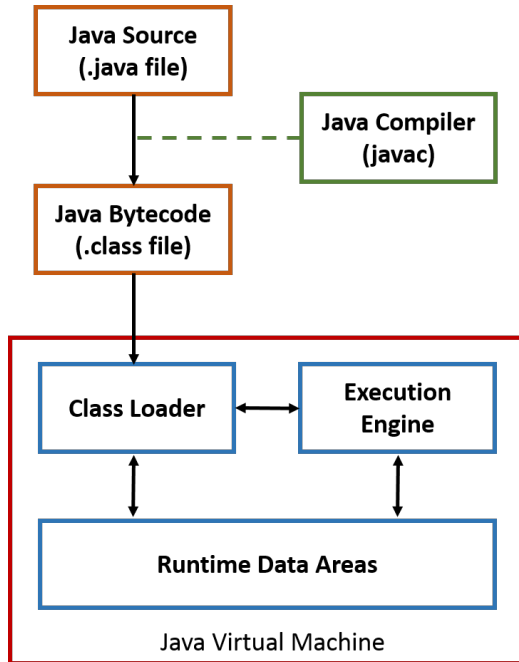
The JVM is a stack machine and essentially prescribes a complete execution environment for programming languages targeted at the JBC instruction set, such as Java, with the purpose of being independent of the underlying hardware and OS. Platform independence is achieved since the JVM can be implemented for the platforms of interest and may even be independent of an OS. This is achieved with the premise that the programs are not utilizing platform-dependent functionality accessed by native methods or using non-standard runtime libraries.

The Java program is compiled into JBC, which can be uploaded to an arbitrary JVM running on an arbitrary processor. While this is very important, it also complicates the problem of calculating the WCET because although the bytecodes are always the same, they can be executed in different JVM machines and processors, which can alter the needed processor cycles and the time.

The process of a Java code is shown in the following figure (Figure 4.1) which is explained in detail in the official JVM specification [219]. In total, JBC is



composed of 256 bycodes. But not all of the possible 256 opcodes are used because 51 are reserved for future use.



**Figure 4.1:** Steps of Java code execution process

To illustrate how JBC is represented, we are going to present several examples to illustrate the procedure. We will see a representation of the JBC using the corresponding mnemonics.

### Example 1

The following code example shows a simple `if-else` comparing two integer parameters.

```

1 public int greaterThan(int intOne, int intTwo) {
2     if (intOne > intTwo) {
3         return 0;
4     } else {
5         return 1;
6     }
7 }
  
```

**Programming Code 4.1:** Example `if-else` in Java

This method results in the following bytecode:

```

1 0: iload_1
2 1: iload_2
3 2: if_icmple      7
4 5: iconst_0
5 6: ireturn
6 7: iconst_1
7 8: ireturn

```

**Programming Code 4.2:** Example if-else in bytecodes

First the two parameters are loaded onto the operand stack using `iload_1` and `iload_2`. Then, `if_icmple` compares the top two values on the operand stack. This operand branches to byte code 7 if `intOne` is less than or equal to `intTwo`. Notice this is the exact opposite of the test in the if condition of the Java code because if the byte code test is successful, the execution branches to the `else` block where, as with the Java code, if the test is successful the execution enters the if block. In other words `if_icmple` is testing whether the if condition is not true and jumping over the if-block. The body of the if block is byte code 5 and 6, the body of the `else` block is byte code 7 and 8.

### Example 2

This example shows a simple `while` loop.

```

1 public void whileLoop() {
2     int i = 0;
3     while (i < 2) {
4         i++;
5     }
6 }

```

**Programming Code 4.3:** Example if-else in Java

This method results in the following bytecode:

```

1 0: iconst_0
2 1: istore_1
3 2: iload_1
4 3: iconst_2
5 4: if_icmpge     13
6 7: iinc         1, 1
7 10: goto        2
8 13: return

```

**Programming Code 4.4:** Example if-else in bytecodes

The `while` loops consist of conditional branch instructions such as `if_icmpge` or `if_icmplt` and a `goto` statement. The conditional instruction branches the execution to the instruction immediately after the loop and therefore terminates

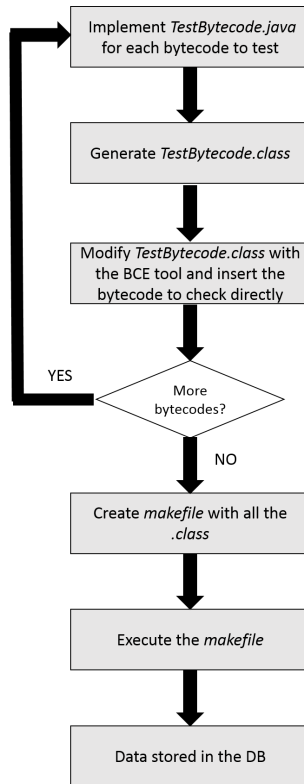
the loop if the condition is not met. The final instruction in the loop is a `goto` that branches the bytecode back to the beginning of the loop ensuring the bytecode keeps looping until the conditional branch is met. The `iinc` instruction is one of the few instruction that updates a local variable directly without having to load or store values in the operand stack. In this example the `iinc` instruction increases the first local variable `i` in 1. Java bytecode generation has to follow stringent rules [220] in order to pass the class file verification of the Java Virtual Machine (JVM). Those restrictions lead to an analysis friendly code; e.g. the stack size is known at each instruction. The control flow instructions are well defined. Branches are relative and the destination is within the same method. In the normal program, there is no instruction that loads a branch destination in a local variable or onto the stack. Detection of basic blocks in Java bytecode and construction of the CFG is thus straightforward. The JVM must not employ dynamic class loading features, since this introduces unpredictable behaviour. Instead, a JVM suitable for real-time systems should employ static linking or linking at program initialization. All loops must be bounded in order to conduct the WCET analysis. This can be solved using techniques such as the discriminant function analysis (DFA) [288] [252] or manual annotations. The general approach has been to provide fixed loop bounds according to the specific functionality the system serves; however, should the system be used in another context it may require the loop bounds to be derived again. Therefore another approach suggested in citehunt06 consists of annotating loop bounds using the Java Modelling Language (JML) [69] and using the KeY tool [39] citebeckert11 to determine the loop bounds symbolically. This means that the bounds depend on certain input, and when this input is available the correct bounds can be derived relatively easily. However, such an analyses, and even the DFA itself, requires much effort in order to be implemented. Instead, we have chosen to explicitly declare loop bounds using annotations in the code. An extension of the Java annotation mechanism for WCET-related annotations is proposed in [154]. The next chapter will discuss the implementation issues and explain how to include annotations in the code of the agents.

#### 4.3.1.2

#### Test generation

As mentioned before, the Java program is compiled into JBC, which can be uploaded to an arbitrary JVM running on an arbitrary processor. This is the issue that highly complicates the problem. If we consider a real-time system with a set of computational nodes, each node has its own configuration and possibly its own JVM as well. We follow the steps shown in Figure 4.2 to

obtain the program that should be executed in each node for testing:



**Figure 4.2:** Steps of the node evaluation

We use the standard Java library `System.currentTimeMillis`. There is also another standard method to consider `System.nanoTime` but the accuracy and precision of the former justifies our choice since `System.nanoTime` cannot guarantee nanosecond accuracy [211]. First, we create Java classes for each bytecode. We can see an example in the following code 4.5.

```

1 public class TestDLoad
2 {
3     public static void main(String args [])
4     {
5         long time1, time2, tme3;
6         long result;
7
8         time1=System.currentTimeMillis ();
9         for (int i=0;i<1000000;i++)
10            test ();
11
12        time2=System.currentTimeMillis ();
  
```

```

13     for(int j=0;j<1000000;j++)
14         test2();
15
16     time3=System.currentTimeMillis();
17
18     result=(tiempo3-tiempo2)-(tiempo2-tiempo1);
19
20     //store result in the database
21 }
22
23 //modify this method in the .class
24 //execute 10000 times the bytecode
25 public static void test()
26 {
27 }
28
29 //modify this method in the .class
30 //execute 50000 times the bytecode
31 public static void test2()
32 {
33 }
34 }

```

**Programming Code 4.5:** Java class for bytecode testing

To ensure that the `for` loop is not affecting the calculations, we program two different `test` functions. The difference is calculated in line 293 of the code, so we know the time of performing the preset number of the bytecodes without the interference of the loop.

To modify and create the `.class` file including the edited bytecodes, we use the tool JBE [310]. It is an open-source bytecode editor suitable for viewing and modifying java class files. For verification and exporting the class files, JBE uses the the Bytecode Engineering Library by Apache's Jakarta project [72]. The advantages of this tool are that all changes are directly applied to the class file, it uses the standard mnemonics for JVM opcodes such as instruction names and the type information; exceptions and annotations are also written as they appear in class files.

```

1  0: lconst_0
2  1: lconst_1
3  2: lcmp
4  3: lconst_0
5  4: lconst_1
6  7: lcmp
7  [...]
8  150001: return

```

**Programming Code 4.6:** `lcmp` bytecode test

It is important to note that there are bytecodes that cannot be tested alone, i.e., we need to mix them with other bytecodes, after which we can perform the calculations needed to know the time execution of the bytecode alone. One example is the bytecode `lcmp` that compares two longs values. In the `test` function, we should include `lconst_0` and `lconst_1` as shown in 4.6.

This is because the bytecode `lcmp` needs to have two long values on the stack that we must push onto the stack previously. After the execution of the bytecode, it leaves an integer value in the stack that is not possible to use with the `lcmp` bytecode. This forces us to push two long values onto the stack again. As we know the execution time the bytecodes `lconst_0` and `lconst_1`, we can perform the calculation in the `TestLConst0.java` before the final value is inserted. In the code 4.6 we have two methods, the first one with the bytecode `lcmp` inserted 10000 times and in the second one, inserted 50000 times.

This data is stored in a database. When it is read in the next step, the values are stored in a global table  $\mathcal{R}(I_{205}, N_{total})$  where each row corresponds to each bytecode and the column is the cost in the  $N$  nodes of the system.

### 4.3.2

### Code evaluation

High-Level WCET Analysis consists of the evaluation of the code that carries out a task. For this to succeed, the control flow of the task must itself be analyzable. Therefore, it must not contain constructs which make this impossible. Furthermore, constructs which result in very pessimistic WCET estimates should preferably be avoided as well. Some of these are described in [294]. The most classic examples are recursion and unbounded loops.

The general steps of the process are included in Figure 4.3. They will be explained in the next subsections, except for the node evaluation, which has been already presented.

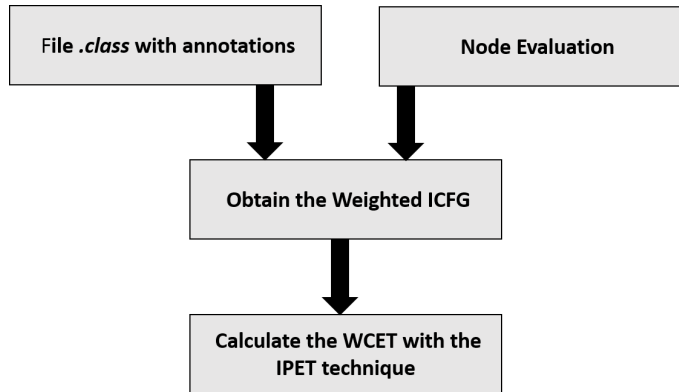


Figure 4.3: Steps of the WCET calculation

#### 4.3.2.1

#### How to obtain the weighted ICFG

A CFG can be used as the foundation for different program analyses such as for WCET analysis and for compiler optimization to e.g. determine infeasible paths [5]. Therefore, before introducing the actual CFG, it is convenient to establish the definition of what we are going to consider a basic block of code.

**Definition 8** *A basic block is a block enclosing a number of instructions that do not contain any jumps or jump targets. The only jump target is when the basic block starts and the only jump is at the end of the basic block. Call instructions are normally not considered as branches since the control flow continues immediately after the call has been processed. However, it is important to note that since it is impossible to guarantee whether the control flow continues immediately after processing the call, it must be considered as a basic block itself [5].*

A CFG of a program represents all the paths that are traversable during execution, thereby capturing the control flow of the program. The CFG is based on a directed graph and hence can be seen as an abstract representation of the control flow where each vertex in the graph represents a basic block. Connecting the vertices by edges captures the jumps in the control flow [119]. Formally, a CFG is defined as:

**Definition 9** *A CFG is described as a directed graph  $G = (V, E, i)$ , where the vertices,  $V$ , correspond to basic blocks and edges  $E \subseteq V \times V$  connect two basic blocks  $v_i, v_j \in V$  if  $v_j$  is executed immediately after  $v_i$ .  $i \in V$  represents*

the start vertex, denoted as the source, which has no incoming edges, that is:  $\nexists v \in V$  talque  $(v, i) \in E$  [223].

An execution path is defined as a sequence of interconnected basic blocks. Formally defined:

**Definition 10** An execution path  $\Upsilon$  through a control flow graph  $G = (V, E, i)$  is defined as a sequence of basic blocks  $(v_i, \dots, v_n) \in V^*$ , with  $v_1 = i$  and  $\forall j \in 2, \dots, n$  talque  $(v_j, v_{j+1}) \in E$  [223].

In our case, we will focus the problem on the analysis of the Java bytecodes, so the definition can be more specific as [395] explains:

**Definition 11** An Extended Control Flow Graph (eCFG) of a Java method  $M$  is a 4-tuple  $(V, A, s, t)$  where  $(V, A)$  is a simple diagram such as that  $V$  is a set of vertices which represent possible flow of control between basic blocks in  $M$ , and  $A \subseteq V \times V$  is a set of arcs which represent possible flow of control between basic blocks in  $M$ .  $s \in M$  is a unique vertex, called start vertex which represent the entry point of  $M$ , such that in-degree  $s = 0$  and  $T \subset V$  is a set of vertices, called termination vertex which represents the exit points of  $M$ , such that for any  $t \in T$  out-degree  $t = 0$  and for any  $v \in V (v \neq s \ \& \ v \neq t)$ .  $\exists t \in T$  such that there exists at least one path form  $s$  to  $v$  and at least ones path from  $v$  to  $t$ .

Traditional control flow analysis treats an individual statement of a program as a vertex in the CFG. However, when analyzing Java bytecode, using a basic block as a vertex in the CFG is considered more appropriate as the number of instructions is very large. Thus, according to the definition 11 of [395], vertices represent basic blocks and arcs represent the control of flow between the basic blocks. The CFG contains one unique vertex  $s$  to represent the entry point of a method and a set of vertices  $T$  to represent the termination of the method because the control flow may terminate at multiple program points due to handling exceptions during the execution.

It is now the moment to introduce the rules for generation the CFG. First of all, the main vertices have to be identified. This is the vertex from which one or more arcs has to start. A main vertex is always:

- the first instruction of a method and the first instruction of every handler of the method.
- all the instructions that correspond to unconditional branch: `goto`, `goto_w`, `jsr`, `jsr_w`, `ret`.



- all the instructions that are the target of a conditional branch: `ifeq`, `iflt`, `ifle`, `ifne`, `ifgt`, `ifge`, `ifnull`, `ifnonnull`, `if_icmpeq`, `if_icmpne`, `if_icmplt`, `if_icmpgt`, `if_icmple`, `if_icmpge`, `if_acpeq`, `if_acmpne`, `lcmp`, `fcmpl`, `fcmpg`, `dcmpl`, `dc,pg`.
- all the instructions that are the target of a compound conditional branch or a return: `tableswitch`, `lookupswitch`.
- all the instructions that immediately follow a conditional or unconditional branch: `ireturn`, `lreturn`, `freturn`, `dreturn`, `return`.
- all the instructions that immediately follow an instruction that explicitly throw an exception: `athrow`.
- all the instructions that immediately follow an instruction that implicitly throw an exception: `aaload`, `aastore`, `anearray`, `arraylength`, `baload`, `bastore`, `caload`, `castore`, `checkcast`, `daload`, `dastore`, `faload`, `fastore`, `getfield`, `getstatic`, `iaload`, `iastore`, `idiv`, `instanceof`, `invokeinterface`, `invokeespecial`, `invokestatic`, `invokevirtual`, `irem`, `laload`, `lastore`, `ldc`, `ldc_w`, `ldc2_w`, `ldiv`, `lrem`, `monitorenter`, `monitorexit`, `multianewarray`, `new`, `newarray`, `putfield`, `putstatic`, `saload`, `sastore`.

Each main vertex includes a basic block, formed by all the instructions until the next main vertex or the end of the bytecode [395]. Once, we have the basic blocks, the following rules are used to construct the eCFG for the method. If  $u \in V$  and  $v \in V$  are two basic blocks:

- If  $v$  follows  $u$  in the bytecode and  $u$  does not terminate in a unconditional branch, then add an arc  $(u, v)$ .
- If the last instruction of  $u$  is a conditional or unconditional branch to the first instruction in  $v$ , then add an arc  $(u, v)$ .
- Always add an arc  $(u, v)$  from the basic block of each `tableswitch` or `lookupswitch` to the basic block of each instruction that is defined as the target in the switch.
- When there is a subroutine, then two arcs have to be added. One arc  $(u_1, v_1)$  from the basic block of the `jsr` or `jsr_w` to the basic block of target. Other arc  $(u_2, v_2)$  from the basic block containing the corresponding `ret` instruction to the basic block of the instruction that immediately follows the call.
- add an arc  $(u, v)$  from the basic block of each instruction that may throw an exception to the entry basic block of every exception handler that covers the region of the bytecode in which the instruction appears.

- add an arc  $(u, v)$  from the basic block of each instruction that may throw an exception to a dummy basic block that represents abnormal completion of the method, i.e. the situation where a thrown exception is not caught by any handler of the method.

In order to determine the WCET, it is needed that all the included arcs  $(u_i, v_i)$  are correctly annotated. This means that they must be bounded with the number of times that it is possible to go through the arc in the execution path  $\Upsilon$ . This frequency  $d_i$  is used in section 4.3.2.2 when the Implicit Path Enumeration Technique (IPET) technique is introduced.

Zhao in [395] includes a new definition to extend the eCFG to a complete or partial Java program.

**Definition 12** *An Interprocedural Control Flow Graph (ICFG) for a partial or complete Java program is a tuple  $(G_1, \dots, G_k, C, R)$  where  $(G_1, \dots, G_k)$  are flow graphs representing the Java methods in the program.  $C$  is a set of call arcs,  $R$  is a set of return arcs,  $g$  is the final ICFG and  $V_{G_i}$  is the set of vertex in the graph  $G_i$ . A ICFG of a partial program satisfies the following conditions:*

- *There is a one-to-one onto mapping between  $C$  and  $R$ . Each call arc is of the form  $(u, v_{1G_i}) \in C$  and the corresponding return arc is of the form  $(v_{1G_i}) \in R$ , where  $u \in V_{G_i}$ , for some  $G_i \in g$  and  $v_{1G_i}$  are the initial and final vertices, respectively, of some  $G_j \in g$ .*
- *$g$  contains two distinguished vertices: an initial vertex  $v_{1G} = v_{1G_i}$  and a final vertex  $v_{FG} = v_{FG_i}$ ,  $G_i \in g$ .*

In this way, an ICFG for a partial or complete Java program consists of (i) a set of CFGs, each one representing the control flow of a method in the program, and (ii) some call and return arcs that are linked together. The difference between an ICFG for a partial program and a complete program is that for a partial program the start vertex may be the entry vertex of any method in the partial program, whereas in the complete program the start vertex must be the start point of the `main` method. After the complete ICFG, if any vertex cannot be reached from the entry vertex  $s$  has to be eliminated.

#### 4.3.2.2

#### Calculation the WCET with the IPET technique

The Path-Based WCET Analysis consists of transforming the ICFG into a weighted graph, in which the execution cost of each individual basic block is used as weights. Standard graph algorithms can then be used to identify the

longest path in the graph, which in turn results in the WCET of an analysed task [102] [374]. One specific approach proposed in [102] consists of four steps where the final three are primarily used to increase precision.

1. Find the longest path in the graph using a standard graph algorithm.
2. Check if the path is feasible.
3. If the path is not feasible, exclude it from the graph and return to the first step.
4. When the longest feasible path is found, its length is the WCET of the task.

Note that the complexity of analysing such a graph is exponential with the depth of conditional statements and thus poses some difficulty in analysing large systems [164]. As this method can take a long time to calculate, we use the improvement called IPET introduced by Li and Malik in [218]. This technique is applied with the constraints 1 and 2 that will be explained in the next pages.

In principle, the WCET is calculated by summarising the worst-case execution frequencies of the basic blocks multiplied with their costs. Thanks to the table obtained in the previous step,  $\mathcal{R}(\mathcal{I}_{205}, N_{total})$  where each row  $\mathcal{I}_i$  corresponds to each bytecode, and the column is the cost in the  $N$  nodes of the system, we already know the cost of each bytecode for each node. Then, we need first to calculate the cost  $\tau$  of each basic block  $BB$  for each node  $k$ :

$$\tau_{BB} = \sum_{j=1}^M \mathcal{R}_j^k \quad (4.1)$$

where  $M$  is the total number of instructions in the basic block  $BB$  y  $\mathcal{R}_j^k$  represents the cost of each individual instruction  $j$  in the node  $k$ .

Again, we obtain a table  $\mathcal{R}^*(BB_{total}, N_{total})$  where each row corresponds with each  $BB$  and each column with one of the nodes of the system then, in each cell of the table, we have the cost of the basic block  $BB$  in the node  $N$   $\tau_{BB}^N$  in the node  $N$ .

Finally, we can define the following ILP problem:

$$WCET_P = \max \sum_{i=1}^I \tau_{BB}^N x_i \quad (4.2)$$

where  $B$  is the number of basic blocks,  $x_i$  is the execution frequency of the basic block  $i$  and thanks to the previous table  $\mathcal{R}^*$ , we know the value of  $\tau_{BB}^N$ . The WCET have to be calculated for each node  $n$  of the system.

To calculate the WCET of each  $BB$ , the bounds for the execution frequencies of the basic blocks must be determined. This requires the annotations, which were previously explained, and the resulting equation is maximised. Thus, the bounds must be determined for the execution frequencies; if not, the result of maximising the equation would be infinite. The bounds can be expressed using a set of constraints expressed for the control flow of the program. These constraints are:

- **Structural Constraints:** Constraints on the control flow of the program such as branches. These can be extracted from the CFG.
- **Functional Constraints:** Constraints on the behaviour of the control flow, such as loop bounds. These can be extracted from the annotations.

**Constraint 1** *Simple structural constraints can be expressed by equations stating the execution frequencies each basic block  $x_0, \dots, x_B$  as the sum of all its incoming transitions or the sum of all outgoing transitions. Therefore, the frequency  $x_i$  for a given block  $i$  is the equation:*

$$x_i = \sum_{j \in \mathcal{I}_i} d_j = \sum_{k \in \mathcal{O}_i} d_k \quad (4.3)$$

*in which  $\mathcal{I}_i$  is the set of all incoming transitions to basic block  $i$ , and  $\mathcal{O}_i$  is the set of all outgoing arcs from basic blocks  $i$ . By repeating this proces for each basic block, a set of constraints such as the following can be constructed.*

### Example 3

An example is shown in Figure 4.4. The basic block are denoted as  $B_0, \dots, B_5$  and the execution frequencies as  $x_0, \dots, x_5$  and each transition between basic blocks have also their frequencies  $d_0, \dots, d_0$

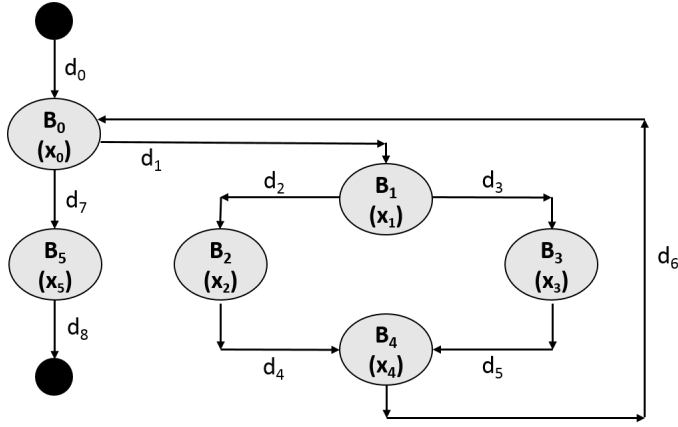


Figure 4.4: ICFG Example

The structural constraints obtained from the ICFG are:

$$\begin{aligned}
 x_0 &= d_0 + d_6 = d_1 + d_7 \\
 x_1 &= d_1 = d_2 + d_3 \\
 x_2 &= d_2 = d_4 \\
 x_3 &= d_3 = d_5 \\
 x_4 &= d_4 + d_5 = d_6 \\
 x_5 &= d_7 = d_8
 \end{aligned} \tag{4.4}$$

**Constraint 2** A common functional constraint is loop bounds. A loop consists of a loop head, which is the entry basic block to the loop, and the loop body which is the basic blocks executed as part of the loop. A functional constraint is that the number of edges from the loop body to the loop head must be less than or equal to the number of incoming edges to the loop head, which do not originate from the loop body, multiplied with the loop bound. This is expressed in the equation [320]:

$$\sum_{j \in \mathcal{C}_h} d_j \leq n \sum_{k \in \mathcal{E}_h} d_k \tag{4.5}$$

where  $\mathcal{C}_h$  is the set of arcs from loop body to loop head, and  $\mathcal{E}_h$  is the set of incoming arcs entering the loop head not originating from the loop body. Finally,  $n$  is the loop bound.

According to the Figure 4.4, the functional constraint can be expressed as (4.6):

$$d_6 \leq n \cdot d_0 \quad (4.6)$$

The above equations (4.4, 4.6) are then solved using an ILP solver, which finds the execution frequencies resulting in the maximum value, in this case, the WCET. Solving these equations is an NP-complete problem in the general case, even though Li and Malik in [218] have shown that the nature of the problem very often results in equations which can be solved in polynomial time. We use the Simplex Algorithm to obtain the solutions of the liner programming according to the constraints (4.4, 4.6) and the objective function (4.2).

Finally, we will have a final vector  $FV$  that will be used in the global scheduling. This  $FV$  stores the  $WCET_P$  for each node  $n$  of the system. The entire procedure can be summarised in the following algorithm (2):

---

**Algorithm 2** Code Evaluation
 

---

```

1: procedure HIGHLEVEL EVALUATION( $\mathcal{R}[I_{205}, N_{total}]$ )
2:    $ICFG \leftarrow generateInterproceduralControlFlowGraph(file.class)$ 
3:    $ICFG_w \leftarrow getWeights(ICFG)$ 
4:   for  $n = 1$  to  $N$  do
5:     for  $BB = 1$  to  $totalBB$  do
6:        $\tau_{BB} \leftarrow costBB(\mathcal{I}_1, \dots, \mathcal{I}_n)$ 
7:        $\mathcal{R}^*[BB, n] \leftarrow \mathcal{R}^*[\tau_{BB}, n]$ 
8:        $WCET_{BB} \leftarrow simplexMethod(\sum_{i=1}^N \tau_i x_i, constraints)$ 
9:        $WCET_P \leftarrow WCET_P + WCET_{BB}$ 
10:       $FV(n) \leftarrow WCET_P$ 
11:    end for
12:  end for
13:  return  $FV$ 
14: end procedure

```

---

4.3.3

 Statistical improvements of the WCET in  
 execution
 

---

The successive executions of the task enable to improve the precision of the measure by statistical techniques. This means that the estimation of the execution time can be carried out statically if there are enough samples of the execution time of the task.

Selecting the WCET according to the analysis of the bytecodes provides an upper bound in execution time; however, this upper bound is a limit which is difficult to reach. A statistical analysis would narrow the variation interval of

the WCET, thus avoiding the use of large upper bounds. This process can be carried out by applying a confidence interval analysis to the execution times obtained during the different iterations. For example, a statistical distribution such as *t* – *student* could be used with a sample containing more than 30 values. The main problem is that the sample should meet several conditions for the confidence intervals to be realistic. Consequently, we opted for other alternatives such as the estimation of WCET through the analysis of atypical values, which were obtained from previous executions. An atypical value  $a_i$  or  $a_j$  is defined according to the following equations (4.7) (4.8):

$$a_i < Q_1 - 3 \cdot (Q_3 - Q_1) \quad (4.7)$$

$$a_j < Q_3 + 3 \cdot (Q_3 - Q_1) \quad (4.8)$$

where  $Q_1$  and  $Q_3$  represent the first and third quartile in the sample.

The WCET value is calculated according to the upper value, and is defined as follows:

$$WCET = Q_3 + 3 \cdot (Q_3 - Q_1) \quad (4.9)$$

## 4.4

## Model of scheduling and task allocation

---

Figure 11.3 shows the global RTS. The FPS algorithm is used for the scheduling inside each node and the proposed scheduling and task allocation model is applied to the global scheduling.

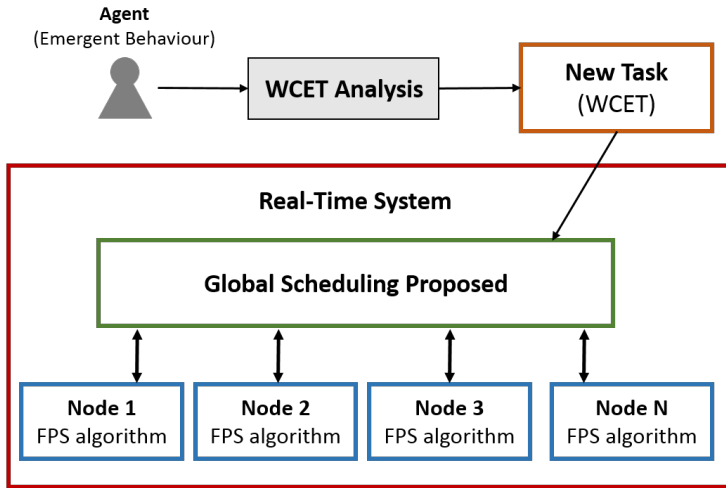


Figure 4.5: Overview of the RTS

An alternative to use heuristics is to employ algorithms that attempt to capture the timing behavior of all tasks in the system. The goal of these algorithms is to provide a high degree of schedulability resource utilization and an a priori verification of timing correctness for all tasks in the system. These algorithms are more restrictive than the heuristic methods in that they require more knowledge about task utilization, arrival patterns, and resource requirements [335]. The well-developed algorithms in this class are for the priority-driven, preemptive scheduling of periodic tasks with hard deadlines. These algorithms assign priorities based on task deadlines, available slack time or, in our case, frequency (FPS).

Each task is characterized by the following parameters:  $C$  is the worst case execution time (WCET);  $D$  is the task deadline;  $I$  is the notation for the possible interferences or the maximum delays allowed due to the communication (in case the task requires programming);  $\rho$  indicates the utilization of the task in each computational node;  $\mathcal{P}$  is the minimum time between two consecutive arrivals of the task (period);  $H$  is the importance or valuation of the task inside the system (just taken into account if the nodes are full).

## 4.5

### Local scheduling

As can be observed in Figure 11.3, the scheduling inside each node is carried out with the FPS algorithm. Some studies [63] [285] [379] have demonstrated



the advantages that it offers compared to the EDF algorithm:

- It is more easily implementable. The parameter used for scheduling, (the priority) is static at the moment that the scheduling is done (although it may vary in successive iterations) while the EDF requires more computation time and therefore, it may delay the plan. Moreover, its tendency to produce deviations is greater.
- It is easy to introduce tasks without a specific deadline, simply assigning them a priority. However, assigning an arbitrary deadline is more abstract and artificial.
- The notion of priority can include other aspects such as a rating on the urgency or importance of the task, however, deadlines are immovable.
- In an overload situation, the FPS has a more reasonable behavior since the malfunction will be given on those tasks with less priority. However, with EDF, the most critical tasks with a longer deadline may be affected and cause a domino effect on the remaining tasks, producing much greater delays.

However, EDF has an advantage over FPS. The utilization factor is higher in the nodes but in a system of hard real-time system, the main purpose is to ensure that the deadline for all tasks is fulfilled; therefore, having a lower utilization factor is considered an acceptable consequence. As a result of these considerations, we selected the FPS planning model for our system. The following section will explain in detail how it also applies to task distribution.

## 4.6

## Global scheduling

---

The model for task planning and allocation is presented in this section. The problem of planning, in its broadest sense, is to assign an appropriate order of execution to a set of tasks seeking specific criteria: efficiency, quality, time, cost, etc. in an organization [10]. In our case, the main criteria is the fulfilment of the time constraint when executing the task. According to Ojeda [268], effective planning depends on two essential tasks: programming and allocation.

Thanks to optimization, we are able to obtain a task allocation with the minimum number of nodes. We identify the place of execution of each task, after which the roles required to the agents are assigned. In case of overload, the system will modify the nodes in which the tasks are executed; otherwise

it could be necessary to increase the number of nodes. If no more nodes are available, the system uses the parameter  $H$  that indicates the criticality of a task.

One of the basic constraints of the optimization model is derived from the FPS feasibility test. To verify the feasibility of a plan that takes into account time constraints with the FPS, we apply the utilization test proposed by Lui and Layland [221]. It is the most restrictive test as well as the easiest to apply, which reduces the computation time. This test is based on the following observation:

Let  $\mathcal{T}$  be a set of periodic tasks. The longest response time for any task occurs when all tasks are activated simultaneously. This moment is called critical moment [237].

**Definition 13** *If the task set is synchronous (all tasks have the same phase), the initial critical moment is the one that occurs at zero moment.*

This indicates that when the critical moment occurs, then the worst-case load appears in the node. In a static priority preemptive system, where a task can be interrupted by another with a higher priority, the critical moment is the most difficult time to guarantee all the task deadlines. Theorem 1 shows that if deadlines can be guaranteed for releases starting at critical time, they can be guaranteed, by implication, for the lifetime of the system [22].

The basic schedulability conditions for RM proposed by Liu and Layland were derived for a set of  $\mathcal{T}$  periodic tasks under the assumptions that all tasks start simultaneously at time  $t = 0$ , relative deadlines are equal to periods  $D_{i,k} = kP_i$  and tasks are independent. Under such assumptions, a set of  $\mathcal{T}$  of periodic tasks is schedulable by the FPS algorithm if the equation 4.10 is true.

**Theorem 1** *The set of tasks  $\mathcal{T}$  is schedulable under RM if:*

$$\sum_{i=1}^N \rho_i \leq N(2^{1/N} - 1) \quad (4.10)$$

where:

$$\rho_i = C_i/T_i \quad (4.11)$$

The schedulability bound of FPS is a function of the number of tasks, and it decreases with  $N$  [63]. Moreover, it is important to realize that:

$$\lim_{N \rightarrow \infty} N(2^{1/N} - 1) = \ln 2 \simeq 0.693 \quad (4.12)$$

for large values of  $N$ , the bound asymptotically approach to 69.3%. Then, any combination of  $N$  tasks with an utilization factor  $\rho$  less then 69.3% will be always schedulable following a preemptive priority-based scheduling scheme (where there will be an immediate switch of execution to the higher priority task). That means that any task set can be scheduled by FPS if  $\rho \leq 0.69$ , but not all task sets can be scheduled if  $0.69 < \rho \leq 1$  [63].

For the global scheduling of the system the total number of tasks are taken into account,  $\mathcal{T}$ . Before each replanning and task allocation, this value must be known  $\mathcal{T}$  and the scheduled program will be viable while the  $\mathcal{T}$  remains static. Any change in the set  $\mathcal{T}$  leads to replanning to ensure that the task are still viable.

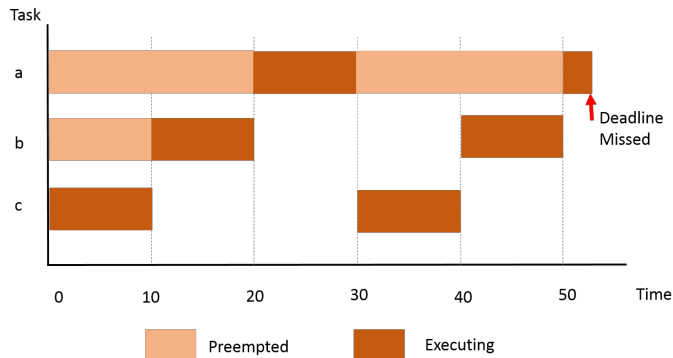
#### Example 4

Consider the following table 4.1 with a set  $\mathcal{T}$  of three tasks:

Task $\mathcal{T}$	Period $\mathcal{P}$	Computation Time $C$	Priority $PI$	Utilization $\rho$
a	50	12	1	0.24
b	40	10	2	0.25
c	30	10	3	0.33

**Table 4.1:** Example FPS

The sum of the utilization is  $\rho = 0.82$ , then the condition  $\rho \leq 0.693$  is not fulfilled. In the Figure 4.6, we can see how the task  $a$  does not fulfil its deadline following a FPS preemptive scheme.



**Figure 4.6:** Timeline for the task set  $\mathcal{T}$

We assume an organizational model where each role is associated with a set of services that determine the capabilities of the agent. With the presented task allocation model, it is possible to obtain a distribution of the different roles required for the teamwork between the available agents in the system's execution nodes.

To perform task allocation we minimize the number of nodes needed:

$$\min \sum_{j=1}^N x^j \quad (4.13)$$

where  $j$  identifies the node,  $N$  is the total number of available nodes and  $x^j = \{0, 1\}$ .

The constraints applied to the problem will now be described, taking into account the following notation:  $x_i^j$  where  $i$  identifies the task and  $j$  the node;  $\mathcal{T}$  is the total number of the tasks in the system;  $\rho_i^j$  is the utilization factor when executing the task  $i$  in the node  $j$ , it is obtained from the equation 11.10.

First, we establish the link between the objective function and the constraints in the following way (11.4):

#### Constraint 1

$$\forall i (x^j \geq x_i^j) \quad (4.14)$$

It means that for each task  $i$ , if a node has a task  $x_i^j$  assigned,  $x^j$  will be 1 adding the value to the final result of the objective function (11.12).

Now we must take into account the utilization factor of each node, so that the tasks assigned to each of the nodes do not exceed this factor. To formulate this constraint 11.4 the equation 4.10 is taken into account.

#### Constraint 2

$$\forall j \sum_{i=1}^{\mathcal{T}} \rho_i^j x_i^j \leq \min \left( \sum_{i=1}^{\mathcal{T}} x_i^j (2^{(1/\sum_{i=1}^{\mathcal{T}} x_i^j)} - 1), 1 \right) \quad (4.15)$$

Expanding the equation according to the number of nodes  $N$ , we have  $N$  inequations with the following form:

$$\begin{aligned}
\rho_1^1 x_1^1 + \rho_2^1 x_2^1 + \rho_3^1 x_3^1 + \dots + \rho_{\mathcal{T}}^1 x_{\mathcal{T}}^1 &\leq \min\left(\sum_{i=1}^{\mathcal{T}} x_i^1 (2^{(1/\sum_{i=1}^{\mathcal{T}} x_i^1)} - 1), 1\right) \\
\rho_1^2 x_1^2 + \rho_2^2 x_2^2 + \rho_3^2 x_3^2 + \dots + \rho_{\mathcal{T}}^2 x_{\mathcal{T}}^2 &\leq \min\left(\sum_{i=1}^{\mathcal{T}} x_i^2 (2^{(1/\sum_{i=1}^{\mathcal{T}} x_i^2)} - 1), 1\right) \\
&\dots \leq \dots \\
\rho_1^N x_1^N + \rho_2^N x_2^N + \rho_3^N x_3^N + \dots + \rho_{\mathcal{T}}^N x_{\mathcal{T}}^N &\leq \min\left(\sum_{i=1}^{\mathcal{T}} x_i^N (2^{(1/\sum_{i=1}^{\mathcal{T}} x_i^N)} - 1), 1\right)
\end{aligned}$$

The next constraint (11.4) establish that the same task  $i$  is not allocated in more than one node.

This constraint may be modified if necessary for replication. If any overload occurs, the task replication is a mechanism widely used in hard real-time that enables any of the nodes where the task is running to be able to finish on time. In our case, we will ensure that all tasks are performed by calculating the minimum number of nodes required for it; if there is an overload, the number of nodes required increases according to 11.12.

### Constraint 3

$$\forall j \sum_{i=1}^N x_i^j = 1 \quad (4.16)$$

The constraint 11.4 guarantee that all the assigned at least to one node.

### Constraint 4

$$\forall j \sum_{i=1, j=1}^{\mathcal{T}, N} x_i^j = \mathcal{T} \quad (4.17)$$

As with a VO, we work with roles that can be associated with different tasks. If the same role must, for example, execute two different tasks, then there must be a way to indicate that those tasks must be assigned to the same node where the agent with the corresponding role will work. With the following constraint (11.4), task  $i$  and task  $k$  will be allocated to the same node  $j$ .

### Constraint 5

$$x_i^j = x_k^j \quad (4.18)$$

Finally, we limit the values of  $x_i^j$  with the constraint 11.4 and 11.4. In this way, they can take only the value 0 or 1. If the node  $j$  has the task  $i$  assigned then  $x_i^j = 1$ , otherwise,  $x_i^j = 0$ .

**Constraint 6**

$$x_i^j \leq 1 \quad (4.19)$$

**Constraint 7**

$$x_i^j, \text{ integer} \quad (4.20)$$

#### 4.6.0.1

#### Method of resolution

With regard to solving assignment problems, there are two alternatives: exact methods and approximate methods. The first consists of solving problems exactly, with the risk of incurring longer computation times. The second involves using approximate methods with a high degree of approximation to the optimal solution but investing less time finding them. The nature of the task allocation problem is combinatorial and therefore the time required to find the optimal solution grows exponentially with the number of tasks considered [359].

These issues are included within an area known as combinatorial optimization. In the best case, these problems are classified as NP (non-deterministic polynomial) [123], which do not have an optimal solution, i.e., there is no algorithm with polynomial complexity to solve them. But a solution can be obtained (not as efficient or optimal), often taking very large spaces of time for small instances [241]. However, most combinatorial optimization problems belong to the family of NP-hard problems (a subclass of NP) [282].

The value of  $x^j$  in the objective function varies according to the task allocation and the constraints, which means that the value cannot be set in advance. It is necessary to establish all possible permutations  $x_i^j$  of the constraints and the permutations of  $x^j$  in the objective function. When facing this kind of problem the easiest approach is the linear programming (LP) or the integer linear constraint [235], but in our case we have non-linear constraints and we should use another method.

Heuristic methods are used to approximate the solution, but they are not exact and difficult time-bounded methods, and the outcome depends on the ability of the designer and their knowledge of the problem to develop. Although, the solutions generated in this way are generally quite good, most of these

algorithms do not adapt well to problems slightly different from those for which they were conceived [312].

Our time constraints do not allow us to use algorithms where many iterations are necessary to reach a solution, nor can we choose heuristics approximated methods. Therefore, we must look for a mechanism based on approximated methods, one that could feasibly be implemented by finding a good solution within a bounded time. It is therefore necessary to use an alternative method to reach a solution. Moreover, most known non-linear programming techniques, such as the multipliers of Lagrange [41] or the Kuhn-Tucker method [209] are problematic in their practical application due to the resulting equations after performing the partial derivatives of the constraints. Consequently, the method chosen is the Method of Approximation Programming (MAP) which is explained below.

#### 4.6.0.2

MAP (Method of approximation programming)

The Frank-Wolfe algorithm [114] finds a feasible solution to a nonlinear program with linear constraints, but since we have nonlinear constraints, we must use its extension, called MAP [24].

The MAP algorithm extends general nonlinear programs by making linear approximations to the constraints as well as the objective function. When the constraints are highly nonlinear, however, the solution to the approximation problem can become far removed from the feasible region since the algorithm permits large moves from any candidate solution. The MAP is a simple modification of this approach that limits the size of any move. As a result, it is sometimes referred to as a *small-step procedure*.

Let  $x^0 = (x_1^0, x_2^0, \dots, x_n^0)$  be any candidate solution to the optimization problem:

$$\max f(x_1, x_2, \dots, x_n) \quad (4.21)$$

subject to:

$$g_i(x_1, x_2, \dots, x_n) \leq 0 \quad (i = 1, 2, \dots, m).$$

Each constraint can be linearized, using its current value  $g_i(x^0)$  plus a linear correction term, as:

$$\hat{g}_i(x) = g_i(x^0) + \sum_{j=1}^n a_{ij}(x_j - x_j^0) \leq 0, \quad (4.22)$$

where  $a_{ij}$  is a partial derivative of constraint  $g_i$  with respect to variable  $x_j$  evaluated at the point  $x^0$ . This approximation is a linear inequality, which can be written as:

$$\sum_{j=1}^n a_{ij}x_j \leq b_i^0 \equiv \sum_{j=1}^n a_{ij}x_j^0 - g_i(x^0), \quad (4.23)$$

since the terms on the righthand side are all constants.

The MAP algorithm uses these approximations, together with the linear objective-function approximation, and solves the linear-programming problem:

$$\max z = \sum_{j=1}^n c_j x_j \quad (4.24)$$

subject to:

$$\begin{aligned} \sum_{j=1}^n a_{ij}x_j &\leq b_i^0 \quad (i = 1, 2, \dots, m) \\ x_j^0 - \delta_j &\leq x_j \leq x_j^0 + \delta_j \quad (j = 1, 2, \dots, n). \end{aligned}$$

the last constraints restrict the step size; they specify that the value for  $x_j$  can vary from  $x_j^0$  by no more than a user-specified constant  $\delta_j$ .

When the parameters  $\delta_j$  are selected to be small, the solution to this linear program is not far removal from  $x^0$ . We might expect then that the additional work required by the line-segment optimization of the Frank-Wolfe algorithm is not worth the slightly improved solution that it provides. MAP operates on this premise, taking the solution to the linear program (4.25) as the new point  $x^1$ . The partial-derivative data  $a_{ij}$ ,  $b_i$  and  $c_j$  is recalculated at  $x^1$ , and the procedure is repeated. Continuing in this manner determines points  $x^1, x^2, \dots, x^k, \dots$  and, as in the Frank-Wolfe procedure, any point  $x^* = (x_1^*, x_2^*, \dots, x_n^*)$  that these points approach in the limit is considered a solution.

Since many of the constraints in the linear approximation merely specify upper and lower bounds on the decision variables  $x_j$ , the bounded-variable version of the simplex method is employed in its solution. Also, the constants  $\delta_j$  are



**Algorithm 3** Steps of the MAP Algorithm

- 1: Let  $x^0 = (x_1^0, x_2^0, \dots, x_n^0)$  be any candidate solution, usually selected to be feasible or near-feasible. Set  $k = 0$ .
- 2: Calculate  $c_j$  and  $a_{ij} (i = 1, 2, \dots, m)$ , the partial derivatives of the objective function and constraints evaluated at  $x^k = (x_1^k, x_2^k, \dots, x_n^k)$ . Let  $b_i^k = a_{ij}x^k - g_i(x^k)$ .
- 3: Solve the linear-approximation problem (4.25) with  $b_i^k$  and  $x_j^k$  replacing  $b_i^0$  y  $x_j^0$  respectively. Let  $x^{k+1} = (x_1^{k+1}, x_2^{k+1}, \dots, x_n^{k+1})$  be its optimal solution. Increment  $k$  to  $k + 1$  and return to step 1.

usually reduced as the algorithm proceeds. The common method to implement this algorithm is to reduce each  $\delta_j$  by between 30 and 50 percent at each iteration.

If we apply this method to our problem, the constraints equations are:

$$\begin{aligned}
 \forall j \quad x^j &\geq x_i^j \\
 \sum_{j=1}^N x_i^j &= 1 \\
 \sum_{i=1, j=1}^{\mathcal{T}, N} x_i^j &= \mathcal{T} \\
 x_i^j &\leq 1 \text{ integer} \\
 x_i^j &\leq x_k^j \\
 \forall j \quad \sum_{i=1}^{\mathcal{T}} \frac{\partial g_j}{\partial x_i^j}(y) x_i^j &\leq \sum_{i=1}^{\mathcal{T}} \frac{\partial g_j}{\partial x_i^j}(y) y_i - g_i(y)
 \end{aligned}$$

where:

$$\begin{aligned}
 \frac{\partial g_j}{\partial x_i^j} &= p_i^j \leq \sum_{i \neq k} x_k^j \frac{-1}{(\sum_{i=1} x_i^j)^2} \cdot 2 \sum_{i=1}^1 \frac{1}{x_i^j} \cdot \ln 2 \\
 &+ 2 \sum_{i=1}^1 \frac{1}{x_i^j} + x_i^j \frac{-1}{(\sum_{i=1} x_i^j)^2} \cdot 2 \sum_{i=1}^1 \frac{1}{x_i^j} \cdot \ln 2 - 1
 \end{aligned}$$

the equation 11.20 is the value of the partial derivative in the y-point.

$$\frac{\partial g_j}{\partial x_i^j}(y) \tag{4.25}$$

$g_i$  is the constraint  $j$ ,  $\mathcal{T}$  is the set of the tasks and  $y$  is the point which corresponds to the current solution.

---

**4.6.0.3**

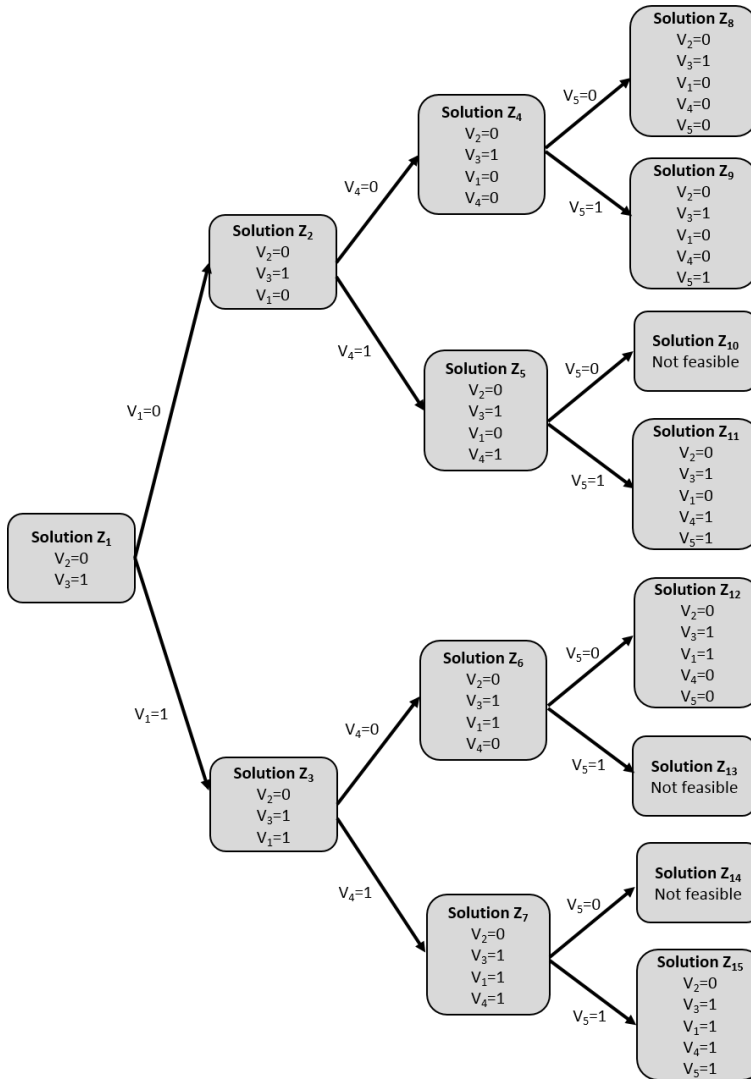
 The Branch and Bound method
 

---

The Branch and Bound method [212] is a type of divide and conquer technique. In the *branch* part, the large problem is divided into a few smaller ones. The conquering part is done by estimate how good a solution we can get for each smaller problems (to do this, we may have to divide the problem further, until we get a problem that we can handle), that is the *bound* part [196].

In our problem, including the constraint 11.4 presented in 4.6, which force the solutions to be integers, is a very costly process in terms of computational time. So, we use the linear programming relaxation [3] to estimate the solution of an integer programming. This means that for an integer programming model  $P$ , we get a reduced linear programming model by dropping the requirement that all variables must be integers. This is called the linear programming relaxation of  $P$ ,  $P_r$ . First, we obtain a valid solution to our  $P_r$  with the MAP method, but we need to approximate the values to the nearest integers  $\{0, 1\}$ . So, the best form to calculate them is using a threshold function  $\mathcal{T} = f(v_i)$  where  $v_i$  are the results after the MAP method and then, use the Branch and Bound method for the rest of the values.

$$\mathcal{T} = f(v_i) = \begin{cases} 0 & \text{if } v_i \leq 0.15 \\ 1 & \text{if } v_i \geq 0.85 \\ \text{BranchAndBound} & \text{otherwise} \end{cases}$$



**Figure 4.7:** Example of the Branch and Bound method to approximate the values to integers

For the case  $f(v_i) = 1$ , the values  $v_i$  that fulfill the threshold  $v_i \geq 0.85$  are arranged in a decreasing order and starting from the higher value, they are approximated to 1 if they comply the constraint 2 of the model.

A graphical example is included in Figure 4.7 with a set of values  $\mathcal{S} = v_1 = 0.35, v_2 = 0.10, v_3 = 0.88, v_4 = 0.72, v_5 = 0.47, v_6 = 0.68$ . Applying the threshold function  $\mathcal{T} = f(v_i)$  the values are  $\mathcal{S} = \{v_1 = 0.35, v_2 = 0, v_3 =$

$1, v_4 = 0.72, v_5 = 0.47\}$ , then the variables  $v_2$  and  $v_3$  have already a result and are fixed at the beginning of the method.

To expand the tree faster is possible to include several constraints at the same time in one branch. The method have two stopping criteria: it examines or "pruned" all the possible nodes or the preset maximum time for getting a solution runs out. In this case, the best value found at that moment will be a valid solution but it may not be the best to the global problem.

As we are trying to minimize the objective function, at the end of the process we choose the  $\min Z_i$  obtained in the last step.

## 4.7

## Conclusions

In this part we have proposed an effective model which is divided in two different steps. First, the model calculates the WCET when an agent desires to form part of a virtual organization and wants to include tasks or behavior in the real time system, i.e. to calculate the WCET in emergent behaviours. And after, using the previous estimation, the model generates a scheduling plan to allocate tasks for the whole system, i.e. a global scheduling. This model has been specially designed for VOs that interact continuously with the environment and have to fulfill real-time constraints. For achieve this, we have applied a combination of mechanisms that all together allow us to present a novel model for the WCET estimation followed by a scheduling and task allocation in real-time VOs.

To sum up, we have used many techniques to fulfil the steps of the model and solve all the difficulties. For the estimation of the WCET, we have studied concepts such as Java bytecodes and the functioning of the JVM, the ICFG, the IPET technique to improve the time estimation, etc. For the scheduling and task allocation we have gone further in the FPS algorithm and its feasibility test, integer linear programming, the MAP algorithm, the Branch and Bound Method, etc. Mixing all these elements, we have obtained a successful method that fulfil the initial hypothesis: *it is possible to develop a scheduling and task allocation model that successfully enables the agents integrated into a VO to fulfil their time constraints. This model will be theoretically formalized and then implemented to evaluate the results.* The next step is to demonstrate with results the effectiveness of the method.

In the next part of the document, we present different versions of the agent-platform PANGEA, where this model has been included. In Part 5, we include the results of the model and its integration in a real case study.

# Part IV

PANGEA+RT PLATFORM



# 5

## PANGEA

---

*New trends in multi-agent systems call for self-adaptation and high dynamics, hence the new model of open MAS or virtual organization of agents. However, as existing agent platforms are not yet equipped to support this behavior, it is necessary to create new systems and mechanisms to facilitate the development of these new architectures. In this chapter, we present PANGEA, an agent platform to develop open multi-agent systems, specifically those including organizational aspects such as virtual agent organizations. The platform allows the integral management of organizations and offers tools to the end user. Additionally, it includes a communication protocol based on the IRC standard, which facilitates implementation and remains robust even with a large number of connections.*

### Contents

---

<b>5.1</b>	<b>Introduction</b>	<b>100</b>
<b>5.2</b>	<b>Related works</b>	<b>102</b>
<b>5.3</b>	<b>PANGEA overview</b>	<b>106</b>
5.3.1	Reorganization and task allocation model	110
<b>5.4</b>	<b>The PANGEA database</b>	<b>112</b>
<b>5.5</b>	<b>The monitoring tool</b>	<b>115</b>
<b>5.6</b>	<b>The norms in PANGEA</b>	<b>117</b>
<b>5.7</b>	<b>The communication module</b>	<b>119</b>
5.7.1	Testing the communication	122
<b>5.8</b>	<b>Testing the SnifferAgent</b>	<b>124</b>
<b>5.9</b>	<b>The subscription model</b>	<b>126</b>
<b>5.10</b>	<b>The Gateway agent</b>	<b>128</b>
5.10.1	Request protocol	129
5.10.2	Subscription protocol	130
5.10.3	Contract-net protocol	131
5.10.4	Inform protocol	131
<b>5.11</b>	<b>Conclusions</b>	<b>132</b>

---



---

## 5.1 Introduction

---

Nowadays, one of the research lines that the multi-agent systems are following is directed toward ensuring that these systems become more open and dynamic. An open MAS [27] should allow for the interaction between heterogeneous agents, which change over time, and architectures and even different languages. Because of their inherent changing nature, we cannot rely on agents' behaviour when it is necessary to establish controls on the basis of norms or social rules. For this reason, and because of the characteristics of open environments, new approaches are needed to support evolutive systems and to facilitate their growth and runtime updates. The dynamics of open environments is one of the reasons that have encouraged the use of VOs. A VO [111] [113] is an open system designed for grouping; it allows for the collaboration of heterogeneous entities and provides a separation between the form and function that define their behaviour.

VOs are conceived as a set of agents with roles and rules that determine their behavior and where previously established capabilities play a crucial role. Possible topologies and organizational aspects well as their communication and coordination mechanisms determine largely the flexibility, dynamism and openness that the multi-agent system can offer. The concept of organization is seen as a promising solution to manage the coordination of the agents and control their behaviours and actions. Every organization needs coordination support to determine explicitly how to organize and carry out the actions and tasks within it. There are different platforms, which will be shown later, that allow for the creation of multi-agent systems. These platforms greatly facilitate the task of working with agents. However, in terms of platforms that allow for the creation of a VO, the number is drastically reduced; it is in fact difficult to find a single platform that covers all the requirements that a VO requires, such as reorganization and adaptation facilities, norm compliance, different organizational topologies, service and role management.

At the same time, distributed multi-agent systems have become increasingly sophisticated in recent years, with a growing potential to handle large volumes of data and coordinate the operations of many organizations [155]. Distributed



intelligent systems are based on the use of cooperative agents, where each agent independently handles a small set of specialized tasks and cooperates to achieve the system-level goals and a high degree of flexibility [145]. Another problem for distributed systems is that they must now be able to accept requests from different devices. There is, furthermore, a rapidly growing use of mobile devices with limitless connection possibilities and computing power. From a practical perspective, multi-agent systems face yet another challenge: obtaining light intelligent agents that can be deployed in this type of device without relinquishing their capabilities.

For the aforementioned reasons, PANGEA introduces a new protocol based on the Internet Relay Chat (IRC) standard to facilitate communication in distributed multi-agent systems. This protocol facilitates cooperation between agents, which is critical in these kinds of systems. It includes a robust communication model that allows intelligent agents to connect from a variety of devices. This communication is easy to implement and enables agents deployed in different devices to cooperate. These agents can be developed both quickly and in any language using the tools provided by the platform. PANGEA also includes facilities for implementing VOs and suborganizations, following any topology and with the appropriate tools for managing the VO itself. It is important to highlight that PANGEA is not a framework or a simple tool; it is a complete platform for the execution and management of VOs. Following an organizational perspective, the most important concepts to consider are:

- Organizations are social entities formed with patterns of communication, responsibilities or tasks, and are regulated by norms and restrictions.
- Entities in an organization are not independent of each other; instead they interact with each other. Different structures are possible and entities are sorted according to the topology of these structures. Some possible topologies are hierarchies, oligarchy, congregations, federations, coalitions, etc.
- Organizations are linked to the norms. There are social norms that define the consequences of the entities' actions, control the access and communications, and are forced to acquire certain behaviors or not allow them.
- Organizations such as open systems are dynamic and should facilitate the input and output of heterogeneous entities, taking different roles by these entities and the group formation.

---

## 5.2 Related works

---

All platforms for creating multi-agent systems that exist to date should be studied according to two principal categories: those that simply support the creation and interaction of agents, and those that permit the creation of virtual organizations with such key concepts as norms and roles. Initially, most of the agents' platforms do not allow VOs. This is the case of FIPA-OS [287], April Agent Platform (AAP) [239] and JASON [52] [53], which its main contribution is the easy with which BDI [76] agents can be implemented [299]. In practice, the platform that is most commonly used to develop multi-agent systems in real case studies is Java Agent DEvelopment Framework (JADE) [40] and Jadex [284]. Jadex is a software framework and a extension of JADE for the creation of goal-oriented agents following the BDI model.

One of the more recent platforms is JIAC [158] [229], a service-aware framework. JIAC V is a Java based agent framework with an emphasis on industrial requirements such as software standards, security, management, and scalability. It combines agent technology with a service oriented approach. Together with the framework, a new language called JADL++ was created; this platform stands out because of its service matching capabilities and an excellent usability. Until now, these platforms can create agents (some with different models), follow their life cycle and manage communication and services. However, in the case of VO, it is necessary to take into account the normative and organizational aspects that the platform itself should provide.

MadKit [150] was one of the first platforms to consider basic organizational aspects. The platform architecture is rooted in the AGR (agent-group-role) model [149] developed in the context of the AALAADIN project. Another pioneering platform in terms of its structural aspect was Jack Teams [225], which introduced the concept of "Team-oriented programming" as an intuitive paradigm to encapsulate coordination activity. The JACK Teams extension introduces the new constructs team, role, teamdata, and teamplan. Various authors have used JACK Teams in their research, notably [107] [181] [44] [81]. The main disadvantage of this platform is that it only allows hierarchical team structures. S-MOISE+ is an organizational middleware that follows the MOISE+ model [168] [165]. In MOISE+ a multi-agent system is specified as an organization, distinguishing three main aspects. The structural aspect is in charge of the structure or topology of the organization. The functional aspect is in charge of the organizational operations and the tasks that should be carried out. And the normative aspect specifies the permission and obligations of each role. The printed material used for this research includes systems

that were developed in conjunction with JASON, using S-MOISE+ as the middleware to achieve a more complete model [167]. Hence the emergence of J-Moise+ [166], which is very similar to S-Moise+ regarding the overall system concepts. AMELI is a middleware that can work with electronic institutions (similar to VO) [331]. An innovative feature of AMELI is its general purpose; because it can interpret any institution specification, organization or topology, it can be regarded as domain-independent, which is the reason why AMELI offers a higher (social) level of abstraction [103].

One of the main disadvantages of platforms geared toward VO is that the concept of service is somewhat diminished, which impacts the management associated with these services and the DF described in the FIPA standard [13]. This need led to the creation of the THOMAS framework [28]. THOMAS is based on the idea that no internal agents exist and architectural services are offered as Web Service (WS). As a result, the final product is entirely independent of any internal agent platform and fully addressed for open multi-agent systems [135]. Until now, all tools used to create, manage and control VOs are frameworks or middlewares that require other agent platforms to be able to integrally develop a VO. This implies that the deficiencies of agent platforms are further diffused if the layer for managing the VO cannot solve the problem. One of the most complete and recent platforms that have been found in the literature review is Janus [122]. Janus is the evolution towards organizations of the platform previously known as TinyMAS (no longer under development.). This platform was specially designed to deal with the implementation and deployment of Holonic multi-agent system (HMAS) [130]. The key aspects handled by the platform are organizations, roles, interactions and capacities [122]. However, it disregards the concept of norm and service. The platform does not explicitly consider the normative aspects of the organizations; they are instead included within the concept of role.

In summary, to deal with all aspects of complex systems, MAS such as VO, it is necessary to deal with multiple levels of abstractions and openness, which is not the case for most solutions [78]. In addition, it is needed to establish controls on the basis of norms or social rules, in the interactions and topologies, these characteristics are also no included in most solutions. Moreover, although the agent frameworks or platforms have similarities, there are subtle differences, too. Each framework uses a different model file syntax and provides different libraries. In our platform, we have tried to use standards that have already demonstrated their robustness and are known within the research community, making the implementation of new architectures easier and highly reliable.

MadKit [150]	The platform architecture is rooted in the AGR (agent-group-role) model [149] and developed in the context of the AALAADIN project. One important characteristic is the Agentification of services [151].
Jack Teams [160] [225]	An extension to JACK Intelligent Agents [62] which provide a team-oriented modelling framework. The JACK Teams extension introduces the new concepts of: team, role, teamdata and teamplan [334] [107].
S-MOISE+ [168]	An organizational middleware that follows the MOISE+ model. A multi-agent system is specified as an organization, distinguishing three main aspects [165] [82]: structural, functional and normative.
J-Moise+ [166]	Very similar to S-Moise+ regarding the overall system concepts. In S-Moise+ agents are programmed in Java (using a very simple agent architecture), while in J-Moise+ they are programmed in AgentSpeak, a programming language based on BDI concepts and thus more suitable for programming agents.
AMELI [106]	A middleware that can work with electronic institutions (similar to VO) [103]. An innovative feature of AMELI is its general purpose (it can interpret any institution specification, organization or topology). This allows it to be regarded as domain-independent, the reason why AMELI offers a higher (social) level of abstraction.
Electronic Institutions Development Environment, EIDE [66]	Consists of the integration of various tools: ISLANDER (a graphical specification language), the middleware AMELI [106], SIMDEI (a debugging and monitoring tool,) and ABuilder (an agent-shell builder that can produce an agent “skeleton” for each agent role).
RICA-J (acronym for RICA-Jade) [324]	Allows programming multiagent systems based on the RICA theory (role/interaction/communicative action). RICA-J is designed as a framework on top of the existing JADE tool that can reuse the middleware aspects required for supporting agent interactions, as well as the different agent abstractions provided.

THOMAS framework [306]	There are no internal agents and architectural services offered as web services. As a result, the final product is entirely independent of any internal agent platform and fully addressed for open multiagent systems [135]. The main components of THOMAS are: Service Facilitator (SF), Organization Manager Service (OMS) and Platform Kernel (PK).
Janus [122]	Janus is the evolution towards organizations of the platform previously known as TinyMAS (no longer under development.). This platform was especially designed to deal with the implementation and deployment of holonic and multiagent systems. Its key feature is that it supports the implementation of the concepts of role and organization as first-class entities (a class in the object-oriented sense) [131].
BVM (Brahms Virtual Machine) [330]	A multi-agent discrete-event engine. It cannot be defined as a platform per se; instead it is a set of development tools to develop and simulate human organizations and work processes. It uses its own language and its main application is the simulation of multiagent models based on norms [329].
ICARO-T [126]	The distinguishing factor of this framework is the use of component patterns for modelling MAS: agent organization pattern, cognitive and reactive agent patterns, and resource patterns. This facilitates the implementation of the agents.

**Table 5.1:** Summary of middlewares or platforms for VOs

The idea of creating PANGEA [392] emerges from the analysis of this table where we find disadvantages to all existing platforms. While Madkit [150] does work with the concept of role, it does not consider a role to be a class entity. In fact, the behavior associated with the role is directly implemented into the agent who assumes the role, which leads to the problem of roles being strongly linked to the agent's architecture. This approach harms the reusability and modularity [106] of organizations. JACK Teams only allows hierarchical team structures. S-MOISE+ lacks a monitoring mechanism to detect whether the agents actually fulfill the goals belonging to their required missions. The printed material used for this research includes systems that were developed in conjunction with JASON, using S-MOISE+ as the middleware to achieve a more complete model [167]. RICA-J uses JADE. Another drawback of this approach, however, is that it defines an agent mainly by means of the role it will

play during its life; such a definition could clash with other agent models and theories forcing the agent developer to mentally shift toward a new definition of agent. AMELI and EIDE can be considered complete platforms. They are oriented toward electronic institutions [105], which is not our desired paradigm, and not VOs.

In general, one of the main disadvantages of platforms geared toward VOs is that the concept of service is somewhat diminished, which impacts the management associated with these services and the DF described in the FIPA standard. In addition to an extension of the DF adapted to the VO, it would also be necessary to have an extension of the AMS to enable the creation of dynamic and reorganizable structures in execution time. THOMAS overcomes this problem, but its monitoring tools are poor and do not provide implementing facilities. Janus is one of the most complete and recent platforms, but it disregards the concept of norm and service. It does not explicitly consider the normative aspects of the organizations; they are instead included within the concept of role. However, in order to allow for more flexible open systems, it is necessary to specify the norms beyond the role. The platform also fails to consider the concept of service as an independent entity and instead includes it within the capacity associated with the roles. In summary, to deal with all aspects of complex systems, whether MAS or VO, it is necessary to deal with multiple levels of abstractions and openness, which is not the case for most existing solutions [78].

At the end of the chapter, the figure 5.21 includes a brief compilation of well-known VOs middlewares or platforms.

---

### **5.3** PANGEA overview

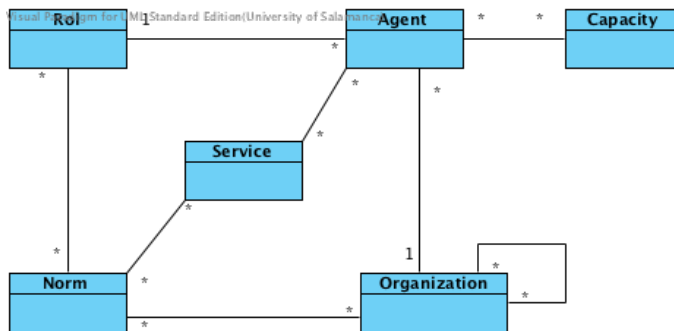
---

As previously mentioned, a platform that can integrally create, manage and control VOs was developed for this study. In general terms, the proposed platform includes the following characteristics:

- Different models of agents, including a BDI and CBR-BDI architecture [76].
- Ability to control the life cycle of agents with graphic tools.
- A communication protocol that allows broadcast communication, multi-cast according to the roles or suborganizations, or agent to agent.
- A debugging tool.

- Module for interacting with FIPA-ACL agents.
- Service management and tools for discovering services.
- Web services.
- Flexibility in allowing organizations with any topology and suborganizations.
- Organization management.
- Services for dynamically reorganizing the organization [391].
- Services for distributing tasks and balancing the workload [391].
- A business rules engine to ensure compliance with the standards established for the proper operation of the organization.
- Java programming and easily extensible.
- Possibility of having agents in various platforms (Windows, Linux, MacOS, Android and IOS)
- Interface to oversee the organizations.

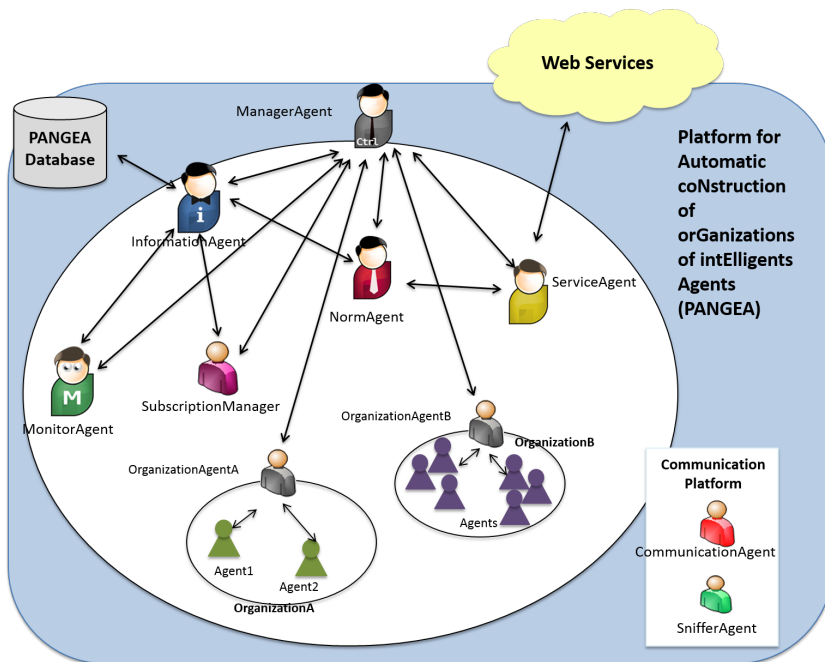
Figure 5.1 displays the principal classes of the system, and illustrates how the roles, norms and the organizations themselves are classes that facilitate the inclusion of organizational aspects. The services are also included as classes completely separate from the agent, facilitating their flexibility and adaption. Capacity determines the reasoning mechanisms available to the agent.



**Figure 5.1:** Main classes of the system

When launching the main container of execution, the communication system is initiated; the agent platform then automatically provides some agents to facilitate the control of the organization. In the Figure 5.2, we can see the agents' roles and their interactions.

- **OrganizationManager**: the agent responsible for the actual management of organizations and suborganizations. It is responsible for verifying the entry and exit of agents, and for assigning roles. To carry out these tasks, it works with the **OrganizationAgent**, which is a specialized version of this agent.
- **OrganizationAgent**: it is a specialized version of the **OrganizationManager**, which is introduced automatically in each suborganization to help the **OrganizationManager** and avoid its overload.
- **InformationAgent**: the agent responsible for accessing the database containing all pertinent system information.
- **ServiceAgent**: the agent responsible for recording and controlling the operation of services offered by the agents. It works as the **Directory Facilitator** defined in the FIPA standar.



**Figure 5.2:** PANGEA agents

- **NormAgent**: the agent that ensures compliance with all the refined norms in the organization.
- **CommunicationAgent**: the agent responsible for controlling communication among agents, and for recording the interaction between agents and organizations.



- SnifferAgent: manages the message history and filters information by controlling communication initiated by queries.

The platform enables two modes of operation. In the first mode, the agents reside in the machine itself, while in the second mode the platform allows for the possibility of initiating all agents in different machines. The latter case has the disadvantage of allowing only minimal human intervention since it is necessary to previously specify the address of the machine where each of the agents are to reside; however it has the advantage of greater system distribution. We have created a service-oriented platform that can take maximum advantage of the distribution of resources. To this end, all services are implemented as Web Services. This makes it possible for the platform to include both a service provider agent and a consumer agent, thus emulating a client-server architecture. The provider agent (a general agent that provide a service) knows how to contact the WS, the rest of the agents know how to contact with the provider agent due to their communication with the ServiceAgent, which contains this informacion about services. In section 3.1, when the communication platform is explained, the Figure 5 shows how a general provider agent (one who has a service to offer) can enter a suborganization and register its services using the specific communicational primitives.

Once the client agent's request has been received, the provider agent extracts the required parameters and establishes contact. Once received, the results are sent to the client agent. Using WS also enables to introduce Service-oriented Application (SOA) [188] into MAS systems and in the platform . SOA is an architectural style for building applications that use services available in a network such as the web. It promotes loose coupling between software components so that they can be reused. Applications in SOA are built based on services. A service is an implementation of a well-defined functionality, and such services can then be consumed by clients in different applications or processes. SOA allows for the reuse of existing services and a level of flexibility that was not possible before in the sense that:

- Services are software components with well-defined interfaces that are implementation-independent. An important aspect of SOA is the separation of the service interface from its implementation. Such services are consumed by agent clients that are not concerned with how these services will execute their requests.
- Services are self-contained and loosely coupled encouraging independence.
- Services can be dynamically discovered
- Composite services can be built from aggregates of other services [378].

Each suborganization or work unit is automatically provided with an OrganizationAgent by the platform during the creation of the suborganization. This OrganizationAgent is similar to the OrganizationManager, but is only responsible for controlling the suborganization, and can communicate with the OrganizationManager if needed. If another suborganization is created hierarchically within the previous suborganization, it will include a separate OrganizationAgent that communicates with the OrganizationAgent from the parent organization. These agents are distributed hierarchically in order to free the OrganizationManager of tasks. This allows each OrganizationAgent to be responsible for a suborganization although, to a certain extent, the OrganizationManager can always access information from all of the organizations. Each agent belongs to one suborganization and can only communicate with the OrganizationAgent from its own organization; this makes it possible to include large suborganizational structures without overloading the AgentManager. All of the OrganizationAgents from the same level can communicate with each other, unless a specific standard is created to prevent this. One possible topology is shown in Figure 5.3, with the ManagerAgent establishing communication with the OrganizationManager.

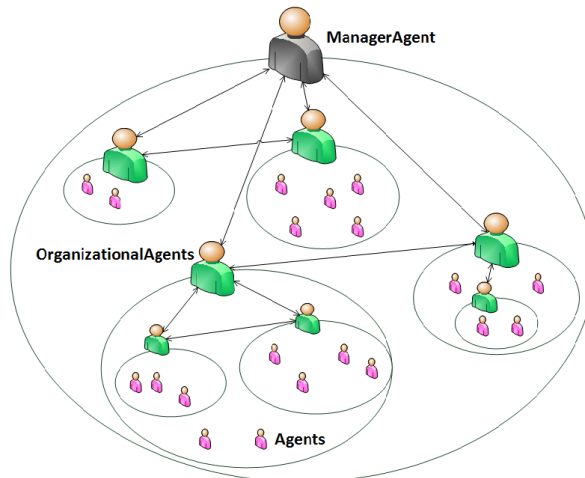


Figure 5.3: OV topology

### 5.3.1

### Reorganization and task allocation model

Along this part of the dissertation, we are explaining the evolution of the PANGEA platform. The first approach of PANGEA (without any extension) has its own reorganization and allocation model published in a previous work

[391]. For this purpose, a genetic algorithm, the queuing theory, and a CBR are used to obtain an efficient distribution.

In the Figure 5.4, the agents' roles involved and their interactions are shown.

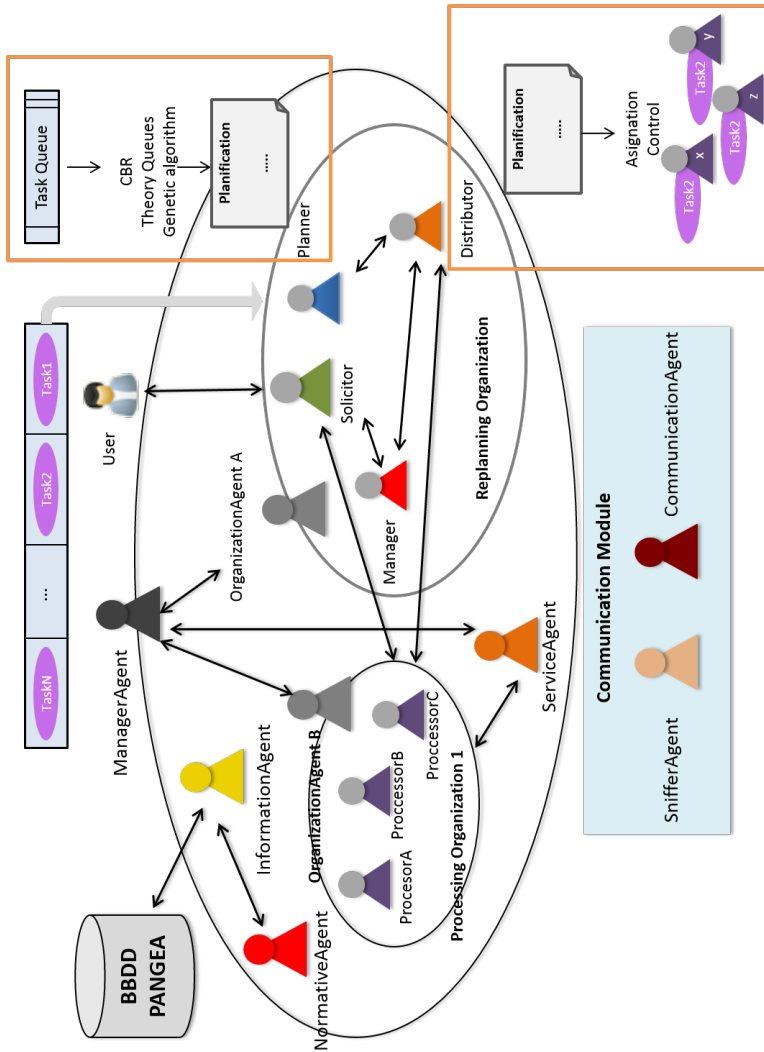


Figure 5.4: Overview of the agents involved

The model proposed in our mentioned work focuses on developing a planning mechanism to coordinate the agents found in the VOs. Thus, first we will set out the roles that these agents can take:

- Processor role. Responsible for carrying out the activities required for

each specific task. For this reason, the responsible agent will specialize depending on the type of tasks the system must solve.

- **Planner role.** Design the overall plan to be implemented by the organization. Sets the number of processor agents and makes the distribution of tasks.
- **Distributor role:** Distributing tasks according to its completion by the agents and checks that each task is being processed within time limits to serve the plan.
- **Manager role:** This agent manages all the information of the task and communicates to the user.

As previously said, the result and the detailed model can be found in [391] but this model cannot be applicable in real-time environments, then it is important to note that PANGEA+RT includes the new model developed specially for real-time in this dissertation.

## 5.4

### The PANGEA database ---

To create an organization, a database consisting of the following information is needed in order to store all the relevant information. The scheme of the database can be seen in the figure 5.5.

*A list of services (SERVICE).* It acts as DF. The purpose of this list is to store all the services that provide the agents or other entities, to others agents that are looking for some service or capability. This is similar to the yellow pages. If necessary, it must also be possible to register a new service in the system. To do so, a service has to be well defined and specified. The following attributes are compulsory for each service:

- Each service must have a unique identifier.
- Name
- Description
- **Provider agents:** all agents that provide this service should be included. It is important to note that the way to perform the task of the service is independent of its definition, so each provider agent can perform the service in a different way or with a different implementation than others.

- Norms associated to the service: in this section, the agent's roles that can consume this service must be specified. It are similar to permissions for each service, and not every agent can access them. Moreover, other norms can be defined, such as dates, agent attributes, number of calls, etc.

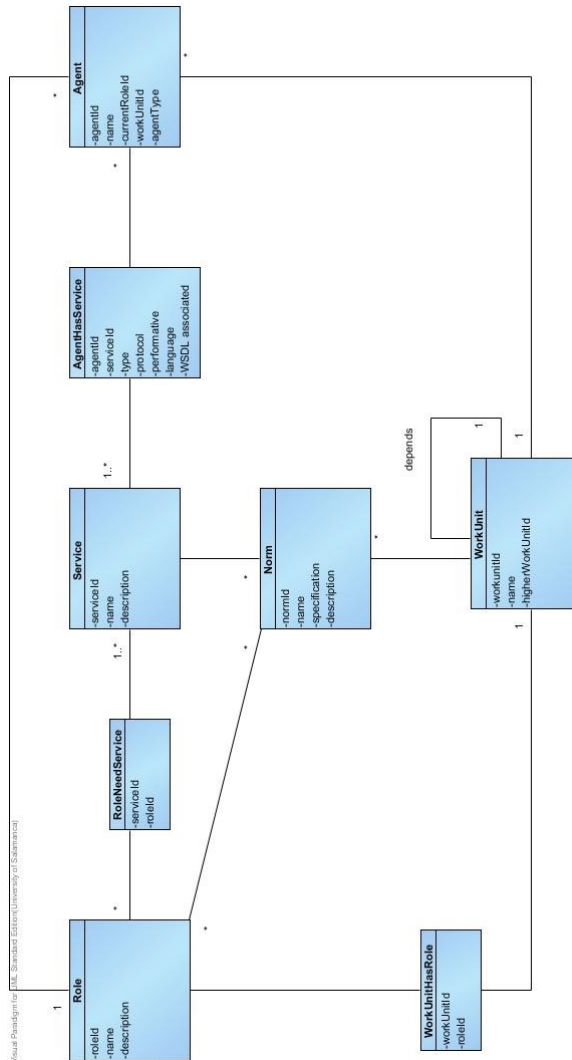
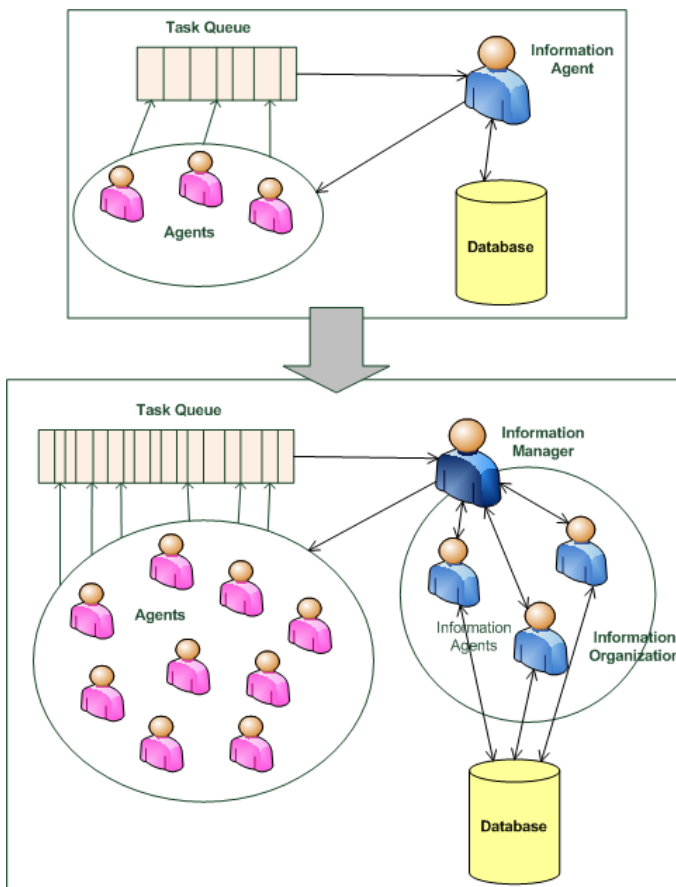


Figure 5.5: Database

*A list of members (AGENT).* All agents that work in the organization must be registered here. Each agent must be uniquely identified including both its local name and the address of the machine on which it is located. A list of roles

(ROLE), which includes the roles that the agents can take. A role must meet a certain goal or goals, so it will need to consume services. Similarly, a role also provides services. When an agent takes a particular role, it has to adapt to the model of that role, i.e., it must be able to provide and do everything that the role in question poses.

*A list of suborganizations or units of work (WORK UNIT).* An organization of agents may have various structures or topologies. It is necessary to know how the topology is, to configure the norms and the communication among the agents. Moreover, it is common for agents to be divided recursively into groups, and each group has a leader.



**Figure 5.6:** InformationAgent replication

The database is managed by the agent InformationAgent. Except for the CommunicationAgent (which has its own replication mechanism) and the OrganizationManager (which has the similar OrganizationAgent to avoid overload),

the remaining agents that manage the platform are controlled by the OrganizationManager when faced with the possibility of work overload. Each agent starts with a queue of messages. If the queue is too long, the agent itself will be responsible for instantiating new agents to share the work, thus creating a new organization. This occurs most commonly with the InformationAgent. This agent, who is in charge of data base access, can find itself overloaded with requests. Using a duplication mechanism, it can create instances of other InformationAgents although they would each be dependent on the initial InformationAgent. The newly created InformationAgents do not have their own task queue; instead the parent InformationAgent is in charge of managing a single task queue that it assigns to the children InformationAgents according to the task distribution mechanism. In this way, the platform would go from being an InformationAgent to an InformationOrganization; that is, an organization that would carry out tasks originally assigned to a single agent. This would avoid the overload or duplication of the message queues for each agent (Figure 5.6).

## 5.5

## The monitoring tool

PANGEA have a tool that can monitor the status of the platform (connected agents and existing organizations) and the messages that are passed between agents or between agents and organizations.

To carry out this almost real-time monitoring, the MonitorAgent was created. This agent is in charge of redirecting all message traffic to the visualization tool. The agent is responsible for listening to all events that occur in the system (agent connection, agent disconnection, an agent joining an organization, message interaction, etc.).

The server architecture was slightly modified for the MonitorAgent tasks. With this new configuration, a message is generated at all points where the system receives an event. In this case, the sender is the OrganizationManager and the receiver the MonitorAgent. It has one of the following formats:

- **[Date and Time] -> JOIN -> [Agent]**  
 2014/03/28 18:30:17->JOIN->suma->#arithmetic  
 The agent agent03 is connected to #initial-world by autojoin.
- **[Date and Time] -> JOIN -> [Agent] -> [Organization]**  
 2014/03/28 18:31:47->JOIN->add->#arithmetic  
 The agent add is connected to the #arithmetic organization.

- [Date and Time] -> PART -> [Agent] -> [Organization]  
2014/03/28 18:33:40->PART->add->#arithmetic  
The agent add leaves the organization #arithmetic.
- [Date and Time] -> QUIT -> [Agent]  
2014/03/28 18:48:03->QUIT->info\_agent  
The agent info\_agent has been disconnected.
- [Date and Time] -> QUIT -> [Agent] -> Agent PING TIMEOUT  
2014/03/28 18:50:27->QUIT->info\_agent->Agent PING TIMEOUT  
The agent info\_agent has been disconnected because it does not reply to the PING messages.
- [Date and Time] -> PRIVMSG -> [Agente Origen] -> [Agente Destino] -> [Message]  
2014/03/28 18:53:50->PRIVMSG->agent18->news\_agent->getnews  
The agent agent18 sends to the agent news\_agent a message with the text getnews asking for the service.
- [Date and Time] -> PRIVMSG -> [Agente Origen] -> [Agente Destino] -> [Message]  
2014/03/28 19:08:44->PRIVMSG->agent09->#p\_subs09->hello  
The agent agent09 sends to the suborganization p\_subs09 a message with the text hello.

Now, there is an agent capable of real-time monitoring everything that happens in the platform. Moreover, as the visualization tool may be used on more than one machine at a time, each instance of this tool is a new agent that we call MonitorClientAgentX, where X is an available random number.

The communication follows the scheme shown in the Figure 5.7.

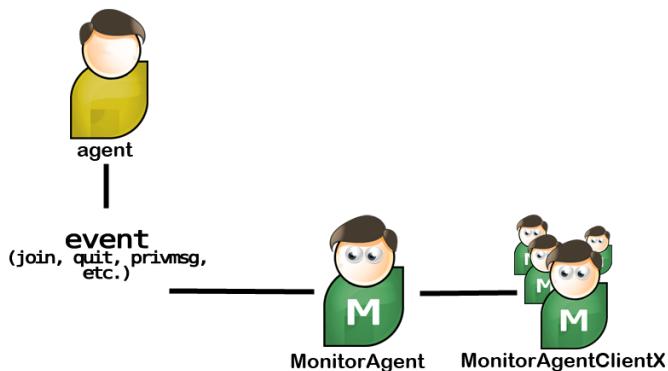


Figure 5.7: Communication scheme for representing events in the monitoring tools



When an agent generates an event, the platform translates it through a message to the `MonitorAgent` and then communicates the event to all the connected `MonitorClientAgentX`. Each instance of a monitoring tool creates a `MonitorClientAgentX` to receive the events.

## 5.6 The norms in PANGEA

The norms that control the system are indispensable in the management of a virtual organization because they are in charge of the organizational aspect. Not all interactions are allowed in an organization, therefore, the PANGEA platform includes the `NormAgent` to provide this functionality. There are different privileges, roles or other criteria that will make a message deliverable or not, according to the established norms (previously or in real-time through an on-line tool).

The norms are stored in an xml file, which will be read by the agent in charge of the rules (`NormAgent`) whenever they change. As a result, it is not necessary to access the database since the request and the response messages to read the norms and the accesses to the database are saved.

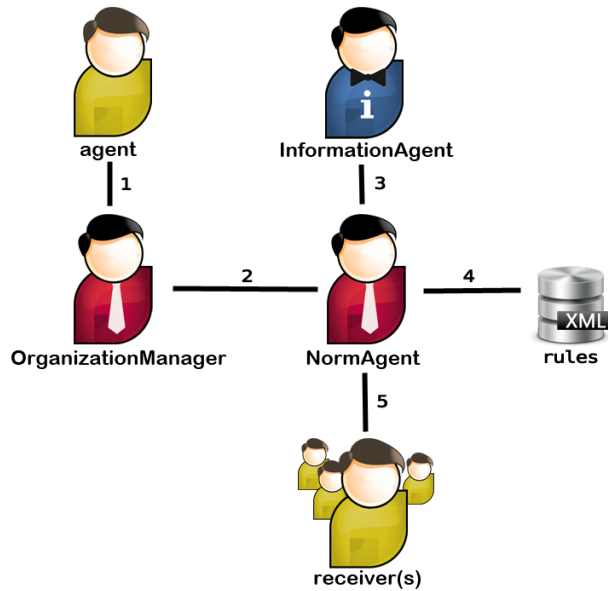
These norms have an xml structure, which is totally customizable with several criteria to consider when deciding whether to deliver the message or not. The structure is defined in a .csv file. For example, a possible structure could be:

```
1 <?xml version="1.0"?>
2 <rules>
3   <rule id="1">
4     <name>Agent A does not communicate with Agent B</name>
5     <description>Description of the norm</description>
6     <from type="Agent">A</from>
7     <to type="Agent">B</to>
8     <precondition></precondition>
9     <postcondition></postcondition>
10    <code>
11      <type>Prohibition</type>
12      <value></value>
13    </code>
14  </rule>
15 </rules>
```

**Programming Code 5.1:** A norm example

Internally the server intercepts all the messages that are generated in the platform and redirects them with the help of the `OrganizationManager`. The

format of these messages will be detailed in the following sections. The analysis performed by the NormAgent will determine if the message can reach the recipient or not. If not, it is not redirected; if so, the message is delivered in a transparent way to each of the recipients, following the scheme that can be seen in the figure 5.8.



**Figure 5.8:** NormAgent scheme

In the IRC server communication, the receipt of messages sent to organizations (or channels) or privately (between agents) is redirected to the NormAgent (if the message meets certain requirements) for further analysis in the following format:

date and time of the message -> sender -> receiver -> message

The NormAgent looks for information about the sender and receiver (which may be an organization), so it would need to contact the InformationAgent to get the information from the database. Once it has all the necessary data to analyze the message (roles, organizations, etc..) it will proceed with the analysis. The NormAgent will then send a message to himself with the same format. This message will be intercepted by the server, which is responsible for delivering the message to the recipients in a transparent manner.

An applet in Java has been implemented to facilitate the management of norms. The interface contacts with the NormAgent to obtain and forward the norm

files and structure files. The tool makes it possible to query the norms that the NormAgent is applying. Moreover, it offers the possibility of modifying or adding new norms in execution time. The NormAgent will read them when they are sent, at which time their norms will be applied.

The screenshot shows the 'MONITORING TOOL' interface. At the top, there are three tabs: 'MONITOR', 'SUBSCRIPTION MODEL', and 'RULES MANAGER'. Below these, there are five buttons: 'RELOAD RULES', 'SHOW RULES', 'ADD RULE', 'SAVE RULES', and 'SHOW STRUCTURE'. The 'ADD RULE' form is active, displaying the following fields:

Rule	
Rule -id-	
Name	This is the name of the rule as shown on the rules list
Description	This is the description to explain the rule meaning
From	Sender name (nick, channel...)
From -type-	Agent, organization, role...
To	Destination name (nick, channel, role...)
To -type-	Agent, organization, role...
Precondition	Must be true
Postcondition	Must happen
Code	
Type	permission / prohibition
Value	Block of code

At the bottom of the form is an 'Add rule' button. The footer of the interface includes the text '© 2012 BISITE Research Group - bisite@usal.es' and the 'BISITE' logo.

**Figure 5.9:** Addition of a new norm in the Monitoring tool

## 5.7

## The communication module

This section will focus on describing the communication platform and protocol. PANGEA will not pretend to present a new communication protocol; instead it will introduce the IRC protocol within multi-agent systems. This protocol is widely used in other distributed environments and has already demonstrated its reliability and robustness. What is proposed is its use within the platform, providing advantages, such as ease of implementation and reliability, given that it has been widely used in online communities with good functionality. As observed in Figure 5.2, the communication platform includes two main agents: the CommunicationAgent and the SnifferAgent. The first is in charge of checking the connections to confirm that the agents are online and see which ones have disconnected. It is also in continual communication with

the NormAgent to ensure that the agents respect the lines of communication and comply with the standards. The SnifferAgent is in charge of recording all communication, offers services so that other agents can obtain history information, and facilitates the control of information flow for programmers and users.

The IRC protocol was used to implement communication. IRC is an internet protocol for simultaneous text messaging or conferencing. This protocol is regulated by 5 standards: RFC1459 [267], RFC2810 [194], RFC2811 [195], RFC2812 [193] and RFC2813 [192]. It is designed primarily for group conversations in discussion forums and channel calls, but also allows private messaging for one on one communications, and data transfers, including file exchanges. The protocol in the OSI model is located on the application layer and uses TCP or alternatively TLS. An IRC server can connect with other IRC servers to expand the user network. Users access the IRC networks by connecting a client to a server. There have been many implementations of clients, including mIRC or XChat. The original protocol is based on flat text (although it was subsequently expanded), and used TCP port 6667 as its primary port, or other nearby ports (for example TCP ports 6660-6669, 7000). The standard structure for an IRC server network is a tree configuration. The messages are routed only through those nodes that are strictly necessary; however, the network status is sent to all servers. When a message must be sent to multiple recipients, it is sent to a multidiffusion; that is, each message is sent to a network link only once. This is a strong point in its favor compared to the no-multicast protocols such as SimpleMail Transfer Protocol (SMTP) or the Extensible Messaging and Presence Protocol (XMPP).

One of the most important features that characterize the platform is the use of the IRC protocol for communication among agents. This allows for the use of a protocol that is easy to implement, flexible and robust. The open standard protocol enables its continuous evolution. There are also IRC clients for all operating systems, including mobile devices. All messages include the following format: prefix command **command** <parameters> \r \n. The prefix may be optional in some messages, and required only for entering messages; this is one of the original commands from the IRC standard.

NICK <agent>	Specifies the name of the agent with which it wishes to initiate communication. Must contain at least 5 characters. If the Nick is currently in use, the server will return an error message.
VERSION	Returns information regarding the version of the server to which we are connected.

PONG	Term used to maintain the client connection to the server. If an agent does not periodically pong the server, it will be forced to disconnect.
QUIT	Used to abandon the architecture and force the client to disconnect.
WHOIS <agent>	Displays information about the agent, provides information about the organizations to which it belongs, if it has been identified in the system, etc.
PART <org>	Term used by agents to abandon a specific organization in execution time.
PRIVMSG <org;texto>	Used to send a message to an agent in the system, where parm1 is the receiving agent and texto the text string to be sent. If parm1 is an organization, the message is sent to all agents belonging to it.
LIST	Displays the system organizations and their description.

Table 5.2: IRC Primitives

The Figure 5.11 illustrates the message flow required for an agent to enter an organization. These messages use the PRIVMSG command followed by the parameters indicated by the arrows in the diagram.

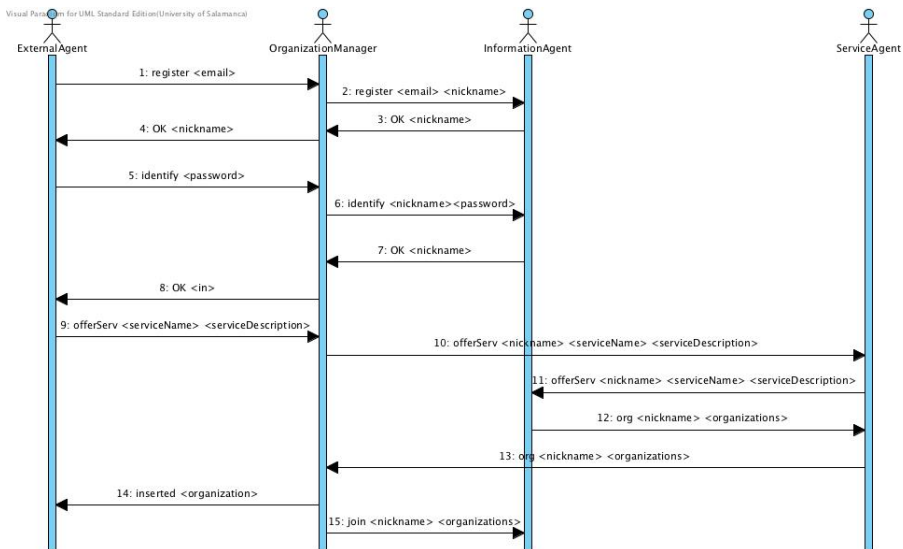


Figure 5.10: Sequence of steps for an agent to enter an organization

Another advantage in using IRC involves the ease in implementing communication. The platform's code generating tool makes it possible to easily create an outline of an agent, with the communication code requiring few lines of code. The code 5.2 displays the code for an agent in C#. It is clear that the functionality of the code consists of associating different events to the `OnQueryMessage` method, intercepting when an agent receives a message or enters an organization, and effectively handling that action from the `OnQueryMessage` method. The `Connect` method specifies the host and the communication server port, which is responsible for enabling all agents to connect and communicate. The `OnRawMessage` event is responsible for intercepting all server responses.

```
1 private void connect (Object sender , EventArgs e){
2   irc.OnJoin += new JoinEventHandler(OnQueryMessage);
3   irc.OnQueryMessage += new IrcEventHandler(OnQueryMessage);
4   irc.OnRawMessage += new IrcEventHandler(OnRawMessage);
5   irc.Connect(host.Text , 6667);
6   irc.Login(agent ,Text , null);
7   irc.Listen();
8 }
```

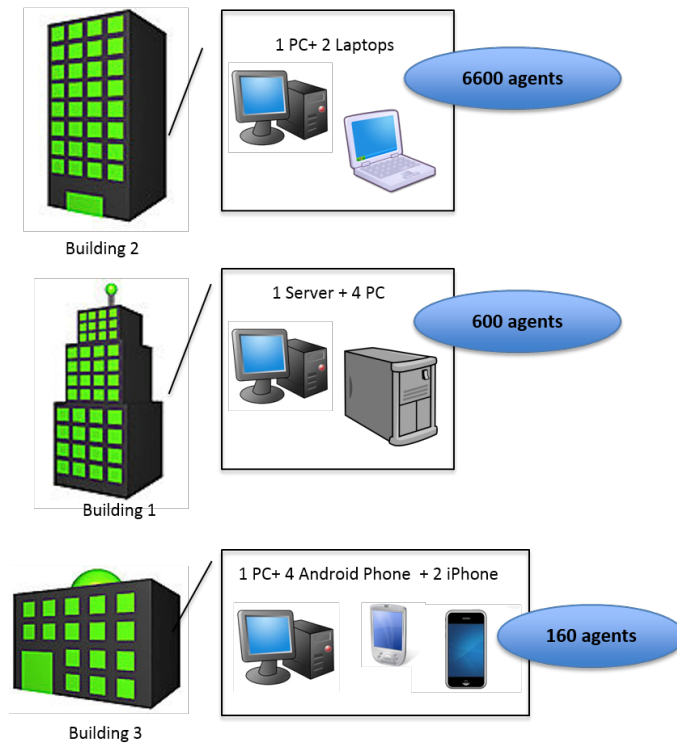
**Programming Code 5.2:** Example of the connection code for an C# agent

### 5.7.1

#### Testing the communication

In order to test communication among the agents deployed in the platform, a test case was designed, as shown in figure 5.11. The first building contains a machine with an Intel Core 2 CPU 6600 2.4GHz processor with 4Gb RAM and a 64 bit operating system connected to a network with a symmetrical speed of 2 Mb. The communication server is installed on this machine. Located in the same building but on a different network, there are 4 PCs each containing 15 agents. In another building located 4 kms away, 2 laptops, each containing 300 agents, and a PC with 6000 agents are connected. In a third building, 10 agents join the platform from a connected PC; 4 agents are connected from Android Smartphones, and 2 from iPhones. The purpose of the study is to test communication. Consequently, the 7360 agents involved do not carry out complex computational tasks; instead they simply request information from web-based news services or other basic services configured within the platform.

Two tests were carried out. The first was performed on a Monday with agents active during a 24 hour period, while the second was performed on a Friday with agents functioning during an 18 hour period, so that different network conditions could be analyzed.



**Figure 5.11:** Test Case

As previously explained, the agents can be developed in any language that uses sockets to enable communication. The table 5.3 displays the number of agents according to the language implemented, and the number of down agents during the test execution. The third column lists the number of resent messages. Since all messages should arrive to the destination agent, the platform configures time-outs to resend the messages. This column, therefore, represents the number of messages that were unable to be sent on the first attempt, and have remained in the server's time-out system waiting to be sent and receive confirmation. Resending and deleting duplicates is automatically managed by the IRC protocol.

The evolution of the number of messages that the server transmitted during the first test can be seen in Figure 5.12. For each hour, the average number of messages is shown.

Agent Type	No. of deployed agents	No. of down agents		No. of resend agents	
		Test 1	Test 2	Test 1	Test 2
.NET	2100	21	44	28	18
C#	2000	35	32	47	40
Objective C	20	0	2	6	5
Phyton	1400	10	8	26	35
Java	1800	12	13	23	23
Java Android	40	2	5	9	12

Table 5.3: Test results

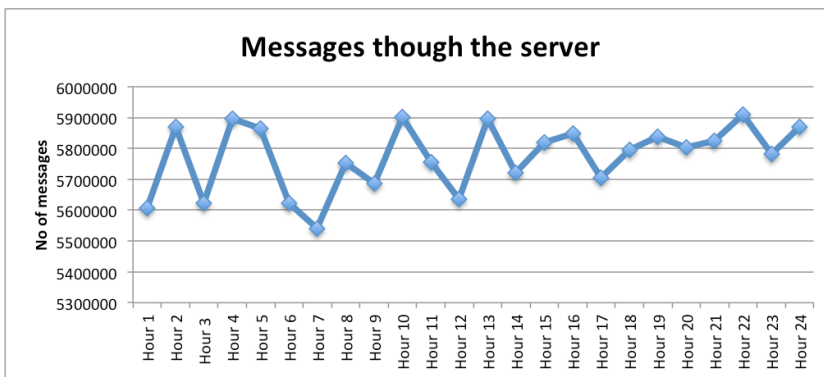


Figure 5.12: Messages though the server

PANGEA has great potential to create distributed systems as VOs. One of the greatest advantages of this system is the communication platform that, by using the IRC standard, offers a robust and widely tested system that can handle a large number of connections and ensure scalability.

## 5.8

### Testing the SnifferAgent

The platform we have developed can create a general type of organization, and includes the possibility of creating open and highly dynamic systems. In order to test the SnifferAgent, a case study was prepared to simulate a working environment. Four organizations were created to simulate four different departments within a company: accounting (composed of 4 accounting agents, one manager and 2 secretaries); quality control (composed of 2 evaluating agents and two training specialist agents); technical services (composed of 6



technical agents); and customer service (composed of 8 telephonist agents). According to the role of each agent, there are specific services offered that allow them to resolve the queries they receive. In one possible case, the client agent contacts the telephonist agent, which simply receives the requests and redirects it to the agent qualified to resolve the request. The telephonist agent extracts the key words from the message sent by the client and contacts the Services Agent to determine which agent can address the required service. If the message contains the keyword “invoice”, the query will be handled by the Accounting agent; if the keyword is “switch on” it will be handled by the Technical agent. Once the client is in contact with the appropriate agent, the agent can communicate with other agents in its organization to carry out the task.

Four 30-minute simulations were performed with 20 different types of requests randomly provided. Studying the Evaluation and Sniffer agents it was possible to observe how both the simulation and message flow unfolded. Focusing specifically on the SnifferAgent, it is possible obtain summary charts and diagrams, and specific numbers. Once the query is made, the SnifferAgent consults the database, filters the data and returns a URL that displays the desired data.

It is possible to obtain the number of each type of message that a specific agent has received. Each message includes a tag that identifies the type of message, which makes it possible to filter information, which can be seen in the figure 5.13.

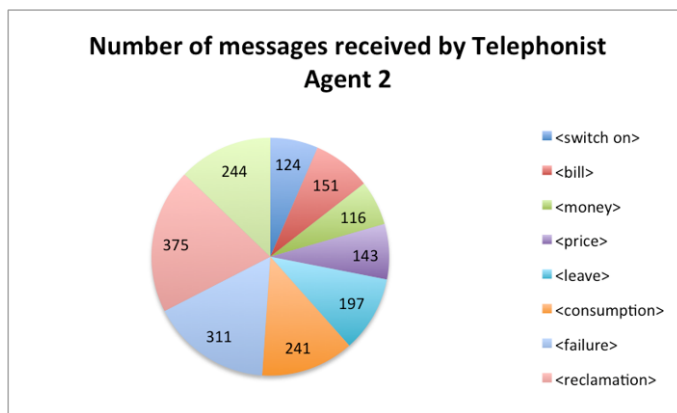
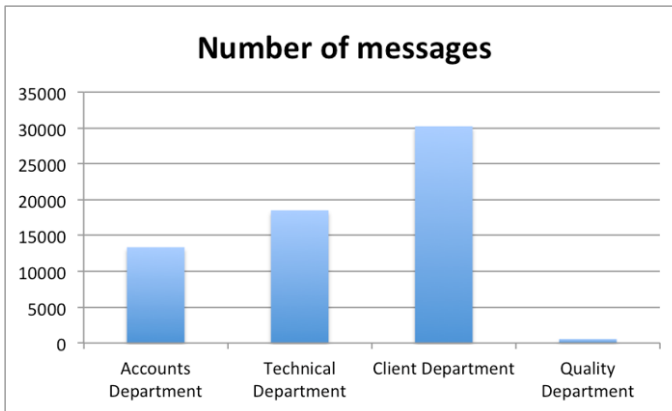


Figure 5.13: Diagram generated by the SnifferAgent (I)



**Figure 5.14:** Diagram generated by the SnifferAgent (II)

It is also possible to obtain a diagram of messages according to organization instead of agents, as shown in the figure 5.14. Using the message identifier, it is also possible to see which agents processed a given request; using the Evaluation agents we can determine the number of requests processed by each agent.

We can conclude that the architecture we are developing has great potential to create open systems, and more specifically, virtual agent organizations. This architecture includes various tools that make it easy for the end user to create, manage and control these systems. One of the greatest advantages of this system is the communication platform that, by using the IRC standard, offers a robust and widely tested system that can handle a large number of connections, and that additionally facilitates the implementation for other potential extensions. Furthermore, the use of the Communication and SnifferAgent agents offers services that can be easily invoked to study and extract message information.

## 5.9

### The subscription model

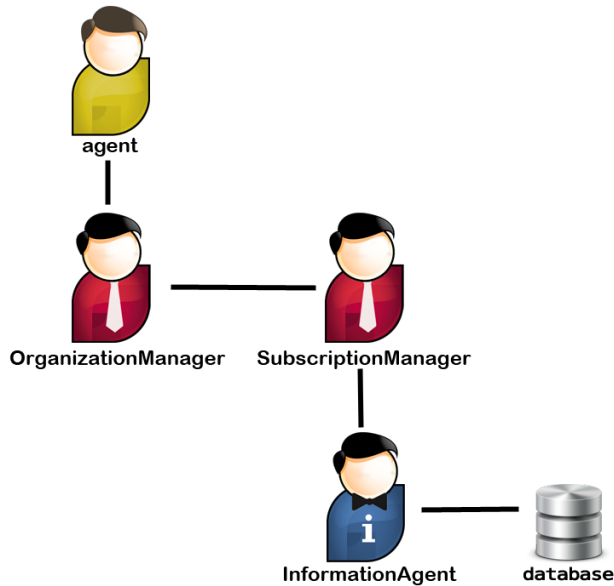
The platform allows creating organizations that are referred to, in terms of the IRC protocol, as channels. In such organizations, all agents, by default, have the same status, can send messages to all other agents in the organization, and receive everything that the other agents have sent to the organization (to which they belong). In the case of a subscription model, the concept of "pseudo-virtual organization" emerges in the platform. It is no more than an organization where only the creator of the virtual organization can send

messages, i.e., offering a subscription, while the remaining users will only be able to read the messages of the creator.

This new concept was developed using IRC protocol options as they meet the requirements perfectly without any modification or adaptation needed. The IRC protocol offers the possibility of establishing different configurations for organizations and agents that are located inside them (by default, free configuration). Among these modes are the following:

- Mode +o (agent): This mode enables an agent to be the "operator" or "moderator" in an organization. The moderator may have different tasks, but for this model only one, which will be explained later, is relevant.
- Mode +m (organization): This mode is also called "moderated", which allows only the agents configured as "moderator" (+ o) to send messages to the organization. These agents have "voice" (+ v).

The subscription model has been implemented using these modes. By default, the agents that have created an organization and the organization itself is +m. Thus, only the creator can send messages to all the member agents of the virtual organization while the remaining members cannot send anything. Recall that when an agent sends a message to an organization of any kind through IRC, it reaches all the agents within the organization (excluding themselves), which that the scheme is perfectly suitable for our needs. An agent was included to use the subscription model of PANGEA. This agent, called SubscriptionManager, is responsible only for managing subscriptions. The SubscriptionManager can communicate and is coordinated with the rest of the agents in charge of the working platform, such as the OrganizationManager (in charge of organizations) or the InformationAgent (responsible for accessing the information in the database). The working scheme is the same as in case of the ServiceAgent. In order to offer a subscription the agents must contact the OrganizationManager, which will then contact the SubscriptionManager. The SubscriptionManager performs the appropriate actions including the communication with the InformationAgent for queries to the database.



**Figure 5.15:** Communication lines among agents for the Subscription model

Among the options offered by the SubscriptionManager are:

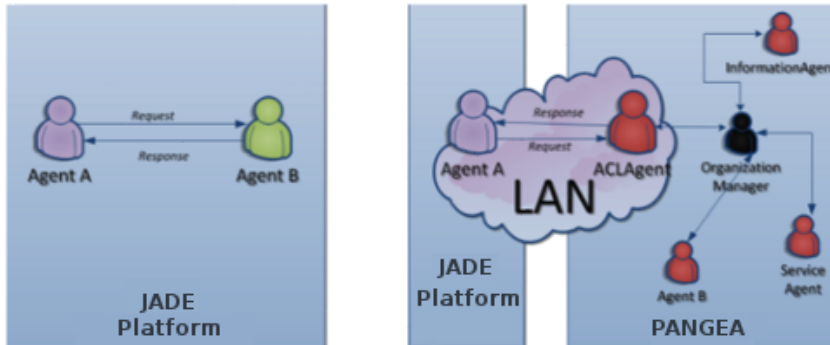
- Offersubs: enables an agent to offer a subscription. Since then, the subscription will become part of the platform and can be found by other agents. The creator will automatically enter with the mode +o whenever it logs into the platform.
- Findsubs: enables an agent to find subscriptions in the platform according to pre-established values.
- Subscribe: enables an agent to subscribe to an existing subscription channel. At that moment, when the agent logs into the platform, it will automatically enter the virtual organization associated with the subscription
- Unsubscribe: enables an agent to remove its subscription.

## 5.10

### The Gateway agent

The gateway arises from the need for external agents to communicate with PANGEA. The external agent must send a FIPA-ACL object to the IP address

of the PANGAEA MAS through the 6668 port. The object must have all the necessary fields for good communication. This includes the ontology, content, sender, etc. The ACLAgent, which is deployed in PANGAEA, is responsible for making inside-outside communication. This special agent is responsible for converting FIPA-ACL messages into PANGAEA messages. Additionally, it also carries out the requested operations and must return the results using an ACL object.



**Figure 5.16:** Communication between FIPA and PANGAEA agents

The Gateway agent initially takes care of the operations based on the request, inform, subscription, and contract-net protocols. The subsections below show these operations based on the protocols. JAVA introspection is used to differentiate the previously protocols.

### 5.10.1

#### Request protocol

This kind of operation is used in order to get a result of a known WS offered in the PANGAEA platform. In the next figure it is possible to observe the workflow that is followed in this protocol. First of all, the external agent sends the object to the ACLAgent. The ACLAgent then sends a message to OrganizationManager to get a service result. The OrganizationManager talks with the InformationAgent if there is a service with a specific name. If there is no service with the same description, the ACLAgent collects the message, and then asks the manager Organization, creating an ACLHelper agent. The manager returns the ACLHelper agent's name responsible for that service to the ACLAgent. Finally ACLAgent sends the response to the external agent. The sequence can be seen in the figure 5.17.

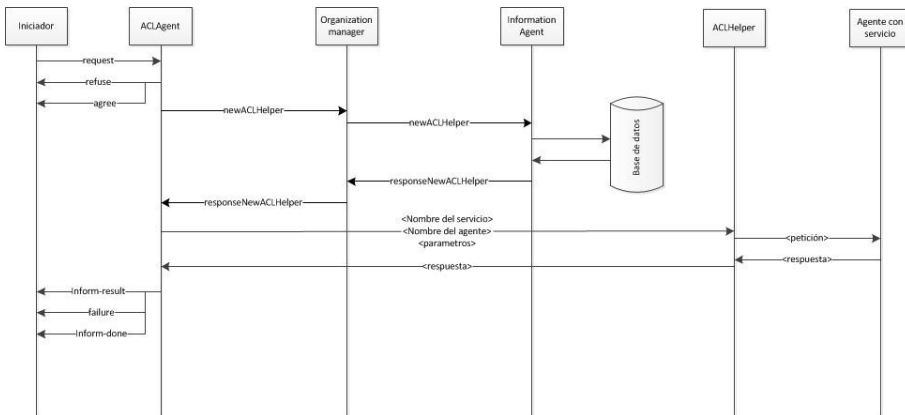


Figure 5.17: Request protocol

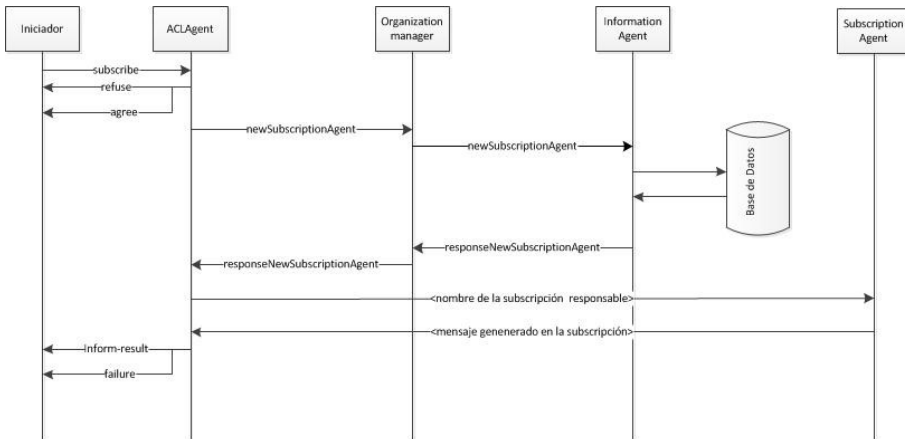


Figure 5.18: Subscription protocol

5.10.2

Subscription protocol

The External agent requests a subscription. The ACLAgent will return it all the news or modifications made in that subscription. The initiator agent sends an ACL message to the ACLAgent with the desired subscription information. The ACLAgent searches for the subscription agent directory if there is already an agent responsible for that subscription. If one does not exist, it will ask to create one and will then send it to it the subscription data. Finally, for each message arriving at the subscription, the ACLAgent sends a message to each foreign agent associated with it. The sequence can be seen in the figure 5.18.

### 5.10.3 Contract-net protocol

In this protocol, the external agent can search, find and execute a specific service. To do so, it is necessary to send multiple messages. Since it is necessary to know the name of the service which will be executed, a CFP message (Call for proposals) is sent with a brief description of the service needed in the content field. Several answers will be sent to the external agent. The agent will choose one of the answers, be executed. The sequence can be seen in the figure 5.19.

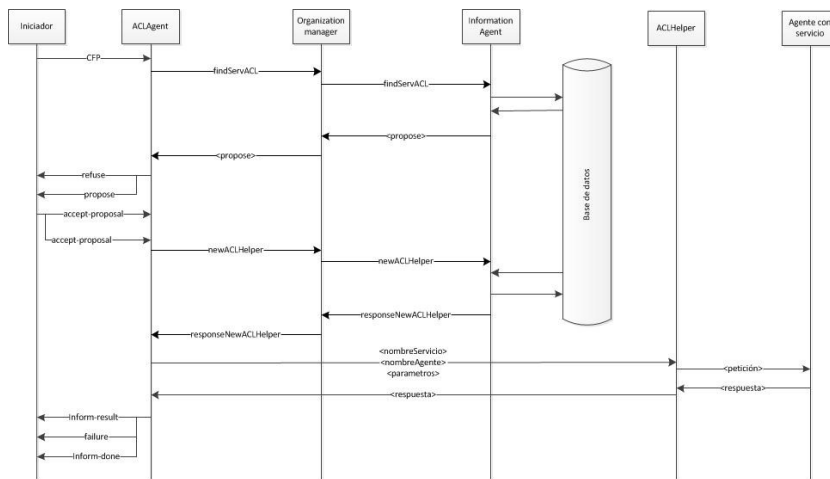


Figure 5.19: Contrat-net protocol

### 5.10.4 Inform protocol

The inform protocol is used to inform PANGEA that a new service is offered. The external agent sends a message with the content right format, and will then receive an answer (refuse, failure, inform-done). The workflow is shown in the following image. The external agent wants to register a new service in PANGEA. It sends an object to the ACLAgent with the service description. The ACLAgent sends it to the OrganizationManager and it registers the service. Finally, ACLAgent sends a response to the external agent. The sequence can be seen in the figure 5.20.

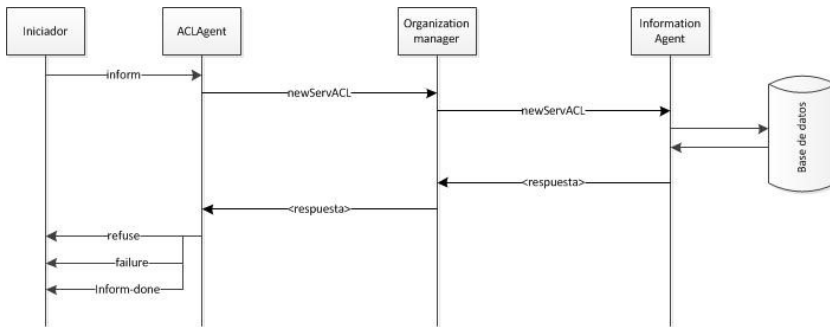


Figure 5.20: Inform protocol

## 5.11 Conclusions

PANGEA is a complete and innovative platform. We can conclude that PANGEA has great potential to create open multi-agent systems, and more specifically, VOs. One of the greatest advantages of this system is the communication platform that, by using the IRC standard, offers a robust and widely tested communication system that can handle a large number of connections and ensure scalability. Moreover, it offers compatibility with FIPA thanks to the GatewayAgent.

This protocol also offers reliability. In the tests carried out, the deployed agents were able to send and receive all messages without losses. Furthermore, the use of the Communication and Sniffer agents offers services that can be easily invoked to study and extract message information.

Another reason that justifies the scalability of the platform is the way of modelling the services as SOA architecture compliant and using WS. The platform offers an IDE, which facilitates the implementation process. It automatically offers the skeleton of an agent and the communication between agents can be implemented with few lines of code. The platform admits mobile agents and agents in any programming language, it is not necessary to learn a new language in order to use it.

This chapter was focused on explaining the implementation of PANGEA. But this is the first step because PANGEA is just the first artifact of the global platform:

1. PANGEA: VO-oriented platform.
2. PANGEA+R: collaborative-robotics-oriented platform.



3. PANGEA+RT: real-time-oriented platform.

At the end, we will have a complete platform which can deal with organizational aspects, can be used in the robotics field and support teams of robots and handle time constraints. In the next chapter, we will see the improvements made to extend PANGEA to use it in the robotics field and with a most appropriate communication protocol.

Finally, we present a summary of the studied platforms (Figure 5.21).

Characteristics	JASON	JADE	MadKit	JACK Teams	S-MOISE+	AMELI	THOMAS	JANUS	JIAC	PANGEA
Plugin or IDE for developers	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Agent models included	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
BDI model	Yes	Jadex	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
VO support	No	No	ADR model	Teams	Yes	Electronic Institutions	Yes	Yes	No	Yes
Different topologies of open MAS	No	No	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes
FIPA Compliant	-	Yes	AALAADI N model	FIPA JACK extension	-	Yes	Yes	-	Yes	Yes
Programming language	Agent Speak	Java	Java	Different languages	Java/JASO N	Java	Java	Different languages	Java/JADL++	Different languages
Communication language	JADE/SAC I	FIPA-ACL	Specific protocol	FIPA JACK extension	SACI	FIPA-ACL	JADE	JXTA protocols	Adapted FIPA-ACL	IRC
SOA architecture (services)	No	No	No	No	No	No	Yes	Yes	Yes	Yes
Web Services	No	No	No	No	No	No	No	Yes	Yes	Yes
Reorganization Services	Yes	No	No	Yes+JACK Plan Language	Yes	No	No	Yes	No	Yes
Balance workload Services	No	No	No	No	No	No	No	No	No	Yes
Planification Services	Yes	No	No	Yes+JACK Plan Language	No	No	No	Yes	No	Yes
Sniffer or Debugging tool	JADE	Yes	Yes	Yes	JADE/JAS ON	Yes	JADE	Yes	No	Yes
Graphic tool for monitoring	JADE	Yes	Yes	Yes	JADE/JAS ON	Yes	JADE	Yes	No	Yes
Different OS and mobiles devices	Yes	No	No	No	No	No	Yes	Yes	No	Yes

Figure 5.21: Comparison of the most used VOs platforms

# 6

## PANGEA+R

---

*MAS are commonly used in robotics; however, many issues arise when joining the agent technology, the robotics field and the collaboration requirement. In this chapter, we evaluate the existing platforms that are used for developing agent groups for robotics, and for multi-agent robotic systems trying to focus on the collaboration aspect. We then analyze the requisites and justify the proposal of a new platform called PANGEA+R, an evolution of the previous existing PANGEA platform, especially designed to operate in robotic environments. The purpose of PANGEA+R is to apply virtual organizations of agents to the field of robotics in order to foster collaboration and the work between distributed robotic systems and agents. PANGEA+R was built as an evolution of the PANGEA platform, including a new communication protocol called MQTT. MQTT is a lightweight broker-based publish/subscribe messaging protocol designed to be open, simple, lightweight and easy to implement. +R was designed as a repository of services frequently used in robotics with the intention of creating a collaborative environment and fosters the reutilization.*

### Contents

---

<b>6.1</b>	<b>Introduction</b>	<b>136</b>
<b>6.2</b>	<b>Platforms and middlewares for robotic systems</b>	<b>137</b>
<b>6.3</b>	<b>Multi-agent robotics systems</b>	<b>141</b>
<b>6.4</b>	<b>PANGEA+R platform</b>	<b>142</b>
6.4.1	The growing need for cooperation in Robotics	143
6.4.2	The contribution of VOs in Robotics	144
6.4.3	Main characteristics of PANGEA+R	146
<b>6.5</b>	<b>Communication module</b>	<b>148</b>
6.5.1	Message format	150
6.5.2	Command messages examples	152
6.5.3	Servers and clients	156
<b>6.6</b>	<b>Conclusions</b>	<b>158</b>

---

---

## 6.1 Introduction

---

Open systems are systems in which the structure is able to change dynamically. The components of the systems are not known a priori, change over time and may be heterogeneous. Open MAS have been gaining relevance over the last years given the importance of open, dynamic and adaptive systems as well as the internal capacities of MAS, which make them appropriate for fulfilling these needs. An open MAS must allow the participation of heterogeneous agents with different architectures and even languages [389]. A key concept when working with open MAS is the concept of organization, and the ability to dynamically re-organize the internal components of the system. The agents in a MAS based on organizational concepts, such as the VOs, work in coordination and exchange services and information; they need to be able to negotiate, collaborate and reach agreements, and can perform other more complex social actions.

Robotics is now facing a similar problem. Current trends no longer consist of a single robot dedicated to a single task, but groups of robots working in increasingly dynamic environments, with more exchange of information and more specific skills that must be shared to achieve a global goal. Most of the existing works in this line of research focus on robot groups seeking a common goal, where each of them has a different perspective of the environment and some specific skills. But this approach can go further, and it is possible to have systems with groups of heterogeneous robots that can interact with each other and share resources.

Many studies highlight the suitability of using MAS in the field of robotics [113] [278] [383]. The authors in [176] highlight the ability of implementing software which is more reusable, scalable and flexible, while maintaining parallelism, robustness and modularity. Most of the existing works focus on the concept of emergence, and pay attention to biological behaviors. Interaction standards, social abilities and cooperation are areas that have already been fully developed in the field of MAS and can be easily applied to groups of robots, mainly oriented to supervise emergent behaviors and coalition formation. Formally, a collection of two or more mobile robots working together is termed a team or society of multiple mobile robots [253]. In these cases, multi-robot approaches must show a collaborative behaviour. Thus multi-agent, multi-robot approaches study how to achieve cooperation by using the multi-agent paradigm. It should be noted that in this case the robot itself can be an agent or a collection of agents. The multi-agent paradigm is then implemented inside the robot architecture and is also used outside to control the team, society or organization [177]. Aspects

such as coordination and communication justify the importance of a VO in robot teams. PANGEA+R combines two concepts: self-organization [90] and re-organization [391]. The former refers to emergent behavior. The latter refers to organizational aspects that can be managed from a higher perspective. Thus, we make use of emergent programming [90] to supervise and control emergent behaviors that can occur in a robotic society. Collaboration between agents, whether or not they are robotic agents, must be supported at a low level by the capacity of self-organization in an emergent and natural way. Furthermore, the high-level contextual information obtained by considering organizational aspects will make it possible to study the formation and regulation of VOs, either in the form of coalitions, hierarchies or simple working groups, to carry out specific tasks within the framework of an organization regulated by norms.

PANGEA+R presents the necessary infrastructure to enable heterogeneous agents to work collaboratively, solving problems at a low level (so that the communication between agents and the implementation are facilitated) and, simultaneously, supporting control problems and/or high level supervision. These are the two main reasons that led us to develop the PANGEA+R platform, a platform that enables collaboration among heterogeneous, robotic and non-robotic agents, supporting everything from the implementation to the supervision of the life cycle and formation of groups. For this purpose, a review of existing platforms that develop robotic systems was carried out. As discussed below, there are numerous studies in this field. However, their limitations lead us to justify the creation of our PANGEA+R platform. It is specifically intended to cover the field of robotics and collaborative robotics using MAS.

## Platforms and middlewares for robotic systems

### 6.2

Robotics middleware are developed “to manage the heterogeneity of the hardware, improve software application quality, simplify software design, and reduce development costs” [99]. Every middleware contains logic, mechanisms or algorithms designed to solve specific problems. Their integration with other components can save time in development and promote reuse. A survey of Robot Development Environments (RDEs) [206] described nine open middlewares, evaluated and compared from various points of view. Another review [246] presents a short overview of several research projects in middleware for robotics, and their main objective. Mohamed et al. [247] provide a thorough study with

different criteria for evaluating networked robot middleware. Furthermore, in [256], some middleware frameworks for robotics are addressed, including their technologies within the field of multirobot systems. The main challenges for middlewares are presented in [246]. These challenges can be summarized as follows:

- Facilitate the process of developing and encouraging abstraction, even with simple interfaces.
- Provide efficient communication and simple interoperability required to facilitate interaction between modules.
- Encourage the integration and reuse of software.
- Provide the system with enough computing resources, allowing for the replication or distribution of tasks to balance the workload.
- Provide heterogeneity abstractions: any robotic system contains many heterogeneous hardware and software components.
- Support communication with other systems.
- Avoid rewritten implementations: There are many often-needed robot services that should be provided by robotic middleware, which allows the reuse of the modules offering these functionalities.
- Provide automatic service discovery with different configurations.
- Support embedded components and low-resource-devices: robots in many situations use or interact with embedded devices that may have several limitations such as limited power, small memory, limited operating system functionalities and limited connectivity.

Another study [373] mentions more necessary requirements. The most important is openness and organization, which happen to be the main characteristics of virtual organizations of agents representing open MAS. In the context of distributed software systems, openness is the property by which services provided by a system adhere to standardized protocols which formalize their syntax and semantics [368]. Other characteristics derived from the concept of openness include:

- **Portability:** The property of being able to function in different execution environments without modification. Examples include different hardware and software platforms and their respective constraints regarding available (parallel) processing, memory and network resources.

- **Flexibility:** The ease with which the structure of a software system can be changed, e.g., by adding new components or altering behaviors of system parts.
- **Interoperability** The ability to function in conjunction with other systems designed for the same domain. Many components written for one middleware already exist and can be used in other middlewares if both have sufficient interoperability qualities.
- **Scalability.** Scalable systems foster the integration of components. The ability to scale the size of a system implies sufficient efficiency in its processing.
- **Organization:** The extent to which a system can be administered, which is developed by several organizational structures with potentially overlapping or conflicting aims and guidelines.

Table 6.1 includes a brief compilation of well-known robotics-oriented middlewares. In these cases the concept of agent is not taken into account.

Miro [101] [361]	Improves the software development process for mobile robots and enables the interaction of CORBA between robots and enterprise systems using the distributed object paradigm.
Orca [233]	Enables software reuse in robotics using component-based development.
Player/Stage System [207]	Provides a development platform that supports different robotic hardware, and provides common services needed by different robotic applications.
ORiN [245]	Provides an interface for accessing and controlling robotic systems from PCs
ASEBA [231]	“It allows distributed control and efficient resources utilization of robots with multiprocessors”.
Orocos [203]	Develops a general purpose modular framework for robot and machine control.
Pyro [46]	“It provides a programming environment for easily exploring advanced topics in artificial intelligence and robotics without having to worry about the low level details of the underlying hardware”.
OpenRTMaist [8]	Provides efficient development for robotic systems by proposing a modular software structure platform and “simplifies the process of building robots by simply combining selected modules

OPROS [180]	“It establishes a component based standard software platform for the robot which enables complicated functions to be developed easily by using the standardized components in the heterogeneous communication network”.
CLARAty [262]	A reusable robotic framework to enable integration, maturation, and demonstration of advanced robotic technologies, from multiple institutions on NASA’s rover platforms in support of its technology programs (Mars and Intelligent Systems).
ROS [296]	It provides the operating system’s services such as “hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management”.
SmartSoft [316]	Implements sensorimotor systems based on communication patterns as the central means to achieve decoupling at various levels; supports model-driven software development ERSP. “Provides cutting edge technologies for vision, navigation, and system development”.
Webots [244]	“It provides a rapid prototyping environment for modeling, programming and simulating mobile robots”.
RoboFrame [281]	Covers the special needs of autonomous lightweight robots such as dynamical locomotion and stability.
Carnegie Mellon Navigation (CARMEN) Toolkit [249] [360]	“CARMEN is an open-source collection of software for mobile robot control. CARMEN is modular software designed to provide basic navigation primitives including: base and sensor control, logging, obstacle avoidance, localization, path planning, and mapping. Communications between CARMEN programs is handled using a separate package called IPC.” [360]

**Table 6.1:** Robotics-oriented middlewares and platforms

As previously mentioned, in our proposal is important to include the agent technology in the robotics systems since it can provide many advantages. For this reason, the next section presents some solutions that combine agents and robotics.



**6.3**Multi-agent robotics systems

---

In this section, we will focus on robotic systems based on MAS, formally referred to as Multi-agent Robotic System (MARS), and a platform called Yet Another Robot Platform (YARP) [186]. Initially, this platform focused on the development of humanoid robots, although its mechanism of communication and modularity have made this platform extend its development to teams of robots with multiple CPUs or even systems to control [243]. This section should also make mention of the SWARM [186] system, which is considered the precursor of robotics as a collaborative distributed system made up of a large number of autonomous robots. From this project, the term "Swarm intelligence" was coined to define the ability of multiple unintelligent entities that, working together, exhibit an intelligent collective behavior. It is a homogeneous architecture in which interaction is limited to the nearest neighbors [186]. This platform is based in the concept of emergence, but does not take into account the organizational perspective. As mention in the introduction, our proposal is focused on collaborative robotics using MAS. There are only two platforms that allow the use of agents and are focused on collaboration. The first is Cooperative Architecture for Intelligent Mobile Robots (ACROMOVI) [260]. It has an embedded agent-based architecture for the development of collaborative applications for teams of heterogeneous mobile robots. It is developed over the JADE platform in Java for collaborative robot teams. According to [260] the main purpose of ACROMOVI is to add a layer where a set of agents supervise and control access to the agents at the lower layer. In the lower layer, there are a set of components that access to the physical parts of the robots, such as the sonar, the sensors; and other special components. ACROMOVI allows the reuse of code by means of native components, thanks to agent-based distributed architecture. The main disadvantage is not allowing agents or robots programmed in C or C + +. This platform also does not take into account the concept of organization or aggrupation formation. Consequently, supervision and control is limited to individual agents since no organizational concepts are taken into account. The second platform is called ICARO-T [126]. The distinguishing factor of this framework is its use of component patterns for modelling MAS: agent organization pattern, cognitive and reactive agent patterns and resource patterns. They are described in UML, include consistent Java code, and are used in the development of a team of cooperating robots for achieving different tasks. According to the publication [148], ICARO-T offers the possibility to model MAS cooperation in two ways: (i) following AMAS theory [136], where task responsibility is assigned according to the most suitable cost evaluation to achieve the goal. This assignment is agreed upon

between team nodes by exchanging messages containing these estimates. Or (ii) a hierarchical model where a coordinator assigns the task team members estimates and then assigns the task to the most suitable agent based on the cost evaluations. This platform includes the concept of organization, but does not take into account most of the organizational aspects required to manage virtual organization. The platform ICARO is integrated with the INGENIAS Development Kit (IDK) [127] with the development of two IDK modules (code generator and code update) for the implementation of ICARO reactive agent applications. Moreover, no tools are developed to facilitate the implementation of other agent architectures, and in an intelligent and adaptative systems, other architectures such as deliberative architectures [75], are essential. In conclusion, after an exhaustive review, there is no platform that combines the needs that are required for the development of a collaborative MARS. In the next section, we present the reasons that led us to create the PANGEA+R platform.

## 6.4

### PANGEA+R platform

---

The importance of having a good platform arises from the need for a scalable system. Prototypes are initially configured with a few agents, which offer limited services, and robots (each made up by agents). However, as they are expected to have more applications, further extensions and modifications will be necessary. Moreover, it is becoming increasingly necessary to have mechanisms for collaboration (communication and service management) and facilities for reusing the robotic agents. As a final benefit, this platform offers all the advantages of the MAS and VOs, such as regulatory mechanisms of high-level organizational aspects, which include grouping entities and supervision. Currently, MAS applied to robotic environments are based on the concept of emergence and, in the majority of the cases, they perform reorganizations based on the concept of adaptation [304]. Thus, the agents adapt to the changes that occur in the environment; the strategy that the agents use to adapt to changes and reorganizes is primarily based on collaborative aspects. However, when working with systems that have been defined according to organizational requirements and social norms, it becomes necessary to design new models to monitor the behavior of the MAS and provide mechanisms for reorganization at a higher level. It would be possible to design systems with reactive capabilities and emergent behaviors at low level and organizational capabilities in the high level. With PANGEA+R it is possible to establish a gateway between the two levels by using the concept of emerging programming and the previous work done in relation to VOs with self-organizing capabilities.

To solve the problem of the Emergent Programming, the theory presented in [136] and called AMAS (Adaptative Multi-Agent Systems) was chosen. It uses mechanisms based on self-organization and collaboration to develop advanced adaptative systems. This theory highlights the importance of providing agents with regulated behaviors (in our case, thanks to the programming norms in the VOs) and fostering collaboration and interaction between agents and the environment. So, PANGEA+R provides a high level control that enables the extraction of information on emerging behaviours of robots, environmental norms, objectives and tasks, human interaction and contextual information. With this information, PANGEA+R provides a self-organizing mechanism for the agents. If any action were necessary at a low level, incentives are provided to the agents that allow them to evolve by adapting to a new organizational model.

**6.4.1**

## The growing need for cooperation in Robotics

Human and autonomous robots with different capabilities will soon need to collaborate on tasks that can be performed much more efficiently by working together. The philosophy of developing a collaborative agent system is to create a system that interconnects all the collaborative agents allowing them to function beyond the individual abilities of any one of its members [120]. Having a single robot with multiple capabilities can lead to a waste of resources, computing overhead and lower efficiency. Applying the method known as "divide and conquer", we can deduce that different robots, each one with its own settings, form a more flexible, robust and low implementing cost system [260]. Even, if tasks are too complex for a single robot, multiple robots can perform these tasks more effectively by working together [11]. Until now, most robotic teams were formed by homogeneous entities. A review of existing literature finds cases of exploration areas [277], playing soccer [117] [362], joint surveillance [35] [191], cooperative hunting [383], etc.

However, the biggest disadvantage of these devices is that the robots are very similar to each other, equipped with the same set of sensors and actuators, meaning that they offer limited benefits when performing complex tasks. In collaborative robotics, robots are now simpler, less equipped than before. As a result, less expensive equipment is needed because efficiency lies in collective intelligence and collaborative work.

Depending on the level of heterogeneity, robots in a team are jointly classified as weakly or strongly heterogeneous [198]. When the robots only differ in their capabilities, they are not identical but are still commonly considered to

form a homogeneous robot team [277]. The main difficulty comes when the robots are equipped with different sensing, perception, motion and onboard computing capabilities. In this case, for example, an application with a strongly heterogeneous robot team was developed for aerial surveillance [269], where different robot types (a blimp, an airplane and a helicopter) cooperatively monitor a rural area detecting forest fires.

With PANGEA+R, heterogeneous agents deployed on different operating systems and implemented with different languages can be integrated into groups formally regulated by the platform. In addition, another recurring issue in robotics and collaboration is reuse. With the provision of services that PANGEA+R offers, it is possible for mechanisms, algorithms, methods or other frequently used tools to form part of the service repository that provides an agent and/or group. Therefore, thanks to the supervision offered by e-agents in PANGEA+R, it is possible to form groups that benefit and foster reuse.

#### 6.4.2

#### The contribution of VOs in Robotics

Until now, different platforms or middlewares for robotics or MAS have been presented. In some cases, they are specialized in VO or collaborative robotic tasks; however, none of them combines all the advantages of a VO for collaborative robotics, especially with regard to heterogeneous teams and organizations. The use of MAS composed of mobile robots that perform tasks in a non-structured environment offers several advantages, such as:

- **Robustness:** defined as resistance to a malfunctioning of any of the robots. A simple robot failure does not keep the remaining robots from performing correctly.
- **Specialization:** creating different functional types to suit different required characteristics in the same environment for the performance of a certain global task. One example is the possibility of specializing in performing sub-tasks.
- **Implementation of functions that robots are not capable of performing individually.** From simple robots with limited decision capabilities, the collective work may reach high complexity tasks.
- **Communication between robots that leads to a better understanding of the environment.** While it is not essential for communication to exist, most multi-robot systems include this capability in varying degrees.

- Easy robot programming. We assume that the programming is done for a team composed of single robots. The power lies in working together.

As previously mentioned, collaboration must be supported at a low level by an infrastructure that enables effective communication and formation of groups working to achieve global and local goals. While there are many studies that propose models for coalition formation, such as [328] [297] [313] [306] [179], they propose methods or techniques; none of them explain at the lower level, the needs of these mechanisms, or propose a generic platform that supports such proposals. The coalitions, which are so frequently used in robotics and multi-agent systems, are considered a type of VOs. According to [306], coalitions are defined within the VO as "a temporary group of agents to achieve a particular goal. These coalitions are dissolved when they reach the goal, since the need for grouping no longer exists, or when a critical number of agents leave the group. Internally, they are usually represented as a flat structure or with a leader or representative, and externally as a single atomic entity".

In our proposal, the coalition and other group formations are done through a formalized model in [304]. We use cooperative agents, each one capable of establishing plans dynamically in order to reach its objectives. Additionally, there is a global mechanism that can optimally assign activities to the agents so that they can work in a coordinated manner. The global mechanism considers the global objective of the society, as well as its norms and roles. The model is presented in [304] in detail.

The importance of monitoring mechanisms is highlighted in the definition of [129]. Thanks to the VOs, the control and supervision can be carried out by using different mechanisms. With the concept of normativity, there is a set of norms governing the operation of a group in any organization. These norms may affect individual and collective behaviors, communications and interaction between agents and even different organizations, and access to services. Since the definition of norms is an important point that greatly facilitates overall control, a platform oriented to the collaborative work of groups cannot neglect this aspect. This possibility is offered in PANGEA+R, not only by its implementation but also through a graphical interface for easy management and control.

Another important point is the description of groups. Clarifying this, groups can be formed according to various characteristics, taking the global objective into account. Therefore, it is important to allow the input/output of entities that group dynamically, and, if necessary, to control and limit such access. Groups can also have different topologies, largely determined by the objective to achieve and the high or low need for interaction between agents. With PANGEA+R it is possible to define groups with different topologies, or to define derivative

groups or subgroups. This fact, coupled with the efficiency of norms, enables the control and monitoring of the functioning of each organization working within the established limits and according to its goals.

As mentioned, the input and output of entities must be fluid and, at the same time, be controlled. In this sense, the concept of role takes on a special relevance. Roles are the mechanism used to manage access to the system and the different organizations. As in the previous point, and thanks to the norms related to the roles, the access and interaction of entities with other entities or groups is limited to specific roles. In PANGEA+R, each agent has a role for every moment, although it is possible for the roles to change over time, which allows the agents to get in and out of different organizations but always with the appropriate permission.

Finally, there is a concept of service that is identified with the skills, abilities or behavior that each agent offers. Services are intrinsically linked to roles, because in the first instance, unless there is a norm indicating otherwise, a certain role offers a compendium of services. PANGEA+R has mechanisms for management, control and service discovery, which easily make it possible to request entities capable of performing different tasks if so required by an organization to meet its objective.

Using PANGEA+R in the field of collaborative robotics can greatly facilitate the implementation of monitoring and control processes thanks to the characteristics of VOs. As previously mentioned, the studies on collaboration through grouping, and especially the creation of coalitions among robotic entities, have solid models; however, they are not implemented on a platform that facilitates the infrastructure at a low-level and a high-level supervision. This requires more effort in verifying the collaboration models that can be avoided with the use of PANGEA+R. The next section presents some characteristics of the proposed PANGEA+R platform to cover the presented issues.

### 6.4.3

### Main characteristics of PANGEA+R

---

After identifying the mentioned shortcomings, we developed a platform called PANGEA with the necessary facilities for developing new architectures based on VOs. The PANGEA platform [393] for the development of VO and the integral management of open MAS has been used as the base platform for the design of architectures applied to different fields such as environmental intelligence, development of tools for handicapped people, and energy management. The new +R middleware is proposed to extend this platform for its use in robotics, highlighting the software reusability and allowing the programmer

to integrate native software components (computer vision libraries, navigation modules, location, etc.). PANGEA [144] [392] models agent capabilities as Web services and has the needed functionality to manage them. By adding the +R middleware, the functionality of these services is extended to agents with robotic tasks such as algorithms, drivers and mechanisms often used in robotic systems, thus avoiding continuous reimplementations. Furthermore, the different collaborative entities can offer services and acquire easily developed complex robotic systems based on the request and composition of existing services within the +R middleware. Figure 6.1 shows the mentioned layer +R added to PANGEA.

Thus, as with MAS, robotics needs social entities capable of cooperation and able to operate in highly dynamic and unpredictable environments. To overcome these obstacles, the entities must have large amounts of resources at their disposal. These resources can be managed similarly to a collaborative repository. The purpose of the +R middleware is to extend this platform to include its use in robotics, highlighting software reusability and allowing the programmer to integrate native software components (computer vision libraries, navigation modules, location, etc.). The relevant characteristics of the PANGEA platform and the performance of the +R middleware are shown below.

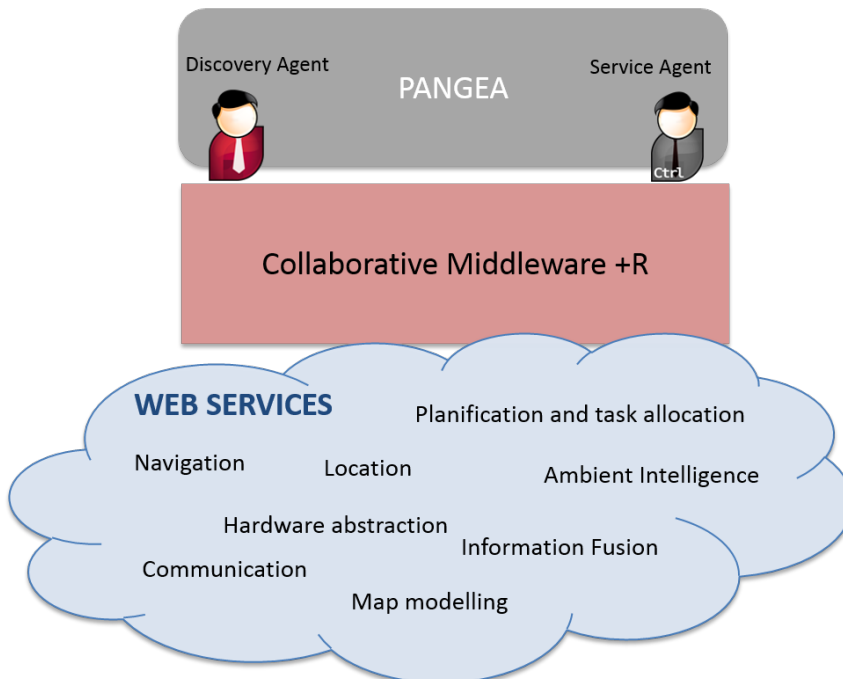


Figure 6.1: Software tool of GECKO

The +R middleware was designed as a repository of specific services oriented to the development of robotics, which facilitates the development of collaborative robot teams. The functionality of managing the life cycles of agents and organizations with their rules and roles does not differ from previous versions of PANGEA, available at [144]. However, changes were made for the entry of new entities (as they must provide specific features). Additionally, the process was defined for the management of services.

## 6.5 Communication module

The PANGEA+R communication module is improved using the Message Queue Telemetry Transport (MQTT) [170] instead of the IRC protocol. MQTT is a lightweight broker-based publish/subscribe messaging protocol designed to be open, simple, lightweight and easy to implement. These characteristics make it ideal for use in constrained environments. It was originally developed by IBM and its partners from the industrial sector (Dr Andy Stanford-Clark of IBM, and Arlen Nipper of Eurotech, in 1999). Since, the protocol has been opened to open source community and has significant growth in popularity.

There are two main specifications for MQTT [170]:

- MQTT v3.1 specification – the primary MQTT specification is available, for royalty-free implementation. This protocol enables a publish/subscribe messaging model in an extremely lightweight way. It is useful for connections with remote locations where a small code footprint is required and/or network bandwidth is at a premium.
- MQTT-SN v1.2 specification – MQTT for Sensor Networks is aimed at embedded devices on non-TCP/IP networks, such as Zigbee. MQTT-SN is a publish/subscribe messaging protocol for Wireless Sensor Networks (WSN), with the aim of extending the MQTT protocol beyond the reach of TCP/IP infrastructure for Sensor and Actuator solutions.

Currently, only the first specification is used since all our agents run in an infrastructure with a TCP/IP connection. But, in further versions we will extend it to allow its use in embedded reactive architectures.

MQTT is a publish/subscribe, extremely simple and lightweight messaging protocol, designed for constrained devices and low-bandwidth, high-latency or unreliable networks. The design principles are to minimise network bandwidth and device resource requirements whilst also attempting to ensure reliability



and some degree of assurance of delivery. MQTT defines three levels of Quality of Service (QoS):

- Level 0: The broker/client will deliver the message once, with no confirmation.
- Level 1: The broker/client will deliver the message at least once, with confirmation required.
- Level 2: The broker/client will deliver the message exactly once by using a four step agreement.

The QoS defines how hard the broker/client will try to ensure that a message is received. Messages may be sent at any QoS level, and clients may attempt to subscribe to topics at any QoS level. This means that the client chooses the maximum QoS it will receive. For example, if a message is published at QoS 2 and a client is subscribed with QoS 0, the message will be delivered to that client with QoS 0. If a second client is also subscribed to the same topic, but with QoS 2, then it will receive the same message but with QoS 2. For a second example, if a client is subscribed with QoS 2 and a message is published on QoS 0, the client will receive it on QoS 0. Higher levels of QoS are more reliable, but involve higher latency and have higher bandwidth requirements.

Other important characteristics of the protocol are [172]:

- A small transport overhead (the fixed-length header is just 2 bytes), and protocol exchanges minimised to reduce network traffic
- A messaging transport that is agnostic to the content of the payload: The protocol does not require that the content of messages be in any particular format.
- The publish/subscribe message pattern to provide one-to-many message distribution and decoupling of applications. MQTT uses a publish/subscribe messaging pattern that has loose coupling. Clients do not need to be aware of the existence of other devices. They just need to care about the content to be delivered or received
- Open and royalty-free for easy adoption and adaptation for the wide variety of devices, platforms, and operating systems that are used at the edge of a network.
- MQTT is highly scalable, it is possible to create systems that involve hundreds or even thousands of remote devices.

### 6.5.1 Message format

MQTT allows different messages formats, in this section we will present the message format and the adaptations that we adopt to include the protocol in our platform.

The message header for each MQTT command message contains a fixed header. The figure 11.4 shows the format.

bit	7 6 5 4	3	2 1	0
byte 1	Message Type	DUP flag	QoS level	Retain
byte 2...N	Remaining Length			
byte N+1...M	Variable Header: depends on the Message Type			
byte M+1...K	Payload: depends on the Message Type			

**Figure 6.2:** MQTT Message Format

The fields are:

- Message Type: Represented as a 4-bit unsigned value (Table 11.1).
- Dup: This flag is set when the client or server attempts to re-deliver a PUBLISH, PUBREL, SUBSCRIBE or UNSUBSCRIBE message. This applies to messages where the value of QoS is greater than zero, and an acknowledgment is required. When the DUP bit is set, the variable header includes a Message ID.
- QoS: indicates the level of assurance for delivery of a PUBLISH message. The values were previously defined in section.
- Retain: This flag is only used on PUBLISH messages. When a client sends a PUBLISH to a server, if the Retain flag is set to 1, the server should hold on to the message after it has been delivered to the current subscribers. When a new subscription is established on a topic, the last retained message on that topic should be sent to the subscriber with the Retain flag set. If there is no retained message, nothing is sent.
- Remaining Length: Represents the number of bytes remaining within the current message, including data in the variable header and the payload. It must be at least one byte.

The numbers used in the field "Message Type" of the figure 11.4 are shown in the following table 11.1.

MNEMONIC	NUMBER	DESCRIPTION
CONNECT	1	Client request to connect to the server.
CONNACK	2	Connect Acknowledgment
PUBLISH	3	Publish message
PUBACK	4	Publish Acknowledgment
PUBREC	5	Publish received (part 1)
PUBREL	6	Publish release (part 2)
PUBCOMP	7	Publish complete (part 3)
SUBSCRIBE	8	Client Subscribe request
SUSACK	9	Subscribe Acknowledgment
UNSUBSCRIBE	10	Client Unsubscribe request
UNSUBACK	11	Unsubscribe Acknowledgment
PINGREQ	12	PING request
PINGRESP	13	PING response
DISCONNECT	14	Client is disconnecting

**Table 6.2:** MQTT message types

The following types of MQTT command message have a payload:

- **CONNECT:** The payload contains one or more UTF-8 encoded strings. They specify a unique identifier for the client, a Will topic and message and the User Name and Password to use. All but the first are optional and their presence is determined based on flags in the variable header.
- **SUBSCRIBE:** The payload contains a list of topic names to which the client can subscribe, and the QoS level. These strings are UTF-encoded.
- **SUBACK:** The payload contains a list of granted QoS levels. These are the QoS levels at which the administrators for the server have permitted the client to subscribe to a particular Topic Name. Granted QoS levels are listed in the same order as the topic names in the corresponding SUBSCRIBE message.
- **PUBLISH:** The payload part of a PUBLISH message contains application-specific data only. No assumptions are made about the nature or content of the data.

The Message Identifier (Message ID) field is only present in messages where the QoS bits in the fixed header indicate QoS levels 1 or 2 and is included in the variable header. The Message ID is a 16-bit unsigned integer that must be unique amongst the set of in process messages in a particular direction of communication. It typically increases by exactly one from one message to the next, but is not required to do so. A client will maintain its own list of Message

IDs separate to the Message IDs used by the server it is connected to. It is possible for a client to send a PUBLISH with Message ID 1 at the same time as receiving a PUBLISH with Message ID 1.

UTF-8 is an efficient encoding of Unicode character-strings that optimizes the encoding of ASCII characters in support of text-based communications. In MQTT, strings are prefixed with two bytes to denote the length, the string length is the number of bytes of encoded string characters, not the number of characters.

---

### 6.5.2 Command messages examples

---

Some types of MQTT command messages also contain a variable header component. It resides between the fixed header and the payload. The message identifier is present in the variable header of the following MQTT messages: PUBLISH, PUBACK, PUBREC, PUBREL, PUBCOMP, SUBSCRIBE, SUBACK, UNSUBSCRIBE, UNSUBACK.

It is impossible to specify all of them in this document, so, we will include the three most used. All of them contains three fields in the following order: fixed header, variable header and payload.

---

#### 6.5.2.1 CONNECT command

---

It is used when a client requests a connection to a server. When a TCP/IP socket connection is established from a client to a server, a protocol level session must be created using a CONNECT flow. In the Figure 6.3 the format of this type of message is presented.

The flags involved in the connection are:

- Clean session flag: if not set (0), then the server must store the subscriptions of the client after it disconnects. If set (1), then the server must discard any previously maintained information about the client and treat the connection as "clean". The server must also discard any state when the client disconnects.
- Keep Alive timer: The Keep Alive timer, measured in seconds, defines the maximum time interval between messages received from a client. It enables the server to detect that the network connection to a client has dropped, without having to wait for the long TCP/IP timeout. The

client has a responsibility to send a message within each Keep Alive time period. In the absence of a data-related message during the time period, the client sends a PINGREQ message, which the server acknowledges with a PINGRESP message.

		Description	7	6	5	4	3	2	1	0	
<b>Fixed Header</b>	byte 1	Message Type, DUP flag, QoS level, RETAIN	0	0	0	1	x	x	x	x	
	byte 2...N	The length of the variable header (12 bytes) plus the length of the payload	Remaining Length								
<b>Variable Header</b>	<b>Protocol Name</b>	byte 1	Length MSB (0)	0	0	0	0	0	0	0	
		byte 2	Length MSB (6)	0	0	0	0	0	1	1	0
		byte 3	'M'	0	1	0	0	1	1	0	1
		byte 4	'Q'	0	1	0	1	0	0	0	1
		byte 5	'I'	0	1	0	0	1	0	0	1
		byte 6	's'	0	1	1	1	0	0	1	1
		byte 7	'd'	0	1	1	0	0	1	0	0
		byte 8	'p'	0	1	1	1	0	0	0	0
	<b>Protocol Version</b>	byte 9	Version 3	0	0	0	0	0	0	1	1
	<b>Connect Flags</b>	byte 10	User name (1) Password flag (1) Will RETAIN (0) Will QoS (01) Will flag(1) Clean Session (1)	1	1	0	0	1	1	1	x
	<b>Keep Alive Timer</b>	byte 11	Keep Alive MSB (0)	0	0	0	0	0	0	0	0
		byte 12	Keep Alive MSB (10)	0	0	0	0	1	0	1	0
	<b>Payload</b>	byte 1	Client Identifier	String Length MSB							
		byte 2		String Length LSB							
byte 3...N		UTF Encoded Character Data									
byte 1		Will Topic	String Length MSB								
byte 2			String Length LSB								
byte 3...N			UTF Encoded Character Data								
byte 1		Will Message	String Length MSB								
byte 2			String Length LSB								
byte 3...N			UTF Encoded Character Data								
byte 1		User Name	String Length MSB								
byte 2			String Length LSB								
byte 3...N			UTF Encoded Character Data								
byte 1		Password	String Length MSB								
byte 2			String Length LSB								
byte 3...N	UTF Encoded Character Data										

Figure 6.3: CONNECT Message Format

- Will flag: it defines that a message is published on behalf of the client by the server when either an I/O error is encountered by the server during

communication with the client, or the client fails to communicate within the Keep Alive timer schedule. Sending a Will message is not triggered by the server receiving a DISCONNECT message from the client. If the Will flag is set, the Will QoS and Will Retain fields must be present in the Connect flags byte, and the Will Topic and Will Message fields must be present in the payload.

- Will QoS: A connecting client specifies the QoS level in the Will QoS field for a Will message that is sent in the event that the client is disconnected involuntarily. The Will message is defined in the payload of a CONNECT message. If the Will flag is set, the Will QoS field is mandatory, otherwise its value is disregarded.
- Will Retain flag: The Will Retain flag indicates whether the server should retain the Will message which is published by the server on behalf of the client in the event that the client is disconnected unexpectedly. The Will Retain flag is mandatory if the Will flag is set, otherwise, it is disregarded.
- User name and password flags: A connecting client can specify a user name and a password, and setting the flag bits signifies that a User Name, and optionally a password, are included in the payload of a CONNECT message. If the User Name flag is set, the User Name field is mandatory, otherwise its value is disregarded. If the Password flag is set, the Password field is mandatory, otherwise its value is disregarded.

#### **6.5.2.2** SUBSCRIBE command

---

The SUBSCRIBE message allows a client to register an interest in one or more topic names with the server. Messages published to these topics are delivered from the server to the client as PUBLISH messages. The SUBSCRIBE message also specifies the QoS level at which the subscriber wants to receive published messages. In the fixed header, SUBSCRIBE messages use QoS level 1 to acknowledge multiple subscription requests. In the variable header another QoS field is included, this field is equivalent to saying "I would like to receive messages on this topic at the QoS at which they are published". This means a publisher is responsible for determining the maximum QoS a message can be delivered at, but a subscriber is able to downgrade the QoS to one more suitable for its usage. The QoS of a message is never upgraded. The Figure 6.4 shows the format of the message.

		Description	7	6	5	4	3	2	1	0
<b>Fixed Header</b>	byte 1	Message Type, DUP flag, QoS level, RETAIN	1	0	1	1	0	0	1	x
	byte 2...N	The length of the variable header (12 bytes) plus the length of the payload	Remaining Length							
<b>Variable Header</b>	byte 1	Message Identifier	String Length MSB							
	byte 2		String Length LSB							
<b>Payload</b>	byte 1	Topic Name	String Length MSB							
	byte 2		String Length LSB							
	byte 3...N		UTF Encoded Character Data							
	byte 1	Requested QoS	x	x	x	x	x	x	0	1
	byte 1	Topic Name	String Length MSB							
	byte 2		String Length LSB							
	byte 3...N		UTF Encoded Character Data							
byte 1	Requested QoS	x	x	x	x	x	x	1	0	

Figure 6.4: SUBSCRIBE Message Format

		Description	7	6	5	4	3	2	1	0
<b>Fixed Header</b>	byte 1	Message Type, DUP flag, QoS level, RETAIN	0	0	1	1	0	0	1	0
	byte 2...N	The length of the variable header (12 bytes) plus the length of the payload	Remaining Length							
<b>Variable Header</b>	byte 1	Topic Name	String Length MSB							
	byte 2		String Length LSB							
	byte 3...N		UTF Encoded Character Data							
	byte 1	Message Identifier	String Length MSB							
byte 2	String Length LSB									
<b>Payload</b>	byte 1	Data for publishing	String Length MSB							
	byte 2		String Length LSB							
	byte 3...N		UTF Encoded Character Data							

Figure 6.5: PUBLISH Message Format

**6.5.2.3**

## PUBLISH command

A PUBLISH message is sent by a client to a server for distribution to interested subscribers. Each PUBLISH message is associated with a topic name (also known as the Subject or Channel). The topic name (a UTF-encoded string) is the key that identifies the information channel to which payload data is published. Subscribers use the key to identify the information channels on which they want to receive published information. This is a hierarchical name space that defines a taxonomy of information sources for which subscribers can

register an interest. A message that is published to a specific topic name is delivered to connected subscribers for that topic. If a client subscribes to one or more topics, any message published to those topics are sent by the server to the client as a PUBLISH message. In the Figure 6.5 the format of the message is presented.

### 6.5.3 Servers and clients

Another important feature is that different clients and servers exist in different languages (<http://mqtt.org/software>), such as Java, C, Arduino, Javascript, etc.

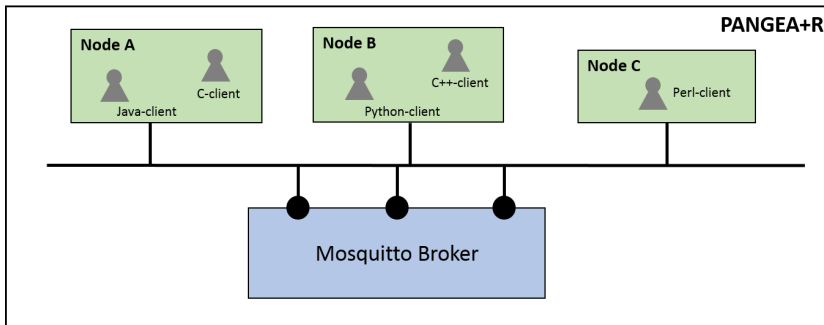


Figure 6.6: Communication with the MQTT protocol

### MQTT brokers

An MQTT broker is a server that implements the MQTT protocol. It mediates communication between MQTT client applications, such as those running in remote sensors and other devices, and the enterprise integration layer. Really Small Message Broker (RSMB) is a small, freely available C implementation of an MQTT broker. It also is an advanced MQTT V3 client application. Developers usually use it as a hub to store and forward messages from MQTT clients to a backend MQTT broker. We use Mosquitto (<http://mosquitto.org/>). It is a small, no-cost, open source implementation of an MQTT broker that supports the MQTT V3.1 protocol. Mosquitto replicates the functionality of RSMB. In the figure 6.6 our architecture is shown.

Mosquitto supports 3 types of connections:

- Port 1883: it is the standar port for communications without encrypting.
- Port 8883: with SSL/TLS encrypting.



- Port 8884: with SSL/TLS encrypting and requires certificate.

It is easily installable both in any Linux distribution including the Raspbian. In the part V of this dissertation, more information can be found.

### MQTT clients

An MQTT client (also called a client application) collects information from a telemetry device, connects to a messaging server, and uses a topic string to publish the information in a way that allows other clients or applications to retrieve it. An MQTT client also can subscribe to topics, receive publications associated with those topics, and issue commands to control the telemetry device. Client libraries can simplify the process of writing MQTT client applications. The Eclipse tools facilitate designing and developing connectivity solutions between devices and applications, thereby enabling and encouraging more innovative M2M integration. Users can write their own API to interface with the MQTT protocol in their preferred programming language on their preferred platform.

We use the Eclipse Paho project (<http://www.eclipse.org/paho/>) that provides scalable, open-source messaging models for machine-to-machine (M2M) communication, starting with C- and Java-based client implementations of MQTT.

```

1 public void PublishAndSuscribeBlocking() throws Exception {
2     MQTT mqtt = new MQTT();
3     mqtt.setHost("tcp://mosquito.fis.usal.es:1883");
4     BlockingConnection connection = mqtt.blockingConnection()
5         ;
6     connection.connect();
7     Topic[] topics = {new Topic(utf8("PresentationTopic"),
8         QoS.AT_LEAST_ONCE)};
9     byte[] qoses = connection.subscribe(topics);
10    connection.publish("PresentationTopic", "Hello".getBytes
11        (), QoS.AT_LEAST_ONCE, false);
12    Message message = connection.receive();
13    Assert.assertEquals("Hello", new String(message.
14        getPayload()));
15    message.ack();
16    connection.disconnect();
17 }

```

**Programming Code 6.1:** Example connect and subscribe in Java

```

1 broker = \"tcp://mosquito.fis.usal.es:1883\"
2 port = 1883
3 mypid = os.getpid()
4 client_uniq = "pubclient_"+str(mypid)
5 mqttc = paho.Client(client_uniq, False)
6 mqttc.connect(broker, port, 60)

```

```

7   while mqttc.loop() == 0:
8       msg = "test message "+time.ctime()
9       mqttc.publish("timesample", msg, 0, True)
10      print "message published"
11      time.sleep(1.5)
12      pass

```

**Programming Code 6.2:** Example connect and subscribe in Python

PANGEA+R and the MQTT enables to connect an agent in many other languages, but without this two are the easiest.

The agents of the platform are clearly reduced in comparison with the agents that compose the PANGEA platform, the reason is the introduction of the MQTT communication protocol. The allowed us to remove the CommunicationAgent, the SnifferAgent and the SubscribeAgent since all their functions are automatically supported by the protocol and the Mosquitto broker. The GatewayAgent is also no more needed because the link between MQTT and FIPA-ACL is not beneficial and can cause dangerous delays.

## 6.6

## Conclusions

Many platforms have been presented in this work. As a summary, a table reflecting the main characteristics required since the robotics point of view is presented below (Figure 6.7). The PANGEA+R platform is included to demonstrate that the inclusion of the robotics characteristics specific to VOs can lead to many improvements, allowing for one single platform which includes all the main characteristics. Looking at these tables, we propose a new middleware, which enable a connection with the PANGEA platform, creating as its final product the PANGEA+R platform together with the included new message protocol.

The main question that arises is: why another platform? One reason is the capacity of improvement that VO can provide to robotic middlewares where the importance of a solid distributed infrastructure with supervision capacities is the main feature. Another reason is the need to foster collaboration among heterogeneous robotic agents, thus facilitating the implementation and the reutilization of frequently used methods, hardware control or algorithms. The integration of our PANGEA platform with the collaborative +R middleware forms an integral system. PANGEA+R enables the creation of complex teams of cooperative agents and robotic agents. The power of multiple heterogeneous robots working on the same task are still being studied and growing. This kind

of platform can allow the robotics community to share software infrastructures based on common requirements and objectives, which is clearly an important goal to reach in order to avoid the duplication of efforts. Finally, we present a summary of the platforms and middlewares (Figure 6.7).

At the end, we can conclude that we have a complete platform that can manage VOs with different topologies and taking into account organizational aspects. Any kind of agent is supported including robotics and embedded agents thanks to the lightweight MQTT protocol. In the next chapter, the last milestone is presented, to include real-time constraints with the proposed scheduling and task allocation model.

Platform	Distributed Architecture	Introspection and management tools	Hardware drivers and interfaces	Robotics algorithms	Simulation	Real-time oriented	MAS/VO oriented	Languages	Communication standard	Discovery services
CARMEN	Yes	No	Yes	Yes	Yes	No	No	C, Java	No robotics or MAS standard (IPC Communicating module)	No
CLARATY	Yes	Yes	Yes	Yes	No	No	No	C	No robotics or MAS standard	No
MIRO	Yes	No	Yes	No	No	No	No	C	CORBA	No
ORCA	Yes	No	Yes	No	No	No	No	C++	No robotics or MAS standard	No
Player / Stage System	No	No	Yes	Yes	Yes	No	No	C, C++, Java, Python	No robotics or MAS standard	No
ASEBA	No	Yes	No	No	No	No	No	AESL	No robotics or MAS standard	No
Orocos	Yes	No	Yes	Yes	No	Yes	No	C++, Python, Simulink, Orocos, Scripting Language	No robotics or MAS standard	No
OpenRTMaist	Yes	Yes	Yes	Yes	No	No	No	C++, Java, Python	RTC	No
ROS	Yes	Yes	Yes	Yes	Yes	No	No	C, C++, Python, Java, Lisp, Octave	No robotics or MAS standard	No
Webots	No	Yes	Yes	No	Yes	No	Yes (MAS)	C, C++, Python, Java	No robotics or MAS standard	No
RoboFrame							No		No robotics or MAS standard	No
YARP	Yes	Yes	Yes	No	Yes	No	Yes (MAS)	C, C++, Java, Python, Octave	No robotics or MAS standard	No
ACROMOVI	Yes	Yes (JADE)	No	No	No	No	Yes (MAS)	Java	FIPA	No
ICARO-T	Yes	Yes	No	Yes	No	No	Yes (VO)	Java	FIPA	Yes
PANGEA+R	Yes	Yes	No	Yes	Yes	Yes (PANGEA+RT)	Yes (VO)	All allowing sockets	IRC	Yes

Figure 6.7: Comparison of the most used VOs and Robotics platforms

# 7

## PANGEA+RT

---

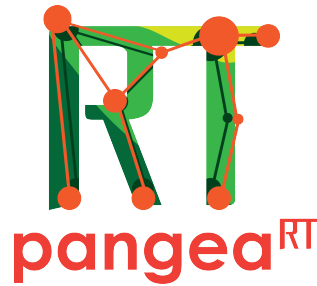
*In this chapter we present the last milestone of the desired platform, that we name as PANGEA+RT. We have included the WCET estimation, scheduling and task allocation model defined in the part III of this dissertation. The inclusion of this model lead us to other modifications and extensions. We have removed some agents that are not necessary and cannot be temporal bounded but we have included some requirements of the real-time systems. The agent that plays the role PlannerAgent is the most important change since it is responsible for the scheduling and task allocation in the real-time system. In this chapter we deal with real-time execution environments according to the Real-Time Specification for JAVA and with the virtual machines also for real-time and Java. Moreover, we include a mechanism to annotate the behaviour of the agent in order to perform the WCET analysis, moreover, this mechanism follows the current standards to make our proposal more accessible to the users. Finally, we show some examples of a basic agent implementation. With all these characteristics we have developed an innovative platform since there is no other platform suitable for VOs with real-time constraints.*

### Contents

---

<b>7.1</b>	<b>Introduction</b>	<b>162</b>
7.1.1	Problems of Java in real-time environments	163
<b>7.2</b>	<b>Real-time specification for Java</b>	<b>166</b>
<b>7.3</b>	<b>Real-time Java virtual machines</b>	<b>169</b>
<b>7.4</b>	<b>Annotation for bounded loops</b>	<b>171</b>
<b>7.5</b>	<b>Agents of the platform</b>	<b>176</b>
<b>7.6</b>	<b>Modification of the classes</b>	<b>178</b>
<b>7.7</b>	<b>Conclusions</b>	<b>181</b>

---



## 7.1

## Introduction

In recent years, most of the researches working on MAS have been using the Java programming language for the implementation of agents. Java is an object-oriented programming language and it is highly suitable for its use by researchers in the development of agents. This is due to the similarities that can be established between agents and objects. In the object-oriented paradigm, a system is seen as a set of objects that communicate one with another to perform internal operations. While, the agent-oriented paradigm envisions a set of agents whose internal operations are based on beliefs, capabilities and actions, and that communicate with other agents using messages based on communication theory.

In their book, Jennings and Wooldridge [183] highlighted three main differences between agents and objects:

- The first is the degree in which agents and objects are autonomous. In this case, we do not think in agents as the method callers between them instead they ask for the realization of actions. In the object-oriented case, the decision falls on the object that invokes the method. For agents, the decision lies with the agent receiving the request.
- The second distinction is that an agent has an autonomous and flexible behavior (reactive, proactive, social).
- The last major difference is that it is considered that each agent has its own flow control. In the object standard model, there is a single flow of control for the entire system.

As Navarro mentioned in its study [258] one of the major advantages of Java is a secure programming language, ie, that the results of every execution are represented by the program source code. This does not indicate that there can be errors in programming, but these are not caused by uncontrollable execution

errors, eg, due to poor allocation of memory, out of memory range (overflow), etc. In other programming languages both memory assignment and release must be done manually. This can cause undetectable errors at compilation time and uncontrollable errors at runtime, being difficult to debug besides the need for the programmer of being aware of what memory area must reserve and release. Java solves this problem by automatically allocating memory when the object is created. In turn, all the memory allocated to objects that are not referenced at the time of the garbage collector (GC) runs will be automatically released by it.

To complement the design of PANGEA+RT that will allow us to take into account real-time constraints and the model developed, in chapter 4 is necessary to consider the following issues:

- To introduce the real-time specification for Java, and evaluate possible real-time virtual machines for Java.
- To enable the class annotation of the classes that represents the agents' behaviour, which must go through the WCET estimation model.
- To extend the model created for PANGEA+R, which mainly implies providing to the main agents communication capabilities inside the new communication mechanism and temporarily limit their behavior.

### 7.1.1

#### Problems of Java in real-time environments

Because of the popularity of Java as a development language, there has been a major effort to address the shortcomings of Java as a real-time development language and to introduce new features into the language to support real-time development. These led, over a number of years, to the development of a real-time specification for Java. This includes both changes to the language and to the Java Virtual Machine (JVM), the run-time system for Java.

The main problems studied in the IBM Toronto Lab and published in [347] are:

- Thread management: Standard Java provides no guarantees for thread scheduling or thread priorities. An application that must respond to events in a well-defined time has no way to ensure that another low-priority thread won't get scheduled in front of a high-priority thread. To compensate, a programmer would need to partition an application into a set of applications that the operating system can then run at different

priorities. This partitioning would increase the overhead of these events and make communication between the events far more challenging.

- **Class loading:** A Java-conformant JVM must delay loading a class until it's first referenced by a program. Loading a class can take a variable amount of time depending on the speed of the medium (disk or other) the class is loaded from, the class's size, and the overhead incurred by the class loaders themselves. The delay to load a class can commonly be as high as 10 milliseconds. If tens or hundreds of classes need to be loaded, the loading time itself can cause a significant and possibly unexpected delay. Careful application design can be used to load all classes at application start-up, but this must be done manually because the Java language specification doesn't let the JVM perform this step early.
- **Garbage collection:** The benefits of GC to application development – including pointer safety, leak avoidance, and freeing developers from needing to write custom memory-management tooling – are well documented. However, GC is another source of frustration for hard RT programmers using the Java language. Garbage collects occur automatically when the Java heap has been exhausted to the point that an allocation request can't be satisfied. The application itself can also trigger a collection. On the one hand, GC is a great thing for Java programmers. Errors introduced by the need to manage memory explicitly in languages such as C and C++ are some of the most difficult problems to diagnose. Proving the absence of such errors when an application is deployed is also a fundamental challenge. One of the Java programming model's major strengths is that the JVM, not the application, performs memory management, which eliminates this burden for the application programmer. On the other hand, traditional garbage collectors can introduce long delays at times that are virtually impossible for the application programmer to predict. Delays of several hundred milliseconds are not unusual. The only way to solve this problem at the application level is to prevent GC by creating a set of objects that are reused, thereby ensuring that the Java heap memory is never exhausted. In other words, programmers solve this problem by throwing away the benefits of the managed memory by explicitly managing memory themselves. In practice, this approach generally fails because it prevents programmers from using many of the class libraries provided in the JDK and by other class vendors, which likely create many temporary objects that eventually fill up the heap.
- **Compilation:** Compiling Java code to native code introduces a similar problem to class loading. Most modern JVMs initially interpret Java methods and, for only those methods that execute frequently, later



compile to native code. Delayed compiling results in fast start-up and reduces the amount of compilation performed during an application's execution. But performing a task with interpreted code and performing it with compiled code can take significantly different amounts of time. For a hard RT application, the inability to predict when the compilation will occur introduces too much nondeterminism to make it possible to plan the application's activities effectively. As with class loading, this problem can be mitigated by using the `Compiler` class to compile methods programmatically at application start-up, but maintaining such a list of methods is tedious and error prone.

Although Java is a programming language suitable for implementing agents, their use in an environment with real-time characteristics is inappropriate, mainly because one of its greatest qualities; the garbage collector. A real-time environment must take all the control over all the tasks that can be executed within the system. When the garbage collector runs, all Java threads are stopped at the moment until the collector completes its task [258].

We should not allow this to happen if we want that the agents' tasks finish according to their deadlines. Another of the reasons that Java is not a suitable language for the implementation of applications with time constraints are:

- Java has no mechanism for analyzing the viability of the plan, necessary if you want to know, a priori, whether a task is schedulable.
- The threads used in Java applications do not have methods to add temporal characteristics to them.
- Java does not resolve the problem of priority inversion in tasks that share resources.
- And in general, it is not intended to run real-time applications, whereby the use of such language is inappropriate.

One solution is to adapt Java in the way that solves the above mentioned problems. Today there are two committees in charge of adapting Java to real-time systems: the first is the Real-Time for Java Expert Group that propose a specification called Real-Time Specification for Java (RTSJ) [140], and the second group is called J-Consortium [74] with its specification Real-Time Core Extensions for Java. The extension proposed by RTSJ is the most known and therefore, is the one chose for this platform.

We must clearly distinguish between the above specifications and execution environments for this it. After selecting RTSJ, we evaluated different available JVMs. Therefore, in this chapter we present in two distinct sections the bases of RTSJ and in the next, the possible JVMs. Later, we deal with the problem of annotating the bounded loops. At the end of the chapter, we present the

agents needed for the execution of the platform and how the classes changed in comparison with the PANGEA platform.

## 7.2

## Real-time specification for Java

---

Because of the popularity of Java as a development language, there has been a major effort to address the shortcomings of Java as a real-time development language and to introduce new features into the language to support real-time development. These led, over a number of years, to the development of a real-time specification for Java. This includes both changes to the language and to the JVM, the run-time system for Java.

The RTSJ extends the JVM and Java class libraries in order to facilitate real-time development with Java. The initial version 1.0 of the specification was conceived in 2001 as JSR 1 and this description is based on the current stable version 2.2 of the specification [273]. The key features that have been included in real-time Java are:

- Precise memory management. A major problem with Java is that garbage collection is unpredictable and, if garbage collection kicks in during an operation, the response time may be affected. Real-time Java introduces the notion of immortal memory where objects are created but never destroyed and scoped memory where objects are destroyed when a process goes out of scope. Neither of these memory types are garbage collected.
- Direct memory access. Programs can access physical memory directly, thus allowing direct hardware interaction in embedded systems.
- Asynchronous communications. This includes asynchronous event handling where the response to events from outside the JVM can be scheduled and asynchronous transfer of control which allows threads to be safely interrupted.
- Precise timing specification. Time may be specified to the nanosecond level of precision.
- Real-time threads. These are processes that are not interrupted by garbage collection and which may be assigned up to 28 priority levels. A lower priority thread cannot block access to a resource that is needed by a higher-priority thread.

Java RTS is similar to Java SE in many ways. The most important are [58]:

- JAVA RTS uses the same unmodified compiler, `javac`.
- Supports many of the same command-line options.
- Is fully bytecode-compatible with Java SE and will run all existing `.class` and `.jar` files that are Java SE 5-compliant.
- Works with most existing Java SE tool, however there are exceptions.

The following gives an overview of some of the changes introduced by the RTSJ [119]:

### Memory management

Java does not provide means for explicitly deallocating memory, instead a garbage collector is normally used in various JVMs. Unfortunately, as noted in [371], garbage collection affects the timing predictability of real-time systems, which in turn complicates WCET analysis. Some research has been conducted on the subject of real-time garbage collection [319] [25], but the RTSJ does not rely solely on this yet. Instead, the RTSJ provides a memory model which avoids garbage collection, providing time predictable memory allocation and deallocation. This new memory model consists of two new memory areas in addition to the heap which can be used when allocating new objects. The two new areas are:

- The Immortal Memory is released when the application ends, therefore all the defined in this type of memory cannot be removed while the application is being executed.
- The Scope Memory is released when there is no object referencing to that memory.

### Schedulable objects and scheduling

Java supports threads and priorities, but the Java specification does not guarantee that the threads are executed in accordance with their priorities if e.g. the underlying OS does not support them [371]. Furthermore, Java does not support expressing concepts such as release patterns in real-time systems. This is solved in the RTSJ through two changes. One is a more general notion of schedulable objects and a number of classes used to specify important properties of these, such as their priority and period. Furthermore, the JVM must be more strict in its notion of priorities such that real-time scheduling policies can be adhered to.

RTSJ enables to create a special class of real time thread, `NoHeapRealTimeThread`. This thread has the characteristic that it can not be located or reference the heap memory area, ie, the one that is affected by the GC. The most important parameters of the real-time thread include:

- **SchedulingParameters:** Provides the parameters needed by the system scheduler for the calculation of planning. The priority of the thread is defined inside this parameter.
- **ReleaseParameters:** This parameter defines the characteristics of the thread, for example, whether the thread is periodic or sporadic. We can use for this the subclasses of **ReleaseParameters** that are **PeriodicParameters**, **AperiodicParameters** and within the latter a special class called **SporadicParameters**. In our case, we are interested in **PeriodicParameters** where the behavior is defined in case of a periodic. This behavior will come defined by the attributes start, period, cost y deadline. The start attribute defines when the thread starts its execution. The period attribute specifies when the thread will be activated again. The cost attribute is the necessary execution time to complete the task defined in the thread in the worst case. And finally, the deadline attribute defines how much time it is available to complete the task.

### Asynchronous event handing and timers

In addition to the threading model, a model of asynchronous event handlers has been added which allows for executing schedulable objects on specific events. These events can be fired from the software itself, hardware interrupts, periodically by timers, etc. This model co-exists with the threading model and provides an alternative approach to executing tasks in a real-time system.

Asynchronous Transfer of Control (ATC) ATC is a mechanism which allows schedulable objects to interrupt each other in order to change their control flow. This mechanism is motivated by the need to implement forms of error recovery and mode change which needs to take effect quickly.

### Scheduling characteristics

The specification RTSJ provides a scheduler that is responsible for checking if the tasks are feasible according to the policy used and managing the execution of schedulable objects. The schedulable objects are: the **RealtimeThread**, the **NoHeapRealtimeThread** and the **AsyncEventHandler** (asynchronous event manager). By default, the Scheduler has the following characteristics:

- 28 priority levels.
- The scheduling algorithm is preemptive (with replacement) and with fixed priorities.
- The policy used to test the feasibility of a particular task will be planning algorithm Rate Monotonic.

### Physical and raw memory access

In real-time systems, and especially embedded real-time systems, it is often necessary to handle physical memory. This may be the case if one needs to interact with memory-mapped devices or differentiate between different types of memory connected to the system. To support this, the RTSJ provides facilities to specify the placement of memory areas and to access memory more directly than allowed in traditional Java.

### Time values and clocks

Time is inherently an important part of real-time systems, thus the RTSJ provides enhancements to Java in the form of a high resolution timer, with support for representing times down to one nano-second, given a hardware clock supporting this. The timer is supported by a concept of clocks which show the passage of time and concepts of absolute and relative time used in various parts of the API.

## 7.3

## Real-time Java virtual machines

We look for a Real-time Java Virtual Machines (RTJVMs) that must be completely RTSJ compliant. For this election, we have studied the most used at this moment. Until 2009 the official RTJVM was the Sun Java RTS [272] but when Oracle abandoned its support and the RTJS was released, different virtual machines has been developed with different levels of success. In this section we present a summary of them and the justification of our choice.

Atego Perc [19]	"Atego Perc Ultra is one of the most deployed embedded and real-time Virtual Machines in the industry. Atego Perc Ultra is a virtual machine and tool set expressly created for demanding embedded and real-time systems requiring Java Standard Edition support. Atego Perc Ultra delivers the ease and efficiency of Java SE without sacrificing integrity, performance, or real-time behavior. The Atego Perc product line offers Ahead-of-Time (AOT) and Just-in-Time (JIT) compilation, remote debug support, deterministic garbage collection and standard graphics." [264]
-----------------	---

JamaicaVM [73]	Developed by the Aicas Company, JamaicaVM is a hard real-time Java VM with a fully preemptable, deterministic garbage collector. Depending on the hardware and operating system a submicrosecond jitter can be achieved. The JamaicaVM toolchain contains an application builder and profiler for optimizing your applications and all the J2SE 6 language features are supported.
IBM WebSphere [171]	WebSphere Real Time bundles real-time capabilities with the standard JVM. It is possible to enable real-time capabilities by using the <code>-Xrealttime</code> option when running the JVM or any of the tools provided. WebSphere Real Time removes these obstacles of a classical JVM by providing: the Metronome Garbage Collector, an incremental, deterministic garbage collector with very small pause times Ahead-Of-Time (AOT) compilation and a priority based FIFO scheduling.
Reference Implementation [353]	The Reference Implementation (RI) de Java is developed by the company TimeSys. This company participates in the definition of RTSJ and is responsible for developing the official reference implementation of RTSJ. TimeSys is authorized by the Java Specification Process Executive Committee to maintain and modify the specification RTSJ. But it can not be used in production environments.
jRate [77]	Developing at the University of California, Irvine. It is an extension of the GNU Compiler for Java (GCJ) and supports the specification RTSJ. One goal of the project is that the user can generate a customize jRate version adapted to his machine and his needs. Currently, it have not a full version that meets all requirements of the specification RTSJ.
Ovm [365]	Ovm is implemented almost exclusively in Java with only small amounts of C for the bootloader and low level facilities. The Ovm project aims to develop a customizable framework for research in virtual machines and object-oriented programming language runtime systems. It has been used to implement a Real-time JVM. The high performance realtime configuration of OVM relies on ahead of time compilation. The entire program is processed to maximize the opportunities for optimization and an executable image is generated for a particular Java application. [15]

**Table 7.1:** Overview of Java Virtual Machines for real-time

Looking for applications that can help to choose one among the presented, we found that just four of the RTJVM are successfully introduced into real applications, for example: US Navy's Aegis Weapon System uses Atego Perc, US Navy's DDG-1000 uses IBM WebSphere and JamaicaVM is in use some avionics platforms. We reject the TimeSys RI because even though it is currently the official one, in the work [77] they show performance results of jRate that illustrate how well it performs compared to the TimeSys RI. Due to this, we decide to restrict our choice in these four. The ATEGO PERC and IBM WebSphere need expensive licence and for research issues is out of our reach. The JamaicaVM and jRate have academic licenses, so we based our finally decision on the quality of the existing documentation for solving implementation issues and the users' opinions. According to these factors, we chose JamaicaVM, highly extended in the Java real-time community.

In our case, the embedded nodes have the single-board computer Raspberry Pi, so PANGEA+RT has been tested with a Debian distribution and the Fiji VM [173]. For the high-level computational nodes we have installed different Linux distributions. But the standard Linux kernel only meets soft real-time requirements: it provides basic POSIX operations for userspace time handling but has no guarantees for hard timing deadlines. With Ingo Molnar's Realtime Preemption patch (RT-Preempt) and Thomas Gleixner's generic clock event layer with high resolution support, the kernel gains hard real-time capabilities. So, we have installed the RT-Preempt patch. This can be consulted in chapter 9 of this document.

## 7.4

## Annotation for bounded loops

---

The temporal analysis of the tasks cannot be made automatic and normally requires source annotations to assist the analysis. Then, in order to integrate the proposed WCET model, it is needed to include a new mechanism to bound the loops in the code. To enable an automatic timing analysis of arbitrary programs, the user is typically forced to support the WCET analyzer with program annotations about the loop iteration counts. In a similar way, recursion depths and target addresses of dynamic calls and jumps, which can not be statically determined, make a user annotation mandatory. If the program execution is dependent on input data, e.g., in sorting algorithms, the user annotations must respect all potential values that may be provided to the program [223].

A number of competing and incompatible styles of annotation have been

developed for real-time Java. The earliest work in WCET annotations for Java comes from Bernat, who proposed a portable WCET analysis tool [43]. In this case, portable refers to language portability: The tool was designed for analyzing Java, C, Ada, and any other language that could be translated to Java bytecode. To achieve such portability, source code must invoke methods in a predefined class whenever a WCET annotation is required. Compared to traditional annotations, this style mingles non-functional metadata (that is, the WCET information) with the normal source code statements, making programs more difficult to read. The XRTJ project [163] implemented WCET annotations in a more traditional way. All annotations appear as comments with the characters `//@` for single lines and `/*@ ... */` for multiple lines [162]. The XRTJ compiler parses these lines and writes them to an XAC (Extensible Annotation Class) file, an XML-like text file that is paired with its class in a real-time Java program. The XRTJ analyzer then reads each XAC file to determine loop bounds, timing modes, and other details necessary to derive the WCET [161]. In more recent work, Schoeberl and Pedersen implemented a WCET analyzer for JOP [318]. This tool introduced yet another syntax for annotations. Although it still used the same `//@` start marker, the syntax of the annotations differed from the XRTJ project, making the tools incompatible with each other.

The Java Modelling Language (JML) is a notation for formally specifying the behavior and interfaces of Java [16] [139] classes and methods. However, the design of design currently has minimal support for specification of space and timing constraints, based on the work of Krone et al. [208]. SafeJML [152] is an extension to the JML for specification of safety critical Java programs. The SafeJML design as a JML extension supports the modular specification of both functional and timing constraints. Although, we face the same problem since it still used the same `//@` start marker.

In the works presented above, the annotations are not comments despite their format. Instead, they are first-class objects in the Java language, requiring special compiler support. They can have parameters and must conform to type-checking rules, for example. Such a substantial change to the language demanded formal review under the Java Community Process (JCP): a public, cooperative system for adopting new technologies as official Java specifications.

The proposed annotation framework was submitted to the JCP as JSR-175 [290] and met final approval in September 2004. That same month, Sun released Java 5, Standard Edition, which included support for JSR-175 annotations. A new edition of the Java Language Specification [4] was published the following year to formalize these annotations and ensure that tools and libraries using them would remain compatible with each other. Today, annotations are a standard,



well-defined part of the Java platform. Currently, the version JSR-308 [291] is the last one but it was expected to be included in the Java SE 8 but it has not yet received official final approval.

One of the advantages of Java annotations and a key difference versus existing WCET annotation frameworks, is that the annotation data is stored directly in files `.class`. Embedding annotations within classes simplifies code management because a separate file format for metadata is not required to be maintained. Also, existing mechanisms for storing, distributing, and deploying class files can readily be used for annotation data. For instance, a build script that packages a class library into a JAR file (Java ARchive) will automatically package annotations, as well.

However, Java annotations cannot be placed on critical source code statements such as loops, rendering the mechanism useless for WCET annotations. The JSR-308 committee is currently investigating the possibility of removing this restriction. Analysis tools for determining WCET may suffer from limited reasoning capabilities or lack of knowledge about the execution context. Developers should be able to supply information to the WCET analysis, allowing a static analyzer to compute a better bound on the WCET. For example:

```

1  @LoopBound(max=100)
2  for (int i=0; i<percent; i++)
3  { ... }
```

**Programming Code 7.1:** Desired annotation in JSR-308

Now Java is becoming viable for hard real-time systems, however, Java annotations are simply too limited. For instance, the `@LoopBound` example shown above is currently impossible because Java annotations simply aren't allowed on loops. An extension of the annotation mechanism to support statements is therefore necessary.

We want to follow the current standards to make our proposal more accessible to the users and since Java SE 5, annotations can only be applied to type uses. This means that annotations can be used anywhere you use a type. For example, in a class instance creation expressions `new`, `cast`, `implements` and `throw` clauses. This form of annotation is called a type annotation (7.2).

```

1  public void Example ()
2  {
3      @AnnotationValid    // legal in Java
4      int i = 10;
5
6      @LoopCount(100)    // illegal in Java
7      for (int k = 0; k < nextFree; k++) {
8          retValue.theItems[k] = theItems[k];
9      }
```

```

10
11     return retValue;
12 }

```

**Programming Code 7.2:** Example of type annotation

Then, we have adapted our annotation mechanism to meet this standard in PANGEA+RT. This lead us to establish the following restriction when programming loops:

*All the methods that include loops should declare the variable used to go through each loop at the beginning of the method and include there the corresponding annotation. As consequence, no duplicate variables can be used in different loops belonging to the same method.*

```

1 public void ExampleRestriction ()
2 {
3     @LoopBoundAnnotation
4     int i;
5     @LoopBoundAnnotation
6     int j;
7     @LoopBoundAnnotation
8     int k;
9
10    for (i=0; i<total; i++)
11    { ... }
12
13    j=0;
14    while (j < 11)
15    { ... }
16
17    for (k=0; k<10; k++)
18    { ... }
19 }

```

**Programming Code 7.3:** Desired annotation in JSR-308

We developed our personalized label using the standard available since SE 5. Creating an annotation is similar to creating an interface. But the annotation declaration is preceded by an `@` symbol. The annotation declaration itself must be annotated with the `@Retention` annotation. The `@Retention` annotation is used to specify the retention policy, which can be `SOURCE`, `CLASS`, or `RUNTIME`.

- `RetentionPolicy.SOURCE` retains an annotation only in the source file and discards it during compilation.
- `RetentionPolicy.CLASS` stores the annotation in the `.class` file but does not make it available during runtime.

- `RetentionPolicy.RUNTIME` stores the annotation in the `.class` file and also makes it available during runtime.

We chose the `RUNTIME` retention policy since we have to evaluate the class files.

An annotation cannot have the `extends` clause. However, annotations implicitly extend the `Annotation` interface but it is needed to include the package `java.lang.annotation`. The body of an annotation consists of method declarations without body. These methods work like fields.

```

1 @Retention(RetentionPolicy.RUNTIME)
2 @interface LoopAnotation
3 {
4     double maxBound();    // Annotation member
5     double minBound();   // Annotation member
6 }

```

**Programming Code 7.4:** Loop-bound annotation

Once an annotation is created, it can be applied. While applying an annotation to a declaration, it is necessary to provide values for its members (7.5):

```

1 public class Test
2 {
3     public void ExampleMyAnnotation ()
4     {
5         @LoopAnotation(maxBound=20,minBound=0)
6         int i;
7         @LoopAnotation(maxBound=11,minBound=0)
8         int j;
9         @LoopAnotation(maxBound=10,minBound=0)
10        int k;
11
12        for (i=0; i<total; i++)
13        { ... }
14
15        j=0;
16        while (j < 11)
17        { ... }
18
19        for (k=0; k<10; k++)
20        { ... }
21    }
22 }

```

**Programming Code 7.5:** Desired annotation in JSR-308

Now, thanks to the Java Reflection property which can be use to load the Java class, call its methods or analysis the class at runtime, the annotations can be queried at runtime (7.6):

```

1 public static void getAnnotations()
2 {
3     Test test=new Test();
4     try
5     {
6         Class c=test.getClass();
7         Method m=c.getMethod("ExampleMyAnnotation");
8
9         LoopAnotation annotation=m.getAnnotation(
10             LoopAnotation.class);
11         double maxBound=annotation.maxBound();
12         double minBound=annotation.minBound();
13         ...
14     }
15     catch(NoSuchMethodException ex)
16     { ... }
17 }

```

**Programming Code 7.6:** Obtaining annotations in runtime

## 7.5

## Agents of the platform

The agents of the platform are clearly reduced in comparison with the agents that compose the PANGEA+R platform. The introduction of the MQTT communication protocol in the previous version PANGEA+R has already allowed us to remove the *CommunicationAgent*, the *SnifferAgent* and the *SubscribeAgent* since all their functions are automatically supported by the protocol and the Mosquitto broker. The *GatewayAgent* is also no more needed because the link between MQTT and FIPA-ACL is not beneficial when working in real-time and can cause dangerous delays.

We only maintain the strictly necessary agents to avoid possible delays and to limit the communication acts:

- *OrganizationManager*: the agent responsible for the actual management of organizations and suborganizations. It is responsible for verifying the entry and exit of agents, and for assigning roles. To carry out these tasks, it works with the *OrganizationAgent*, which is a specialized version of this agent.
- *OrganizationAgent*: it is a specialized version of the *OrganizationManager*, which is introduced automatically in each suborganization to help the

OrganizationManager and avoid its overload. It communicates with the real-time agents in order to verify their functioning and time constraints.

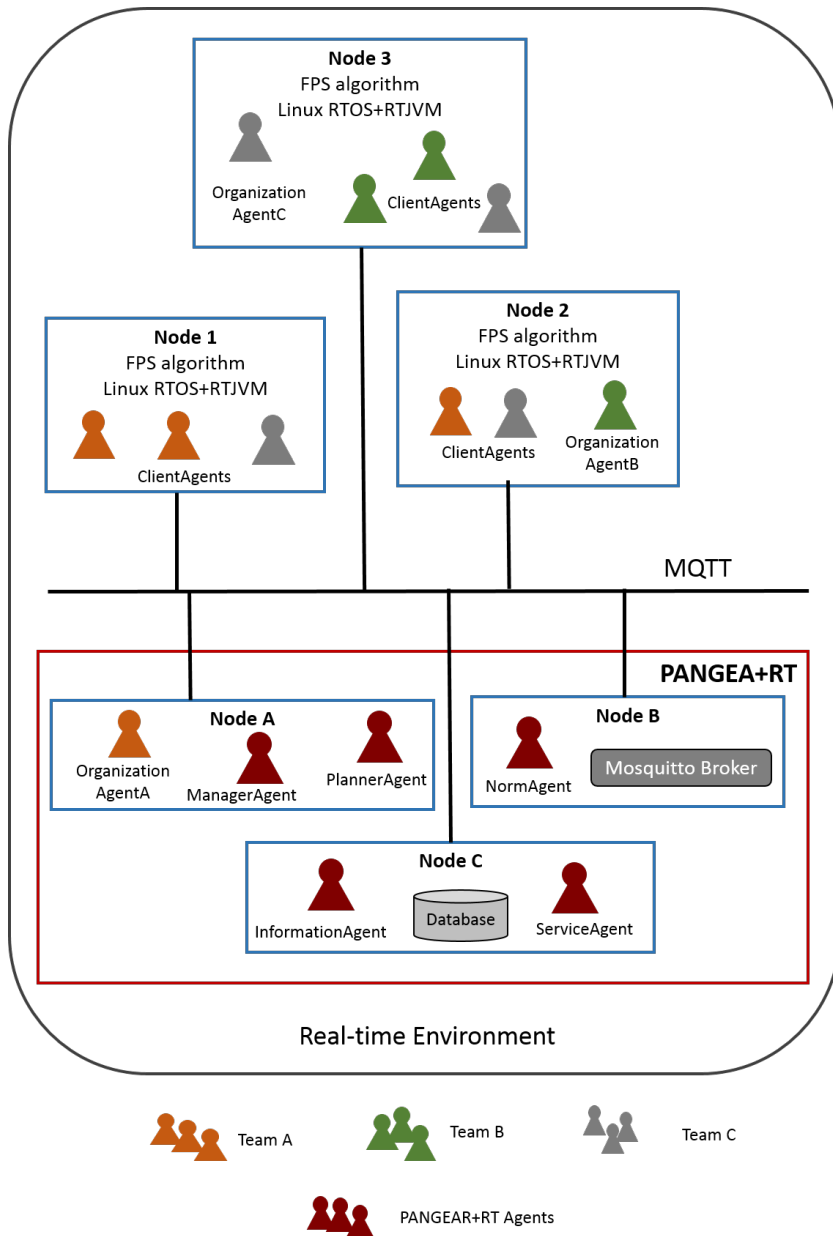


Figure 7.1: Platform Overview

- *InformationAgent*: the agent responsible for accessing the database containing all pertinent system information.

- *ServiceAgent*: the agent responsible for recording and controlling the operation of services offered by the agents. It works as the DF defined in the FIPA standar.
- *NormAgent*: the agent that ensures compliance with all the refined norms in the organization.

Now, we need an extra role called *PlannerAgent*. This role implements the model presented in chapter 4 for the global scheduling and task allocation. It is in contact with the *OrganizationManager*. The *OrganizationManager* controls the entering of new agents and notify the *PlannerAgent* this kind of events. The *PlannerAgent* is also in contact with the *OrganizationAgent*, who notifies when a replanning action is needed in case that any agent had suffered a problem.

In this version of the platform only real-time agents are allowed. The main reason is that the communication between a non-real-time agent and a real-time agent is not time bounded since one of the agents does not follow the real-time constraints. Then, the possible interaction between them can lead to a time fault.

## 7.6

## Modification of the classes

---

As we explained in Chapter 5, in PANGEA each agent extends of an abstract class called **Agent**. For PANGEA+RT, we develop a different abstract class to include all the time constraints, **AgentRT**. The same happens with the abstract class **Behaviour**, now we have the **BehaviourRT** class.

In Figure 11.7 we can see the main classes of the implementation:

- **AgentRT**: it defines all the methods that the real-time agents must implement.
- **MqttCallback**: this is not a class specifically of the platform. But all the agents must extends this class to use the MQTT protocol.
- **RealtimeThread**: as the previous class, this is not a proper class of the platform but is totally needed. All the agents must extend this class to be executed as real-time threads in Java.

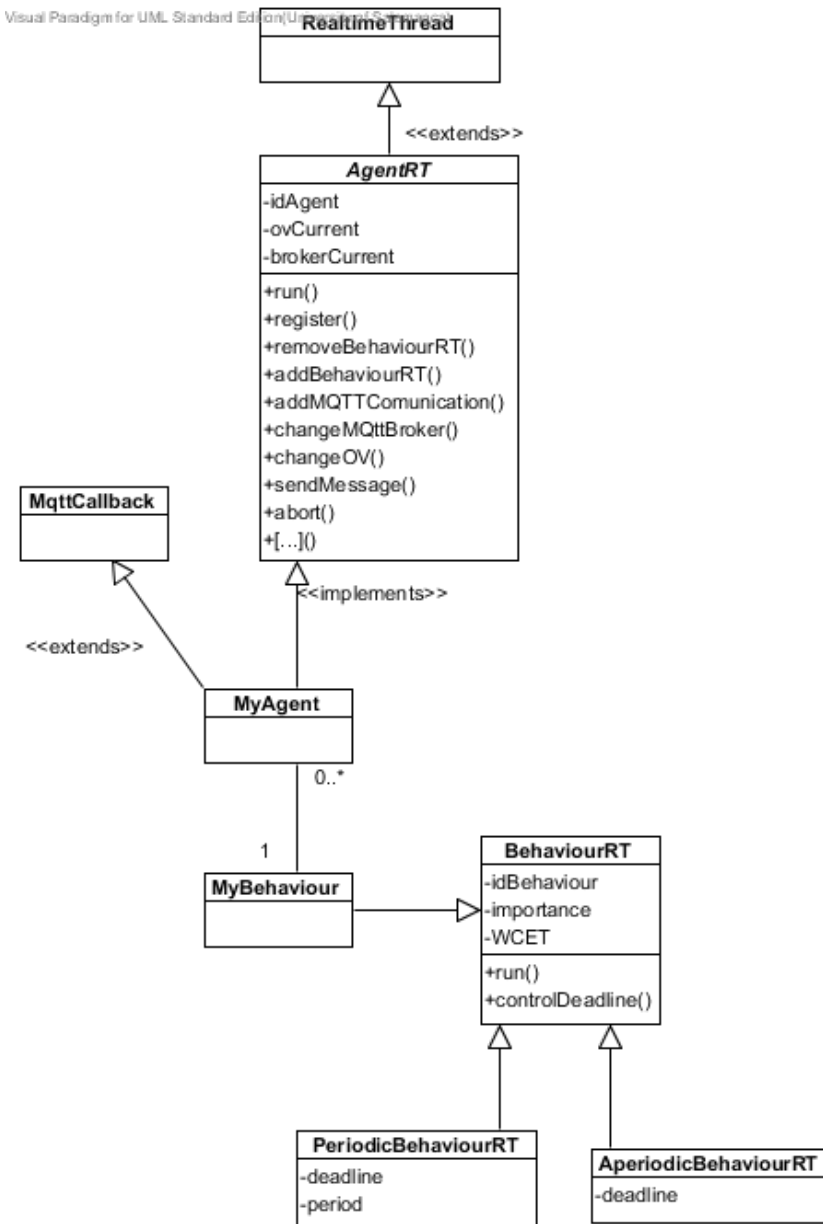


Figure 7.2: Main classes of the platform

- **BehaviourRT**: it defines the behaviour of the agent, that can be periodic or aperiodic.
  - **PeriodicBehaviourRT**: this class specifies a deadline that the behaviour and the period of execution must comply .

- **AperiodicBehaviourRT**: at this moment, PANGEA+RT only allows sporadic and aperiodic task assigning *period = deadline*.

It is important to mention the method `controlDeadline` of the `BehaviourRT` class. If the tasks programmed inside the method `run` are not finish within the time constraints (the specified deadline), `controlDeadline` must generate an event to handle the problem and notify the incident to the `OrganizationAgent`. The attribute `WCET` is also defined in the `BehaviourRT` class. This attribute enables to store the `WCET` estimation made using the model presented in Chapter 4.

Now, we present one simple example with the skeleton of an agent (11.5) an its associated behaviour (11.4).

```

1 public class MyBehaviour extends PeriodicBehaviourRT
2 {
3     public MyBehaviour(int importante , long WCET, long
4         deadline , long period)
5     {
6         //Constructor
7         super();
8     }
9     public void run()
10    { //Task to carry out }
11
12    public controlDeadline()
13    { //Actions if the deadline is not fulfilled }
14 }

```

**Programming Code 7.7:** Example of a simple `MyBehaviour` class

```

1 public class MyAgent extends MqttCallback implements AgentRT
2 {
3     private MyBehaviour behaviour;
4     private BlockingConnection connection;
5
6     public MyAgent(string ovCurrent , string brokerCurrent ,
7         MyBehaviour behaviour)
8     {
9         //Constructor
10        this.behaviour=behaviour;
11        super();
12    }
13
14    public void run()
15    { //Actions before starting the periodic task of the
16        behaviour }
17
18    public void addMQTTCommunication(string brokerCurrent)
19    {
20        //MQTT client

```



```

19     MQTT mqtt = new MQTT();
20     mqtt.setHost(brokerCurrent);
21     BlockingConnection connection = mqtt.
        blockingConnection();
22     connection.connect();
23 }
24
25 public sendMessage(string message, string topic)
26 {
27     connection.publish(topic, message.getBytes(), QoS.
        AT_LEAST_ONCE, false);
28 }
29
30 [...]
31 }

```

**Programming Code 7.8:** Example of a simple `MyAgent` class

The class `MyAgent` will be executed as a real-time Java thread (11.6).

```

1 public static void main (String [] args)
2 {
3     MyAgent agent = new MyAgent();
4     agent.start();
5     try{
6         agent.join();
7     } catch(InterruptedException ie)
8         //Exception
9     }
10 }

```

**Programming Code 7.9:** Real-time Java thread

## 7.7

## Conclusions

We have developed an innovative platform because, as mentioned in the related works in chapter 3, to the best of our knowledge there is no other platform suitable for VOs with real-time constraints. The advantages of PANGEA and PANGEA+R are extended to enables real-time agents to interact within a temporal control.

In PANGEA+RT, by means of the `PlannerAgent`, we have implemented the model presented in the fourth part of this document. Now, the platform can control the arrival of new agents, each one with different behaviours. Despite this diversity, the platform can include the agents in an organization and follow the real-time schedule and distribution of task among the available nodes.

The platform requirements for each node are initially a Linux RT distribution with a real-time patch or a RT-Linux OS. It is also needed a RTJVM.

We have also shown how a basic agent has to be implemented to fulfill its time constraints and to be supported by the platform. We have also explained how to annotate the loops in the Java classes to be properly bounded and get satisfactory WCET estimations.

In the next part of this document, we will see the results including a case study "Collaboration of heterogeneous robots for surveillance tasks developed under the agent-platform PANGAEA+RT" where two kind of robots: HAWKS, a UAV (Unmanned Aerial Vehicle) developed by the BISITE group, and GECKOs, a UGV (Unmanned Ground Vehicle) also developed by the BISITE group. The team's goal is to scan a large peripheral area focusing on surveillance tasks but this system can be also applied to rescue tasks using image processing. All the development of this team is implement with the PANGAEA+RT platform.

# Part V

CASE STUDY AND  
CONCLUSIONS



# 8

## CASE STUDY

---

*In this chapter, we present a case study where all the innovations presented in this PhD dissertation are evaluated. The case study describes the collaboration of heterogeneous robots for surveillance tasks developed under the agent-platform PANGEA+RT. This scenario involves two kind of robots: HAWKs, which are UAVs (Unmanned Aerial Vehicle) developed by the BISITE group, and GECKOs, UGVs (Unmanned Ground Vehicle) also developed by the BISITE research group. The team's goal is to scan a large peripheral area focusing on surveillance tasks using image processing. This case study allows us to mix the concept of VO by team formations, the real-time model proposed in the part III of this document and finally, to test the PANGEA+RT platform which includes all the desired characteristics to develop successfully all the requirements of this case study.*

### Contents

---

<b>8.1</b>	<b>Collaboration of heterogeneous robots for surveillance tasks</b>	<b>186</b>
<b>8.2</b>	<b>Related works</b>	<b>187</b>
<b>8.3</b>	<b>Presentation of the heterogeneous robots</b>	<b>192</b>
<b>8.4</b>	<b>The problem and the VO solution</b>	<b>197</b>
8.4.1	Proposed VO of agents	198
<b>8.5</b>	<b>Collaboration description</b>	<b>202</b>
8.5.1	Calculations for the collaborative movement	204
8.5.2	Common area calculation	208
8.5.3	Calculation of horizontal and vertical movement for the HAWK waypoints	209
<b>8.6</b>	<b>Deployment of the involved agents</b>	<b>211</b>
<b>8.7</b>	<b>Conclusions</b>	<b>213</b>

---

---

## Collaboration of heterogeneous robots for surveillance tasks

---

### 8.1

The philosophy of developing a model of collaborative agents is to create a system that interconnects the agents, allowing their collaborative effort to extend beyond the individual abilities of the members [121]. In general, having a single robot with multiple capabilities can lead to a waste of resources, increased overhead and lower efficiency. Applying the method known as "divide and conquer", we can deduce that different robots, each one with their own settings, form a more flexible, robust and cost efficient system [259]. Indeed, while some tasks can be too complex for a single robot, multiple robots can perform these tasks more effectively by working together [12].

Many studies highlight the suitability of using MAS in the robotics field [346] [35] [197]. Researchers in [175] highlight the benefit of implementing software which is more reusable, scalable and flexible, while maintaining parallelism, robustness and modularity. Interaction standards, social abilities and cooperation are areas that have already been fully developed in the field of MAS and can be easily applied to groups of robots. Formally, a collection of two or more mobile robots working together is termed a team or society of multiple mobile robots [254]. In these cases, multi-robot approaches must show collaborative behaviour. Thus a multi-agent, multi-robot approach focuses on how to achieve cooperation by using the multi-agent paradigm. It should be noted that in this case the robot itself can be either a single agent or a collection of agents. The multi-agent paradigm is implemented inside the robot architecture, and it is also used outside to control the team, society or organization [177].

Possibilities offered by Unmanned Aerial Vehicles (UAVs) have increased in recent years thanks to advances in the technological world, especially aerial imaging. Aerial imaging enables quick monitoring of large areas, making it a very efficient way to solve problems related to surveillance tasks, rescue tasks, tracking animals or people, etc. In our case, not only does the system coordinate UAVs, but Unmanned Ground Vehicles (UGVs) have also been included in the team. The UAVs take aerial images while the UGVs are responsible for taking images of the ground, using their front camera. The main contribution is that each team is made up of a number of heterogeneous agents, some running inside the vehicles and others in the different nodes of the system. Therefore, each team will consist of a HAWK and GECKO and all necessary agents for controlling and monitoring the team. Moreover, many teams can focus on the task and can be coordinated in every moment.

The team's goal in the case study is to scan a large peripheral area of a route chosen by the personnel. Our main goal is to focus on surveillance tasks; however, this system can easily be adapted to be applied to rescue tasks. One of the advantages of the system proposed in the case study is that search time can be reduced as the team of robots helps scan the search area. The current system is precisely focused on providing a solution to help human staff with the surveillance efforts and reducing as much as possible the amount of time invested in the exploration of areas. It is necessary to remark that the PANGEA+RT platform [144] [392], specially designed to manage VOs, was successfully used as the infrastructure at a low level for the development of the system, introducing a new perspective where organizational theory, robotics and MAS are combined to manage real-time scenarios. The system was applied in a real environment to manage surveillance tasks in a leisure park called Valcuevo.

In the robotics field and, specifically, in this case study in which there are vehicles in movement, the real-time aspect takes an important role. As previously mentioned, in this case study we use PANGEA+RT which facilitates the development of the VO by team formations and integrates the model proposed in the part III of this document.

## 8.2

## Related works

---

An examination of the process of collaboration from the point of view of the VO indicates that it is closely linked to the concepts of coordination, communication and cooperation. Collaborative environments are described as a group of entities working together in the same work environment to promote the achievement of a shared goal task. Some features, mentioned in [303], that are needed to cover such environments are:

- **Social Organization:** Refers to the way the groups of agents in the system are organized according to the function or role, characteristics, responsibilities, or needs that they have, taking into account the purpose of their communication with other agents.
- **Cooperation:** Refers to the process of sharing intermediate results in an effort to find a solution for the specific objectives of other agents, while at the same time advancing toward the achievement of the global objectives of the system. In short, cooperation is the mechanism by which the agents work together to achieve a common goal, and define a strategy to achieve this goal [45].

- **Coordination:** This is understood as a set of additional actions which may be performed to achieve a goal which an agent with the same objective would not be able to achieve on its own. Gutknecht and Ferber in [151] define two basic reasons for coordinated actions: (i) Resources are limited. Agents can share resources to develop actions. (ii) Coordination optimizes costs by eliminating unjustified or redundant actions.
- **Communication:** Enables interaction among agents. To this end, not only should all messages exchanged between agents share a common language, but the agents themselves should be able to understand and interpret the knowledge being exchanged and be able to exchange it. This involves the ability of agents to communicate with each other, i.e., exchange information and knowledge in an understandable way. It allows agents to obtain the knowledge they need to decide the sequence of actions that must be executed according to their goals. Since communication enables interaction between agents, the exchanged messages must use a common language and the agents must be able to understand, interpret and redistribute the exchanged knowledge.

Until now, the majority of robotic teams were formed by homogeneous entities. A review of existing literature finds cases of exploration areas [269], soccer matches [118] [362], joint surveillance [35] [191], cooperative hunting [384], etc. But the biggest disadvantage of these devices is that the robots are very similar to each other, equipped with the same set of sensors and actuators, meaning that they offer limited benefits when performing complex tasks. In collaborative robotics, robots are now simpler and less equipped than before. As a result, less expensive equipment is needed because efficiency lies in collective intelligence and collaborative work.

Depending on the level of heterogeneity, robots in a team are jointly classified as weakly or strongly heterogeneous [197]. When the robots only differ in their capabilities, they are not identical but are still commonly considered to form a homogeneous robot team [277]. The main difficulty comes when the robots are equipped with different sensing, perception, motion and onboard computing capabilities. In this case, for example, an application with a strongly heterogeneous robot team was developed for aerial surveillance [269], where different robot types (a blimp, an airplane and a helicopter) cooperatively monitor a rural area to detect forest fires. This system called COMETS (Real-Time Coordination and Control of Multiple Heterogeneous Unmanned Aerial Vehicles) is aimed at designing and implementing a system for cooperative activities using heterogeneous UAVs. Heterogeneity is considered both in terms of aerial vehicles and onboard processing capabilities, ranging from fully autonomous systems to conventional remotely piloted vehicles. The



COMETS system also involves cooperative environmental perception including fire detection and monitoring as well as terrain mapping.

In [68], the authors present a heterogeneous ensemble of mobile robots focused on the development of mapping and exploration tasks. This system is based on the already cited method known as "divide and conquer", because it is formed by many robots with limited processing and sensing abilities. This means that each robot may not be able to execute all components of the mapping and exploration task. The hierarchical system proposed consists of computationally powerful robots (managers) at the upper level and limited capability robots (workers) at the lower levels. This enables resources (such as processing) to be shared.

For surveillance tasks, [384] presents a novel feedback-control law for coordinating the motion of multiple mobile robots for security against invaders in surveillance areas. Each robot in this control law (which can be similar to our norms) has its own coordinate system and it senses a target/invader, other robots and obstacles, to achieve this cooperative behavior without making any collision. There is no centralized controller and each robot has a vector referred to as a formation vector, and the formations are controllable by the vectors.

The work presented in [363] is similar to this since they describe localization-space trails (LOST), a method that enables a team of robots to navigate between places of interest in an initially unknown environment using a trail of landmarks (or waypoints, as in our case). The landmarks are not physical; they are waypoint coordinates generated online by each robot and shared with teammates. Waypoints are specified in each robot's local coordinate system, and contain references to features in the world that are relevant to the team's task and common to all robots. Despite significant divergence of their local coordinate systems, the robots are able to share waypoints, forming and following a common trail. But this system is oriented to indoor environments and makes no mention of agents or any distributed method of computation. Feddema et. al. [108], highlight how a decentralized control theory can be used to analyze the control of multiple cooperative robotic vehicles. They also present three applications of this theory: controlling a formation, guarding a perimeter, and surrounding a facility. The work most similar to ours, regarding teams involving aerial and ground vehicles, is the CROMAT Project architecture, which also implemented cooperative perception and multi-robot task allocation techniques [364]. This paper presents a system for the coordination of aerial and ground robots for applications such as surveillance and intervention in emergency management. Moreover, the paper presents a demonstration with a team of heterogeneous robots (aerial and ground) cooperating in a mission of fire detection and extinguishing. Nevertheless, this system does not mention any organizational aspects or supervision of teams.

In the paper [240], an intelligent cooperative control and path-following algorithm is proposed and tested for a group of mobile robots. The designed fuzzy model employs two behaviors: path following and group cooperation. As the robots move along individual predetermined paths, the designed algorithm adjusts their velocities so that the group arrives at their target points within the same time duration regardless of the length of each individual path. However, necessary to have a predetermined path involving a known environment, which is not the case in the majority of the situations.

Many studies focus on mapping unknown environments and the methods used to navigate through them using a team of simple robots. This is the case of [71]. The authors present a system where minimal assumptions are made about the abilities of the robots on a team. The robots can explore the environment using a random walk, detect the goal location, and communicate among themselves by transmitting a single small integer over a limited distance and in a direct line of sight. Additionally, one designated robot in the system, the navigator acting as a coordinator, can track toward a team member when it is nearby and in a direct line of sight. Another example is presented in [141]. The robots, called Millibots, are configured from modular components that include sonar and IR sensors, camera, communication, computation, and mobility modules. Robots with different configurations use their special capabilities collaboratively to accomplish a given task. They have developed a localization system that uses sonar-based distance measurements to determine the positions of all the robots in the group. Wurm et. al. [380], focus their paper on the key issue of how to assign target locations to the individual robots so that the overall mission time is minimized. To achieve this, they divide the space into segments, for example, corresponding to individual rooms. Instead of only considering frontiers between unknown and explored areas such as target locations, they send the robots to the individual segments with the task of exploring the corresponding area.

Alessandro Renzaglia et. al., in [302] take one step further, presenting a multi-robot three-dimensional coverage of unknown areas. They explain the problem of deploying a team of flying robots to perform surveillance coverage missions over an unknown terrain of complex and non-convex morphology. In such a mission, the robots attempt to maximize the part of the terrain that is visible while keeping the distance between each point in the terrain and the closest team member as small as possible.

In the context of Urban Search and Rescue (USAR), there is no standard robot platform capable of solving all the challenges offered by the environment [30]. The RoboCup Rescue competitions provide a benchmark for evaluating robot platforms for their usability in disaster mitigation and are experiencing ever-increasing popularity [200]. Since its inception, RoboCup Rescue has

been structured into two leagues, the Rescue Robot League and the Rescue Simulation League. Whereas the Rescue Robot League fosters the development of high-mobility platforms with adequate sensing capabilities, e.g., to identify human bodies under harsh conditions, the Rescue Simulation League promotes research in planning, learning, and information exchange in an inherently distributed rescue effort. This study [30] evaluates the groups of robots that have participated in this league.

Currently many studies focus on UAVs. For example, [270] compiles many studies related to teams of UAVs. Maza et al. [238] describe a multi-UAV distributed decisional architecture developed in the framework of the AWARE Project together with a set of tests with real UAVs and WSNs to validate this approach in disaster management and civil security applications. The paper presents the different components of the AWARE platform and the scenario in which the multi-UAV missions were carried out. The missions described in this paper include surveillance with multiple UAVs, sensor deployment and fire threat confirmation. In this case, however, the UAVs can be considered a weakly heterogeneous team since all the team components are similar. In [242] cooperation perception methods for multi-UAV systems are proposed. Each UAV extracts knowledge by applying individual perception techniques [204], and the overall cooperative perception is performed by merging the individual results. This approach requires knowing the relative position and orientation of the UAVs.

In general terms, teams composed of heterogeneous members involve challenging aspects, even for the intentional cooperation approach [271]. After consulting many studies, we realized that:

- There are few studies in which the level of heterogeneity can be considered strong. The robots used in the equipment are very similar, except in a few mentioned examples.
- The use of the agent technology and VOs, which can bring many advantages in cooperation, adaptation and resource management, has not been practically introduced into robotics, mainly focus on the concept of emergence.
- The proposed systems do not use a basis platform for managing low-level emergent behaviors or automatically manage the formation groups. Some studies focused on forming groups or coalitions [297] [146] [179], but they propose methods or techniques and none of them explain the needs of these mechanisms at a lower level, or propose a generic platform that supports such proposals. Hence, the use of our platform PANGAEA+RT.

The UAV and UGV vehicles used in this study will now be presented, followed by a description of the developed system.

### 8.3

## Presentation of the heterogeneous robots

The robot team presented in this work is composed of two types of vehicles with different configurations (heterogeneous), which collaborate autonomously and perform their tasks to reach their final goal. One of the vehicles is an UAV nicknamed HAWK and presented in the figure 8.2, and the other is a UGV nicknamed GECKO that can be seen in the figure 8.1.



**Figure 8.1:** Image of GECKO

One of the most popular technological advances in recent years are the multirotors or multicopters, a type of UAV capable of aerial filming, among other things, as described in [232]. The use of this kind of system to obtain an aerial video with enough quality is a tool that, in combination with image analysis to detect animals in the area, can save considerable time and money in the process of counting animals.

A powerful computer and a good communication system are needed to process the large amount of data obtained by HAWK. The transmission of large amounts of data requires at least one channel with high bandwidth. Most multirotor systems that exist today use analog channels for transmitting flight orders to a multirotor and sending the captured video [357] [153] [298].



**Figure 8.2:** Image of HAWK

The proposed system presents an alternative to these systems, taking advantage of existing technological advances such as powerful antennas and Wi-Fi, to unify the task of sending information such as flight commands, telemetry and high resolution video, through a single digital channel. Currently there are some systems that utilize UAV communication using sockets, but are restricted to use in UAV type aircraft.

This form of communication can both control the multirotor and obtain information about the status of the sensors it carries, such as the Global Positioning System (GPS) or altimeter. Of course, it can also send video images taken by its camera. Furthermore, the use of a computer instead of a radio station as a multirotor control element provides processing capacity that the station does not have; this allows the computer to control the multirotor intelligently without a human pilot. The architecture of the multirotor for obtaining images consists of three main parts: hardware, software and communication protocol.

The hardware part of the system refers to the system running the software and is present at both ends of the communication channel. At one end is the multirotor UAV with 6 engines and 6 arms. The UAV carries a Single Board Computer (SBC) with a 700MHz processor and 512MB memory capable of running a Linux distribution. The connectivity of the SBC allows connecting a camera via Ethernet, a Wi-Fi antenna Universal Serial Bus (USB) and gets the information that the Electronic Speed Controller (ESC) needs to control the motors of multirotor. At the other end of the communication is an access point whose strength depends on the distance to be achieved, and that can cover distances of up to 3 kilometers. This access point will connect the SBC UAV control computer to a high performance laptop with a USB gamepad connected for manual control.

The SBC runs a software that is primarily used to read, process and deliver telemetry, and is responsible for carrying out the calculations for the behavior

and stability of the multirotor. On the other hand, in the computer runs a software specifically developed (in Java) to control the multirotor both manually and autonomously through waypoints. The user can see all the information from the sensors that the multirotor sends in real time, such as information relating to power consumption, intensity and quality of the Wi-Fi signal, image and video (Figure 8.4). This information generates log files in XML format that will serve to make a more accurate and detailed analysis of the captured images, as it will provide additional useful information such as height and GPS.

The UAV configuration consists of a multi-rotor with the power required to transport vision equipment. It must also have sufficient stability to support the presence of adverse weather conditions (such as rain or wind) although, as noted above, its eventual improvement still remains a pending task.

In addition to stability, one reason for choosing a multi-rotor system is its vertical takeoff capability (VTOL), which does not require a large space for its implementation. Furthermore, the mechanical properties of a multi-rotor facilitate its transportability, due in large part to the folding arms system. This makes it ideal for use in leafy areas, as in the case of study for which it was developed.

The components of this equipment can be controlled by a programmed on-board autopilot system with Linux OS, which enables connecting peripherals through its multiple input ports as Ethernet port. This allows any type of IP camera to be connected. The cameras stream the images via a Wi-Fi network to a base station for monitoring and controlling. Under this scheme, different types of cameras for the UAV can be used:


- **Optical Camera:** Due to the presence of a high resolution optical camera, the ground conditions viewed from an aerial perspective can be checked remotely at any time. The status information is available at all times from the best viewing angle. Similarly, the optical camera can be useful for GECKO to record the ground level from different points of view.
- **Thermal camera:** This camera enables a better definition of objects with higher temperatures, making these images more easily visible to the human eye, and more easily identifiable by intelligent automatic detection systems.

The UGV is a 4x4 off-road car (scale 1:8) which can be tele-operated and combined with autonomous control in certain situations. The components of this type of robot are generally the same as those used with the UAV, excluding the thermal camera, thus lowering costs. However, the UGV incorporates an

IP camera with the same characteristics as the UAV. The most significant components are: the controller (onboard computer with Linux), Wi-Fi adapter for USB, IP Camera/Thermal camera, stabilizer, GPS, magnetometer, ultrasonic distance sensors (Sonar), barometric sensor.

The control logic for each type of vehicle is defined independently by the control agent of its vehicle, because they move by different means. Moreover, it is necessary to consider the obvious differences in the structure of the vehicle.

In Figure 8.3, a resume table is presented with the main technical characteristics.



FRAME	Diagonal Wheelbase	Frame Arm Length	Frame Arm Weight (Including Motor, ESC, Propeller)	Center Frame Diameter
		800mm	350mm	304g
FLIGHT PARAMETERS	Center Frame Weight	Bi-pod Size	Bi-pod Weight	Total Weight
	365g	500mm(Length)×415mm(Width)×320mm(Height) (Top width: 145mm)	428g	2.6Kg
MOTOR	Takeoff Weight	Load Weight	Power Battery	Max Power Consumption
	5.0Kg ~ 7.0Kg	0Kg ~ 2.5Kg	LiPo (6S, 10000mAh~15000mAh, 15C(Min))	2100W
ESC	Hover Power Consumption	Hover Time		
	720W(@ Takeoff Weight 6Kg)	Max: 16 min (@10000mAh&6KgTakeoff Weight)		
PROPELLER	Stator Size	KV	Max Power	Weight
	41×14mm	320rpm/V	360W	147g
ESC	Current	Voltage	Signal Frequency	Drive PWM Frequency
	40A OPTO	6S LiPo	30Hz ~ 450Hz	24KHz
PROPELLER	Weight			
	18g			
PROPELLER	Material	Size	Weight	
	Carbon Fiber	15×04in	15g	

# Gecko

FRAME	Scale	Length	Base width (without wheels)	Total width (with wheels)
		1:8	440mm	165mm
DRIVING PARAMETERS	Power Battery	Time		
	LiPo (3S, 3000mAh~8000mAh, 15C(Min))	Max: 43 min		
MOTOR	Size	KV	Max Power	Weight
	60×36mm	3000rpm/V	1700W	240g
ESC	Size	Voltage	Cooling Fan	BEC Output
	58×46×35mm	2-4S LiPo	5V	5.75V@3A
WHEELS	Diameter	Width		
	105mm	41mm		

Figure 8.3: Characteristics of HAWK and GECKO

Both vehicles are permanently connected to the global system, thanks to the agents deployed in the PANGEA+RT platform, and are able to communicate with each other (see section 8.4). For this communication, a Wi-Fi antenna

module was incorporated and installed into every one of the team members [2]. This allows them to exchange a lot more information than they could by using a Radio control module. However, this communication scheme penalizes the system in terms of information exchange, and even more so in the case of radio-frequency. On the other hand, the WiFi communication level of coverage can be increased with more powerful antennas than those used in this case study. In the checkpoint there is another powerful WiFi antenna which, under optimal conditions, allows the signal to reach distances of up to 10 km.

The communication mechanism between all the different agents is implemented using the MQTT protocol [178] as explained in the chapter 6. In this way, the system is ready for a possible extension where one computer can control different multirotors at the same time, and even communicate with each of them to achieve common goals more efficiently [307] [137] [143].

Finally, some screenshots of the software developed for GECKO (Figure 8.5) and for HAWK (Figure 8.4) are presented.

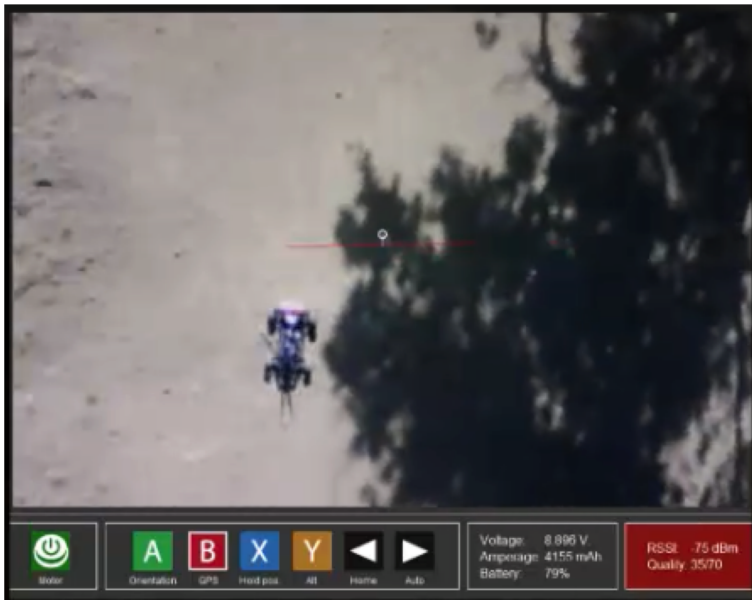


Figure 8.4: Software tool of HAWK



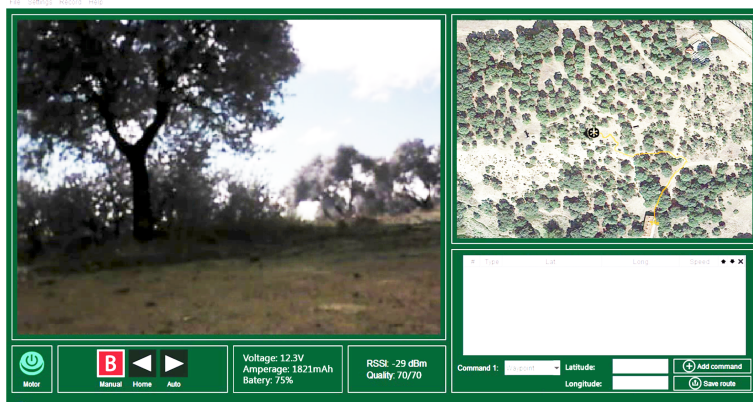


Figure 8.5: Software tool of GECKO

## 8.4

## The problem and the VO solution

In the proposed case study, the team's goal is to scan a large peripheral area of a route chosen by the personnel, in our case, who knows and monitors the Valcuevo park. Our main goal is to focus on surveillance tasks. With this system, search time can be reduced as the team of robots helps scan the search area. The current system is precisely focused on providing a solution to help human staff with their surveillance efforts and reducing the amount of time invested in the exploration of areas as much as possible.

As the system is agent-oriented, the PANGEA+RT platform has been successfully used for the system development. A cooperative-robotic heterogeneous VO was developed to aid in the surveillance tasks taking organizational aspects into account. In this section, the agents that form the VOs will be explained and finally, the steps carried out to define the collaborative work will be detailed.

The studies mentioned in the previous section show different systems for coordinated teams of robots. But none of them combines all the advantages that a VO can offer to collaborative robotics, specifically when focusing on heterogeneous teams and organizations. The use of a MAS composed of mobile robots that perform tasks in a non-structured environment offers several advantages such as robustness, specialization, implementation of functions that robots are not capable of performing individually, communication improvement, and easier robot programming thanks to the reuse. In order to facilitate the interaction among the robots, all of the characteristics offered by MAS, and all of the advantages of VO are included in PANGEA+RT.

Because the PANGEA+RT platform is VO-oriented, we can benefit from its advantages:

- The restrictions of the global objectives or limitations when forming groups can be formalized.
- With the concept of normativity, in any organization it is possible to define a set of norms governing the operation of the group. These norms may affect individual and collective behaviors, communications and interaction between agents and even between different organizations, and the access to the services.
- Description of groups and their topology. The groups can be formed according to various characteristics taking the global objective into account. Therefore, it is important to allow the input/output of entities to form groups dynamically, and if necessary, to control and limit such access.
- PANGEA+RT has mechanisms for management, control and service discovery, which makes it very easy to request entities to perform different tasks if an organization requires it to meet its objective.

Using PANGEA+RT in the field of collaborative robotics can greatly facilitate the implementation of monitoring and control processes thanks to the characteristics of VOs. The platform was used together with the +R middleware to facilitate its use in robotics, highlighting software reusability and allowing the programmer to integrate native software components (computer vision libraries, navigation modules, location, etc.). Moreover, the +RT middleware allows the system to work under real-time constraints.

#### 8.4.1

#### Proposed VO of agents

The agents are immersed in a virtual organization where different roles, norms, capacities and tasks are defined. With the VO approach, different agents can adopt different roles. The role is associated with the capabilities or behavior of the agent. Over time, an agent can play different roles and, thanks to the task allocation provided by PANGEA+RT, the roles can be reassigned to different agents on different nodes of the system in order to improve efficiency or simply to solve communication problems, non-desired stops of a node, etc..

A team must have agents that can play on all the roles presented below. However, there may be several agents playing the same role, increasing the number of agents in the team. This is very common in the case of the *InterfaceAgent* because if either the control interface of HAWK, the control interface of the

GECKO, or the interface controlling all the teams are deployed at more than one node, then all the nodes must include an *InterfaceAgent* for capturing the data from the *ControlAgent* of each robot. With our notation, when an specific agent plays a role, it is named as the role following by the word "Agent". For example, the agent that plays the role *Interface* is called *InterfaceAgent*.

The figure 8.6 presents an overview of the roles that the agents can take. The agents represented in blue are executed in the robots (depending on the proximity to the image that represents HAWK or GECKO). The agents represented in green are executed in any of the available nodes.

The agents and their associated roles required for a minimal team are:

*InterfaceAgent*: this role is responsible for capturing the data needed for the interaction between the personnel and the vehicles. There are three specializations of this role. The *InterfaceGECKOAgent* communicates with its corresponding *ControlAgent* to get the data from the GECKO, and the sensors display them in the interface (Figure 8.5). The *InterfaceHAWKAgent* communicates with its corresponding *ControlAgent* to get the data from the HAWK and sensors display them in the interface (Figure 8.4). Finally, the *TeamInterface* interacts with the *ControlAgent* of all the vehicles and teams and with the *MovementControllerAgent* to get and show the position of all the teams working at every moment. Moreover, it receives an alert from the *AlertColisionAgent* if two vehicles will collide or from the *MovementControllerAgent* in case two different teams explore the same area. Other alerts can be also configured.

*ControlAgent*: this agent runs on the node physically located inside the vehicle. It encapsulates the logic and functionality needed when controlling the specific vehicle. It is based on information received from the sensor agents in conjunction with possible control commands arriving from the interface agents. Therefore, this agent offers a control service to handle the movements of the vehicle. It uses the information obtained by the *SensorAgents*. In the same way that the *SensorAgent*, it has two specializations, the *ControlHAWKAgent* and the *ControlGECKOAgent*.

*SensorAgent*: this agent role refers to each of the vehicle sensors participating in the system. It runs in the node located in the robot vehicle, and may vary depending on the purpose of practical case or vehicle (*SensorHAWKAgent* or *SensorGECKOAgent*). Their mission is to measure environmental information for further interpretation. The most important tasks of the *SensorAgent* used by the system are:

- It is in charge of picking up the wireless signal that reaches the system and avoiding loss of connection.

- Using the sonar, it is in charge of detecting and reporting any obstacles in its path in order to, for example, avoid possible collisions and analyze possible objectives.
- It uses a GPS network to locate the vehicle on the map. The objective is to know its location at all times. This behaviour, which is performed autonomously, supports searches conducted between waypoints. If the GPS gets lost, this agent sends an alert to stop the movement of the robots in its team.
- It carries a magnetometer to indicate the direction of a vehicle using the North Magnetic Pole as a point of reference. This allows predicting the path taken by the vehicle. This behaviour is required to autonomously track the paths.

*MovementControllerAgent*: this role is deployed in any node of the system. It is responsible for performing the calculations for the movement of GECKO and HAWK controlling that the scanned area by HAWK at different consecutive moments does not overlap more than a percentage set by the user in the *TeamInterface*.

*ScanningRouteAgent*: this role is deployed in any node of the system and is responsible for calculating the path that the multirrotor must follow to reach the waypoints. It is possible to configure the speed to reach the points, the height, or a timeout at the corresponding point. Thus, it is possible to explore an area from the air autonomously. This agent receives information from the *MovementControllerAgent* about the horizontal and vertical displacements that HAWK must move to reach the waypoints  $WPN_n$  of the image 8.7. In section 8.5.1 the needed calculations will be explained in detail. Once the displacements are known, this agent configures the  $WPN_n$  to carry out the complete scanning movement  $SM$ , from  $WP1_a$  to  $WP9_a$  according to the Figure 8.7.

*RouteAreaAgent*: this role is deployed in any node of the system. It is responsible for controlling the movements done by the two robots (with their possible errors and deviations) and stores the information until the end of the mission. Then, it presents the results of the flight. For this, it communicates with the *TeamInterface*.

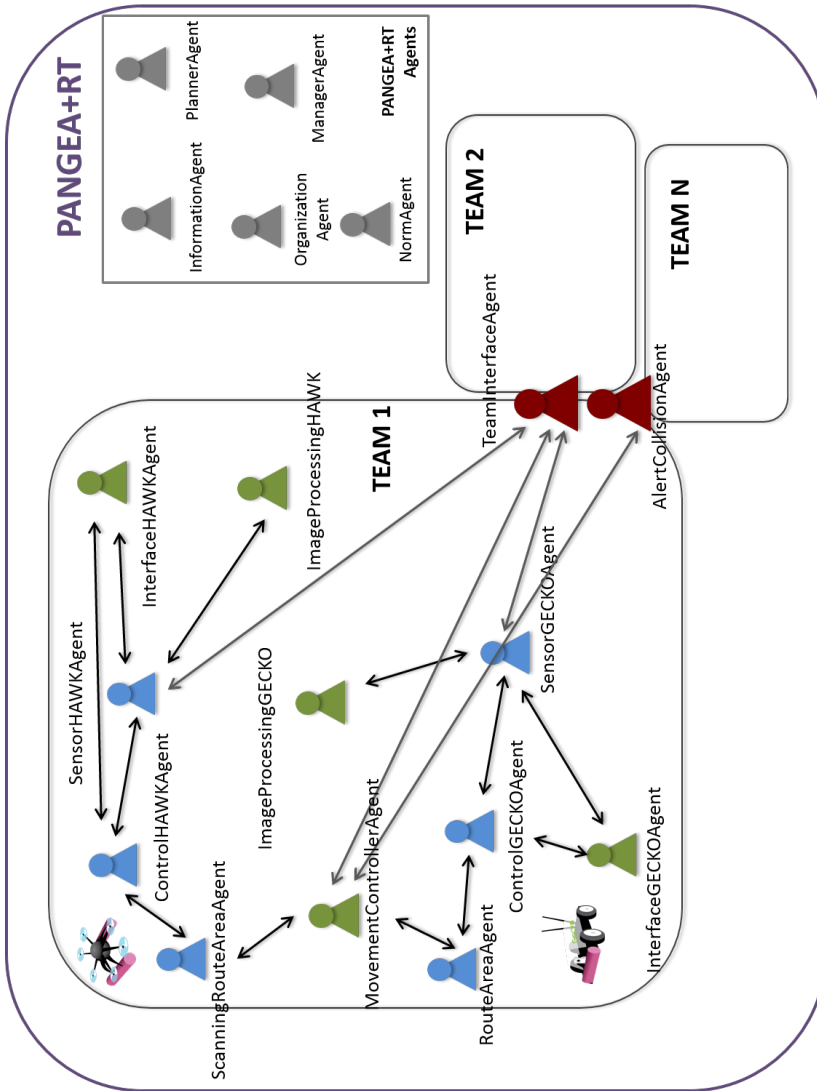


Figure 8.6: Proposed VO of agents

There are also two mandatory roles for each team in charge of the study of the images transmitted by HAWK and GECKO. However, as mentioned, this particular study is presented in [67]. This work is more oriented to rescue tasks, while for surveillance tasks, personnel have devices with the mentioned interfaces where they can view the images that HAWK and GECKO transmit.

*ImageProcessorAgent*: this agent is responsible for recognizing human patterns

in the images. There are also two specializations of this role. For HAWK, the *ImageProcessingHAWKAgent* receives the images from the air with a value  $alpha = 0$ . For GECKO, the *ImageProcessingGECKOAgent*, which has the camera in the front of the UGV, transmits the images with  $alpha = 90$  according to the perspective projection of the Figure 8.9. The images are transmitted from the camera to the agent using the First-person View (FPV), also known as Remote-person View (RPV), a method used to control a radio-controlled vehicle from the driver or pilot's view point [266] [376].

In the system there is also an agent that communicates with the *MovementControllerAgent* of each group. This agent called *AlertColisionAgent* receives the coordinates for where HAWK will be positioned for the next scan. If another HAWK has previously scanned a configurable percentage of that area, the *AlertColisionAgent* launches an alert for a new calculation.

## 8.5

### Collaboration description

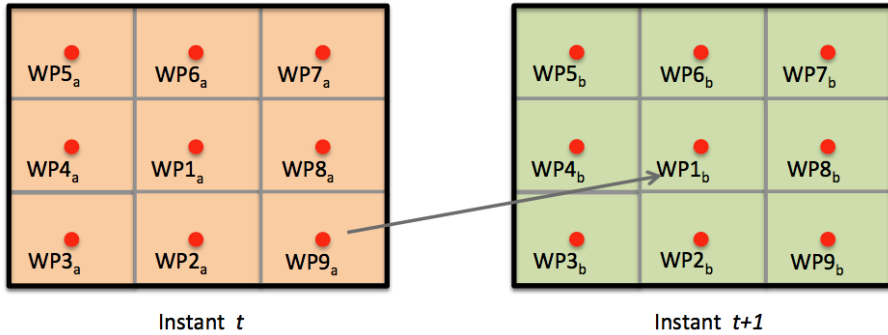
---

This section will show, in general terms, the coordinated movements performed by two robots. Then we will provide information about the agents that formed the team and made the collaboration between all entities of the team possible. The purpose of the team, or the different teams, is to cover an area with predefined routes set by the security personnel. Currently, due to the large area to check, the personnel have to walk all over the park after the closing time. This is a waste of time and human resources that our proposed system can avoid. At the same time, GECKO will be taking front images while HAWK scans the area surrounding GECKO. As explained in section 8.5.1, this area will depend on the flight's altitude and the characteristics of the camera installed in HAWK.

GECKO covers the ground and can be configured in two ways. As it has a front camera, it can be run by the surveillance personnel through the camera and the control software (Figure 8.5). Alternatively, waypoints can be specified to the agent in charge of the autonomous movements of GECKO using the mentioned control software. This agent, *ScanningRouteAgent*, will be explained in section 8.5.1.

HAWK flies automatically over the area surrounding GECKO. The HAWK is always autonomous but alerts can be sent for an emergency stop. The same alarm system controls possible collisions with other teams. If the connection is lost, HAWK stops automatically; and if the disconnection persists, HAWK automatically returns to the starting point of the flight.

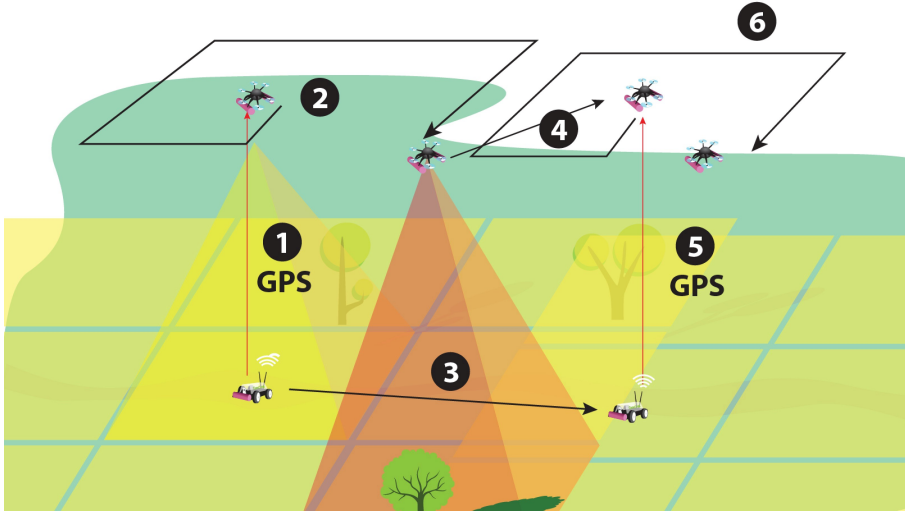
The Figure 8.7 shows the flight path of HAWK and what will now be referred to as the scanning movement,  $SM$ , formed by nine waypoints ( $WP1_a, \dots, WP9_a$ ), all of them calculated automatically by the *ScanningRouteAgent* using the information provided by the *MovementControllerAgent*. Each small rectangle of the image shows the field of view from the waypoints more distant that HAWK can reach, in this way, the combination of the 9 rectangles will form the field of view after performing the the  $SM$ .



**Figure 8.7:** Waypoints that must follow HAWK

The main steps of the process are presented in Figure 8.8. The image is also simplified since all the agents involved in the process are omitted (in the section, these agents will be presented). These steps are:

1. GECKO stops in a GPS coordinate and shares this coordinate with HAWK. Then, HAWK flies to that position until arriving just over GECKO.
2. HAWK scans the area around the initial GPS coordinate (sent by GECKO) to take images
3. While the previous is performing, GECKO begins to move to the next stop point. This stop corresponds to a GPS coordinate which is calculated by the *MovementControllerAgent*, which will be describe in section 8.5.1.
4. HAWK starts its movement towards the place where GECKO is located.
5. HAWK receives the exact new start position where it carry out a new scanning movement,  $SM$ .
6. HAWK repeats the  $SM$  over the surroundings of GECKO.



**Figure 8.8:** General steps of the coordinated movement

As observed in the experiments, the maximum speed of the team is largely determined by the meteorological conditions, since the flying capabilities of HAWK get worse with adverse environmental conditions. It is also reduced by the rectangular *SM* performed by HAWK in each cycle.

Both HAWK and GECKO send the images that they have taken to the agents in charge of analyzing them. These agents will play the role *ImageProcessorAgent* (see section 8.4.1). This step can be very useful for locating people more quickly during search efforts in large areas. The work related to the analysis of the images is not the object of this dissertation but it can be consulted in [67].

### 8.5.1

#### Calculations for the collaborative movement

The algorithm 4 shows the behavior of the agent that plays the role *MovementControllerAgent* during the execution of the surveillance task. The initial parameters that it receives are: the GECKO's position (in a GPS measurement) through its *SensorAgent*, the HAWK's height, also through its *SensorAgent*, and a percentage value of the maximum area that the system will allow to overlap between two consecutive *SM*. This value is chosen by the personnel and get through the *TeamInterfaceAgent*. This algorithm is in permanent execution while GECKO is moving. For each iteration of the loop, the agent requests the GECKO's position from the *SensorAgent*. At the end, a stop



command is sent to the *RouteAgent* of GECKO. This command includes the actual GPS position in which GECKO has to stop and HAWK has to start the next scan. Likewise, this position is sent to agent *ScanningRouteAgent* HAWK to take the position as the center of the next scanning movement.

---

**Algorithm 4** Procedure MovementControllerAgent
 

---

```

1: procedure CALCULATENEWPOSITION( $GPSLat_i, GPSLong_i,$ 
    $HAWKHeight, threshold$ )
2:   repeat
3:      $(GPSLat_f, GPSLong_f, angle) \leftarrow getPositionGECKO()$ 
4:      $percCommonArea \leftarrow commonAreaCalculation(GPSLat_i,$ 
    $GPSLong_i, GPSLat_f, GPSLong_f, HAWKHeight, angle)$ 
5:   until  $percCommonArea > threshold$ 
6:    $sendStopRouteArea(GPSLat_f, GPSLong_f)$ 
7:    $sendScannigRouteAreaAgentPosition(GPSLat_f, GPSLong_f)$ 
8: end procedure

```

---

The algorithm 5 presents the calculations needed to obtain the percentage of the overlapped area between two consecutive positions.

---

**Algorithm 5** Common Area Calculation
 

---

```

1: procedure COMMONAREACALCULATION( $GPSLat_i, GPSLong_i,$ 
    $GPSLat_f, GPSLong_f, HAWKHeight, angle$ )
2:    $C_i \leftarrow cornersProjection2Dto3D()$ 
3:    $scannedArea \leftarrow areaCalculation(C_i)$ 
4:    $(\Delta x_G, \Delta y_G) \leftarrow GPSToXY(GPSLat_i, GPSLong_i, GPSLat_f, GPSLong_f)$ 
5:    $C_f \leftarrow cornersProjection2Dto3D(\Delta x_G, \Delta y_G, angle)$ 
6:    $crossPoints \leftarrow rectangleCrossPoints(C_i, C_f)$ 
7:    $convexValues \leftarrow convexHull(C_i, C_f)$ 
8:    $commonArea \leftarrow polygonArea(convexValues)$ 
9:    $percCommonArea \leftarrow commonArea / scannedArea$ 
10:  return  $percCommonArea$ 
11: end procedure

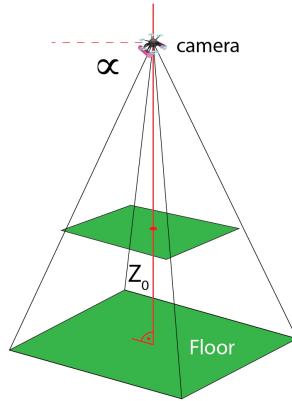
```

---

8.5.1.1

## Corners projection

The first step is to obtain the images captured in different consecutive instants of time. The images are captured by an onboard camera facing the ground at an angle  $\alpha$  and height  $Z_0$ , as shown in Figure 8.9.



**Figure 8.9:** Camera placement

To calculate the common area it is first necessary to calculate the four corners that form the field of view that HAWK reaches after its autonomous scanning movement  $SM$  (Figure 8.7). To calculate the real position of each point, it is necessary to know the intrinsic and extrinsic parameters of the camera. Using descriptive geometry (8.1, 8.2) the position in the space of a pixel in the image is calculated. For this, we take a known dimension. All the points of the images are considered in the surface of the ground, ie,  $Z = 0$ .

$$Y = \frac{((\lambda_y \cdot Z_0 \cdot \phi) + (y_i \cdot \rho \cdot Z_0)) + (y_i \cdot \rho)}{((\lambda_y \cdot \rho) - (y_i \cdot \phi))} \quad (8.1)$$

$$X = \frac{(x_i \cdot Y \cdot \phi) + (x_i \cdot Z_0 \cdot \rho) + (x_i \cdot \lambda_X)}{\lambda_X} \quad (8.2)$$

where  $(x_i, y_i)$  is the point of the images, measured in millimeters, from which the projection will be calculated inside the size of the CCD of the camera;  $(\lambda_x, \lambda_y)$  is the focal length of the camera;  $Z_0$  is the height in millimeters where the camera of the HAWK is located;  $\phi$  y  $\rho$  are the  $\sin \alpha$  and the  $\cos \alpha$ , respectively;  $\alpha$  is the orientation angle of the camera. In our case, the HAWK camera is always oriented to the ground, then,  $\alpha = 0$ . After these calculations, we know the four corners that form the HAWK's field of view after its autonomous movement. In Figure 8.10 the points are  $(C_1, C_2, C_3, C_4)$ . These are stored in the vector  $C_i$ .

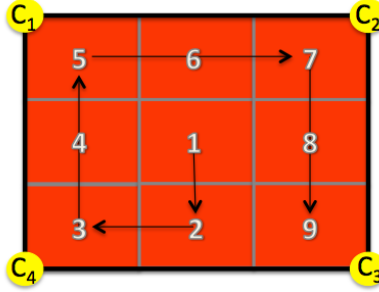


Figure 8.10: Corners of the field of view

The corners that form the field of view in the case of HAWK have to be placed in the position where GECKO is after the movement. They are stored in  $\mathcal{C}_f$ . Because this position is determined by GPS coordinates, they must previously be transformed to Cartesian coordinates using the initial point of GECKO as the origin. The HAWK movement is determined by the values  $(\Delta x_G, \Delta y_G)$  and the change of orientation suffered by GECKO stored in *angle*.

The new field of view will have the same area as the previous one, if the height of flying HAWK is maintained. With the advancement and possible rotation of GECKO, the coordinates of the final corners must be recalculated according to the starting point. To do so, the corresponding translation and rotation matrix must be applied (8.3).

$$\mathcal{C}_f = \begin{bmatrix} x_f \\ y_f \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\text{angle}) & -\sin(\text{angle}) & \Delta x \\ \sin(\text{angle}) & \cos(\text{angle}) & \Delta y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \quad (8.3)$$

Therefore, the end points of the corners used in the coordinate system, which is centered in the initial position of GECKO, are calculated according to the equations (8.4) y (8.5).

$$x_f = x_i \cos(\text{angle}) - y_i \sin(\text{angle}) + \Delta x \quad (8.4)$$

$$y_f = x_i \sin(\text{angle}) + y_i \cos(\text{angle}) + \Delta y \quad (8.5)$$

---

**8.5.2** Common area calculation
 

---

Once the limits of the field of view in both positions are known, the next step is to calculate the points that are visible in both fields. The algorithm 6 presents the steps to calculate the intersections between the two fields of view. To do this, the points where the lines delimiting each one of rectangles (fields of views) intersect must be calculated.

---

**Algorithm 6** Rectangle Cross Points
 

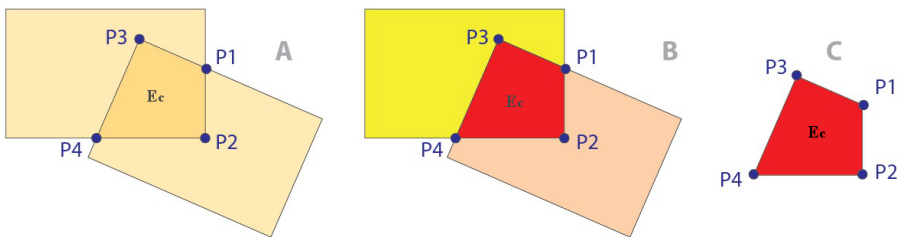
---

```

1: procedure RECTANGLECROSSPOINTS( $\mathcal{C}_i, \mathcal{C}_f$ )
2:   for  $c_i = \mathcal{C}_i[1]$  to  $\mathcal{C}_i[3]$  do
3:      $\mathcal{S}_i \leftarrow \text{segment}(c_i, \text{next}(c_i))$ 
4:     for  $c_f = \mathcal{C}_f[1]$  to  $\mathcal{C}_f[3]$  do
5:        $\mathcal{S}_f \leftarrow \text{segment}(c_f, \text{next}(c_f))$ 
6:        $\mathcal{C}_p \leftarrow \text{segmentCross}(\mathcal{S}_i, \mathcal{S}_f)$ 
7:       if  $\mathcal{C}_p$  then
8:          $E_c \leftarrow \text{cornersInsideP}(\mathcal{C}_i, \mathcal{C}_f, \mathcal{C}_p)$ 
9:       end if
10:    end for
11:  end for
12:   $\text{crossPoints} \leftarrow \{\mathcal{C}_p\} + \{E_c\}$ 
13:  return  $\text{crossPoints}$ 
14: end procedure

```

---



**Figure 8.11:** Corners of the field of view

Two consecutive corners form a linear segment delimiting that field. For each segment of the initial field of view,  $\mathcal{S}_i$ , the crossing points are checked with all the segments of the other rectangle of view,  $\mathcal{S}_f$ ; in Figure 8.11, the points are  $P_1, P_2$ . These points are stored in the vector  $\mathcal{C}_p$ . If there is a cross between two segments, we must determine which segment extremes form the common area,  $E_c$ , in figure 8.11 the points are  $P_2, P_3$ . The result of this process is a

cloud of points that includes the crossing points and the crossing corners of each rectangle that belong to the common area, *crossPoints*.

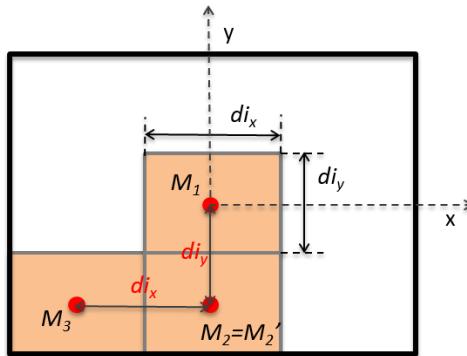
To obtain the polygon  $\mathcal{P}$  with the maximum area formed by a cloud of points, the function of convex hull [98] is used.  $\mathcal{P}$  describes the common points in both fields of views. Later, the area of  $\mathcal{P}$  is calculated, *commonArea*. And finally, the only thing left to calculate is the percentage of the common area with respect to the total scanned area at the initial moment, *percCommonArea* (8.7).

$$\text{percCommonArea} = \frac{\text{commonArea}}{\text{scannedArea}} \cdot 100\% \quad (8.6)$$

Calculation of horizontal and vertical movement for the HAWK waypoints

### 8.5.3

To indicate the ScannigRouteArea, that is, the horizontal and vertical displacement that enables configuring the waypoints of HAWK's navigation, we should again take the following equations into account: (8.1, 8.2). As shown in Figure 8.12, if the rectangle represents the dimensions of the CCD of the camera ( $di_x, di_y$ ), then the displacement from the midpoint  $\mathcal{M}_1$  of the rectangle to the next midpoint  $\mathcal{M}_2$  of the bottom rectangle is equal to  $di_y$ . The same happens for the horizontal displacement, the distance from  $\mathcal{M}_3$  to  $\mathcal{M}_4$  is equal to  $di_x$ .



**Figure 8.12:** Dimensions of the CCD in relation to the displacement of HAWK

Therefore, knowing these measures in the image, the projection can be calculated with (8.1, 8.2). For the vertical distance, and using the center of the coordinates

at point  $\mathcal{M}_1$  where the movement  $SM$  starts, we have  $\mathcal{M}_1 = (x_1, y_1) = (0, 0)$ . Applying the mentioned equations, the new values of the point in the projection are:

$$Y_1 = \frac{\lambda_y \cdot Z_0 \cdot \phi}{\lambda_y \cdot \rho} = \frac{Z_0 \cdot \phi}{\rho} \quad (8.7)$$

$$X_1 = \frac{0}{\lambda_y} = 0 \quad (8.8)$$

The second point corresponds to the point  $\mathcal{M}_2 = (x_2, y_2) = (0, di_y)$ , in the projection according to the equations:

$$Y_2 = \frac{((\lambda_y \cdot Z_0 \cdot \phi) + (di_y \cdot \rho \cdot Z_0)) + (di_y \cdot \rho)}{((\lambda_y \cdot \rho) - (di_y \cdot \phi))} \quad (8.9)$$

$$X_2 = \frac{0}{\lambda_y} = 0 \quad (8.10)$$

Therefore, the vertical distance  $yDist$  between  $\mathcal{M}_1$  and  $\mathcal{M}_2$  is:

$$\begin{aligned} yDist &= \sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2} = \sqrt{0 + (Y_2 - Y_1)^2} = Y_2 - Y_1 = \\ &= \frac{((\lambda_y \cdot Z_0 \cdot \phi) + (di_y \cdot \rho \cdot Z_0)) + (di_y \cdot \rho)}{((\lambda_y \cdot \rho) - (di_y \cdot \phi))} - \frac{Z_0 \cdot \phi}{\rho} \end{aligned} \quad (8.11)$$

where  $di_y$ ,  $\lambda_y$ ,  $\phi$  and  $\rho$  are constants while the camera does not change, and then, the distance only depends on the height  $Z_0$  of the HAWK flying,  $yDist = f(Z_0)$ .

For the horizontal distance, as observed in Figure 8.12, the horizontal displacement corresponds to the distance from  $\mathcal{M}_2$  to  $\mathcal{M}_3$ , which is equal to  $di_x$ . Considering the center of the coordinates again where the movement starts, in the point  $\mathcal{M}_2$ , we have  $\mathcal{M}_2 = (x_2', y_2') = (0, 0)$ . Applying the mentioned equations, the new values of the point in the projection are:

$$Y_2' = \frac{\lambda_y \cdot Z_0 \cdot \phi}{\lambda_y \cdot \rho} = \frac{Z_0 \cdot \phi}{\rho} \quad (8.12)$$

$$X_2' = \frac{0}{\lambda_y} = 0 \quad (8.13)$$

Then, the following point would be the point corresponding to the point  $\mathcal{M}_3 = (x_3, y_3) = (di_x, 0)$  in the projection, according to the equations::

$$Y_3 = \frac{Z_0 \cdot \phi}{\rho} \quad (8.14)$$

$$X_3 = \frac{\left(di_x \cdot Z_0 \cdot \frac{\phi^2}{\rho}\right) + (di_x \cdot Z_0 \cdot \rho) + (di_x \cdot \lambda_X)}{\lambda_X} \quad (8.15)$$

Therefore, the horizontal distance  $xDist$  between  $\mathcal{M}_2$  and  $\mathcal{M}_3$ , taking into account  $Y_2' = Y_3$ , is:

$$\begin{aligned} xDist &= \sqrt{(X_3 - X_2')^2 + (Y_3 - Y_2')^2} = \sqrt{X_3^2 + (Y_3 - Y_2')^2} = X_3 = \\ &= \frac{\left(di_x \cdot Z_0 \cdot \frac{\phi^2}{\rho}\right) + (di_x \cdot Z_0 \cdot \rho) + (di_x \cdot \lambda_X)}{\lambda_X} \end{aligned} \quad (8.16)$$

where  $di_x$ ,  $\lambda_y$ ,  $\phi$  and  $\rho$  are constants and the horizontal distance only depends on the height  $Z_0$  of the flight of HAWK,  $xDist = f(Z_0)$ .

Once we know the functioning and responsibilities of the roles involved in the case study, in the next section we will explain the hardware specifications for the deployment of the robot teams.

## 8.6

## Deployment of the involved agents

For the deployment of the system, we have three different types of nodes with the following characteristics:

- Type 1: Intel Core 2 Duo at 3.00 GHz and 4Gb RAM. As operating system, it has a Fedora 20 kernel 3.10.22 and the real-time patch RT-PREEMPT (patch-3.10.22.rt20). The RTJVM is the JamaicaVM Personal Edition 6.2.
- Type 2: i7-3630QM at 3.40 GHz and 8Gb RAM. As operating system, it has a Fedora 19 kernel 3.12.11 and the real-time patch RT-PREEMPT (patch-3.12.11.rt19). The RTJVM is the JamaicaVM Personal Edition 6.2.

- Type 3: Raspberry Pi Single Board Computer (SBC) at 700MHz processor and 512MB RAM with a Raspbian kernel 3.12.1 and the real-time patch RT-PREEMPT (patch-3.12.1-rt4). The RTJVM is the FijiVM Academic.

We have 10 nodes, the type of each node can be seen in the table 8.1.

<b>Node 1</b>	Type 3 (Raspberry Pi)
<b>Node 2</b>	Type 3 (Raspberry Pi)
<b>Node 3</b>	Type 2 (Intel i7-3630QM)
<b>Node 4</b>	Type 1 (Intel Core 2 Duo)
<b>Node 5</b>	Type 1 (Intel Core 2 Duo)
<b>Node 6</b>	Type 3 (Raspberry Pi)
<b>Node 7</b>	Type 3 (Raspberry Pi)
<b>Node 8</b>	Type 2 (Intel i7-3630QM)
<b>Node 9</b>	Type 1 (Intel Core 2 Duo)
<b>Node 10</b>	Type 1 (Intel Core 2 Duo)

**Table 8.1:** Relation between the nodes and their characteristics

The agents related to the interfaces of the team or the vehicles (*TeamInterfaceAgent*, *InterfaceHAWKAgent*, *InterfaceGECKOAgent*) are executed in independent nodes which are not taken into account since the hard real-time constraints are not applied to the interface applications. The agents of PANGEA+RT itself are executed in another node which is specifically reserved because it is not necessary that these agents interact in real-time. As we explained in the chapter 7, these agents are: the *OrganizationAgent*, the *ManagerAgent*, the *NormAgent*, the *InformationAgent* and the *ServiceAgent*. Nevertheless, the *PlannerAgent* is included in the real-time environment in order to bound its execution time (the MAP algorithm and the Branch and Bound methods are included in the proposed model to reduce the execution time). Its purpose is to generate a new planning and task allocation when a change occurs, i.e., a new agent enters the organization, an agent wants to change its role, a node fails, etc. For this, the *PlannerAgent* uses the remaining computational capacity of any of the available nodes.

In the proposed model, as we explained in chapter 4, we use the basic schedulability condition for the RM proposed by Liu and Layland [221], which is taken into account in the constraint 2 of the model for the task allocation. The schedulability bound of RM decreases with the number of tasks,  $N$ , as Butazzo [63] demonstrate in 8.17.

$$\lim_{N \rightarrow \infty} N(2^{1/N} - 1) = \ln 2 \simeq 0.693 \quad (8.17)$$



This means that the capacity of any node is 69.3% applying our model. When the *PlannerAgent* needs to execute, it uses the remaining percentage of one node. In the next chapter, we will present the results related to the rest of the agents and nodes involved in the case study.

## 8.7

## Conclusions

---

In this chapter we have presented the case study with the vehicles and the agents involved. The vehicles HAWK and GECKO were developed by the BISITE research group and together with the proposed model included in the *PlannerAgent*, another innovative aspect of this proposal is the mechanism developed for their collaboration. All the collaboration was carried out using agents integrated in different VOs and using the infrastructure provided by PANGEA+RT.

We include a basic design of VOs which is limited by the number of available robots and in chapter 10, we include the study of more complex VOs as future work.

This is a real case study since the system can be used for surveillance tasks as we have demonstrated in the Valcuevo leisure park. In the next chapter, we will show the results obtained when the complete system is executing using the proposed model and the PANGEA+RT platform .



# 9

## RESULTS OF THE PROPOSED MODEL

---

*In this chapter, we present the results related with the proposed method and the case study previously explained. The chapter is composed of three main sections, in first place we present some general tests carried out to verify the model for the WCET estimation presented in the section 11.3 of the chapter 4. We dedicated a special section to this with an easier example using the Bubble Sort Algorithm because the generated graphs in the case study are impossible to represent in an image that fits in this document. Later, we include the results obtained when we applied the model with the agents proposed in the case study for the collaboration of heterogeneous robots for surveillance tasks presented in the chapter 8. And as the third part, we show some results of the mathematical collaboration model for scanning areas. This model was also presented in chapter 8 but, specifically in section 8.5. Finally, as conclusions we present some qualitative advantages that the PANGEA+RT provided for the development of the whole case study.*

### Contents

---

<b>9.1</b>	<b>General evaluation of the model . . . . .</b>	<b>216</b>
9.1.1	WCET evaluation . . . . .	216
<b>9.2</b>	<b>Case study results . . . . .</b>	<b>225</b>
<b>9.3</b>	<b>Results of the collaboration search . . . . .</b>	<b>229</b>
<b>9.4</b>	<b>Conclusions . . . . .</b>	<b>232</b>

---

## 9.1 General evaluation of the model

In this section, we have to distinguish two different subsections. The first one is devoted to the evaluation of the WCET predicted with the proposed model and the second part shows the results of the task allocation model.

### 9.1.1 WCET evaluation

For the node evaluation, we use tree nodes each one corresponding with the three types presented in the previous chapter 8. This means node 1 is the type 1, node 2 is type 2 and node 3 is type3.

With the designed program described in chapter 4 we perform the test to obtain the time execution in nanoseconds of each bytecode. In the tables 9.1 and 9.2, we present the results in nanoseconds corresponding to the most used bytecodes in each node. To get these results each bytecode has been executed in 1000 blocks of 5000000 executions per block. The column "Error Rate" corresponds with maximum difference in nanoseconds between the 1000 blocks of executions.

The results of the execution time of some bytecodes are presented in a comparative graph (Figure 9.1) corresponding to the data in tables 9.1 and 9.2. The corresponding bytecode is represented in the x-axis of the graph and the time in nanoseconds taken to execute it in the y-axis .

Bytecode	Node 1	Node 2	Error Rate
iconst_[0-5]	4,7126 ns	3,1612 ns	0,01 ns
lconst_[0-5]	6,9756 ns	4,2986 ns	0,01 ns
lcmp	44,3769 ns	29,2526 ns	0,01 ns
dup	6,0662 ns	3,1816 ns	0,01 ns
bipush	7,3112 ns	5,3088 ns	0,01 ns
sipush	10,9586 ns	8,0544 ns	0,03 ns
dup_x[1-2]	30,5263 ns	17,3002 ns	0,02 ns
if_icmpgt	15,8049 ns	18,5756 ns	0,04 ns
ddiv	56,2395 ns	32,2563 ns	0,04 ns
pop	6,5632 ns	4,9568 ns	0,01 ns
if_eq	16,4586 ns	12,3689 ns	0,03 ns
frem	72,2365 ns	44,287 ns	0,04 ns

**Table 9.1:** Bytecode execution in Nodes 1 and 2

Bytecode	Nodo 3	Error Rate
iconst_[0-5]	25,0796 ns	0,05 ns
lconst_[0-5]	38,0464 ns	0,06 ns
lcmp	195,0205 ns	0,07 ns
dup	32,8319 ns	0,03 ns
bipush	39,0198 ns	0,03 ns
sipush	58,0805 ns	0,04 ns
dup_x[1-2]	160,8704 ns	0,09 ns
if_icmpgt	82,7686 ns	0,09 ns
ddiv	301,813 ns	0,05 ns
pop	35,0369 ns	0,05 ns
if_eq	87,6744 ns	0,06 ns
frem	332,0282 ns	0,05 ns

Table 9.2: Bytecode execution in Node 3

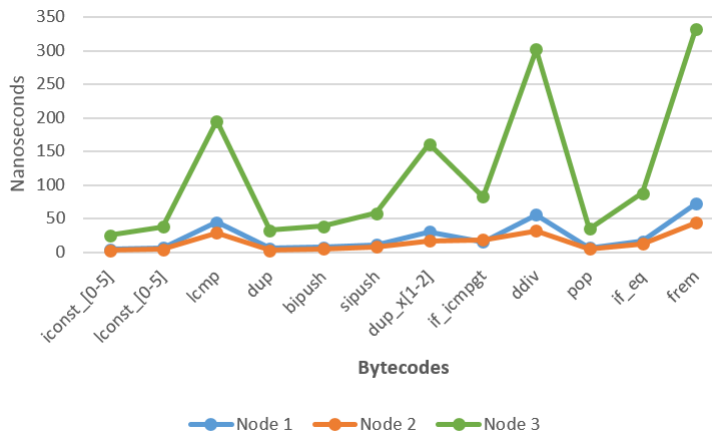


Figure 9.1: Time execution of some bytecodes in node 1, 2 and 3

Now, we evaluate a piece of code of the Bubble Sort Algorithm. First, we present the Java code used (9.1).

```

1 public static void test(){
2     int [] a = new int [10];
3     for (int ind=0;ind<10;ind++){
4         a[ind] = randInt (0,10);
5     }
6     int i, j, v1, v2;
7     timel = System.currentTimeMillis();
8     for (i=9;i>0;i--){
9         for (j=1;j<=i;j++){
10            v1 = a[j-1];
11            v2 = a[j];

```

```

12     if (v1 > v2){
13         a[j] = v1;
14         a[j-1] = v2;
15     }
16 }
17 }
18 time2=System.currentTimeMillis();
19 test = test + time2 - time1;
20 }

```

**Programming Code 9.1:** Test in Java

An example of some executions in the node 2 gives the following results (Figure 9.3). In the first column the time of the best case is presented, in this case the 10 element-array is fully ordered and the code included in the if-comparison is never executed. In the second column the time of the worst case is presented. This case happens when the 10 elements array is fully disordered and the code included in the if-comparison is always reached. In the last column, we present some executions with the array randomly initialized. Each row correspond with the WCET of 5000 tests.

Test	Best Case	Worst Case	Random Case
WCET 1	5589 ns	8695 ns	7475 ns
WCET 2	5574 ns	8654 ns	7436 ns
WCET 3	5584 ns	8692 ns	7391 ns
WCET 4	5500 ns	8670 ns	7333 ns
WCET 5	5524 ns	8692 ns	7491 ns
WCET 6	5568 ns	8668 ns	7455 ns
WCET 7	5562 ns	8679 ns	7500 ns
WCET 8	5559 ns	8690 ns	7499 ns
WCET 9	5578 ns	8682 ns	7498 ns
WCET 10	5529 ns	8687 ns	7402 ns
<b>Error Rate</b>	0.089 ns	0.041 ns	0.166 ns

**Table 9.3:** Example of some executions

In table 9.4, we show a brief description of the bytecodes included to understand the flow and in the second column the WCET estimation in the node 2 (necessary to the next step) is presented.

Bytecodes	WCET Node 2	Description
invokestatic	171 ns	invoke a static method, where the method is identified by method reference index in constant pool (in the example #14)

putstatic	110.5158 ns	set static field to value in a class, where the field is identified by a field reference index in constant pool (in the example #15)
bipush	5.3088 ns	push a byte onto the stack as an integer value (in the example value 9 at the beginning of the loop)
istore_1	5.9975 ns	store int value into variable 1
iload_1	4.3539 ns	load an int value from local variable 1
ifle	16.7540 ns	if value is less than or equal to 0, branch to instruction at branchoffset (in the example the offset is +50, to the instruction 88 that finish the initial for-loop)
icons_1	3.1612 ns	load the int value 1 onto the stack
istore_2	5.9975 ns	store int value into variable 2
if_icmpgt	26.5756 ns	if value1 is greater than value2, branch to instruction at branchoffset (in the example, the offset is 37 which lead to the instruction 82, the end of the second for-loop)
aload_0	6.0978 ns	load a reference onto the stack from local variable 0
if_icmple	22.2732 ns	if value1 is less than or equal to value2, branch to instruction at branchoffset (in the example, the offset is 14 to reach the instruction 76, the end of the if-comparison)
iinc	10.8821 ns	increment local variable #index by signed byte const (next iteration of the loop)
goto	4.1158 ns	goes to another instruction at branchoffset (in the example, -36 to reach the second for-loop or -48 to the first for-loop)

**Table 9.4:** Bytecodes involved in the example

With this data, we can build the annotated control flow graph (*aCFG*). The arcs contain labels showing how often the between the two basic block (vertex), *BB* are taken. If  $N$  is size of the array and  $N = 10$ , then the number of iterations of the outer for-loop is one less than the array size:  $F_1 = N - 1 = 9$ .

The inner for-loop is executed (9.1):

$$F_2 = \sum_{i=1}^{F_1} i = \frac{F_1(F_1 + 1)}{2} \quad (9.1)$$

The figure 9.2 corresponds to the generated bytecodes after the initialization of the array. It can be seen the two calls to the method `System.currentTimeMillis()` at the beginning and at the end of the ordering process. In the example,  $F_2 = 45$  times. The outer for-loop consist on the BBs from  $B_3$  to  $B_8$  and the inner `for_loop` consist on the BBs from  $B_5$  to  $B_7$ .

```

Specific info:
Bytecode Exception table Misc Code Editor
17 28 invokestatic #14 <java/lang/System/currentTimeMillis()J>
18 31 putstatic #15 <PruebaRandom/tiempol J>
19 34 bipush 9
20 36 istore_1
21 37 iload_1
22 38 ifle 88 (+50)
23 41 iconst_1
24 42 istore_2
25 43 iload_2
26 44 iload_1
27 45 if_icmpgt 82 (+37)
28 48 aload_0
29 49 iload_2
30 50 iconst_1
31 51 isub
32 52 iaload
33 53 istore_3
34 54 aload_0
35 55 iload_2
36 56 iaload
37 57 istore_4
38 59 iload_3
39 60 iload_4
40 62 if_icmple 76 (+14)
41 65 aload_0
42 66 iload_2
43 67 iload_3
44 68 iastore
45 69 aload_0
46 70 iload_2
47 71 iconst_1
48 72 isub
49 73 iload_4
50 75 iastore
51 76 iinc 2 by 1
52 79 goto 43 (-36)
53 82 iinc 1 by 255
54 85 goto 37 (-48)
55 88 invokestatic #14 <java/lang/System/currentTimeMillis()J>

```

Figure 9.2: Screenshot of the bytecodes in the JBE tool

The figure 9.3 shows the corresponding graph. As explained in chapter 4, the



complexity of analysing a graph is exponential depending on the depth of the conditional statements. As this method can take a long time to calculate, we use the improvement called IPET (see section 4.3.2.2).

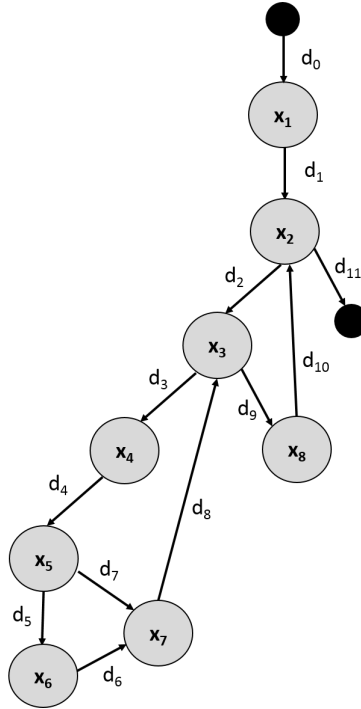


Figure 9.3: CFG of the example

We define the following ILP problem according to the constraints 1 and 2 and the objective function according to the equation 4.2. Then, the objective function is:

$$\begin{aligned}
 WCET_P = \max & (\tau_1 x_1 + \tau_2 x_2 + \tau_3 x_3 + \\
 & + \tau_4 x_4 + \tau_5 x_5 + \tau_6 x_6 + \tau_7 x_7 + \tau_8 x_8)
 \end{aligned}
 \tag{9.2}$$

where we can substitute the cost  $\tau_{BB}$  of each  $BB$  and the equation of the objective function is:

$$\begin{aligned}
 WCET_P = \max & (11.3063x_1 + 21.104x_2 + 35.60x_3 + 46.975x_4 + \\
 & 30.981x_5 + 46.975x_6 + 14.9921x_7 + 14.9921x_8)
 \end{aligned}
 \tag{9.3}$$

The structural constraints are:

$$\begin{aligned}
 x1 &= d0 ; x1 = d1 ; x2 = d1 ; x2 = d2 + d11 ; \\
 x3 &= d2 + d8 ; x3 = d3 + d9 ; x4 = d3 ; x4 = d4 ; \\
 x5 &= d4 ; x5 = d5 + d7 ; x6 = d5 ; x6 = d6 ; \\
 x7 &= d7 + d6 ; x7 = d8 ; x8 = d9 ; x8 = d10
 \end{aligned} \tag{9.4}$$

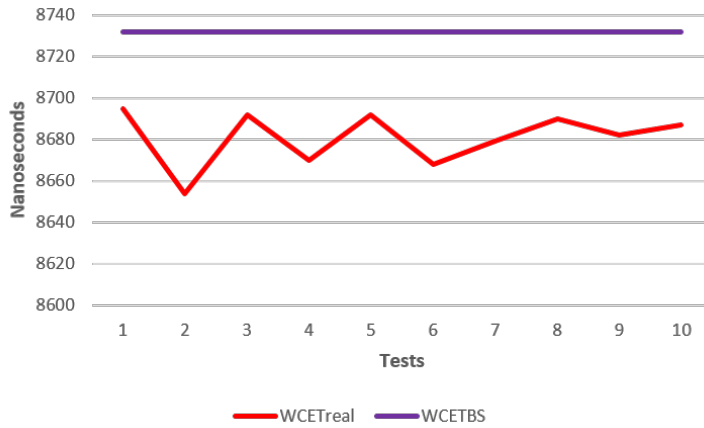
And finally, the functional constraints are:

$$d10 \leq 8 * d1 ; d8 \leq 8 * d2 \tag{9.5}$$

We solve this IPL problem with the Simplex Algorithm obtaining a value of  $WCET_P = 8732.35$  with corresponds with the nanoseconds of the worst execution time. The comparison between the estimated and the real execution time is shown in table 9.5 and figure 9.4.

Test	$WCET_{BS}$	Worst Case	Error Rate
WCET 1	8732 ns	8695 ns	0.037 ns
WCET 2		8654 ns	0.078 ns
WCET 3		8692 ns	0.040 ns
WCET 4		8670 ns	0.062 ns
WCET 5		8692 ns	0.040 ns
WCET 6		8668 ns	0.064 ns
WCET 7		8679 ns	0.053 ns
WCET 8		8690 ns	0.042 ns
WCET 9		8682 ns	0.050 ns
WCET 10		8687 ns	0.045 ns
<b>Average</b>	8732 ns	8681.5 ns	0.0505 ns

**Table 9.5:** Comparison between the real WCET execution and WCET estimation



**Figure 9.4:** Real WCET execution and WCET estimation

#### 9.1.1.1

#### Statistical adaptation of the WCET

The Bubble Sort Algorithm was used again to improve the estimated WCET with the statistical method. The execution time of this algorithm in the worst case, that is, with sorted in the opposite direction values, gives a value of 8695 ns. On the other hand the execution time in the best case, that is, with the elements sorted, was 5500. The execution time cannot be established beforehand because it depends on the data. The number of elements to be sorted was fixed at 100. The system generated 30 sets of random values, after which the bubble algorithm was applied. This process was repeated 30 times and the execution time was stored. The system then calculates the execution time of 30 new sets of values. The WCET was predicted using confidence intervals and atypical values. The average confidence interval with unknown variance was calculated using a t-student and a normal distribution. The WCET is established by the upper bound in the average confidence intervals with  $\alpha = 0.01$ .

The Tables 9.6 and 9.7 shows the obtained values. The t-student column shows the upper bound obtained from the last 30 execution times, the next column shows the difference with the real value. The WCET is similarly calculated using a normal distribution. The atypical column shows the WCET using atypical values. In this case, the system does not obtain negative values, which means that the system does not any have predictions out of range. However, if the system received a completely unordered list, this value would probably

have exceeded the estimated WCET; if several unordered lists arrived, the WCET would adapt to the new executions.

WCET (ns)	<i>t-student</i>	Difference	Normal	Difference
7416	7451.22896	35.2289595	7448.94947	32.9494676
7597	7448.84527	-148.154726	7446.60111	-150.398893
7422	7458.52503	36.5250273	7455.95101	33.9510149
7421	7459.37655	38.3765544	7456.81346	35.8134584
7472	7461.0146	-10.9854015	7458.53978	-13.4602191
7334	7460.1068	126.106803	7457.65367	123.653667
7393	7457.23689	64.2368862	7454.70778	61.7077768
7518	7453.02248	-64.9775214	7450.54945	-67.4505466
7429	7454.1112	25.1112041	7451.60319	22.6031942
7442	7451.0122	9.01220297	7448.56839	6.56839434
7355	7452.34366	97.3436615	7449.89683	94.8968284
7374	7449.89236	75.8923608	7447.4123	73.4123001
7372	7448.2587	76.2587041	7445.76357	73.763567
7340	7448.91851	108.918515	7446.55512	106.555119
7344	7446.31073	102.310734	7443.89478	99.8947828
7308	7445.00951	137.009507	7442.54455	134.544551
7617	7442.21846	-174.781536	7439.65513	-177.344868
7508	7452.89949	-55.1005132	7449.94929	-58.050706
7478	7458.15306	-19.8469365	7455.14628	-22.8537216
7445	7458.64914	13.6491394	7455.63197	10.6319699
7500	7461.50437	-38.4956316	7458.55239	-41.4476065
7469	7466.12817	-2.87182815	7463.15405	-5.84595025
7504	7468.55537	-35.4446262	7465.57356	-38.4264405
7491	7471.90587	-19.0941288	7468.87694	-22.1230591
7488	7474.97016	-13.0298444	7471.91752	-16.0824753
7430	7478.0407	48.0406992	7474.97577	44.9757664
7444	7479.96625	35.9662527	7477.02064	33.020638
7458	7481.16738	23.1673812	7478.23544	20.2354396
7435	7481.02036	46.020356	7478.08974	43.0897374

**Table 9.6:** Values of the statistical adaptation of the WCET (I)

Atypical	Difference
7668	252
7652	55
7668	246
7657.5	236.5

Atypical	Difference
7651.5	179.5
7651.5	317.5
7657.5	264.5
7637	119
7637	208
7603	161
7603	248
7641	267
7669.5	297.5
7665	325
7670.5	326.5
7679.5	371.5
7700.5	83.5
7726.5	218.5
7768.5	290.5
7768.5	323.5
7743	243
7751	282
7763	259
7781	290
7819	331
7839	409
7807.5	363.5
7807.5	349.5
7807.5	372.5

**Table 9.7:** Values of the statistical adaptation of the WCET (II)

## 9.2

## Case study results

As explained in the previous chapter and shown in figure 8.6, we have 9 agents per team and the *AlertCollisionAgent* that is the only one for both teams. We have as maximum 10 nodes of computation as explained in the previous chapter.

We have to add four more constraints per team. This is due to the node 1 corresponds to the Raspberry Pi of GECKO and the agents *ControlGECKOAgent* and *SensorGECKOAgent* must be executed in such node. Moreover, the node 2

corresponds to the Raspberry Pi of HAWK and the agents *ControlHAWKAgent* and *SensorHAWKAgent* must be executed in this node. The constraints are expressed in the equation 9.6.

$$\begin{aligned} x_1^1 = 1, x_2^1 = 1, x_3^2 = 1, x_4^2 = 1 \\ x_{10}^6 = 1, x_{11}^6 = 1, x_{12}^7 = 1, x_{16}^7 = 1 \end{aligned} \quad (9.6)$$

The figure 9.5 presents the utilization factor  $\rho_i^j$  where:  $i$  represents the task and  $j$  the computational node. The results are obtained with the proposed WCET estimation model. The value  $-1$  is assigned to the nodes where the task cannot be executed according to the equations 9.6.

With these values, we can obtain the results of the proposed task allocation model shown in the table 9.8. The nodes appear in the first column and the tasks allocated to such node in the second column. For the third column, we calculate final utilization factor of the node,  $\rho^j$ , using the values  $\rho_i^j$  of the previous figure 9.5 and with the formula 9.7.

$$\rho^j = \sum_i^{N_j} \rho_i^j \quad (9.7)$$

where  $N_j$  is the number of tasks allocated in the node  $j$ .

The last column corresponds with the remaining capacity of each node,  $\phi^j$ . The theorem 1 of Liu and Layland and the equation 4.10 is used to the calculation (formula 9.8).

$$\phi^j = \begin{cases} N_j(2^{1/N_j} - 1) - \rho^j & \text{if } N > 0 \\ 1 & \text{if } N = 0 \end{cases} \quad (9.8)$$

	Node 1		Node 2		Node 3		Node 4		Node 5		Node 6		Node 7		Node 8		Node 9		Node 10	
	Type 3	Type 3	Type 3	Type 3	Type 2	Type 2	Type 1	Type 1	Type 1	Type 1	Type 3	Type 3	Type 3	Type 3	Type 2	Type 2	Type 1	Type 1	Type 1	Type 1
<b>TEAM 1</b>	0.02	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Agent 1 ControlGECKOAgent	0.13	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Agent 2 SensorGECKOAgent	-1	0.02	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Agent 3 ControlHAWKAgent	-1	0.13	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Agent 4 SensorHAWKAgent	1.85	1.85	0.41	0.36	0.41	0.36	0.36	0.36	0.36	1.85	1.85	1.85	1.85	0.41	0.41	0.36	0.36	0.36	0.36	0.36
Agent 5 ImageProcessingHAWKAgent	1.85	1.85	0.41	0.36	0.41	0.36	0.36	0.36	0.36	1.85	1.85	1.85	1.85	0.41	0.41	0.36	0.36	0.36	0.36	0.36
Agent 6 - ImageProcessingGECKOAgent	1.72	1.72	0.37	0.33	0.37	0.33	0.33	0.33	0.33	1.72	1.72	1.72	1.72	0.37	0.37	0.33	0.33	0.33	0.33	0.33
Agent 7 ScanningRouteAgent	0.98	0.98	0.22	0.18	0.22	0.18	0.18	0.18	0.18	0.98	0.98	0.98	0.98	0.22	0.22	0.18	0.18	0.18	0.18	0.18
Agent 8 - MovementControllerAgent	0.93	0.93	0.18	0.15	0.18	0.15	0.15	0.15	0.15	0.93	0.93	0.93	0.93	0.18	0.18	0.15	0.15	0.15	0.15	0.15
Agent 9 RouteAreaAgent	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Agent 10 ControlGECKOAgent	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Agent 11 SensorGECKOAgent	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Agent 12 ControlHAWKAgent	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0.02	-1	-1	-1	-1	-1	-1
Agent 13 SensorHAWKAgent	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0.13	-1	-1	-1	-1	-1	-1
Agent 14 ImageProcessingHAWKAgent	1.85	1.85	0.41	0.36	0.41	0.36	0.36	0.36	0.36	1.85	1.85	1.85	1.85	0.41	0.41	0.36	0.36	0.36	0.36	0.36
Agent 15 ImageProcessingGECKOAgent	1.85	1.85	0.41	0.36	0.41	0.36	0.36	0.36	0.36	1.85	1.85	1.85	1.85	0.41	0.41	0.36	0.36	0.36	0.36	0.36
Agent 16 ScanningRouteAgent	1.72	1.72	0.37	0.33	0.37	0.33	0.33	0.33	0.33	1.72	1.72	1.72	1.72	0.37	0.37	0.33	0.33	0.33	0.33	0.33
Agent 17 MovementControllerAgent	0.98	0.98	0.22	0.18	0.22	0.18	0.18	0.18	0.18	0.98	0.98	0.98	0.98	0.22	0.22	0.18	0.18	0.18	0.18	0.18
Agent 18 RouteAreaAgent	0.93	0.93	0.18	0.15	0.18	0.15	0.15	0.15	0.15	0.93	0.93	0.93	0.93	0.18	0.18	0.15	0.15	0.15	0.15	0.15
Agent 19 AlertCollisionAgent	1.99	1.99	0.45	0.39	0.45	0.39	0.39	0.39	0.39	1.99	1.99	1.99	1.99	0.45	0.45	0.39	0.39	0.39	0.39	0.39
<b>TEAM 2</b>																				

Figure 9.5: Results of the utilization factor  $\rho_i^j$ 

The value of the objective function is 9 and the node 5 is free of tasks. The nodes 1, 2, 6 and 7 are the ones with more remaining capacity ( $\phi^j = 0.67$ ), nevertheless, if the table in the figure 9.5 is consulted, there is no possibility to assign any other agent because of two possible reasons: (i) all the remaining tasks in these nodes have  $\phi_i^j = -1$  what means that the tasks cannot be assigned to such nodes or (ii) all the remaining tasks in these nodes have

Node	Agent ( $\rho_i^j$ )	Utilization Factor $\rho^j$	Current Remaining Capacity $\phi^j$
1	1 (0.02) and 2 (0.13)	0.15	0.67
2	3 (0.02) and 4 (0.13)	0.15	0.67
3	7 (0.37) and 16 (0.37)	0.74	0.088427
4	8 (0.18), 9 (0.15), 17 (0.18) and 18 (0.15)	0.66	0.096828
5	Empty	0	1
6	10 (0.02) and 11 (0.13)	0.15	0.67
7	12 (0.02) and 13 (0.13)	0.15	0.67
8	19 (0.45)	0.45	0.55
9	6 (0.36) and 15 (0.36)	0.72	0.108427
10	5 (0.36) and 14 (0.36)	0.72	0.108427

**Table 9.8:** Results of the task allocation model

their  $\phi_i^j$  with a value that would exceed the allowed utilization factor, i.e.,  $\rho^j > 2^{1/N_j} - 1$ .

In the node 8 there is only one task  $i = 19$ . In order to test the model, we manually adapt the values of  $\phi_{19}^j$  to verify that in this case, the task is allocated to a different node and the number of nodes needed is reduced to 8 instead of 9. In the table 9.9, we show the adapted values.

Agent 19		
AlertCollisionAgent		
	$\rho_{19}^j$ Before	$\rho_{19}^j$ After
<b>Node 3</b>	0.45	0.08
<b>Node 4</b>	0.39	0.06
<b>Node 5</b>	0.39	0.06
<b>Node 8</b>	0.45	0.08
<b>Node 9</b>	0.39	0.06
<b>Node 10</b>	0.39	0.06

**Table 9.9:** Modification of the  $\rho_{19}^j$

With these new values, the value of the objective function is 8. This means that two nodes suffer modifications, the task 19 is now allocated to the node 4



and the node 8 is now empty. This is the best solution in this case. The results are shown in the table 9.10.

Node	Agent ( $\rho_i^j$ )	Utilization Factor $\rho^j$	Current Remaining Capacity $\phi^j$
4	8 (0.18), 9 (0.15), 17 (0.18), 18 (0.15) and 19 (0.06)	0.72	0.023492
8	Empty	0	1

**Table 9.10:** Results of the scheduling and task allocation model

### 9.3

## Results of the collaboration search

The results presented in this section correspond to the model explained in section 8.5 of the chapter 8. The notation used is the same that in such section.

The system was tested in the natural park of Valcuevo, owned by the Spanish bank Caja España-Duero. This park is located in the municipality of Valverdón, approximately 6 km away from the center of Salamanca city in Spain. It is used for leisure and familiar activities due to its characteristics. The prototype was used in this park for surveillance tasks and was used at closing time, to make sure that no one was left locked inside the enclosure. Given the size and the topographic features of the park, the prototype is ideal for this task, moreover, the only security guard takes a long time to walk around the whole park.

The performance tests were conducted under favorable environmental conditions (minimal winds with relatively clear skies and no harsh climate). The main problem of the system is the duration of the batteries used for HAWK. Since HAWK can carry about 5 kilos, we chose to add longer lasting batteries, but the maximum flight time achieved was 19 minutes (without counting the time of the flight back to the starting point). The battery life depends heavily on flying height, since wind resistance is greater at higher elevations and the meteorological conditions are worse in general. The optimal velocity of HAWK is between 3 and 5 m/s to avoid any instability when performing the movement  $SM$  and to capture images with the enough quality.

The Table 9.11 presents the relationship between the height  $Z_0$  of HAWK and the characteristics of different cameras with a field of view with an area of ( $FOV$ ) and a total field of view after completing the  $SM$  ( $FOV_{SM}$ ). In the first case, the camera used is the GoPro Hero 3 with the following

characteristics:  $CCDdiagonal = 7.81mm$ ;  $\lambda = 2.98mm$ . In the second case, the camera Logitech HD C920 was chosen with the following characteristics:  $CCDdiagonal = 8.46mm$ ;  $\lambda = 3.67mm$ .

The results of the image analysis are not good when the height exceeds  $Z_0 = 7m$ . Consequently, the parameter  $\lambda$  must be increased, which means the camera must simply zoom in. With these latest cases, in the Table 9.12 we can see how  $FOV_{SM}$  varies according to the values of  $\lambda$ .

		$Z_0 = 4m$	$Z_0 = 5m$	$Z_0 = 6m$	$Z_0 = 7m$
CASE 1	$FOV(m^2)$	53.94	84.25	121.30	165.08
	$FOV_{SM}(m^2)$	485.47	758.32	1091.8	1485.8
CASE 2	$FOV(m^2)$	41.74	65.20	93.86	127.74
	$FOV_{SM}(m^2)$	375.70	586.82	844.82	1149.7

**Table 9.11:** Results obtained by modifying the hight of HAWK and the camera

		$Z_0 = 3m$		$Z_0 = 5m$	
GoPro Hero 3 $CCDdiagonal = 7.81mm$	$\lambda$	3,18	3,38	3,18	3,38
	$FOV_{SM}$	239.95	200.41	665.98	556.17
Logitech HD C920 $CCDdiagonal = 8.46mm$	$\lambda$	3,87	4,07	3,87	4,07
	$FOV_{SM}$	190.19	163.84	527.78	54.63

		$Z_0 = 7m$	
GoPro Hero 3 $CCDdiagonal = 7.81mm$	$\lambda$	3,18	3,38
	$FOV_{SM}$	1304.9	1089.7
Logitech HD C920 $CCDdiagonal = 8.46mm$	$\lambda$	3,87	4,07
	$FOV_{SM}$	1034	8414.4

**Table 9.12:** Results obtained by modifying the hight and the focal length,  $\lambda$

Both Tables 9.11 and 9.12 show that using the GoPro Hero 3 enables the device to fly higher and, therefore, to avoid more obstacles while maintaining the image quality for better analysis.

The Table 9.13 corresponds to the values  $(xDist, yDist)$  that the MovementControllerAgent should provide to the ScanningRouteAgent in order to calculate the waypoints of the autonomous movement  $SM$ .

The Table 9.14 shows a summary that compiles data taken from some tests, where  $LM_{HAWK}$  refers to the total linear distance travelled by HAWK and  $LM_{GECKO}$  to the total linear distance travelled by GECKO.

As previously explained, the parameter  $percCommonArea$  can be configured by the user and it can be change at any time during the task. Increasing this parameter, we can make sure that no large corners remain without exploration

		$Z_0 = 4m$	$Z_0 = 5m$	$Z_0 = 6m$	$Z_0 = 7m$
CASE 1	$xDist(m)$	8.11	10.14	12.17	14.20
	$yDist(m)$	6.64	8.30	9.96	11.62
CASE 2	$xDist(m)$	7.14	8.92	10.71	12.49
	$yDist(m)$	5.84	7.30	8.76	10.22

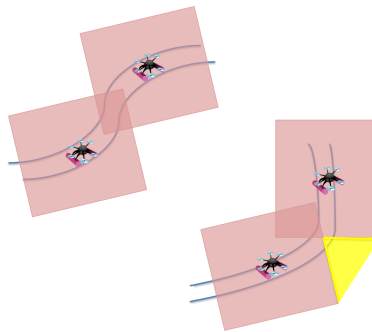
**Table 9.13:** Results of the horizontal and vertical displacements, ( $xDist, yDist$ ), for the calculations of the movement  $SM$

Time of flight	HAWK Velocity	$Z_0$	$LM_{HAWK}$	$LM_{GECKO}$	TotalScannedArea
19min	3m/s	5m	3422m	1885m	27077m <sup>2</sup>
		7m	3431m	1901m	38287m <sup>2</sup>
18min	4m/s	5m	4328m	2223m	35827m <sup>2</sup>
		7m	4319m	2397m	52636m <sup>2</sup>
17min	4m/s	5m	4080m	1958m	33537m <sup>2</sup>
		7m	4107m	1918m	47047m <sup>2</sup>
16min	5m/s	5m	4791m	2203m	39957m <sup>2</sup>
		7m	4800m	2164m	53972m <sup>2</sup>
15min	5m/s	5m	4508m	2235m	37475m <sup>2</sup>
		7m	4541m	2482m	55188m <sup>2</sup>

**Table 9.14:** Summary of flying results

between consecutive  $FOV_{SM}$  movements. But in this case, more time is required and for this time the total scanned area is also reduced.

In Figure 9.6, we can see what we call corners without exploration and observe how they depend on the route that GECKO follows and the stop points. In both images, the  $percCommonArea$  is the same; however, in the upper image, GECKO follows a route almost without curves or turns while in the lower image, the curve means that the  $angle$  of GECKO changes. The yellow indicates the corner without exploration, which even affects a part of the path.



**Figure 9.6:** Changes of angle according to the path

---

## 9.4 Conclusions

---

The most important issue that we have probed in this chapter is the suitability and efficiency of the proposed model. We have shown the results of several applications of the model with different values and situations. It is now possible to distribute agents working in the same team in the most suitable computational resources while the real-time constraints are always complied. Moreover, the scalability of the system is remarkable; it enables new agents with new tasks to enter the VOs and join the team or even add new teams.

From the point of view of the collaboration, we have also developed a successful system tested in real conditions. Given the characteristics quality/price of the vehicles, the results are good. Currently, we have only two teams, but once the future work proposed in 9.4 is more advanced, more teams will be added for testing. We have also demonstrated that the model proposed for the collaborative scanning area is correct and fulfill its objectives.

The advantages of using VOs are well documented in our previous studies [308] [305]. Traditional MAS development methodologies are not suitable for developing systems where the members dynamically change because they assume a fixed number of agents that are specified during the system analysis phase. It then becomes necessary to have an infrastructure that can use the concept of agent technology in the development process, and apply decomposition, abstraction and reorganization methods. To solve this, the concept of VOs emerged as a paradigm to apply organizational aspects and emergent behaviours to open systems, which are characterized by their dynamic nature. The use of VOs in robotics, and especially in heterogeneous teams, can be beneficial for the following reasons:

- The concept of normativity: in any organization or group there is a set of norms governing the operation of the group. These norms may affect individual and collective behaviors, communications and interaction between agents and even between different groups and organizations, and access to the services.
- The ability to define groups with different topologies, ie, coalitions, hierarchical, etc
- The ability to create different roles, which is one helpful mechanism used to manage access to the system and the different organizations.
- The ability regulate the access to the services, which are identified with the skills, abilities or behavior that each agent offers.

For these same reasons, the use of the PANGEA+RT platform is also an advantage. PANGEA+RT covers all the mentioned issues and provides the following desired characteristics:

- PANGEA+RT can facilitate the implementation of monitoring and controlling processes for teams of agents. Additionally, organizational mechanisms can take advantage of the continuous monitoring of emergent behaviours and interaction with users, to define more precise decision making mechanisms.
- Robustness: if an agent controlling the movement of a robot fails, this agent can be easily replaced and does not prevent the remaining robots from performing correctly.
- Specialization: creating different functional types (roles) that specialize in one sub-task.
- Decision capabilities: the collective work may reach high complexity tasks, and organizational mechanisms can take advantage of the continuous monitoring of emergent behaviours and interaction with users, to define more precise decision making mechanisms.
- Easy robot programming because the objective is for them to work together. This means that the individual mechanisms of each robot can be simpler since the efficiency of the process lies in the group.
- Efficient communication and simple interoperability required to (i) facilitate interaction between agents and (ii) design open robotic societies where new agents or robots can be dynamically incorporated to the system or leave it.
- Encourage the integration and reuse of software without having to rewrite implementations.
- Provide the system with enough computing resources, allowing for the replication or distribution of tasks to balance the workload.
- Provide automatic service discovery with different configurations. A service oriented approach facilitates the development of open systems and the reusability of resources.
- Provide mechanisms to control emergent behaviours and to reconfigure the groups and coalitions of agents.

In summary, the most important milestones that have been achieved with this case study are:

- To demonstrate the proper functioning of the model for the WCET estimation and task allocation proposed in the chapter 4.
- To develop a complete system with heterogeneous agents and robotic entities over the PANGEA+RT platform. The platform has facilitated the development and it manages the formation of new groups as well as the allocation of tasks through the several available nodes.
- End user products, including monitoring and controlling interfaces for HAWK and GECKO, were also implemented, in addition to the interface for controlling all the teams that participate in the task.
- Team members are able to work together to scan areas. And moreover, that different teams work together to share information.
- An integral system consisting of a group of heterogeneous robots for surveillance was developed. This system significantly helps the personnel dedicated to this task.
- All these milestones have been achieved using an agent technology with organizational concepts which have not yet been introduced in the field of robotics and the real-time.

# 10

## CONCLUSIONS AND FUTURE WORK

---

*This chapter presents the conclusions that were obtained during this research. The main contributions are presented, as well as possible future research, which includes the issues that require improvement and are open to further research.*

### Contents

---

<b>10.1 Conclusions and Main Contributions . . . . .</b>	<b>236</b>
<b>10.2 Future Work . . . . .</b>	<b>238</b>
10.2.1 Future lines related to the model . . . . .	238
10.2.2 Future lines related to PANGEA+RT . . . . .	240
10.2.3 Future lines related to the Case Study . . . . .	241

---

**10.1****Conclusions and Main Contributions**

---

This chapter describes how we achieved the proposed objectives at the beginning of this research to validate and evaluate the initial hypothesis: *it is possible to develop a scheduling and task allocation model that successfully enables the agents integrated into a VO to comply with their time constraints. This model will be theoretically formalized and then implemented to evaluate the results.*

The research presented in this document provides new contributions, and the innovative integration of real-time agents in VOs. From our knowledge, this is the first model and the first agent-platform specifically designed to develop VOs in real-time environments. The process from establishing the hypothesis to obtain results for the proposed model and the implemented platform has been described throughout this document. This complies with the methodology required to successfully validate the initial hypothesis.

The main contributions of this research are described next.

**Study of related works, technologies and methodologies**

At the beginning of this investigation, a study of the RTS and VO paradigms was carried out. This study is a complete compendium with basic literature to start future projects in related areas for new researchers. Moreover, references to other important reference works were also included in order to provide a list of must-read documents. This work is reflected in Part II of this document.

**Design, development and test of the proposed model**

With the main contribution of this dissertation presented in part III, we have fulfilled two objectives:

- Developing an effective model for calculating the WCET when an agent desires to form part of a virtual organization and wants to include tasks or behavior in the real time system, i.e. to calculate the WCET in emergent behaviours.
- Developing a model of scheduling and task allocation for the whole system, i.e. global scheduling.

Moreover, we have shown the good results obtained when applying the model in the Chapter 9.

**Platform for real-time agents' execution**

This contribution was divided into three milestones, each one resulting in a different version of the platform:



- PANGEA: VO-oriented.
- PANGEA+R: collaborative-robotics-oriented.
- PANGEA+RT: real-time-oriented.

We can conclude that we have a complete platform that can manage VOs with different topologies and take organizational aspects into account. PANGEA+R supports any kind of agent including robotics and embedded agents thanks to the lightweight MQTT protocol. And finally, in PANGEA+RT we included real-time constraints with a scheduling and task allocation model.

### **Application and use of the platform in a real problem**

A case study was developed to test the validity of the proposed model and the functioning of the PANGEA+RT platform. The case study was applied in a real environment to surveillance tasks in a leisure park called Valcuevo and is focused on the collaboration of two robots, HAWK and GECKO, both of which were developed by the BISITE group. The team's job is to scan a large peripheral area of a route chosen by the personnel. The main goal is to focus on surveillance tasks; however, this system can be also applied to rescue tasks. With this system, search time can be reduced as the team of robots helps scan the search area. The current system is precisely focused on providing a solution to help human staff with the surveillance efforts and reducing as much as possible the amount of time invested in the exploration of areas.

### **Establishment of new knowledge and future research lines**

The research in the field of the VO started within the BISITE group with a national project called THOMAS; subsequently, two projects have continued in this line:

- THOMAS Project (Métodos Técnicos y Herramientas para Sistemas Multiagente Abiertos), TIN 2006-14630-C03-03, funded by the Spanish Ministry of Science and Innovation.
- OVAMAH Project (Organizaciones Virtuales Adaptativas: Mecanismos, Arquitecturas y Herramientas), TIN 2009-13839-C03-03, funded by the Spanish Ministry of Science and Innovation (Projects of non-oriented fundamental Research).
- iHAS Project (Intelligent Social Computing for Human-Agent Societies), TIN 2012-36586-C03-03, funded by the Spanish Ministry of Science and Innovation (Projects of non-oriented fundamental Research).

The last project will allow us to continue and extend the development of this research, as explained in the next section, and to apply it to other scenarios presented in the project.

We had a special interest in disseminating the experiences and progress of this research during its development, from its earliest stages to its final form, through various publications and attendance at conferences, workshops, etc. that can be consulted in the Appendix of this document.

## 10.2 Future Work

---

This dissertation opens new lines of research that can be classified into three groups. The first group includes issues related to the developed model shown in part III, the second group is formed by the points related to the agent-platform presented in part IV, and finally, the third group is composed of issues related to the case study in chapter 8. They are presented next.

### 10.2.1 Future lines related to the model

---

#### **Extent the model to bounded communication**

Scheduling message communication in a network is difficult since some communication media such as Ethernet do not guarantee bounded communication delay at the media access level and do not provide priority-based arbitration. In order to guarantee that the timing requirements of all tasks are met, the communication delay between a sending task queuing a message, and a receiving task accessing that message, must be bounded. This total delay is termed the end-to-end communication delay. Tindell [354] defines the four major components that must be taken into account in an end-to-end communication delay:

1. the generation delay: the time taken for the application task to generate and queue the message
2. the queuing delay: the time taken by the message to gain access to the communications device after being queued
3. the transmission delay: the time taken by the message to be transmitted on the communications device
4. the delivery delay: the time taken to process the message at the destination processor before finally delivering it to the destination task

Currently, in our model we use a  $I$  parameter to include possible delays when a task includes communication among agents. We will improve this mechanism according to the previously mentioned points and in some guidelines presented by [26]. The main challenges in this field is to develop a formal model to the delay calculation.

### **Handle aperiodic and sporadic tasks**

The scheduling problem for aperiodic and sporadic tasks is very different from that for periodic tasks. Scheduling algorithms for aperiodic tasks must be able to guarantee the deadlines for hard deadline aperiodic tasks and provide good average response times for soft deadline aperiodic tasks, even though the occurrence of the aperiodic requests are non-deterministic. The aperiodic scheduling algorithm must also accomplish these goals without compromising the hard deadlines of the periodic tasks.

Two common approaches for servicing soft deadline aperiodic requests are background processing and polling tasks. Background servicing of aperiodic requests occurs whenever the processor is idle (i.e., not executing any periodic tasks and no periodic tasks pending). If the load of the periodic task set is high, then the utilization left for background service is low, and background service opportunities are relatively infrequent. Polling consists of creating a periodic task for servicing aperiodic requests.

In the future, we will study these proposals to adapt them to our work. Although these types of tasks in our system are currently converted to periodic tasks, this solution obtains worse performance results.

### **Resource limited systems**

In our model, we suppose that enough nodes are available for all the tasks. In the case study, we show a scenario where we overload the system and we must use the parameter  $H$ , which indicates the importance or valuation of the task inside the system (see section 11.4). But this may result in some tasks with a lower level of importance being missed.

We will improve the proposal under conditions of limited or changing resource availability where the system can exhibit an inefficient and erratic behavior, or low or unacceptable utility. When a resource failure occurs, the system may not have sufficient resources to redistribute important tasks. This limitation will be taken into account in the future.

### **Supported Languages in Class Annotation**

Agents implemented in various languages can be deployed in the PANGEA+R platform, but PANGEA+RT has only implemented the mechanism for annotating classes in Java. Currently, if a real-time agent written in another language

wants to be integrated into the platform, it must provide the WCET with an explanation of its tasks.

The next step in this line is to include the class annotation for C++ in PANGEA+RT. So far, most of the frameworks to annotate in C++ are oriented to information-flow security and policies [311], compilation efficiency [286] [6] or executing the code in a type-safe manner [351].

Starting from the "The annotated C++ reference manual" [100] we will adapt it to our needs, specifically, the loop boundaries.

---

**10.2.2****Future lines related to PANGEA+RT**

---

**Distributed Java**

This line is closely related to the Distributed Real-Time Specification for Java (DRTSJ) [289]. As its creators mention in [7], the main goal is to enhance RTSJ with trans-node real-time end-to-end support, using REMOTE METHOD INVOCATION (RMI) as the distribution mechanism, specifically:

- The primary aim is to offer end-to-end timeliness by guaranteeing that an application's trans-node has its timeliness properties explicitly employed during resource management. Provide for and express propagation, resource acquisition/contention and storage and failure management in the programming abstraction.
- The second goal is to enhance the programming model with control flow facilities that model real-time operations composed of local activities. The activity-control actions include: suspend, abort, changes propagation, failure propagation, consistency mechanisms, and event notification. The result of this activity is distributable real-time threads.

**Hard real-time communication protocol**

Real-time extensions to standard switched Ethernet widen the field of computer networking into the time-critical domain. These technologies have started to establish in process automation, where Ethernet-based communication infrastructures are challenged by particularly hard real-time constraints. Some examples are the OMNeT++ INET framework [344] for simulating real-time Ethernet with high temporal accuracy. The FlexRay system [345] is based on time-triggered Ethernet, which is implemented as a replacement of current in-vehicle bus-systems. Further it is shown that a switched system has advantages in bandwidth utilization over a shared bus, when using group communication.

Another possibility is the Real Time Messaging Protocol (RTMP) [321] [322] developed by Macromedia for soft real-time systems.

We will delve more deeply into this topic and study all the possibilities to include an existing protocol in the platform or create an adapted protocol if the existing ones do not meet our requirements.

**10.2.3**

## Future lines related to the Case Study

All the mentioned future lines will improve the case study and will enable new features. But, current points to start working in this practical approach are:

- Improving the HAWK features to allow more autonomous flight time.
- Automatic avoidance of obstacles. The height of the flight for HAWK is currently set up over any obstacle, while for GECKO, the waypoints are defined to avoid obstacles.
- Introducing a mechanism that is based on the route or the preset area by and with which the user can coordinate various teams obtaining optimum movements. Currently, each team is configured separately, although the teams do coordinate to avoid studying the same area.
- Performing a correlation study; an evaluation of how the configurable variables affect the analysis of the images, such as the height and speed of the HAWK flight.



# Part VI

RESUMEN





# 11

## RESUMEN

---

*Este último capítulo corresponde al resumen en castellano dónde se incluyen las principales aportaciones de esta tesis. Se presenta el modelo propuesto dividido en dos partes, la estimación del WCET y la planificación y asignación de tareas respetando las restricciones de tiempo real. También se incluye una descripción de la plataforma de agentes PANGEA+RT que permite ejecutar VOs en un entorno de tiempo real con el modelo propuesto y además, incluyen todas las características necesarias para la formación de equipos desde el punto de vista organizacional. Una vez explicado el modelo y la plataforma, se presenta el caso de estudio centrado en la colaboración de robots heterogéneos para tareas de vigilancia. Finalmente, se muestran los resultados del caso de estudio con la evaluación del modelo propuesto y las conclusiones.*

### Contents

---

<b>11.1</b>	<b>Introducción</b>	<b>246</b>
<b>11.2</b>	<b>Conceptos básicos y trabajos relacionados</b>	<b>248</b>
11.2.1	Análisis del WCET	249
11.2.2	Planificación en tiempo real y distribución de tareas	250
<b>11.3</b>	<b>WCET en comportamientos emergentes</b>	<b>253</b>
11.3.1	Evaluación de los nodos	254
11.3.2	Evaluación del código	257
11.3.3	Adaptación estadística del WCET en tiempo de ejecución	261
<b>11.4</b>	<b>Planificación y distribución de tareas</b>	<b>262</b>
<b>11.5</b>	<b>La plataforma PANGEA+RT</b>	<b>268</b>
11.5.1	El protocolo de comunicación	268
11.5.2	Agentes de PANGEA+RT	272
11.5.3	Modificación de las clases	274
<b>11.6</b>	<b>Caso de estudio</b>	<b>277</b>
11.6.1	Despliegue de los agentes propuestos	280
11.6.2	Resultados	281
<b>11.7</b>	<b>Conclusiones</b>	<b>285</b>

---

## 11.1 Introducción

El diseño de un sistema de tiempo real (STR) es una actividad donde es necesario desarrollar una planificación meticulosa y una gestión efectiva de múltiples recursos. Estos recursos deben funcionar de forma predecible para asegurar que las tareas que se ejecuten en el sistema cumplen los requisitos temporales y con el funcionamiento esperado del sistema.

Los recursos deben ser asignados teniendo en consideración tanto aspectos funcionales como no funcionales (temporales) del sistema y el coste del diseño. El control y asignación de los recursos en ejecución dentro de un STR requiere una reacción rápida a los cambios de la carga de trabajo y comprobaciones eficientes para la planificabilidad, incluso cuando hay muchas limitaciones a controlar. Hasta ahora, los límites tradicionales han sido conservadores cumpliendo que, siempre y cuando el conjunto de tareas no supere el factor de utilización, se cumplirá que todas las tareas acaben dentro de sus plazos de ejecución. Alternativamente, si un conjunto de tareas supera el factor de utilización, se puede dar el caso en el que algunas tareas agoten sus plazos.

Los STR difieren de la mayoría de los sistemas de computación, debido a las restricciones temporales que tienen que satisfacer. Las tareas en un STR se asocian a limitaciones de tiempo explícitas que deben cumplirse para el correcto comportamiento. El tiempo se vuelve un elemento no funcional importante porque la corrección del sistema no sólo depende de la ejecución funcional correcta de las tareas sino también en el momento de finalización de las mismas.

Se debe distinguir entre la planificación y la gestión de recursos, a pesar de que están estrechamente relacionadas. La planificación es el aspecto de la gestión de los recursos de que trata de asegurar que un conjunto de tareas cumple con sus requisitos temporales. La gestión de recursos en su totalidad, tiene que ver con las decisiones de más alto nivel que determinan qué recursos llevarán a cabo una tarea específica.

Los algoritmos de planificación de tareas son generalmente independientes del marco de gestión de recursos. En este trabajo, se ha combinado planificabilidad y asignación de recursos ya que una vez se estiman los límites de utilización y el peor tiempo de ejecución para las tareas (WCET), las decisiones de gestión de recursos se pueden hacer dentro de la llamada región de planificabilidad. Esto se considera como un problema NP-hard: solucionable en teoría, pero casi imposible de resolver en la práctica. No obstante, se propone una combinación

de métodos matemáticos que llevan a una buena solución de acuerdo con las restricciones temporales.

En esta tesis se pretende que los STR hagan uso del paradigma de agentes. La investigación actual centrada en el diseño de sistemas multi-agente (MAS) desde el punto de vista organizativo está ganando terreno. La idea que prevalece es que el modelado de las interacciones de un MAS no puede limitarse al agente y sus capacidades de comunicación, sino que requiere de la ingeniería de la organización. Los conceptos de normas [386], instituciones [105] y estructuras sociales [279] nacieron de la idea de que es necesario un mayor nivel de abstracción, independiente del agente, y la capacidad de definir explícitamente la organización en la que los agentes residen. Los agentes de un MAS basado en conceptos organizativos trabajan en coordinación e intercambian servicios e información que necesitan para ser capaces de negociar, colaborar y realizar otras acciones sociales más complejas. El término acuñado para estos sistemas es Organizaciones Virtuales (OV) [111]. La dinámica de los entornos abiertos es una de las razones que han alentado el uso de OV. Hoy en día, los MAS deben ser más abiertos y dinámicos. En un MAS abierto [27] como una OV se debe permitir la interacción entre agentes heterogéneos, que cambian con el tiempo y con arquitecturas e incluso diferentes lenguajes de programación. Debido a su propia naturaleza cambiante, no podemos confiar en el comportamiento de los agentes cuando es necesario establecer controles sobre la base de normas o reglas sociales. Por esta razón, y debido a las características de los entornos abiertos, se necesitan nuevos enfoques para apoyar a los sistemas evolutivos y para facilitar el crecimiento en tiempo de ejecución. Por tanto, una OV [111] [113] es un sistema abierto diseñado para la agrupación que permite la colaboración de entidades heterogéneas y proporciona una separación entre la forma y la función que definen su comportamiento.

En este estudio, se trata de conjugar la apertura y el dinamismo necesarios en las OV y las restricciones que imponen los STR. Esta no es una tarea trivial, ya que el primer paradigma no es estricto, como el término "abierto" indica, pero el segundo paradigma debe cumplir con estrictas limitaciones. En resumen, el modelo que se presenta permite definir las acciones que una organización de agentes debe llevar a cabo dentro de un plazo de tiempo determinado, teniendo en cuenta los cambios que se puedan producir durante la ejecución de un plan en particular. Se trata de una planificación en tiempo real dentro de una OV.

Con esta tesis se pretende cumplir principalmente con dos objetivos:

- Proponer un modelo efectivo para el cálculo del tiempo de ejecución de una tarea en el peor caso posible cuando un agente nuevo desea unirse a una OV proporcionando un nuevo comportamiento o servicio en tiempo real, es decir, calcular el WCET en un comportamiento emergente y

teniendo en cuenta restricciones temporales. Esta medida se recalculará y ajustará mediante técnicas estadísticas en las sucesivas ejecuciones.

- Proponer un modelo de planificación y distribución de tareas para el sistema, es decir, un nuevo sistema de planificación global.

Hasta el momento, no existe ningún modelo de tiempo real especialmente desarrollado para su uso en sistemas abiertos como las OV. Convencionalmente, los modelos de tiempo real se aplican a sistemas cerrados donde todas las variables y los componentes son conocidos a priori. Sin embargo, este modelo presenta limitaciones cuando se trabaja con sistemas abiertos que incorporan conceptos organizativos, ya que se necesita capacidad para evolucionar y adaptarse a las necesidades de los sistemas actuales. Por tanto, la hipótesis inicial que se plantea en este trabajo de tesis doctoral es que: *es posible desarrollar un modelo de planificación y asignación de tareas que permita a los agentes pertenecientes a una OV rematar con éxito sus tareas dentro de las limitaciones temporales. Este modelo será formalizado teóricamente e implementado para evaluar los resultados.*

Además, tomando como referencia el estudio y la búsqueda exhaustiva de trabajos llevada a cabo en este trabajo de tesis doctoral, no se conoce la existencia de ninguna plataforma de ejecución para agentes que soporte las restricciones del tiempo real y a la vez, los mecanismos para el control y gestión de las OV. Estos hechos proporcionan la motivación necesaria para el desarrollo de la plataforma de agentes denominada PANGEA+RT.

## 11.2

### Conceptos básicos y trabajos relacionados

Esta sección está dividida en dos subsecciones ya que nos enfrentamos a dos problemas por separado. Primero, se debe afrontar la predicción del WCET y para ello, se muestran algunas de las publicaciones más relevantes relacionados con este ámbito. La segunda parte de la sección está dedicada a la planificación global y la distribución de tareas.

**11.2.1** Análisis del WCET

---

Los problemas relacionados con el cálculo del WCET han sido estudiados desde diferentes perspectivas. En [327] Shaw presenta un conjunto de esquemas temporales para calcular los tiempos mínimo y máximo de ejecución mediante reglas y árboles de decisión. En [295] [218] se propone el cálculo mediante IPL y [147] se centra en el uso del control del flujo de ejecución para detectar límites en los bucles. En términos de coste de computación se presentan interesantes propuestas en [4] [80] [18]. En particular, en [261] se presenta un *framework* que regula el conjunto de recursos en tiempo real. También el proyecto Mobility, Ubiquity and Security (MOBIUS) [32] trata con la planificación y la distribución de tareas.

Si centramos el estudio en el lenguaje Java, surgen dificultades como la falta de un reloj con precisión para tiempo real o el concepto de plazo. Iniciativas como las especificaciones Safety Critical Java (SCJ) [182] y Predictable Java (PJ) [49] pretenden establecer a Java como un lenguaje de peso dentro de los sistemas de tiempo real. La nueva especificación incluye, entre otros, un nuevo modelo de programación que es más susceptible de ser temporal correcto. La corrección temporal puede garantizarse mediante el uso de la información a partir de un análisis WCET para realizar análisis de planificabilidad. Con respecto a Java, el análisis del WCET se complica por la presencia de la Máquina Virtual de Java (JVM), ya que ésta introduce una capa adicional entre la propia aplicación y la plataforma de hardware subyacente. Para mitigar esta complejidad, los trabajos se ha centrado en la eliminación de esta capa intermedia mediante la implementación de la JVM en hardware, consiguiendo de este modo la ejecución del código nativo de Java, los llamados bytecodes (JBC) [318]. Esta línea ha sido una de las más exploradas para experimentar con Java en el desarrollo de sistemas de tiempo real.

Bernat [43] fue el primero en considerar el análisis del WCET a nivel de bytecodes. Bernat argumenta que la correcta representación intermedia de un programa en bytecodes de Java, que también puede ser generada a partir de compiladores para otros lenguajes (por ejemplo, ADA), es altamente adecuada para una herramienta de análisis del WCET. En dicho trabajo, se utilizan anotaciones de Java y Ada para guiar el análisis del WCET a nivel de bytecodes. El trabajo se extendió para abordar el análisis del tiempo a bajo nivel donde es dependiente de la máquina [36].

En [293] se presenta un mecanismo portable del WCET. Este análisis abstracto se lleva a cabo durante el desarrollo y genera información abstracta que se usa posteriormente. El análisis concreto se efectúa en la máquina de ejecución final

reemplazando estos valores abstractos en las fórmulas del WCET propuestas. En [37] se presenta un mecanismo que muestra cómo la información sobre el WCET puede ser embebida en la clase compilada de Java, es una extensión de los trabajos [43] [36]. Sin embargo, las medidas obtenidas no pueden garantizar los límites temporales superiores de manera segura. En [228], el autor describe el diseño y las capacidades de la herramienta TetaSARTS que de manera estática lleva a cabo un análisis del WCET y verifica que las aplicaciones cumplen la especificación nombrada anteriormente, Safety Critical Java (SCJ).

Por último, Bogholm [48] presenta una selección de herramientas que ayudan a los desarrolladores de aplicaciones de tiempo real estricto a verificar que los programas se ajustan a un perfil en tiempo real en Java, y que se satisfacen las limitaciones de recursos específicos de la plataforma. El problema de los trabajos presentados es que son frameworks cerrados que no podemos incluir en nuestro entorno de OV's y adaptarse a las restricciones que imponen cada una de las organizaciones.

En definitiva, fomentando la capacidad de Java y su entorno de ejecución para sistemas críticos de tiempo real con mecanismos deterministas de comportamiento y planificación facilitará que Java se convierta en un importante lenguaje y entorno de ejecución en el desarrollo de sistemas de tiempo real [119].

### 11.2.2

## Planificación en tiempo real y distribución de tareas

---

En esta sección se introducen algunos conceptos básicos sobre tiempo real y se mencionan los estudios más relevantes relacionados con el campo de aplicación de esta tesis.

Existen dos tipos básicos de esquemas de planificación: estática, donde las predicciones se hacen antes de la ejecución y cuando todas las tareas se conocen; y dinámica, donde las decisiones se toman en tiempo de ejecución. Sin embargo, esta clasificación es muy estricta y como se propone en esta tesis, un modelo considerado como estático, el FPS, puede ser adaptado e incluido en ambientes dinámicos. Esto obliga a introducir un control teniendo en cuenta los posibles cambios a lo largo del tiempo y acotando temporalmente este proceso de replanificación. Es necesario tener en cuenta que para un sistema de tiempo real estricto donde las tareas son críticas y todas tienen un plazo de respuesta introducir un modelo de planificación dinámica puro puede conllevar un mal

funcionamiento. En la propia definición de planificación dinámica establecida por [70], se mencionan las siguientes características:

- No es posible garantizar los plazos de respuesta.
- No es adecuada para sistemas críticos.
- No se puede establecer una planificación estable.

En los últimos años, los sistemas de planificación basados en prioridades han recibido una especial atención en la comunidad científica, alcanzando un alto grado de madurez y reconocimiento [339] [348] [213] [326] [59] [21]. En [23] y [325] se pueden encontrar dos interesantes estudios históricos sobre el progreso en la planificación con prioridades fijas. Además, una revisión de varios trabajos [133] [275] [51] [116] [133] [275] [51] [116] muestra que la planificación dinámica es muy dependiente del ambiente y del contexto de ejecución. También es importante resaltar que los métodos propuestos están basados en sistemas iterativos con gran complejidad computación o con heurísticas que no pueden asegurar el tiempo de computación y por lo tanto, las tareas no están estrictamente acotadas en el tiempo.

Un sistema de planificación está caracterizado por dos aspectos:

- El algoritmo de planificación: determina el orden de acceso a los recursos disponibles en el sistema.
- El modelo de análisis: se encarga de calcular y predecir el comportamiento en el tiempo del sistema, es decir, si los requisitos y restricciones temporales están garantizados en todos los casos posibles. Para asegurarse, debe tenerse en cuenta el escenario en el peor de los casos.

El Rate-Monotonic (RM) o también conocido como algoritmo Fixed Priority Scheduling (FPS) y sus extensiones son algoritmos de planificación estáticos y representan el mayor paradigma de planificación para tiempo real. C.L. Liu y J.W. Layland propusieron por primera vez el RM en su trabajo [221]. De manera simple, consiste en asignar las máximas prioridades a las tareas con los periodos más cortos, esto es, la prioridad de cada tarea es inversamente proporcional a su periodo. Para implementar esta política de planificación, se debe establecer que el tiempo de respuesta de cada tarea es igual a su periodo.

El Earliest Deadline First Scheduling (EDF) es el segundo paradigma para la planificación de tiempo real. sobre ciertas condiciones, el EDF [221] [341] es un algoritmo óptimo para la planificación si existen suficientes recursos para la computación. Sin embargo, el coste y los entornos impredecibles hacen imposible garantizar que los recursos del sistema sean siempre suficientes. En este caso, cuando existen situaciones de sobrecarga, el rendimiento del EDF se degrada

rápidamente. El algoritmo de planificación conocido como Spring [396] puede garantizar la correcta ejecución de tareas mediante un sistema de admisión en línea y que es aplicable en entornos con recursos insuficientes. Muchos otros algoritmos [341] se han diseñado para operar de esta forma. Estos algoritmos basados en sistemas de admisión controlados representan el tercer paradigma en la planificación de tiempo real. Sin embargo, hay problemas del mundo real que no son fácilmente soportados dentro de estos tres paradigmas.

Los algoritmos como el EDF, el RM y el Spring pueden soportar conjuntos de tareas con características complejas (tales como plazos, restricciones de precedencia, recursos compartidos, jitter, etc) pero todos ellos son algoritmos con programación de bucle abierto. Bucle abierto se refiere al hecho de que una vez que se establece la planificación no se ajustan teniendo en cuenta una continua retroalimentación. Mientras que los algoritmos de programación en bucle abierto pueden funcionar bien en ambientes predecibles en los que las cargas de trabajo pueden ser modeladas con precisión (por ejemplo, sistemas de control de procesos tradicionales), pueden tener mal rendimiento en ambientes impredecibles, es decir, los sistemas cuyas cargas de trabajo no se pueden modelar con precisión.

En los últimos años, una nueva categoría de aplicaciones de tiempo real no-estrictas que se ejecutan en ambientes impredecibles está rápidamente siendo implantada [343]. Algunos algoritmos adaptativos han sido recientemente desarrollados [1] [38] [251] [355] [202] [201] como un enfoque efectivo en términos de coste para garantizar un rendimiento aceptable en dichos ambientes impredecibles. El problema es que estas propuestas son apropiadas para tareas no-críticas sobre todo en el campo de la comunicación y sistemas multimedia, mientras que nuestro estudio de centra en tareas críticas.

Los primeros estudios en planificación de tiempo real se centraban en evitar en su totalidad cualquier efecto indeseable como la sobrecarga o la pérdida de plazos, en cambio, los sistemas adaptativos de tiempo real se diseñan para manejar dinámicamente y con éxito estos efectos. El problema principal es que el propio acto de tratar estos efectos implica la posibilidad de que se produzcan, lo que propiamente se puede considerar un efecto indeseable pero a la vez inevitable en sistemas dinámicos de tiempo real. Desde nuestro punto de vista, la necesidad de conjugar una planificación estricta en sistemas adaptativos está todavía en su fase inicial de estudio. Para ser estrictos y encontrar trabajos relacionados con tareas críticas, debemos recurrir al campo del control y la automatización. Diversos trabajos han optado por aplicar la teoría del control [115]. Por ejemplo, trabajos como [350] [314] [358] [257] presentan técnicas de planificación flexibles para mejorar el control de sistemas digitales. Estas técnicas están diseñadas específicamente para el control de sistemas y son difícilmente aplicables a escenarios de tiempo real en sistemas



adaptativos. También otros trabajos [343] [367] [199] [54] presentan algoritmos de planificación que basan su adaptación en arquitecturas de QoS para sistemas multimedia y de comunicación.

Después del estudio de los trabajos mencionados, podemos aseverar que proponer un modelo de planificación y asignación de tareas en tiempo real y además en un ambiente dinámico y adaptativo como son las OV, es un área por explorar. Los métodos revisados no pueden ser aplicados en este campo debido a una falta de capacidad para asegurar tiempo de computación con una solución efectiva hablando de sistemas de tiempo real estrictos. Para lograr esto, el sistema debe estar basado en modelos deterministas con un fundamento matemático sólido para validar el procedimiento.

Finalmente, después del extenso estudio realizado y consultadas diversas fuentes de información, no es posible afirmar que existe hasta el momento, ningún modelo unificado para el desarrollo de un sistema adaptativo y dinámico con una planificación local y global donde los nodos de computación necesarios se recalculen en tiempo real para lograr la capacidad del sistema de adaptación dinámica.

### 11.3

## Modelo de estimación del WCET en comportamientos emergentes

Como se ha mencionado, en los sistemas de tiempo real crítico no puede permitirse que se produzcan oérridas de plazos en la realización de sus tareas. Para asegurar que se cumpla esta, se debe aplicar un análisis de planificabilidad de manera que determine si un conjunto de tareas pueden ser incluidas en un sistema sin que excedan su tiempo de ejecución y terminen dentro del plazo previsto. El denominado WCET de una tarea es el parámetro clave de dicho análisis de planificabilidad.

Existen dos tipos análisis del WCET:

- *Análisis de alto nivel*: se refiere al control de flujo de la tarea. Este flujo se representa mediante un grafo dirigido (GFC) con vértices denominados bloques básicos y arcos. Cada vértice representa un conjunto de instrucciones que se ejecutan secuencialmente. Los bloques básicos se conectan mediante arcos que representan las posibles ramas de ejecución. A este nivel se supone que se conoce el coste de cada bloque básico, ya que se ha obtenido mediante el análisis a bajo nivel. El GFG es preferible generarlo a partir de un programa compilado donde ya se pueden

tener en cuenta las optimizaciones introducidas por el propio compilador, esto es importante ya que dichas optimizaciones afectan directamente al WCET [50].

- *Análisis de bajo nivel*: este análisis se refiere al tiempo de ejecución de un bloque básico en una máquina específica. Esto implica el cálculo de cada una de las instrucciones individuales en la máquina donde finalmente se ejecutará la tarea. El primer problema, reside en que estos datos no son proporcionados por los vendedores y además existen diversos métodos que utilizan para reducir el tiempo de ejecución como la memoria caché o las tuberías.

Estos análisis se complican más por la presencia de un sistema operativo e incluso una máquina virtual, ya que añaden capas adicionales entre la lógica del propio programa y el hardware [119].

### 11.3.1

### Evaluación de los nodos

La evaluación de los nodos se refiere al análisis WCET de bajo nivel donde las especificaciones hardware tienen un papel crucial. Esto implica que se debe estimar el tiempo de ejecución de cada instrucción en el nodo donde se va a ejecutar, lo que no es sencillo.

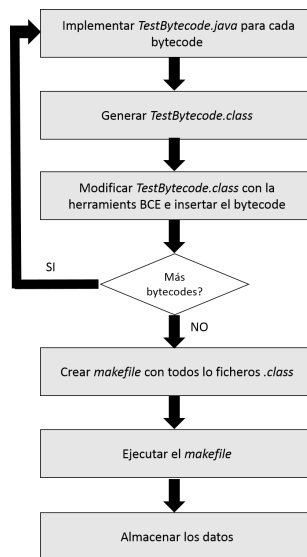


Figure 11.1: Pasos para la evaluación de los nodos

Un programa Java se compila en los denominados javabytes (JBC), que se pueden cargar en una máquina virtual de Java (JVM) arbitraria a su vez en ejecución en un procesador arbitrario. Este es el aspecto principal que complica el problema. Si consideramos un sistema en tiempo real como un conjunto de nodos de cálculo, cada nodo tiene su propia configuración y, posiblemente, su propia JVM también. Los pasos para obtener el programa que debe ser ejecutado en cada nodo para el análisis de los WCET se muestran en la figura 11.1.

Las pruebas se realizan con la biblioteca estándar de Java `System.currentTimeMillis`. Existe también otra biblioteca a considerar `System.nanoTime` pero la precisión de la primera justifica la elección ya que `System.nanoTime` no garantiza la precisión en nanosegundos como se prueba en [211]. Primero, se crean las clases de Java para cada bytecode. En el código 11.1 se muestra un ejemplo de evaluación de nodos.

```
1 public class TestDLoad
2 {
3     public static void main(String args [])
4     {
5         long time1, time2, tme3;
6         long result;
7
8         time1=System.currentTimeMillis();
9         for(int i=0;i<1000000;i++)
10            test();
11
12        time2=System.currentTimeMillis();
13        for(int j=0;j<1000000;j++)
14            test2();
15
16        time3=System.currentTimeMillis();
17
18        result=(tiempo3-tiempo2)-(tiempo2-tiempo1);
19
20        //store result in the database
21    }
22
23    //modificar los m{'e}todos en el fichero .class
24    public static void test()
25    {
26    }
27    public static void test2()
28    {
29    }
30 }
```

**Programming Code 11.1:** Clase Java para la evaluación de un bytecode

Para asegurarnos de que el bucle `for` no afecta a los cálculos, se programan

dos funciones de `test` diferentes. La diferencia se calcula en la línea 293 del código, así se conoce el tiempo de ejecución del número de bycodes predefinidos sin que interfiera el bucle.

Para modificar y crear el fichero `.class` incluyendo los bycodes editados se usa la herramienta JBE [310]. Es un editor de bycodes abierto y adecuado para consultar y modificar archivos `.class`. Para verificar y exportar las clases, JBE utiliza la biblioteca Bytecode Engineering del proyecto Apache's Jakarta [72]. Las principales ventajas que nos llevan a elegir esta herramienta es que todos los cambios se aplican directamente al fichero compilado de la clase, que usa los mnemónicos estándar para los JVM opcodes y las excepciones y otras anotaciones también son visibles en la interfaz de la herramienta.

```

1  0: lconst_0
2  1: lconst_1
3  2: lcmp
4  3: lconst_0
5  4: lconst_1
6  7: lcmp
7  [...]
8 1500001: return

```

**Programming Code 11.2:** Prueba para el bytecode `lcmp`

Es importante remarcar que hay bycodes que no se pueden probar de manera independiente, es decir, que dependen de otros bycodes para tener los operandos necesarios. Por tanto, para determinar el valor de estos bycodes se necesita conocer previamente el tiempo de ejecución de los que sí pueden ser independientes. Un ejemplo es el bytecode `lcmp` que compara dos valores de tipo `long`. Esto fuerza a introducir `lconst_0` y `lconst_1` en la función `test` como se muestra en 11.2.

Esto se debe a que el bytecode `lcmp` necesita dos valores de tipo `long` en la pila y deben introducirse antes de su ejecución. Después de la ejecución, `lcmp` dejan un entero en la pila pero que no es posible usarlo en las siguientes ejecuciones del bytecode por el tipo. Esto nos obliga a introducir repetidamente dos nuevos valores de tipo `long`. Como previamente conocemos el tiempo de ejecución del bytecode `lconst_0`, podemos calcular el valor en `TestLConst0.java` antes de que éste sea almacenado. Estos datos son introducidos en una base de datos. Cuando se utilizan en el siguiente paso se recuperan de una tabla global  $\mathcal{R}(I_{205}, N_{total})$  donde cada fila corresponde con un bytecode y la columna es el coste en el nodo  $N$  del sistema.

### 11.3.2 Evaluación del código

El análisis del WCET conlleva el estudio del código de programación por tanto, es necesario que el código no contenga estructuras que hagan imposible seguir el flujo de análisis. Además, los bucles no acotados o la recursión pueden tener como resultado evaluaciones pesimistas [294].

Los pasos generales para este proceso se muestran en la Figura 11.2.

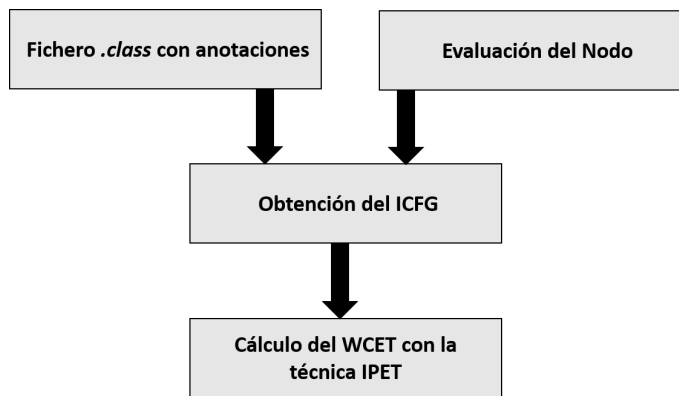


Figure 11.2: Pasos para el cálculo del WCET

Ahora se introducen las reglas principales para la generación del GFC. En primer lugar, se identifican todos los vértices principales, éstos corresponden con las siguientes intrucciones: `goto`, `goto_w`, `jsr`, `jsr_w`, `ret`, `ifeq`, `iflt`, `ifle`, `ifne`, `ifgt`, `ifge`, `ifnull`, `ifnonnull`, `if_icmpeq`, `if_icmpne`, `if_icmplt`, `if_icmpgt`, `if_icmple`, `if_icmpge`, `if_acpeq`, `if_acmpne`, `lcmp`, `fcmpl`, `fcmpg`, `dcmpl`, `dc,pg`, `tableswitch`, `lookupswitch`, `ireturn`, `lreturn`, `freturn`, `dreturn`, `return`, `athrow`, `aaload`, `aastore`, `anearray`, `arraylength`, `baload`, `bastore`, `caload`, `castore`, `checkcast`, `daload`, `dastore`, `faload`, `fastore`, `getfield`, `getstatic`, `iaload`, `iastore`, `idiv`, `instanceof`, `invokeinterface`, `invokeespecial`, `invokeestatic`, `invokevirtual`, `irem`, `laload`, `lastore`, `ldc`, `ldc_w`, `ldc2_w`, `ldiv`, `lrem`, `monitorenter`, `monitorexit`, `multianewarray`, `new`, `newarray`, `putfielf`, `putstatic`, `saload`, `sastore`.

Cada vértice incluye un bloque básico de instrucciones formado por todas las intrucciones hasta el siguiente vértice o el final de los bytewords [395]. Ahora se deben construir el eCFG. Si  $u \in V$  y  $v \in V$  son dos bloques básicos:

- Si  $v$  sigue a  $u$  en el fichero de bycodes y  $u$  no termina en una rama incondicional entonces se debe añadir un arco  $(u, v)$ .
- Si la última instrucción de  $u$  es un rama condicional o incondicional a la primera instrucción en  $v$  se debe añadir un arco  $(u, v)$ .
- Se debe añadir un arco  $(u, v)$  desde un bloque `tableswitch` o `lookupswitch` al bloque básico de cada instrucción definido como objetivo en el `switch`.
- Cuando hay un subrutina, entonces se deben de añadir dos arcos. Un arco  $(u_1, v_1)$  desde el bloque básico de `jsr` o `jsr_w` al bloque básico objetivo. Y otro arco  $(u_2, v_2)$  desde el bloque básico que contiene la instrucción `ret` al bloque básico de la instrucción que sigue inmediatamente en el flujo de ejecución.
- Se añade un arco  $(u, v)$  desde el bloque básico de cada instrucción susceptible de lanzar una excepción a la entrada del bloque básico del correspondiente manejador.
- Finalmente, se añade un arco  $(u, v)$  desde el bloque básico de cada instrucción que sea susceptible de lanzar una excepción a un bloque que represente el final anómalo si la excepción no tiene manejador.

Zhao en [395] incluye una definición para extender el eCFG en un programa Java.

**Definition 14** *Un ICFG de un programa Java parcial o completo es una tupla  $(G_1, \dots, G_k, C, R)$  donde  $(G_1, \dots, G_k)$  son grafos de flujo que representan los métodos del programa.  $C$  es el conjunto de los arcos de llamada/salida, y  $R$  es el conjunto de arcos de retorno/entrada. Un ICFG de un programa parcial satisface las siguientes características:*

- *Existe una relación uno a uno entre  $C$  y  $R$ . Cada llamada es un arco de la forma  $(u, v_{1G_i}) \in C$  y su correspondiente arco de retorno de la forma  $(v_{1G_i}) \in R$ , donde  $u \in V_{G_i}$ , y  $G_i \in g$  y  $v_{1G_i}$  son el vértice inicial y final respectivamente, de algún  $G_j \in g$ .*
- *$g$  contiene dos vértices distintos: uno inicial  $v_{1G} = v_{1G_i}$  y otro final  $v_{FG} = v_{FG_i}$ ,  $G_i \in g$ .*

### 11.3.2.1

### Cálculo del WCET mediante la técnica IPET

El método para el análisis del WCET basado en rutas consiste en transformar el ICFG en un grafo ponderado, donde el WCET de cada bloque básico se

utiliza como peso y los arcos se anotan con la frecuencia de ejecución. La propuesta específica explicada en [102] consiste en cuatro pasos:

1. Encontrar la ruta más larga en el grafo utilizando un algoritmo estándar para grafos.
2. Comprobar que la ruta es posible.
3. Si la ruta no es posible, excluirla del grafo y volver al primer paso.
4. Cuando la posible ruta más larga se encuentre, su longitud corresponde con el WCET de la tarea.

La complejidad para el análisis de un grafo de este tipo es exponencial y dependiente de la profundidad del grafo lo que supone una dificultad en tareas grandes [164]. Esto significa que puede ser necesaria gran cantidad de tiempo de ejecución para resolverlo, por lo que en nuestro caso usamos una mejora denominada Implicit Path Enumeration Technique (IPET) y presentada por Li y Malik en [218].

En principio, el WCET se calcula sumando las frecuencias de ejecución en el peor caso por sus costes. Gracias a la tabla obtenida en el paso previo,  $\mathcal{R}(\mathcal{I}_{205}, N_{total})$  donde cada fila  $\mathcal{I}_i$  corresponde con un bytecode y las columnas corresponden con el coste en cada nodo  $N$  del sistema, se conoce el coste de cada bytecode en los diferentes nodos. Por lo tanto, ahora es necesario calcular el coste de cada bloque básico  $BB$  en cada nodo  $k$ :

$$\tau_{BB} = \sum_{j=1}^M \mathcal{R}_j^k \quad (11.1)$$

donde  $M$  es el número total de instrucciones en el bloque básico  $BB$  y  $\mathcal{R}_j^k$  representa el coste de cada instrucción individual  $j$  en el nodo de computación  $k$ .

De nuevo, se obtiene una tabla  $\mathcal{R}(BB_{total}, N_{total})$  donde cada fila corresponde a cada  $BB$  y la columna es el coste  $\tau_{BB}$  en el nodo  $N$ .

Finalmente, se puede definir el siguiente problema ILP:

$$\max WCET_{BB} = \sum_{i=1}^N \tau_i x_i \quad (11.2)$$

donde  $N$  es el número de bloques básicos,  $x_i$  es la frecuencia de ejecución del bloque básico  $i$  y gracias a la tabla previa  $\mathcal{R}$ , conocemos el valor de  $\tau_i$ . El WCET se debe calcular para cada nodo  $k$  del sistema.

Para calcular el WCET para cada  $BB$ , debe establecerse los límites en la frecuencia de ejecución de cada bloque usando la anotación de los arcos del grafo. La ecuación 11.2 resultante de aplicar el método IPET se maximiza para quedarnos con el peor de los casos. Los límites pueden expresarse usando una serie de restricciones basándose en el control del flujo del programa. Con la técnica IPET, éstas son:

- Restricciones estructurales: se refieren a aquellas restricciones relacionadas con el flujo como pueden ser ramas condicionales. Se pueden obtener del ICFG.
- Restricciones funcionales: se refieren a las restricciones en el comportamiento del flujo de control como los bucles. Se pueden obtener de las anotaciones del código.

### Restricción 1:

Las restricciones estructurales pueden expresarse mediante ecuaciones estableciendo la frecuencia de ejecución de cada bloque básico  $x_0, \dots, x_B$  como la suma de todos los arcos de entrada o la suma de todos los arcos de salida. Por tanto, la frecuencia  $x_i$  para un bloque dado  $i$  es la ecuación:

$$x_i = \sum_{j \in \mathcal{I}_i} d_j = \sum_{k \in \mathcal{O}_i} d_k \quad (11.3)$$

donde  $\mathcal{I}_i$  es el conjunto de todos los arcos o llamadas de entrada a un bloque básico  $i$ , y  $\mathcal{O}_i$  es el conjunto de todos los arcos o llamadas de salida desde un bloque básico  $i$ . Mediante la repetición de este proceso para cada bloque básico, se construyen el conjunto de las restricciones estructurales.

### Restricción 2:

La restricción funcional se refiere a la acotación de los bucles. Un bucle consiste en una condición de entrada y el cuerpo que constituye un bloque básico. En una restricción funcional el número de arcos desde el cuerpo del bucle a la condición debe ser menor o igual que el número de arcos de entrada a la condición que no se inician en el cuerpo, multiplicado por el valor que acota cada arco. Esto se expresa en la siguiente ecuación [320]:

$$\sum_{j \in \mathcal{C}_h} d_j \leq n \sum_{k \in \mathcal{E}_h} d_k \quad (11.4)$$

donde  $\mathcal{C}_h$  es el conjunto de arcos desde el cuerpo del bucle a la condición del bucle y  $\mathcal{E}_h$  es el conjunto de arcos de entrada a la condición del bucle que no se originan en el cuerpo del bucle. Finalmente,  $n$  es el valor que acota el número de veces que se puede ejecutar cada arco.



Las ecuaciones resultantes (11.3.2.1, 11.3.2.1) se resuelven usando ILP, donde se toma el máximo valor de las frecuencias obteniendo así el WCET. Resolver estas ecuaciones generalmente es un problema de complejidad NP-completo, aunque Li and Malik en [218] muestran que la naturaleza del problema resulta frecuentemente en ecuaciones que pueden resolverse con un orden de complejidad polinomial. En nuestro caso, se usa el algoritmo del Simplex para obtener las soluciones para el problema de programación lineal planteado en las restricciones (11.3.2.1, 11.3.2.1) y la función objetivo (11.2).

El WCET del programa completo es la suma de todos los  $BB$  que lo forman (11.5):

$$WCET_P = \sum_{b=1}^M WCET_b \quad (11.5)$$

Finalmente, se obtiene un vector  $FV$  que será necesario para la planificación global. Este  $FT$  almacena el  $WCET_P$  para cada nodo  $N$  del sistema. El procedimiento completo puede resumirse en el siguiente algoritmo (7):

---

**Algorithm 7** Análisis del código
 

---

```

1: procedure EVALUACIONALTONIVEL( $\mathcal{R}(I_{205}, N_{total})$ )
2:    $ICFG \leftarrow generateInterproceduralControlFlowGraph(file.class)$ 
3:    $ICFG_w \leftarrow getWeights(ICFG)$ 
4:   for  $n = 1$  to  $N$  do
5:     for  $BB = 1$  to  $totalBB$  do
6:        $\tau_{BB} \leftarrow costBB(\mathcal{I}_1, \dots, \mathcal{I}_n)$ 
7:        $\mathcal{R}(BB, n) \leftarrow \mathcal{R}(\tau_{BB}, n)$ 
8:        $WCET_{BB} \leftarrow simplexMethod(\sum_{i=1}^N \tau_i x_i, constraints)$ 
9:        $WCET_P \leftarrow WCET_P + WCET_{BB}$ 
10:       $FV(n) \leftarrow WCET_P$ 
11:    end for
12:  end for
13:  return  $FV(N)$ 
14: end procedure

```

---

11.3.3

 Adaptación estadística del WCET en tiempo de ejecución
 

---

Las sucesivas ejecuciones de la tarea permiten mejorar la precisión de la estimación mediante técnicas estadísticas. Esto significa que la estimación

puede ajustarse estadísticamente si hay suficientes muestras de ejecución de la tarea.

El análisis estadístico permite estrechar el intervalo de variación del WCET estimado y del WCET efectivo en ejecución, evitando así el uso de un límite superior demasiado amplio. Este proceso se puede llevar a cabo aplicando un análisis del intervalo de confianza a la muestra de las diferentes ejecuciones. La distribución estadística *t - student* se puede usar si la muestra contiene más de 30 valores pero el problema reside en que la muestra debe cumplir varias restricciones para que el intervalo de confianza sea realista. Por tanto, se opta por otra alternativa como es el análisis de valores atípicos. Un valor atípico  $a_i$  o  $a_j$  se define de acuerdo a las siguientes ecuaciones (11.6) (11.7):

$$a_i < Q_1 - 3 \cdot (Q_3 - Q_1) \quad (11.6)$$

$$a_j < Q_3 + 3 \cdot (Q_3 - Q_1) \quad (11.7)$$

donde  $Q_1$  y  $Q_3$  representan el primer y el tercer cuartil de la muestra, respectivamente.

El valor de WCET se calcula de acuerdo la valor superior y se define de la siguiente manera:

$$WCET = Q_3 + 3 \cdot (Q_3 - Q_1) \quad (11.8)$$

## Modelo de planificación en tiempo real y distribución de tareas

### 11.4

La figura 11.3 muestra el STR global. El algoritmo FPS se usa para la planificación dentro de cada nodo y el modelo propuesto se aplica para la planificación global.

Usando técnicas de optimización computacional, el modelo propuesto obtiene una distribución de tareas con el número mínimo de nodos. A cada nodo se le asignan los agentes con los roles responsables de realizar cada tarea. En caso de sobrecarga en algún nodo, el sistema replanifica y si no existen más nodos disponibles el sistema usa el parámetro  $H$  que indica la criticidad e importancia de una tarea.

Cada tarea se caracteriza por los siguientes parámetros:  $C$  es el WCET;  $D$  es el plazo para completar la ejecución;  $I$  indica posibles interferencias en la comunicación que puedan causar retraso;  $\rho$  indica el factor de utilización de cada tarea en un nodo;  $\mathcal{P}$  es el periodo mínimo entre dos llegadas consecutivas de la tarea (periodo);  $H$  es la importancia o criticidad (sólo se tiene en cuenta en caso de recursos insuficientes). El número total de tareas en el sistema se denota con  $\mathcal{T}$ . Este valor se conoce antes de cada replanificación y distribución es valor y el plan y la distribución es viable mientras  $\mathcal{T}$  no se altere. Cualquier cambio en el conjunto de tareas  $\mathcal{T}$  conlleva una replanificación para asegurar que las tareas son todavía planificables cumpliendo las restricciones temporales.

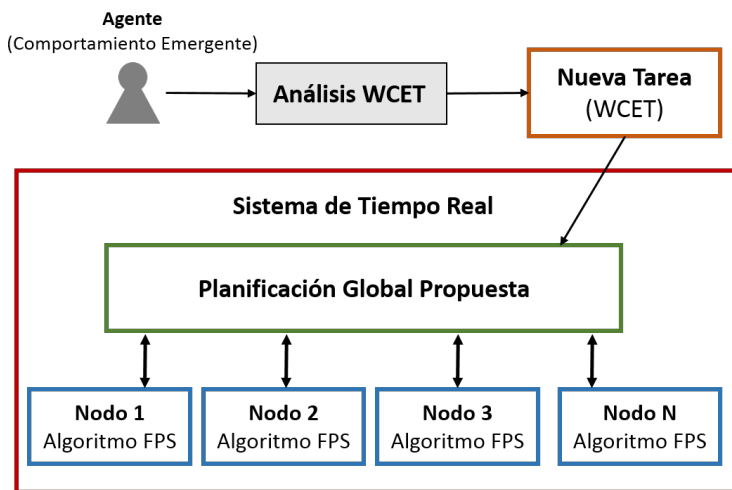


Figure 11.3: Vista general del STR

Una de las restricciones básicas del modelo de optimización deriva del test de viabilidad del FPS. Para verificar la viabilidad de un plan tomamos el test de utilización propuesto por Lui y Layland [221]. Es el test más restrictivo pero a su vez el más fácil de aplicar lo que reduce el tiempo de ejecución. Según este test, un conjunto de tareas periódicas  $\mathcal{T}$  es planificable con el algoritmo FPS si se cumple la ecuación 11.9.

**Theorem 2 Teorema 1:** *El conjunto de las tareas  $\mathcal{T}$  es planificable con el RM si:*

$$\sum_{i=1}^N \rho_i \leq N(2^{1/N} - 1) \quad (11.9)$$

donde:

$$\rho_i = C_i/T_i \quad (11.10)$$

El límite de planificabilidad del FPS depende del número de tareas y decrece con  $N$  [63]. Además, es importante resaltar que:

$$\lim_{n \rightarrow \infty} N(2^{1/N} - 1) = \ln 2 \simeq 0.693 \quad (11.11)$$

para valores altos de  $N$ , el límite se acerca asintóticamente a 69.3%. Esto significa que cualquier tarea puede ser distribuida en un nodo que siga el FPS si  $\rho \leq 0.69$ , pero no todas las tareas pueden ser distribuidas y por lo tanto, seguir un modelo planificable si  $0.69 < \rho \leq 1$  [63].

Se asume un modelo organizacional donde cada rol tiene asignados un conjunto de servicios que determinan las capacidades del agente. Con el modelo de distribución de tareas presentado, es posible obtener una distribución de los diferentes roles necesarios para completar el trabajo dentro del tiempo establecido.

Para realizar la distribución de tareas, se minimiza el número de nodos necesarios:

$$\min \sum_{j=1}^N x^j \quad (11.12)$$

donde  $j$  identifica el nodo,  $N$  es el número total de nodos disponibles y  $x^j = \{0, 1\}$ .

Las restricciones que deben aplicarse al problema se describen a continuación teniendo en cuenta la siguiente notación:  $x_i^j$  donde  $i$  identifica la tarea y  $j$  el nodo;  $\mathcal{T}$  es el número total de tareas en el sistema;  $\rho_i^j$  es el factor de utilización de la tarea  $i$  en el nodo  $j$ , el cual se obtiene a partir de la ecuación 11.10.

Primero se establece el vínculo entre la función objetivo y las restricciones de la siguiente manera:

### Restricción 3:

$$\forall i (x^j \geq x_i^j) \quad (11.13)$$

Esto significa que por cada tarea  $i$ , si el nodo tiene una tarea asignada  $x_i^j$ , se sumará 1 a la función objetivo.

Ahora se debe tener en cuenta el factor de utilización de cada nodo, de manera que el total de las tareas asignadas al nodo en cuestión no superen dicho factor. Para formular esta restricción 4 se debe tener en cuenta la ecuación 11.9.

**Restricción 4:**

$$\forall j \sum_{i=1}^{\mathcal{T}} \rho_i^j x_i^j \leq \min\left(\sum_{i=1}^{\mathcal{T}} x_i^j (2^{(1/\sum_{i=1}^{\mathcal{T}} x_i^j)} - 1), 1\right) \quad (11.14)$$

Desarrollando la ecuación, se tienen  $N$  inecuaciones de la siguiente forma:

$$\begin{aligned} \rho_1^1 x_1^1 + \rho_2^1 x_2^1 + \rho_3^1 x_3^1 + \dots + \rho_{\mathcal{T}}^1 x_{\mathcal{T}}^1 &\leq \min\left(\sum_{i=1}^{\mathcal{T}} x_i^1 (2^{(1/\sum_{i=1}^{\mathcal{T}} x_i^1)} - 1), 1\right) \\ \rho_1^2 x_1^2 + \rho_2^2 x_2^2 + \rho_3^2 x_3^2 + \dots + \rho_{\mathcal{T}}^2 x_{\mathcal{T}}^2 &\leq \min\left(\sum_{i=1}^{\mathcal{T}} x_i^2 (2^{(1/\sum_{i=1}^{\mathcal{T}} x_i^2)} - 1), 1\right) \\ &\dots \leq \dots \\ \rho_1^N x_1^N + \rho_2^N x_2^N + \rho_3^N x_3^N + \dots + \rho_{\mathcal{T}}^N x_{\mathcal{T}}^N &\leq \min\left(\sum_{i=1}^{\mathcal{T}} x_i^N (2^{(1/\sum_{i=1}^{\mathcal{T}} x_i^N)} - 1), 1\right) \end{aligned}$$

La siguiente restricción (5) establece que la misma tarea  $i$  no se asigna a más de un nodo. Esta restricción se debe modificar en caso de permitir la replicación. En caso de error, la replicación es un mecanismo ampliamente usado en sistemas de tiempo real críticos.

**Restricción 5:**

$$\forall j \sum_{i=1}^N x_i^j = 1 \quad (11.15)$$

La restricción 6 garantiza que todas las tareas se asignen, al menos, a un nodo.

**Restricción 6:**

$$\forall j \sum_{i=1, j=1}^{\mathcal{T}, N} x_i^j = \mathcal{T} \quad (11.16)$$

Con las OV se trabaja con roles que pueden estar asociados a diversas tareas. Si el mismo rol debe ejecutar dos tareas diferentes con la siguiente restricción (7), la tarea  $i$  y la tarea  $k$  se asignan al mismo nodo  $j$ .

**Restricción 7:**

$$x_i^j = x_k^j \quad (11.17)$$

Finalmente, los valores de  $x_i^j$  se limitan con la restricciones 8 y 9. De esta forma, sólo pueden tomar el valor 0 ó 1. Si el nodo  $j$  tiene asignada la tarea  $i$  entonces  $x_i^j = 1$ , sino,  $x_i^j = 0$ .

**Restricción 8:**

$$x_i^j \leq 1 \quad (11.18)$$

**Restricción 9:**

$$x_i^j, \text{ entero} \quad (11.19)$$

**11.4.0.1**

## Método de resolución

El algoritmo de Frank-Wolfe [114] encuentra posibles soluciones a un problema de programación no lineal con restricciones lineales, pero en nuestro problema también contamos con restricciones no lineales por lo que debemos usar su extensión denominada MAP [24]. El algoritmo MAP extiende un problema general de programación no lineal haciendo aproximaciones lineales de las restricciones y de la función objetivo.

**Algorithm 8** Pasos del algoritmo MAP

- 1: Sea  $x^0 = (x_1^0, x_2^0, \dots, x_n^0)$  un solución candidata, normalmente posible o "casi" posible. Sea  $k = 0$ .
- 2: Calcular  $c_j$  y  $a_{ij}$  ( $i = 1, 2, \dots, m$ ), las derivadas parciales de la función objetivo y las restricciones evaluadas en  $x^k = (x_1^k, x_2^k, \dots, x_n^k)$ . Sea  $b_i^k = a_{ij}x^k - g_i(x^k)$ .
- 3: Solucionar el problema como una aproximación lineal (4.25) con  $b_i^k$  y  $x_j^k$  sustituyendo por  $b_i^0$  y  $x_j^0$  respectivamente. Sea  $x^{k+1} = (x_1^{k+1}, x_2^{k+1}, \dots, x_n^{k+1})$  su solución óptima, incrementar  $k$  a  $k + 1$  y volver al paso 1.

Si se aplica este método a nuestro problema, las ecuaciones de las restricciones

son:

$$\begin{aligned}
 \forall j \quad x^j &\geq x_i^j \\
 \sum_{j=1}^N x_i^j &= 1 \\
 \sum_{i=1, j=1}^{\mathcal{T}, N} x_i^j &= \mathcal{T} \\
 x_i^j &\leq 1 \text{ integer} \\
 x_i^j &\leq x_k^j \\
 \forall j \quad \sum_{i=1}^{\mathcal{T}} \frac{\partial g_j}{\partial x_i^j}(y) x_i^j &\leq \sum_{i=1}^{\mathcal{T}} \frac{\partial g_j}{\partial x_i^j}(y) y_i - g_i(y)
 \end{aligned}$$

donde:

$$\begin{aligned}
 \frac{\partial g_j}{\partial x_i^j} &= p_i^j \leq \sum_{i \neq k} x_k^j \frac{-1}{(\sum_{i=1} x_i^j)^2} \cdot 2 \sum_{i=1}^1 \frac{1}{x_i^j} \cdot \ln 2 \\
 &+ 2 \sum_{i=1}^1 \frac{1}{x_i^j} + x_i^j \frac{-1}{(\sum_{i=1} x_i^j)^2} \cdot 2 \sum_{i=1}^1 \frac{1}{x_i^j} \cdot \ln 2 - 1
 \end{aligned}$$

la ecuación 11.20 es el valor de la derivada parcial en el punto  $y$ .

$$\frac{\partial g_j}{\partial x_i^j}(y) \tag{11.20}$$

$g_i$  es la restricción  $j$ ,  $\mathcal{T}$  es el conjunto de tareas y  $y$  es el punto que corresponde con la solución actual.

#### 11.4.0.2

#### El método Branch and Bound

El método Branch and Bound [212] es una técnica de tipo "divide y vencerás". Para nuestro problema incluyendo la restricción 9 de la sección 11.4 fuerza a la solución a ser valores enteros. Nuestros experimentos muestran que esto es un proceso costoso en términos de tiempo de computación. Por ello, se usa la programación lineal relajada [3] para estimar una solución a la programación lineal pura. Esto significa que para una modelo de programación entera  $P$  se

obtiene un modelo reducido eliminando la restricción de que todas las variables deben ser enteras. Esto se denomina programación lineal relajada de  $P$ ,  $P_r$ . Primero, se obtiene una solución válida para el  $P_r$  con el método MAP pero posteriormente, es necesario aproximar los valores al entero  $\{0, 1\}$  más cercano. Para ello, se usa una función umbral  $\mathcal{T} = f(v_i)$  donde  $v_i$  es el resultado después de aplicar el método MAP y se usa el método Branch and Bound para el resto de resultados.

$$\mathcal{T} = f(v_i) = \begin{cases} 0 & \text{if } v_i \leq 0.15 \\ 1 & \text{if } v_i \geq 0.85 \\ \text{BranchAndBound} & \text{en otro caso} \end{cases}$$

Para el caso  $f(v_i) = 1$ , la valores  $v_i$  que cumple la condición umbral  $v_i \geq 0.85$  se ordenan decrecientemente y empezando por el valor más alto se aproximan a 1 mientras se cumplan las restricciones del modelo.

Si se desea expandir el árbol de manera más rápida es posible incluir varias restricciones en una misma ramificación. Se establecen dos criterios de parada para este método: cuando todas las posibilidades se han explorado o estableciendo un máximo temporal predeterminado para obtener una solución. En este último caso, la mejor solución obtenida hasta es momento será una solución válida pero puede que no la mejor solución al problema global.

Dado que se pretende minimizar la función objetivo, al final del proceso se elige el  $\min Z_i$  obtenido en el último paso.

## 11.5

### La plataforma PANGEA+RT

PANGEA+RT es la evolución de la plataforma PANGEA [393] [144] [392] hacia la robótica y el tiempo real. En los siguientes apartados se explicarán las modificaciones y extensiones incluidas en este trabajo de investigación dentro la plataforma PANGEA para conseguir su adaptación.

#### 11.5.1

#### El protocolo de comunicación

El módulo de comunicación de PANGEA+RT se mejora usando el protocolo MQ Telemetry Transport (MQTT) [170] en vez de el IRC que usaba



PANGEA. MQTT es un protocolo ligero basado en un modelo de suscripción con un servidor (broker) y clientes. Está diseñado para ser abierto, simple y fácil de implementar, lo que lo convierte en ideal para su uso en ambientes restringidos.

Algunas características del protocolo son [172]:

- La trama es pequeña (la longitud fija mínima son 2 bytes) y los intercambios de mensajes del sistema son mínimos por lo que el tráfico de red se reduce.
- Mediante la configuración de la calidad del servicio (QoS) es posible definir cómo de estrictos serán los controles entre el cliente y servidor para asegurarse de la recepción de los mensajes.
- El protocolo no determina el formato en el que se incluye el contenido específico del mensaje.
- El protocolo utiliza un patrón de publicación y suscripción lo que favorece el bajo acoplamiento. Los clientes no tienen que conocerse previamente entre sí.
- Abierto y no es necesaria licencia, por lo que es fácilmente adaptable a multitud de servicios, plataformas y sistemas operativos.
- Altamente escalable para permitir crear sistemas con multitud de dispositivos remotos.

#### 11.5.1.1

#### Formato básico del mensaje

MQTT permite diferentes tipos de formatos para los mensajes, pero todos los mensajes contienen una cabecera fija. La tabla de la figura 11.4 muestra su formato.

bit	7	6	5	4	3	2	1	0
byte 1	Tipo Mensaje				Flag DUP	Nivel QoS		Retain
byte 2...N	Longitud Restante							
byte N+1...M	Cabecera Variable: depende de Tipo Mensaje							
byte M+1...K	Carga Significativa: depende de Tipo Mensaje							

Figure 11.4: MQTT Message Format

Los campos incluidos son:

- Tipo de mensaje: se representa como un entero sin signo con 4 bits, los diferentes tipos pueden verse en la tabla 11.1.
- Dup: se establece este flag a 1 cuando el cliente o el servidor intentan reenviar un mensaje de tipo PUBLISH, PUBREL, SUBSCRIBE o UNSUBSCRIBE. Se aplica a mensajes donde el valor de QoS es mayor que 0 y por lo tanto, se requiere un mensaje de tipo acuse de recibo. Si se activa este bit, la cabecera variable debe incluir un identificador de mensaje.
- QoS: indica el nivel de garantía para la entrega de un mensaje PUBLISH.
- Retain: Se usa sólo en los mensajes tipo PUBLISH. Cuando un cliente envía un mensaje PUBLISH a un servidor, si este flag está activado el servidor debe almacenar el mensaje después de que sea enviado a los clientes suscritos. Cuando se establece una nueva suscripción, el último mensaje almacenado debe ser enviado al nuevo suscriptor.
- Remaining Length: Represents the number of bytes remaining within the current message, including data in the variable header and the payload.

Los números que se usan en el campo tipo de mensaje se muestran a continuación 11.1:

MNEMONIC	NUMBER	DESCRIPTION
CONNECT	1	Un cliente solicita la conexión con un servidor
CONNACK	2	Acuse de recibo Connect
PUBLISH	3	Comando para publicar un mensaje
PUBACK	4	Acuse de recibo Publish
PUBREC	5	Publish recibido (parte 1)
PUBREL	6	Publish publicado (parte 2)
PUBCOMP	7	Publish completado (parte 3)
SUBSCRIBE	8	Un cliente solicita unirse a un tema
SUSACK	9	Acuse de recibo Subscribe
UNSUBSCRIBE	10	Un cliente solicita cancelar una suscripción
UNSUBACK	11	Acuse de recibo Unsubscribe
PINGREQ	12	solicitud PING
PINGRESP	13	respuesta PING
DISCONNECT	14	Un cliente solicita la desconexión

**Table 11.1:** MQTT message types

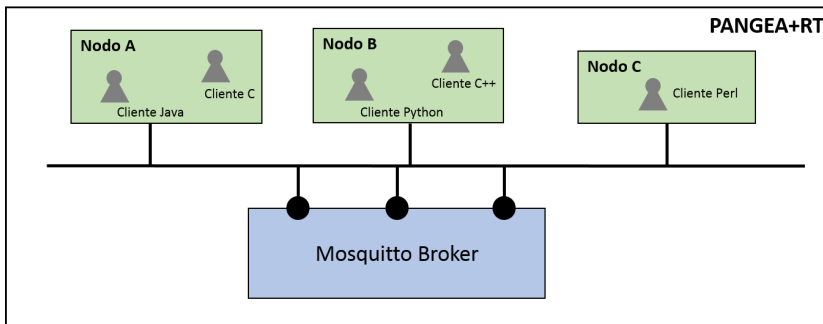
Cada uno de estos mensajes contiene además de una cabecera fija y una cabecera variable. Además, algunos de ellos contiene lo que se denomina carga significativa del mensaje, es decir, el mensaje en sí, como CONNECT, SUBSCRIBE, SUBACK y PUBLISH.

## 11.5.1.2 Servidores y clientes

En la figura 11.5 se muestra la arquitectura de comunicación con el protocolo MQTT.

**MQTT brokers:** un broker MQTT es un servidor que implementa el protocolo y hace de intermediario entre los clientes. Really Small Message Broker (RSMB) es la implementación del broker programada en C que se usa en PANGEA+RT.

**MQTT clientes:** para los clientes existen librerías que simplifican el proceso de desarrollo. En nuestro caso, se usa el proyecto Eclipse Paho (<http://www.eclipse.org/paho/>) que permite clientes en C y Java.



**Figure 11.5:** Arquitectura de comunicación con el protocolo MQTT

Otra característica importante es que clientes y servidores pueden estar implementados en diferentes lenguajes (<http://mqtt.org/software>), como Java, C, Arduino, Javascript, etc. La introducción del protocolo MQTT permite eliminar algunos agentes de la versión previa, PANGEA, como el CommunicationAgent, el SnifferAgent y el SubscribeAgent ya que sus funciones son automáticamente soportadas por el protocolo y el broker Mosquitto. El GatewayAgent tampoco es necesario porque el enlace entre MQTT y FIPA-ACL no es beneficioso ya que puede causar retrasos peligrosos para el correcto funcionamiento.

La sección de código 11.3 corresponde con un ejemplo de conexión y suscripción en Java.

```

1 public void PublishAndSuscribeBlocking() throws Exception {
2     MQTT mqtt = new MQTT();
3     mqtt.setHost("tcp://mosquito.fis.usal.es:1883");
4     BlockingConnection connection = mqtt.blockingConnection()
5     ;
6     connection.connect();

```

```

6   Topic[] topics = {new Topic(utf8("PresentationTopic"),
7       QoS.AT_LEAST_ONCE)};
8   byte[] qoses = connection.subscribe(topics);
9   connection.publish("PresentationTopic", "Hello".getBytes
10      (), QoS.AT_LEAST_ONCE, false);
11  Message message = connection.receive();
12  Assert.assertEquals("Hello", new String(message.
13      getPayload()));
14  message.ack();
15  connection.disconnect();
16 }

```

**Programming Code 11.3:** Conexión y suscripción en Java

### 11.5.2

### Agentes de PANGEA+RT

Como se ha explicado en la sección anterior, en PANGEA+RT el número de agentes predeterminados para el funcionamiento de la plataforma se reduce y sólo se mantienen los estrictamente necesarios.

- **OrganizationManager:** es el agente responsable de la gestión y control de la organizaciones y suborganizaciones. Verifica la entrada y salida de agentes y la asignación de roles. Trabaja junto al OrganizationAgent que es una especialización de este agente.
- **OrganizationAgent:** se introduce automáticamente en cada suborganización para ayudar al OrganizationManager y evitar sobrecargas. Se comunica con los agentes de tiempo real para verificar su funcionamiento y las restricciones temporales.
- **InformationAgent:** es el agente responsable del acceso a los datos y la información del sistema.
- **ServiceAgent:** es el encargado de controlar y asignar los servicios ofrecidos por los agentes. Funciona como el denominado Directory Facilitator (DF) definido en el estándar FIPA.
- **NormAgent:** asegura el cumplimiento de todas las normas definidas para el sistemas y las diferentes organizaciones.

En la figura 11.6 se puede observar la arquitectura para la distribución de los agentes de PANGEA+RT.

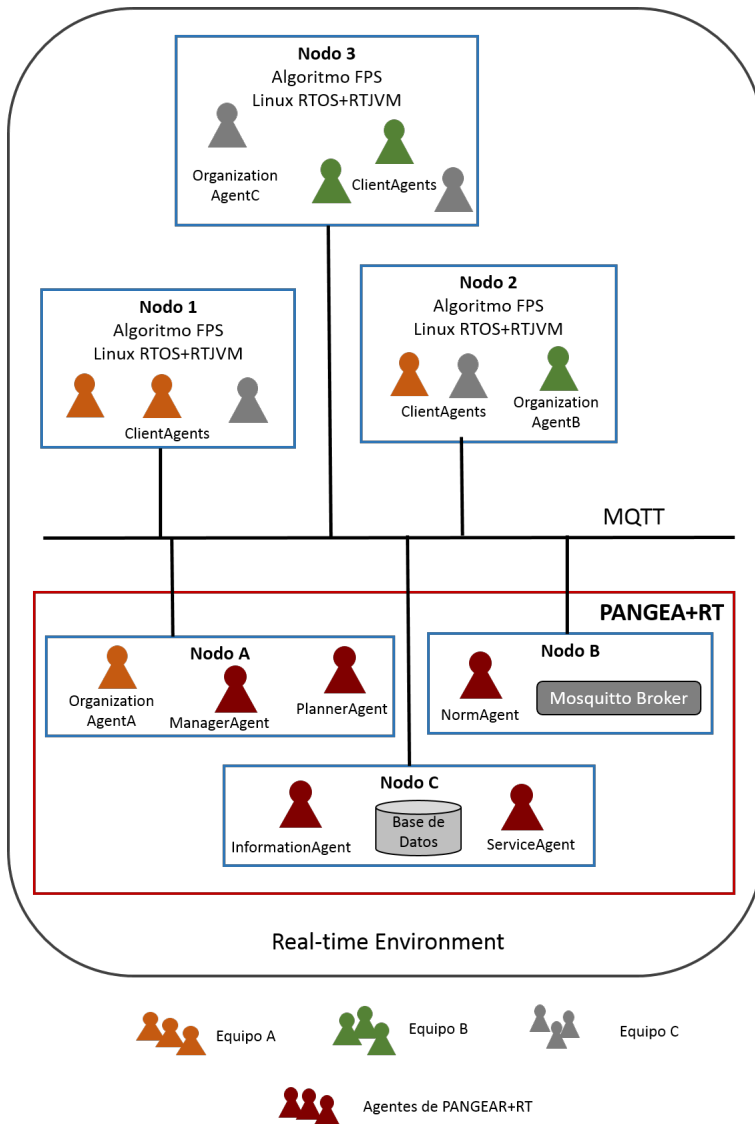


Figure 11.6: Agentes de PANGEA+RT

Ahora es necesario un nuevo rol al que denominamos PlannerAgent. Este rol es ejecutado por un agente que implementa el modelo presentado en la sección 11.4 para la planificación y la asignación de tareas. El OrganizationManager controla la entrada de nuevos agentes en el sistema y notifica al PlannerAgent el evento. El PlannerAgent está también en contacto con el OrganizationManager en caso de que la planificación falle.

En esta versión de la plataforma sólo se permiten agentes de tiempo real. La principal razón es que la comunicación entre un agente sin restricciones temporal y otro con ellas puede conllevar un mal funcionamiento si el agente sin restricciones se retrasa y no cumple la planificación. Por lo tanto, la interacción entre ellos se elimina para evitar fallos.

### 11.5.3

### Modificación de las clases

En PANGEA cada agente extiende de la clase abstracta denominada **Agent**. Para PANGEA+RT se crea una nueva clase abstracta que incluye las restricciones temporales, **AgentRT**. Lo mismo ocurre con la clase abstracta **Behaviour**, ahora reconvertida a la clase **BehaviourRT**.

En la figura 11.7 se pueden observar las principales clases de la implementación:

- **AgentRT**: define todos los métodos que los agentes de tiempo real deben implementar.
- **MqttCallback**: no es una clase específica de la plataforma pero todos los agentes deben extender de esta clase para usar el protocolo MQTT.
- **RealtimeThread**: esta clase tampoco es propia de la plataforma pero todos los agentes deben extender de ella para ejecutarse como hilos de tiempo real de Java.
- **BehaviourRT**: define el comportamiento del agente donde se incluye la tarea a realizar, puede ser periódica o aperiódica.
  - **PeriodicBehaviourRT**: se utiliza esta clase donde se especifica el plazo que debe cumplir el comportamiento de un agente con una tarea periódica junto con el periodo de ejecución.
  - **AperiodicBehaviourRT**: hasta el momento, PANGEA+RT sólo permite tareas espórativas y aperiódicas asignando *period = deadline*.

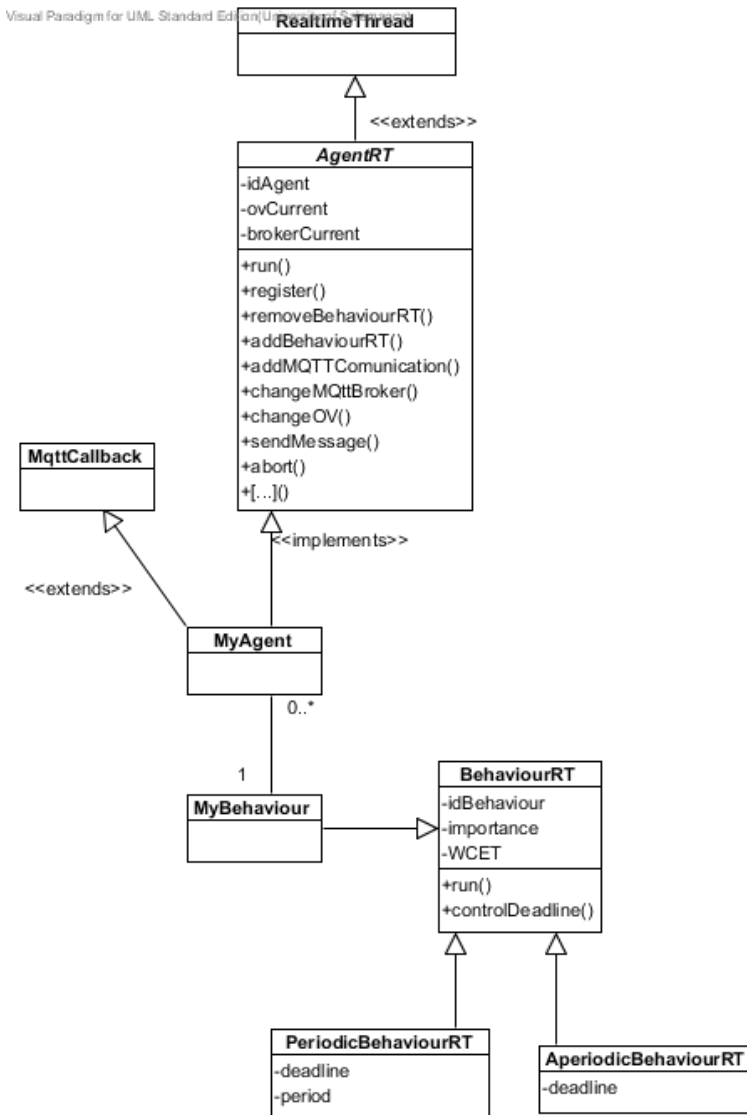


Figure 11.7: Clases principales de la plataforma

Es importante destacar el método `controlDeadline` de la clase **BehaviourRT**. Si las tareas programadas dentro del método `run` no finalizan cumpliendo las restricciones temporales (con el plazo deseado), `controlDeadline` genera un evento que debe ser tratado por un manejador y notificar la incidencia al **OrganizationAgent**. El atributo `WCET` permite almacenar la estimación temporal obtenida con el modelo previamente explicado.

A continuación, se presenta el esqueleto de una clase de un agente (11.5)

asociada a su comportamiento (11.4).

```

1 public class MyBehaviour extends PeriodicBehaviourRT
2 {
3     public MyBehaviour(int importante, long WCET, long
4         deadline, long period)
5     {
6         //Constructor
7         super();
8     }
9     public void run()
10    { //Task to carry out }
11
12    public controlDeadline()
13    { //Actions if the deadline is not fulfilled }
14 }

```

**Programming Code 11.4:** Ejemplo de una clase MyBehaviour básica

```

1 public class MyAgent extends MqttCallback implements AgentRT
2 {
3     private MyBehaviour behaviour;
4     private BlockingConnection connection;
5
6     public MyAgent(string ovCurrent, string brokerCurrent,
7         MyBehaviour behaviour)
8     {
9         //Constructor
10        this.behaviour=behaviour;
11        super();
12    }
13
14    public void run()
15    { //Actions before starting the periodic task of the
16        behaviour }
17
18    public void addMQTTComunication(string brokerCurrent)
19    {
20        //MQTT client
21        MQTT mqtt = new MQTT();
22        mqtt.setHost(brokerCurrent);
23        BlockingConnection connection = mqtt.
24            blockingConnection();
25        connection.connect();
26    }
27
28    public sendMessage(string message, string topic)
29    {
30        connection.publish(topic, message.getBytes(), QoS.
31            AT_LEAST_ONCE, false);
32    }
33
34    [...]
35 }

```



---

**Programming Code 11.5:** Ejemplo de la clase básica `MyAgent`

La clase `MyAgent` se ejecuta como un hilo de tiempo real de Java (11.6).

```
1 public static void main (String [] args)
2 {
3     MyAgent agent = new MyAgent();
4     agent.start();
5     try{
6         agent.join();
7     } catch(InterruptedExcepcion ie)
8         //Exception
9     }
10 }
```

**Programming Code 11.6:** Hilo de tiempo real de Java

## 11.6 Caso de estudio

---

En este apartado, se explica el caso de estudio práctico para evaluar el modelo y la plataforma desarrollados. El caso de estudio se basa en la colaboración de robots heterogéneos para tareas de vigilancia. Existen dos tipos de robots involucrados: HAWKS, que son UAVs (Unmanned Aerial Vehicle) y GECKOs, UGVs (Unmanned Ground Vehicle) desarrollados por el grupo BISITE. La meta del equipo o equipos involucrados es escanear grandes áreas de terreno, con este sistema el tiempo de búsqueda se reduce si lo comparamos con recursos humanos. Para el desarrollo de la infraestructura a bajo nivel se ha utilizado la plataforma PANGEA+RT [144] [392]. Y el sistema se ha probado en un parque de ocio llamado Valcuevo en la localidad de Salamanca.

En el campo de la robótica y especialmente, en este caso donde hay vehículos en movimiento, cumplir con las restricciones de tiempo real es importante. Este caso de estudio nos permite combinar el concepto de OV mediante la formación de equipos, el modelo propuesto para la planificación y distribución de tareas y la plataforma PANGEA+RT.

Los agentes trabajan dentro de sus correspondientes organizaciones dependiendo del equipo del que forman parte, para ello se configuran las normas y las capacidades asociadas a roles de los agentes. Cada agente puede tomar diferentes roles y gracias a la distribución de tareas, los roles pueden ser reasignados a diferentes agentes en diferentes nodos mejorando la eficiencia o simplemente

para resolver problemas de comunicación, paradas indeseadas de un nodo, etc.

Un equipo debe tener como mínimo los roles que se presentan a continuación. Pero puede suceder que varios agentes tomen el mismo rol, incrementando el número de agentes en el sistema. Esto es común en el caso del *InterfaceAgent* ya que la monitorización de HAWK, de GECKO y la del equipo puede realizarse desde varios nodos que deben incluir el *InterfaceAgent* para capturar los datos del *ControlAgent* de cada robot.

Los agentes necesarios para formar un equipo mínimo son:

*InterfaceAgent*: es el responsable de capturar los datos necesarios para la interacción entre los usuarios y los vehículos. Hay tres especializaciones de este rol. El *InterfaceGECKOAgent* se comunica con el correspondiente *ControlAgent* para obtener los datos de GECKO, y la información de sus sensores se muestra en la interfaz. El *InterfaceHAWKAgent* se comunica con el de HAWK para mostrar los datos en la interfaz. Finalmente, el agente *TeamInterface* interactúa con el *ControlAgent* de todos los vehículos y equipos y con el *MovementControllerAgent* para obtener y mostrar la posición de los vehículos en cada momento. Además, este agente recibe una alertas del agente *AlertColisionAgent* en caso de que dos equipos vayan a colisionar o del *MovementControllerAgent* si sus áreas de escaneo se solapan.

*ControlAgent*: este agentes se ejecuta dentro de cada vehículo. Encapsula la lógica y funcionalidad necesaria para controlar el vehículo. Recibe información de los agentes encargados de sondear los sensores y se comunica también con los agentes interfaz. Por tanto, es el agente base para el control de los movimientos y captura de imágenes del vehículo. *SensorAgent*: este agente se ejecuta también dentro de cada vehículo y tiene diferentes especializaciones dependiendo del vehículo y del sensor. Su principal misión es captar información del medio para su interpretación.

*MovementControllerAgent*: este rol puede desplegarse en cualquier nodo del sistema y es el responsable de realizar los cálculos correspondientes para el movimiento sincronizado de GECKO y HAWK.

*ScanningRouteAgent*: este rol también puede desplegarse en cualquier nodo del sistema y se encarga de calcular los mencionados waypoints que debe seguir HAWK. Recibe la información del agente *MovementControllerAgent*.

*RouteAreaAgent*: éste también puede desplegarse en cualquier nodo del sistema y controlas las posibles desviaciones y errores que se produzcan en el movimiento sincronizado. También se encarga de almacenar la información para presentar en la interfaz los resultados finales de la misión.

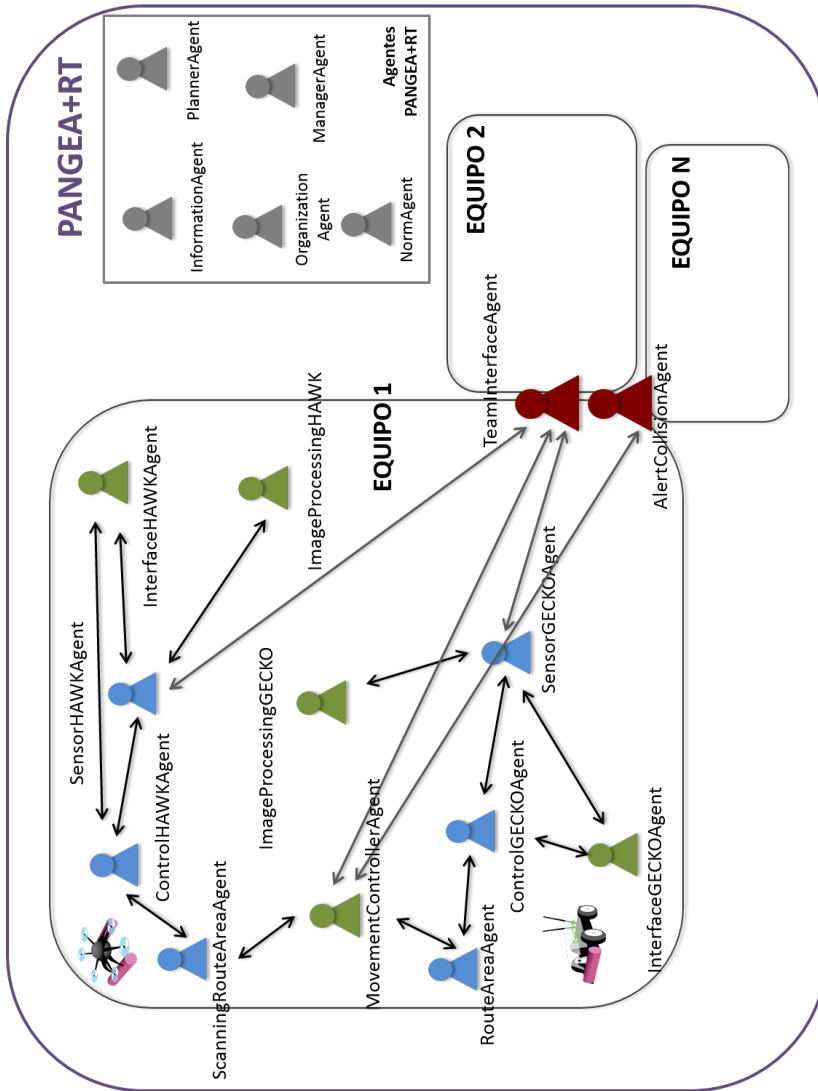


Figure 11.8: OV propuesta para el despliegue del caso de estudio

Existen también dos roles obligatorios que se encargan del procesamiento de las imágenes que transmiten HAWK y GECKO. Sin embargo, el caso particular del tratamientos de imágenes se presenta en el trabajo [67].

*ImageProcessorAgent:* se encarga de reconocer patrones de figuras humanas en las imágenes. Existen también dos especializaciones de este agente, *ImageProcessingHAWKAgent* y *ImageProcessingGECKOAgent*, y puesto que las imágenes

que capta HAWK son hacia el suelo y las que capta GECKO son frontales. Las imágenes se transmiten de la cámara al agente usando el protocolo FPV, conocido también como RPV [266] [376], muy usado para controlar vehículos por radiofrecuencia desde el punto de vista del piloto.

En el sistema se incluye un agente con rol *AlertColisionAgent* que se comunica con el *MovementControllerAgent* de cada equipo. Este agente recibe las coordenadas en las que HAWK se posicionará en el próximo movimiento de escaneo. Si otro HAWK ha escaneado previamente un porcentaje superior al configurado por el usuario, el *AlertColisionAgent* lanza una alerta para realizar un nuevo cálculo de la zona a escanear.

La figura 11.8 presenta un esquema general de los roles que los agentes pueden tomar. Los agentes en azul pueden desplegarse en cualquier nodo del sistema mientras que los agentes en verde deben ejecutarse dentro del vehículo correspondiente (HAWK o GECKO, dependiendo de la proximidad del agente al robot en el dibujo).

### 11.6.1

#### Despliegue de los agentes propuestos

Para la ejecución de los agentes se dispone de 3 tipos diferentes de nodos con las siguientes características:

- Tipo 1: Intel Core 2 Duo a 3.00 GHz y 4Gb de RAM. El sistema operativo es un Fedora 20 kernel 3.10.22 y el parche de tiempo real instalado es RT-PREEMPT (patch-3.10.22.rt20). La RTJVM es la JamaicaVM Personal Edition 6.2.
- Tipo 2: i7-3630QM a 3.40 GHz y 8Gb de RAM. El sistema operativo es un Fedora 19 kernel 3.12.11 y el parche de tiempo real instalado es RT-PREEMPT (patch-3.12.11.rt19). La RTJVM es la JamaicaVM Personal Edition 6.2.
- Tipo 3: Raspberry Pi Single Board Computer (SBC) con procesador a 700MHz y 512MB de RAM con el kernel Raspbian 3.12.1 y el parche de tiempo real RT-PREEMPT (patch-3.12.1-rt4). La RTJVM es la FijiVM Academic.

Se dispone de 10 nodos, el tipo de nodo que determina sus características se muestra en la tabla 11.2.

Los agentes de las interfaces (*TeamInterfaceAgent*, *InterfaceHAWKAgent*, *InterfaceGECKOAgent*) se ejecutan en nodos diferentes ya que las restricciones de tiempo real no se aplican. Los propios agentes de la plataforma PANGEA+RT

<b>Nodo 1</b>	Tipo 3 (Raspberry Pi)
<b>Nodo 2</b>	Tipo 3 (Raspberry Pi)
<b>Nodo 3</b>	Tipo 2 (Intel i7-3630QM)
<b>Nodo 4</b>	Tipo 1 (Intel Core 2 Duo)
<b>Nodo 5</b>	Tipo 1 (Intel Core 2 Duo)
<b>Nodo 6</b>	Tipo 3 (Raspberry Pi)
<b>Nodo 7</b>	Tipo 3 (Raspberry Pi)
<b>Nodo 8</b>	Tipo 2 (Intel i7-3630QM)
<b>Nodo 9</b>	Tipo 1 (Intel Core 2 Duo)
<b>Nodo 10</b>	Tipo 1 (Intel Core 2 Duo)

**Table 11.2:** Relación entre los nodos y sus características

también se ejecutan en otro nodo específico ya que tampoco es necesaria su actuación en tiempo real. Como se ha explicado en la sección 11.5, estos agentes son: el *OrganizationAgent*, el *ManagerAgent*, el *NormAgent*, el *InformationAgent* y el *ServiceAgent*. Sin embargo, el *PlannerAgent* se incluye en el sistema de tiempo real para que el tiempo de ejecución del modelo incluyendo el algoritmo MAP y el método Branch and Bound puedan acotarse temporalmente. Para esto, el *PlannerAgent* utiliza el porcentaje de utilización restante en alguno de los nodos.

En el modelo propuesto, se utiliza la condición de planificabilidad para el RM ó FPS propuesta por Liu y Layland [221], y que se tiene en cuenta en la restricción 4 del modelo. El límite de planificabilidad decrece con el número de tareas  $N$ , como Butazzo [63] demuestra en 11.21.

$$\lim_{N \rightarrow \infty} N(2^{1/N} - 1) = \ln 2 \simeq 0.693 \quad (11.21)$$

Esto significa que la capacidad de cada nodo aplicando el modelo propuesto es del 69.3%. Cuando el *PlannerAgent* necesita tiempo de ejecución, usa el porcentaje restante de cada nodo. En la próxima sección, se presentan los resultados relacionados con los agentes y nodos del caso de estudio.

## 11.6.2

## Resultados

Como se ha explicado previamente y cómo se muestra en la figura 11.8, cada equipo está compuesto por 9 agentes y el agente *AlertCollisionAgent* que es común para todos los equipos. Se dispone como máximo de 10 nodos presentados en la tabla 11.2.

Por cada equipo se añaden cuatro restricciones más. Esto se debe a que el nodo 1 corresponde a la Raspberry Pi de GECKO y los agentes *ControlGECKOAgent* y *SensorGECKOAgent* se deben ejecutar obligatoriamente en este nodo. Además, el nodo 2 corresponde a la Raspberry Pi de HAWK y los agentes *ControlHAWKAgent* y *SensorHAWKAgent* deben obligatoriamente ejecutarse en este nodo. Las restricciones se muestran en la ecuación 11.22.

$$\begin{aligned} x_1^1 = 1, x_2^1 = 1, x_3^2 = 1, x_4^2 = 1 \\ x_{10}^6 = 1, x_{11}^6 = 1, x_{12}^7 = 1, x_{16}^7 = 1 \end{aligned} \quad (11.22)$$

La figura 11.9 presenta el factor de utilización  $\rho_i^j$  donde:  $i$  representa la tarea y  $j$  el nodo de computación. Los resultados se obtienen mediante la aplicación del modelo propuesto de estimación para el WCET. El valor  $-1$  se asigna a los pares tarea-nodo donde dichas tareas no pueden ejecutarse de acuerdo con las ecuaciones 11.22.

Con estos valores, se puede aplicar el modelo propuesto para la planificación y asignación de tareas, los resultados se muestran en la tabla 11.3. En la primera columna se encuentran los nodos y la segunda el conjunto de tareas asignadas al correspondiente nodo. En la cuarta columna, se encuentra el valor final del factor de utilización,  $\rho^j$ , cuyo cálculo se realiza utilizando los valores  $\rho_i^j$  de la figura 11.9 y la fórmula 11.23.

$$\rho^j = \sum_i^{N_j} \rho_i^j \quad (11.23)$$

donde  $N_j$  es el número de tareas asignadas al nodo  $j$ .

La última columna corresponde con la capacidad sobrante en cada nodo,  $\phi^j$ . Para este cálculo se utiliza el teorema de Liu y Layland y la ecuación 11.9, así se obtiene la siguiente fórmula 11.24:

$$\phi^j = \begin{cases} N_j(2^{1/N_j} - 1) - \rho^j & \text{if } N > 0 \\ 1 & \text{if } N = 0 \end{cases} \quad (11.24)$$

	Nodo 1		Nodo 2		Nodo 3		Nodo 4		Nodo 5		Nodo 6		Nodo 7		Nodo 8		Nodo 9		Nodo 10	
	Tipo 3	Tipo 3	Tipo 3	Tipo 3	Tipo 2	Tipo 2	Tipo 1	Tipo 1	Tipo 1	Tipo 1	Tipo 3	Tipo 3	Tipo 3	Tipo 3	Tipo 2	Tipo 2	Tipo 1	Tipo 1	Tipo 1	Tipo 1
<b>EQUIPO 1</b>	0.02	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Agente 1 ControlGCKOAgent	0.13	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Agente 2 SensorGCKOAgent	-1	0.02	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Agente 3 ControlHAWKAgent	-1	0.13	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Agente 4 SensorHAWKAgent	1.85	1.85	0.41	0.41	0.36	0.36	0.36	0.36	0.36	1.85	1.85	1.85	1.85	0.41	0.41	0.36	0.36	0.36	0.36	0.36
Agente 5 ImageProcessingHAWKAgent	1.85	1.85	0.41	0.41	0.36	0.36	0.36	0.36	0.36	1.85	1.85	1.85	1.85	0.41	0.41	0.36	0.36	0.36	0.36	0.36
Agente 6 - ImageProcessingGCKOAgent	1.72	1.72	0.37	0.37	0.33	0.33	0.33	0.33	0.33	1.72	1.72	1.72	1.72	0.37	0.37	0.33	0.33	0.33	0.33	0.33
Agente 7 ScanningRouteAgent	0.98	0.98	0.22	0.22	0.18	0.18	0.18	0.18	0.18	0.98	0.98	0.98	0.98	0.22	0.22	0.18	0.18	0.18	0.18	0.18
Agente 8 - MovementControllerAgent	0.93	0.93	0.18	0.18	0.15	0.15	0.15	0.15	0.15	0.93	0.93	0.93	0.93	0.18	0.18	0.15	0.15	0.15	0.15	0.15
Agente 9 RouteAreaAgent	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Agente 10 ControlGCKOAgent	-1	-1	-1	-1	-1	-1	-1	-1	-1	0.02	0.02	0.02	0.02	-1	-1	-1	-1	-1	-1	-1
Agente 11 SensorGCKOAgent	-1	-1	-1	-1	-1	-1	-1	-1	-1	0.13	0.13	0.13	0.13	-1	-1	-1	-1	-1	-1	-1
Agente 12 ControlHAWKAgent	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Agente 13 SensorHAWKAgent	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Agente 14 ImageProcessingHAWKAgent	1.85	1.85	0.41	0.41	0.36	0.36	0.36	0.36	0.36	1.85	1.85	1.85	1.85	0.41	0.41	0.36	0.36	0.36	0.36	0.36
Agente 15 ImageProcessingGCKOAgent	1.85	1.85	0.41	0.41	0.36	0.36	0.36	0.36	0.36	1.85	1.85	1.85	1.85	0.41	0.41	0.36	0.36	0.36	0.36	0.36
Agente 16 ScanningRouteAgent	1.72	1.72	0.37	0.37	0.33	0.33	0.33	0.33	0.33	1.72	1.72	1.72	1.72	0.37	0.37	0.33	0.33	0.33	0.33	0.33
Agente 17 MovementControllerAgent	0.98	0.98	0.22	0.22	0.18	0.18	0.18	0.18	0.18	0.98	0.98	0.98	0.98	0.22	0.22	0.18	0.18	0.18	0.18	0.18
Agente 18 RouteAreaAgent	0.93	0.93	0.18	0.18	0.15	0.15	0.15	0.15	0.15	0.93	0.93	0.93	0.93	0.18	0.18	0.15	0.15	0.15	0.15	0.15
Agente 19 AlertCollisionAgent	1.99	1.99	0.45	0.45	0.39	0.39	0.39	0.39	0.39	1.99	1.99	1.99	1.99	0.45	0.45	0.39	0.39	0.39	0.39	0.39
<b>EQUIPO 2</b>																				

Figure 11.9: Resultados del factor de utilización  $\rho_i^j$

El valor de la función objetivo es 9 y el nodo 5 se queda libre de tareas. Los nodos con mayor capacidad sobrante ( $\phi^j = 0.67$ ) son el 1, 2, 6 y 7, sin embargo, si se consulta la tabla de la figura 11.9 no existe la posibilidad de asignar otro agente a esos nodos ya que todas las tareas tienen  $\phi_i^j > 0.67$ .

Al nodo 8 se le asigna únicamente una tarea,  $i = 19$ . Para verificar el modelo, manualmente se adaptan los valores de  $\phi_{19}^j$  para comprobar que en este caso,

Nodo	Agente ( $\rho_i^j$ )	Factor de Utilización $\rho^j$	Capacidad Sobrante Actual $\phi^j$
1	1 (0.02) and 2 (0.13)	0.15	0.67
2	3 (0.02) and 4 (0.13)	0.15	0.67
3	7 (0.37) and 16 (0.37)	0.74	0.088427
4	8 (0.18), 9 (0.15), 17 (0.18) and 18 (0.15)	0.66	0.096828
5	Empty	0	1
6	10 (0.02) and 11 (0.13)	0.15	0.67
7	12 (0.02) and 13 (0.13)	0.15	0.67
8	19 (0.45)	0.45	0.55
9	6 (0.36) and 15 (0.36)	0.72	0.108427
10	5 (0.36) and 14 (0.36)	0.72	0.108427

**Table 11.3:** Resultados del modelo de distribución de tareas

la tarea se asigna a otro nodo y el número de nodos disminuye a 8. En la tabla 11.4 se muestran los nuevos valores.

Agent 19 AlertCollisionAgent		
	$\rho_{19}^j$ Antes	$\rho_{19}^j$ Después
<b>Nodo 3</b>	0.45	0.08
<b>Nodo 4</b>	0.39	0.06
<b>Nodo 5</b>	0.39	0.06
<b>Nodo 8</b>	0.45	0.08
<b>Nodo 9</b>	0.39	0.06
<b>Nodo 10</b>	0.39	0.06

**Table 11.4:** Modificación del parámetro  $\rho_{19}^j$

Con estos valores, el valor de la función objetivo es 8. Esto significa que 2 nodos sufren modificaciones, la tarea 19 se asigna al nodo 4 y el nodo 8 queda vacío. Para este caso, la solución obtenida es la mejor. Los resultados se muestran en la tabla 11.5.



Nodo	Agente ( $\rho_i^j$ )	Factor de Utilización $\rho^j$	Capacidad Sobrante Actual $\phi^j$
4	8 (0.18), 9 (0.15), 17 (0.18), 18 (0.15) and 19 (0.06)	0.72	0.023492
8	Empty	0	1

**Table 11.5:** Resultados del modelo de planificación y distribución de tareas

## 11.7 Conclusiones

En este trabajo de tesis doctoral se ha propuesto un modelo efectivo de estimación del WCET y de planificación y distribución de tareas. En primer lugar, con el modelo se calcula el WCET de las tareas asociadas a un agente que desea entrar a formar parte de una OV teniendo en cuenta restricciones de tiempo real. Una vez hecha dicha estimación, el modelo genera una planificación y una distribución de tareas que permite minimizar el número de nodos del sistema. Este modelo se diseñó específicamente para OVs donde los agentes llevan a cabo tareas con restricciones temporales lo que supone una novedad objetiva con respecto al estado del arte de la técnica.

Esta investigación aporta nuevas contribuciones hacia la integración de agentes con comportamientos emergentes que deben trabajar dentro de una OV de tiempo real. Después de un estudio exhaustivo de la literatura existente, éste es el primer modelo que cumple con dichas características desarrollado para este tipo de sistemas. El proceso seguido desde la propuesta de la hipótesis hasta la implementación de la plataforma y la obtención de resultados cumple con la metodología exigida para un trabajo de investigación. La evaluación del modelo propuesto y las comparativas realizadas en el estado del arte, permiten afirmar que, en la actualidad, existen diversas plataformas de agentes, algunas orientadas a OVs ó a robótica pero ninguna combina tiempo real con OVs que cumplan con todas las características organizacionales.

En el marco de este trabajo se han usado diversas técnicas para conseguir resolver los pasos del modelo y las dificultades que éste presenta. Así, para la estimación del WCET, se han aplicado conceptos como los Java bytecodes y profundizado en el funcionamiento de una JVM. Se han utilizado técnicas de modelado de grafos como el ICFG y la técnica IPET para mejorar la estimación. Para la planificación y la distribución de tareas se ha aplicado el algoritmo FPS y su test de viabilidad, programación no lineal, el algoritmo MAP, el método Branch and Bound, etc. Con todos estos elementos hemos obtenido

un método novedoso y que, tras ser evaluado, ha proporcionado resultados que permiten concluir que el modelo permite demostrar la hipótesis inicial planteada: *es posible desarrollar un modelo de planificación y asignación de tareas que permita a los agentes pertenecientes a una OV rematar con éxito sus tareas dentro de las limitaciones temporales. Este modelo será formalizado teóricamente e implementado para evaluar los resultados.*

También como parte de este trabajo doctoral se ha diseñado y desarrollado la plataforma PANGEA+RT que proporciona soporte al modelo propuesto y obteniendo tres artefactos diferentes durante el proceso:

- PANGEA: orientada a OV.
- PANGEA+R: orientada a robótica colaborativa.
- PANGEA+RT: orientada a tiempo real.

Por lo tanto, se puede concluir que PANGEA+RT es una plataforma completa para gestionar OV con diferentes topologías y teniendo en cuenta los diferentes aspectos organizacionales. Puede incluir cualquier tipo de agente, incluso robóticos como en el caso de estudio gracias al protocolo ligero MQTT. También, incluye el modelo de planificación y distribución de tareas que permite desarrollar OVs en tiempo real cumpliendo con la dinamicidad y apertura de estos sistemas.

Finalmente, como se ha mostrado con el caso de estudio, se han obtenido buenos resultados experimentales que revelan el buen funcionamiento no sólo del modelo propuesto sino de la plataforma PANGEA+RT.

# LIST OF ACRONYMS

---

- ACL** Agent Communication Language. 22
- AI** Artificial Intelligence. 49, 50
- AMS** Agent Management System. 52, 54, 106
- APRIL** Agent Process Interaction Language. 52
- BDI** Belief-desire-intention agent architecture. 18, 19, 101, 106
- CFG** Control Flow Graph. 67, 70, 75, 76, 80
- DF** Directory Facilitator. 52, 103, 106, 112
- DMA** Deadline Monotonic Analysis. 41
- DPS** Dynamic Priority Scheduling. 44
- DRTSJ** Distributed Real-Time Specification for Java. 240
- eCFG** Extended Control Flow Graph. 76, 77, 78
- EDF** Earliest Deadline First Scheduling. 41, 42, 43, 45, 52, 65, 84, 85
- FCFS** First Come First Served. 40
- FIPA** Foundation for Intelligent Physical Agents. 19, 22, 54
- FPS** Fixed Priority Scheduling. 42, 43, 44, 61, 64, 65, 83, 84, 85, 86, 87
- FPV** First-person View. 201, 279
- GPS** Global Positioning System. 193, 194, 200, 203, 204, 207
- GRM** Global Rate-Monotonic. 44, 45
- HMAS** Holonic multi-agent system. 103
- ICFG** Interprocedural Control Flow Graph. 78, 81, 258
- ILP** Integer Linear Programming. 62, 79, 82, 221, 259
- IPET** Implicit Path Enumeration Technique. 78, 79, 220
- IRC** Internet Relay Chat. 101, 118, 119, 120, 121, 123, 124, 126, 127, 132, 148

- JADE** Java Agent DEvelopment Framework. 101, 104, 105
- jART** java Agent for Real-Time. 54
- JML** Java Modelling Language. 172
- JVM** Java Virtual Machine. 163, 164, 165, 166, 167, 169
- KQML** Knowledge Query and Manipulation Language. 22, 52
- MAP** Method of Approximation Programming. 91, 92, 94
- MARS** Multi-agent Robotic System. 140
- MAS** Multi-agent System. 4, 5, 7, 10, 11, 15, 19, 20, 21, 22, 23, 24, 25, 27, 28, 29, 33, 35, 49, 50, 51, 52, 53, 61, 99, 100, 103, 104, 106, 109, 128, 135, 136, 137, 138, 140, 142, 144, 146, 147, 162, 186, 197, 232
- MQTT** Message Queue Telemetry Transport. 148, 149, 150, 151, 152, 156, 157, 158, 159, 237
- Pfair** Proportionate-Fair Planning. 45, 46
- RDE** Robot Development Environment. 137
- RM** Rate Monotonic. 41, 42, 44, 65, 86, 212, 263
- RMI** Remote Method Invocation. 240
- RPV** Remote-person View. 201, 279
- RR** Round Robin. 40
- RSMB** Really Small Message Broker. 156
- RTA** Real-time Agent. 50, 51, 52
- RTAI** Real-time Artificial Intelligence. 53
- RTAIS** Real-Time Artificial Intelligence Systems. 49, 51
- RTJVM** Real-time Java Virtual Machine. 169, 170, 181
- RTMAS** Real-time Multi-Agent System. 50, 51
- RTS** Real-time System. 4, 5, 6, 8, 10, 34, 35, 36, 37, 40, 41, 43, 48, 49, 50, 52, 53, 54, 55, 60, 83, 236
- RTSJ** Real-Time Specification for Java. 165, 166, 167, 168, 169, 240
- SOA** Service-oriented Application. 109, 132

- 
- UAV** Unmanned Aerial Vehicle. 186, 188, 191, 192, 193, 194
- UGV** Unmanned Ground Vehicle. 186, 191, 192, 194, 201
- VO** Virtual Organization of agents. v, vii, 4, 5, 6, 7, 8, 9, 10, 11, 15, 27, 28, 29, 33, 39, 55, 59, 61, 63, 66, 89, 96, 100, 101, 102, 103, 104, 105, 106, 111, 124, 132, 134, 136, 142, 144, 145, 146, 158, 159, 161, 181, 185, 186, 187, 191, 197, 198, 213, 231, 232, 236, 237
- WCET** Worst Case Execution Time. 10, 11, 41, 59, 60, 62, 63, 67, 68, 70, 74, 75, 78, 79, 80, 82, 83, 84, 163, 167, 171, 173, 180, 182, 236, 239, 259, 260, 261
- WS** Web Service. 103, 109, 129, 132
- WSN** Wireless Sensor Networks. 148, 191



# BIBLIOGRAPHY

---

- [1] T. F. Abdelzaher and N. Bhatti. Web server qos management by adaptive content delivery. In *Quality of Service, 1999. IWQoS'99. 1999 Seventh International Workshop on*, pages 216–225. IEEE, 1999.
- [2] J. H. Ahnn. *The Robot control using the wireless communication and the serial communication*. PhD thesis, Cornell University, 2007.
- [3] F. A. Al-Khayyal. Generalized bilinear programming: Part i. models, applications and linear programming relaxation. *European Journal of Operational Research*, 60(3):306–314, 1992.
- [4] E. Albert, G. Puebla, and M. Hermenegildo. Abstraction-carrying code. In *Logic for Programming, Artificial Intelligence, and Reasoning*, pages 380–397. Springer, 2005.
- [5] F. E. Allen. Control flow analysis. In *ACM Sigplan Notices*, volume 5, pages 1–19. ACM, 1970.
- [6] L. O. Andersen. *Program analysis and specialization for the C programming language*. PhD thesis, University of Copenhagen, 1994.
- [7] J. S. Anderson and E. D. Jensen. Distributed real-time specification for java: a status report (digest). In *Proceedings of the 4th international workshop on Java technologies for real-time and embedded systems*, pages 3–9. ACM, 2006.
- [8] N. Ando, T. Suehiro, and T. Kotoku. A software platform for component based rt-system development: Openrtm-aist. In *Simulation, Modeling, and Programming for Autonomous Robots*, pages 87–98. Springer, 2008.
- [9] M. Annunziato and P. Pierucci. The emergence of social learning in artificial societies. In *Applications of evolutionary computing*, pages 467–478. Springer, 2003.
- [10] M. T. Anticono. A grasp algorithm to solve the problem of dependent tasks scheduling in different machines. In M. Bramer, editor, *Artificial Intelligence in Theory and Practice*, volume 217 of *IFIP International Federation for Information Processing*, pages 325–334. Springer US, 2006.
- [11] T. Arai, E. Pagello, and L. E. Parker. Editorial: Advances in multi-robot systems. *IEEE Transactions on robotics and automation*, 18(5):655–661, 2002.
- [12] T. Arai, E. Pagello, and L. E. Parker. Editorial: Advances in multi-robot systems. *IEEE Transactions on robotics and automation*, 18(5):655–661, 2002.
- [13] E. Argente, S. Giret, S. Valero, V. Julian, and V. Botti. Survey of mas methods and platforms focusing on organizational concepts. *Recent advances in Artificial Intelligence Research and Development*, 113:309, 2004.
- [14] E. Argente, V. Julian, and V. Botti. Multi-agent system development based on organizations. *Electronic Notes in Theoretical Computer Science*, 150(3):55–71, 2006.
- [15] A. Armbruster, J. Baker, A. Cuneo, C. Flack, D. Holmes, F. Pizlo, E. Pla, M. Prochazka, and J. Vitek. A real-time java virtual machine with applications in avionics. *ACM Transactions on Embedded Computing Systems (TECS)*, 7(1):5, 2007.
- [16] K. Arnold, J. Gosling, and D. Holmes. *The Java programming language*, volume 2. Addison-wesley Reading, 2000.

- [17] A. Artikis, L. Kamara, and J. Pitt. Towards an open agent society model and animation. In *Proceedings of the Agent-Based Simulation II workshop, Passau*, pages 48–55, 2001.
- [18] D. Aspinall, S. Gilmore, M. Hofmann, D. Sannella, and I. Stark. Mobile resource guarantees for smart devices. In *Construction and Analysis of Safe, Secure, and Interoperable Smart Devices*, pages 1–26. Springer, 2005.
- [19] Atego. Atego perc virtual machine. <http://www.atego.com/products/atego-perc/>, 2014.
- [20] G. Attiya and Y. Hamam. Task allocation for maximizing reliability of distributed systems: a simulated annealing approach. *Journal of parallel and Distributed Computing*, 66(10):1259–1266, 2006.
- [21] N. Audsley, A. Burns, M. Richardson, K. Tindell, and A. J. Wellings. Applying new scheduling theory to static priority pre-emptive scheduling. *Software Engineering Journal*, 8(5):284–292, 1993.
- [22] N. C. Audsley. Optimal priority assignment and feasibility of static priority tasks with arbitrary start times. Technical report, University of York, England, 1991.
- [23] N. C. Audsley, A. Burns, R. I. Davis, K. W. Tindell, and A. J. Wellings. Fixed priority pre-emptive scheduling: An historical perspective. *Real-Time Systems*, 8(2-3):173–198, 1995.
- [24] M. Avriel. *Nonlinear programming: analysis and methods*. Courier Dover Publications, 2012.
- [25] D. F. Bacon, P. Cheng, and V. T. Rajan. A real-time garbage collector with low overhead and consistent utilization. *ACM SIGPLAN Notices*, 38(1):285–298, 2003.
- [26] J. Baillieul and P. J. Antsaklis. Control and communication challenges in networked real-time systems. *Proceedings of the IEEE*, 95(1):9–28, 2007.
- [27] J. Bajo, J. M. Corchado, V. Botti, and S. Ossowski. Practical applications of agents and mas: methods, techniques and tools for open mas. *Journal of Physical Agents*, 3(2):1–2, 2009.
- [28] J. Bajo, J. A. Fraile, B. Pérez-Lancho, and J. M. Corchado. The thomas architecture in home care scenarios: A case study. *Expert Systems with Applications*, 37(5):3986–3999, 2010.
- [29] J. Bajo, V. Julián, J. M. Corchado, C. Carrascosa, Y. De Paz, V. Botti, and J. F. De Paz. An execution time planner for the artis agent architecture. *Engineering Applications of Artificial Intelligence*, 21(5):769–784, 2008.
- [30] S. Balakirsky, S. Carpin, A. Kleiner, M. I. Lewis, A. Visser, J. Wang, and V. A. Ziparo. Towards heterogeneous robot teams for disaster mitigation: Results and performance metrics from robocup rescue: Field reports. *J. Field Robot.*, 24(11-12):943–967, November 2007.
- [31] J. M. Banús. *Planificación global en sistemas multiprocesador de tiempo real*. PhD thesis, Universitat Politècnica de Catalunya, 2008.
- [32] G. Barthe, L. Beringer, P. Crégut, B. Grégoire, M. Hofmann, P. Müller, E. Poll, G. Puebla, I. Stark, and E. Vétillard. Mobius: Mobility, ubiquity, security: Objectives and progress report. In *In Trustworthy Global Computing'06, LNCS*. Citeseer, 2007.
- [33] M. Bartlett, I. Bate, and D. Kazakov. Challenges in relational learning for real-time systems applications. In *Inductive Logic Programming*, pages 42–58. Springer, 2008.



- [34] S. K. Baruah, N. K. Cohen, C. G. Plaxton, and D. A. Varvel. Proportionate progress: A notion of fairness in resource allocation. *Algorithmica*, 15(6):600–625, 1996.
- [35] M. Batalin and G. S. Sukhatme. Coverage, exploration and deployment by a mobile robot and communication network. *Telecommunication Systems*, 26(2-4):181–196, 2004.
- [36] I. Bate, G. Bernat, G. Murphy, and P. Puschner. Low-level analysis of a portable java byte code wcet analysis framework. In *Real-Time Computing Systems and Applications, 2000. Proceedings. Seventh International Conference on*, pages 39–46. IEEE, 2000.
- [37] I. Bate, G. Bernat, and P. P. Puschner. Java virtual-machine support for portable worst-case execution-time analysis. In *Object-Oriented Real-Time Distributed Computing, 2002. (ISORC 2002). Proceedings. Fifth IEEE International Symposium on*, pages 83–90. IEEE, 2002.
- [38] G. Beccari, S. Caselli, and F. Zanichelli. A technique for adaptive scheduling of soft real-time tasks. *RealTime Systems*, 30(3):187–215, 2005.
- [39] B. Beckert, R. Hähnle, and P. H. Schmitt. *Verification of object-oriented software: The KeY approach*. Springer-Verlag, 2007.
- [40] F. Bellifemine, A. Poggi, and G. Rimassa. Jade—a fipa-compliant agent framework. In *Proceedings of PAAM*, volume 99, page 33. London, 1999.
- [41] R. Bellman. Dynamic programming and lagrange multipliers. *Proceedings of the National Academy of Sciences of the United States of America*, 42(10):767, 1956.
- [42] G. Bernat. *Specification and Analysis of Weakly Hard Real-Time Systems*. PhD thesis, Universitat de les Illes Balears, 1998.
- [43] G. Bernat, A. Burns, and A. Wellings. Portable worst-case execution time analysis using java byte code. In *Real-Time Systems, 2000. Euromicro RTS 2000. 12th Euromicro Conference on*, pages 81–88. IEEE, 2000.
- [44] S. Bisht, A. Malhotra, and S. B. Taneja. Modelling and simulation of tactical team behaviour. *Defence Science Journal*, 57(6):853–864, 2007.
- [45] A. Biswas, S. Sen, and S. Debnath. Limiting deception in groups of social agents. *Applied Artificial Intelligence*, 14(8):785–797, 2000.
- [46] D. Blank, D. Kumar, L. Meeden, and H. Yanco. The pyro toolkit for ai and robotics. *AI magazine*, 27(1):39, 2006.
- [47] G. Boella, J. Hulstijn, and L. Van Der Torre. Virtual organizations as normative multiagent systems. In *System Sciences, 2005. HICSS'05. Proceedings of the 38th Annual Hawaii International Conference on*, pages 192c–192c. IEEE, 2005.
- [48] T. Bøgholm, C. Frost, R. R. Hansen, C. S. Jensen, K. S. Luckow, A. P. Ravn, H. Søndergaard, and B. Thomsen. Towards harnessing theories through tool support for hard real-time java programming. *Innovations in Systems and Software Engineering*, 9(1):17–28, 2013.
- [49] T. Bøgholm, R. Hansen, A. P. Ravn, B. Thomsen, and H. Søndergaard. A predictable java profile: rationale and implementations. In *Proceedings of the 7th International Workshop on Java Technologies for Real-Time and Embedded Systems*, pages 150–159. ACM, 2009.

- [50] T. Bøgholm, H. Kragh-Hansen, P. Olsen, B. Thomsen, and K. G. Larsen. Model-based schedulability analysis of safety critical hard real-time java programs. In *Proceedings of the 6th international workshop on Java technologies for real-time and embedded systems*, pages 106–114. ACM, 2008.
- [51] B. Bonet and H. Geffner. Planning with incomplete information as heuristic search in belief space. In *Proceedings of the Fifth International Conference on Artificial Intelligence Planning and Scheduling, Seattle, WA, AAAI Press*, pages 52–61, 2000.
- [52] R. H Bordini, J. F Hübner, and R. Vieira. Jason and the golden fleece of agent-oriented programming. In *Multi-agent programming*, pages 3–37. Springer, 2005.
- [53] R. H Bordini, J. F. Hübner, and M. Wooldridge. *Programming multi-agent systems in AgentSpeak using Jason*, volume 8. John Wiley & Sons, 2007.
- [54] E. Borowsky, R. Golding, A. Merchant, L. Schreier, E. Shriver, M. Spasojevic, and J. Wilkes. Using attribute-managed storage to achieve qos. In *Building QoS into distributed systems*, pages 203–206. Springer, 1997.
- [55] V. Botti, C. Carrascosa, V. Julián, and J. Soler. Modelling agents in hard real-time environments. In *Multi-Agent System Engineering*, pages 63–76. Springer, 1999.
- [56] M. E. Bratman, D. J. Israel, and M. E. Pollack. Plans and resource-bounded practical reasoning. *Computational intelligence*, 4(3):349–355, 1988.
- [57] R. A. Brooks. A robust layered control system for a mobile robot. *Robotics and Automation, IEEE Journal of*, 2(1):14–23, 1986.
- [58] E. J. Bruno and G. Bollella. *Real-Time Java Programming: With Java RTS*. Pearson Education, 2009.
- [59] A. Burns. Scheduling hard real-time systems: a review. *Software Engineering Journal*, 6(3):116–128, 1991.
- [60] A. Burns and A. Wellings. *Real-Time Systems and Programming Languages: Ada, Real-Time Java and C/Real-Time POSIX*. Addison-Wesley Educational Publishers Inc, USA, 4th edition, 2009.
- [61] A. Burns and A. J. Wellings. *Real-time systems and programming languages*, volume 2097. Addison-Wesley, 1998.
- [62] P. Busetta, R. Rönnquist, A. Hodgson, and A. Lucas. Jack intelligent agents-components for intelligent agents in java. *AgentLink News Letter*, 2(1):2–5, 1999.
- [63] G. C. Buttazzo. Rate monotonic vs. edf: Judgment day. In Rajeev Alur and Insup Lee, editors, *Embedded Software*, volume 2855 of *Lecture Notes in Computer Science*, pages 67–83. Springer Berlin Heidelberg, 2003.
- [64] G. C. Buttazzo. *Hard Real-Time Computing Systems*, volume 24. Springer, 2011.
- [65] D. Capera, J. Georgé, M. Gleizes, and P. Glize. Emergence of organisations, emergence of functions. In *AISB'03 symposium on Adaptive Agents and Multi-Agent Systems*, pages 103–108, 2003.
- [66] H. G. Ceballos, P. Noriega, and F. J. Cantu. Dispatching agents in electronic institutions. In *The First International Workshop on Infrastructure and Tools for Multiagent Systems ITMAS 2010*, page 26, 2010.

- [67] P. Chamoso, W. Raveane, V. Parra, A. González, and S. Rodríguez. Uavs applied to species accounting and monitoring of animals. In igeru Omatu, José Neves, Juan M. Corchado, Juan F. De Paz Santana, and Sara Rodríguez González, editors, *Distributed Computing and Artificial Intelligence, 11th International Conference.*, 2014.
- [68] P. Chand and D. A. Carnegie. Mapping and exploration in a hierarchical heterogeneous multi-robot system using limited capability robots. *Robotics and Autonomous Systems*, 61(6):565 – 579, 2013.
- [69] Y. Cheon and G. T. Leavens. A runtime assertion checker for the java modeling language (jml). In *Proceedings of the International Conference on Software Engineering Research and Practice (SERP'02), Las Vegas, Nevada, USA*, pages 322–328, 2002.
- [70] H. Chetto, M. Silly, and T. Bouchentouf. Dynamic scheduling of real-time tasks under precedence constraints. *Real-Time Systems*, 2(3):181–194, 1990.
- [71] W. Cohen. Adaptive mapping and navigation by teams of simple robots. *Journal of Robotics & Autonomous Systems*, 18:411–434, 1996.
- [72] Apache Commons. Apache commons bcel. Last Access: 2014-04-03.
- [73] Aicas Company. Jamaica virtual machine. <https://www.aicas.com/cms/en/JamaicaVM>, 2014.
- [74] J Consortium. Real-time core extensions for the java platform. *International J Consortium Specification*, 187, 2000.
- [75] J. M. Corchado, M. Glez-Bedia, Y. de Paz, J. Bajo, and J. F. de Paz. Concept, formulation and mechanism for agent replanification: Mrp architecture. *Comput. Intell*, 24(2):77–107, 2008.
- [76] J. M. Corchado, M. Glez-Bedia, Y. De Paz, J. Bajo, and J. F. De Paz. Replanning mechanism for deliberative agents in dynamic changing environments. *Computational Intelligence*, 24(2):77–107, 2008.
- [77] A. Corsaro and D. C. Schmidt. The design and performance of the jrate real-time java implementation. In *On the Move to Meaningful Internet Systems, 2002 - DOA/-CoopIS/ODBASE 2002 Confederated International Conferences DOA, CoopIS and ODBASE 2002*, pages 900–921, London, UK, UK, 2002. Springer-Verlag.
- [78] M. Cossentino, N. Gaud, V. Hilaire, S. Galland, and A. Koukam. Aspec: an agent-oriented software process for engineering complex systems. *Autonomous Agents and Multi-Agent Systems*, 20(2):260–304, 2010.
- [79] G. F. Coulouris, J. Dollimore, and T. Kindberg. *Distributed systems: concepts and design*. pearson education, 2005.
- [80] K. Crary and S. Weirich. Resource bound certification. In *Proceedings of the 27th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 184–198. ACM, 2000.
- [81] Y. H. Daren. Automatic protocol generation based on commitment machines. The University of Western Australia, 2004.
- [82] M. Dastani, N. AM Tinnemeier, and J. C. Meyer. A programming language for normative multi-agent systems. *Multi-Agent Systems: Semantics and Dynamics of Organizational Models*, pages 397–417, 2009.

- [83] K. Dautenhahn. Reverse engineering of societies—a biological perspective. In *Proceedings of the AISB Symposium, Starting from Society—the application of social analogies to computational systems*, 2000.
- [84] P. Davidsson. Categories of artificial societies. In *Engineering Societies in the Agents World II*, pages 1–9. Springer, 2001.
- [85] F. de la Prieta, B. Pérez-Lancho, J. F. De Paz, J. Bajo, and J. M. Corchado. Ovamah: Multiagent-based adaptive virtual organizations. In *Information Fusion, 2009. FUSION'09. 12th International Conference on*, pages 990–997. IEEE, 2009.
- [86] E. Del Acebo and J. L. De la Rosa. Introducing bar systems: a class of swarm intelligence optimization algorithms. In *AISB Convention Communication, Interaction and Social Intelligence*, pages 18–23, 2008.
- [87] C. Dellarocas and M. Klein. Contractual agent societies. In *Social Order in Multiagent Systems*, pages 113–133. Springer, 2001.
- [88] S. K. Dhall. *Scheduling periodic-time-critical jobs on single processor and multiprocessor computing systems*. PhD thesis, University of Illinois at Urbana-Champaign, 1977.
- [89] S. K. Dhall and C. L. Liu. On a real-time scheduling problem. *Operations Research*, 26(1):127–140, 1978.
- [90] G. Di Marzo, M. Gleizes, and A. Karageorgos. Self-organisation and emergence in mas: An overview. *Informatika (03505596)*, 30(1), 2006.
- [91] F. Dignum. Agents, markets, institutions, and protocols. In *Agent Mediated Electronic Commerce*, pages 98–114. Springer, 2001.
- [92] M. V. Dignum. *A model for organizational interaction: based on agents, founded in logic*. University Utrecht, 2003.
- [93] V. Dignum and F. Dignum. A logic of agent organizations. *Logic Journal of IGPL*, 20(1):283–316, 2012.
- [94] V. Dignum, J. Vázquez-Salceda, and F. Dignum. Omni: Introducing social structure, norms and ontologies into agent organizations. In *Programming Multi-Agent Systems*, pages 181–198. Springer, 2005.
- [95] E. W. Dijkstra, E. W. Dijkstra, and E. W. Dijkstra. *Notes on structured programming*. Technological University Eindhoven Netherlands, 1970.
- [96] L. C. DiPippo, V. Fay-Wolfe, L. Nair, E. Hodys, and O. Uvarov. A real-time multi-agent system architecture for e-commerce applications. In *Autonomous Decentralized Systems, 2001. Proceedings. 5th International Symposium on*, pages 357–364. IEEE, 2001.
- [97] L. C. DiPippo, E. Hodys, and B. Thuraisingham. Towards a real-time agent architecture—a whitepaper. In *Object-Oriented Real-Time Dependable Systems, 1999. WORDS 1999 Fall. Proceedings. Fifth International Workshop on*, pages 59–64. IEEE, 1999.
- [98] B. Efron. The convex hull of a random set of points. *Biometrika*, 52(3-4):331–343, 1965.
- [99] A. Elkady and T. Sobh. Robotics middleware: A comprehensive literature survey and attribute-based bibliography. *Journal of Robotics*, 2012, 2012.

- [100] M. A. Ellis and B. Stroustrup. *The annotated C++ reference manual*. Addison-Wesley Longman Publishing Co., Inc., 1990.
- [101] S. Enderle, H. Utz, S. Sablatnög, S. Simon, G. Kraetzschmar, and G. Palm. Miro: Middleware for autonomous mobile robots. *Telematics Applications in Automation and Robotics*, 2001.
- [102] J. Engblom, A. Ermedahl, and F. Stappert. Comparing different worst-case execution time analysis methods. In *Work-in-Progress session of the 21st IEEE Real-Time Systems Symposium (RTSS 2000)*, 2000.
- [103] M. Esteva. *Electronic Institutions: from specification to development*. PhD thesis, Technical University of Catalonia, 2003.
- [104] M. Esteva, D. De La Cruz, and C. Sierra. Islander: an electronic institutions editor. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 3*, pages 1045–1052. ACM, 2002.
- [105] M. Esteva, J. A. Rodríguez-Aguilar, C. Sierra, P. Garcia, and J. L. Arcos. On the formal specification of electronic institutions. In Frank Dignum and Carles Sierra, editors, *Agent Mediated Electronic Commerce*, volume 1991 of *Lecture Notes in Computer Science*, pages 126–147. Springer Berlin Heidelberg, 2001.
- [106] M. Esteva, B. Rosell, J. A. Rodriguez-Aguilar, and J. L. Arcos. Ameli: An agent-based middleware for electronic institutions. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 236–243. IEEE Computer Society, 2004.
- [107] R. Evertsz, M. Fletcher, R. Jones, J. Jarvis, J. Brusey, and S. Dance. Implementing industrial multi-agent systems using jacktm. In *Programming multi-agent systems*, pages 18–48. Springer, 2004.
- [108] J.T. Feddema, C. Lewis, and D.A. Schoenwald. Decentralized control of cooperative robotic vehicles: theory and application. *Robotics and Automation, IEEE Transactions on*, 18(5):852–864, Oct 2002.
- [109] J. Ferber. *Multi-agent systems: an introduction to distributed artificial intelligence*, volume 1. Addison-Wesley Reading, 1999.
- [110] J. Ferber and O. Gutknecht. A meta-model for the analysis and design of organizations in multi-agent systems. In *Multi Agent Systems, 1998. Proceedings. International Conference on*, pages 128–135. IEEE, 1998.
- [111] Jacques Ferber, Olivier Gutknecht, and Fabien Michel. From agents to organizations: An organizational view of multi-agent systems. In Paolo Giorgini, JörgP. Müller, and James Odell, editors, *Agent-Oriented Software Engineering IV*, volume 2935 of *Lecture Notes in Computer Science*, pages 214–230. Springer Berlin Heidelberg, 2004.
- [112] M. Fisher. A survey of concurrent metatemporal language and its applications. In *Temporal Logic*, pages 480–505. Springer, 1994.
- [113] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *International journal of high performance computing applications*, 15(3):200–222, 2001.
- [114] M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2):95–110, 1956.
- [115] G. F. Franklin, J. D. Powell, and A. Emami-Naeini. Feedback control of dynamics systems. *Addison-Wesley, Reading, MA*, 1994.

- [116] E. Frazzoli, M. A Dahleh, and E. Feron. Real-time motion planning for agile autonomous vehicles. *Journal of Guidance, Control, and Dynamics*, 25(1):116–129, 2002.
- [117] M. Friedmann, J. Kiener, S. Petters, D. Thomas, O. Von Stryk, and H. Sakamoto. Versatile, high-quality motions and behavior control of a humanoid soccer robot. *International Journal of Humanoid Robotics*, 5(03):417–436, 2008.
- [118] M. Friedmann, J. Kiener, S. Petters, D Thomas, O. Von Stryk, and H. Sakamoto. Versatile, high-quality motions and behavior control of a humanoid soccer robot. *International Journal of Humanoid Robotics*, 5(03):417–436, 2008.
- [119] C. Frost, C.S. Jensen, Luckow K.S., and B. Thomsen. Wcet analysis of java bytecode featuring common execution environments. Technical report, Aalborg University, 2011.
- [120] J. A. Galicia. *Protocolo de colaboración entre robots autónomos*. PhD thesis, Instituto Politécnico Nacional de México, 2012.
- [121] J.A. Galicia. *Protocolo de Colaboración entre Robots Autónomos*. PhD thesis, Instituto Plitécnico Nacional de México, 2012.
- [122] S. Galland, N. Gaud, S. Rodriguez, and V. Hilaire. Janus: Another yet general-purpose multiagent platform. In *Seventh AOSE Technical Forum, Paris*, 2010.
- [123] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.
- [124] A. Garvey and V. Lesser. A survey of research in deliberative real-time artificial intelligence. *Real-Time Systems*, 6(3):317–347, 1994.
- [125] A. J. Garvey and V. R. Lesser. Design-to-time real-time scheduling. *Systems, Man and Cybernetics, IEEE Transactions on*, 23(6):1491–1502, 1993.
- [126] J. M. Gascuña, A. Fernández-Caballero, and F. J. Garijo. Using icaro-t framework for reactive agent-based mobile robots. In *Advances in Practical Applications of Agents and Multiagent Systems*, pages 91–101. Springer, 2010.
- [127] J. M. Gascuña, E. Navarro, A. Fernández-Caballero, and J. Pavón. Development of a code generator for the icaro agent framework. In *Advances in Artificial Intelligence-IBERAMIA 2012*, pages 402–411. Springer, 2012.
- [128] L. Gasser, C. Braganza, and N. Herman. Mace: A flexible testbed for distributed ai research. *Distributed AI*, 1987.
- [129] L. Gasser and T. Ishida. A dynamic organizational architecture for adaptive problem solving. In *AAAI*, volume 91, pages 185–190, 1991.
- [130] N. Gaud, S. Galland, V. Hilaire, and A. Koukam. An organisational platform for holonic and multiagent systems. *PROMAS-6@ AAMAS*, 8:111–126, 2008.
- [131] N. Gaud, S. Galland, V. Hilaire, and A. Koukam. An organisational platform for holonic and multiagent systems. In *Programming Multi-Agent Systems*, pages 104–119. Springer, 2009.
- [132] M. R. Genesereth and N. J. Nilsson. *Logical foundations of artificial intelligence*, volume 9. Morgan Kaufmann Los Altos, CA, 1987.
- [133] K. Gharachorloo, A. Gupta, and J. Hennessy. *Hiding memory latency using dynamic scheduling in shared-memory multiprocessors*, volume 20. ACM, 1992.
- [134] A. Giret, V. Botti, and S. Valero. Mas methodology for hms. In *Holonic and Multi-Agent Systems for Manufacturing*, pages 39–49. Springer, 2005.

- [135] A. Giret, V. Julián, M. Rebollo, E. Argente, C. Carrascosa, and V. Botti. An open architecture for service-oriented virtual organizations. In *Programming Multi-Agent Systems*, pages 118–132. Springer, 2010.
- [136] M. Gleizes, V. Camps, and P. Glize. A theory of emergent computation based on cooperative self-organization for adaptive artificial systems. In *Fourth European Congress of Systems Science*, 1999.
- [137] J. Gómez-Romero, M. A. Patricio, J. García, and J. M. Molina. Communication in distributed tracking systems: an ontology-based approach to improve cooperation. *Expert Systems*, 28(4):288–305, 2011.
- [138] J. Goossens, S. Funk, and S. Baruah. Priority-driven scheduling of periodic task systems on multiprocessors. *Real-time systems*, 25(2-3):187–205, 2003.
- [139] J. Gosling. *The Java language specification*. Addison-Wesley Professional, 2000.
- [140] J. Gosling and G. Bollella. *The real-time specification for Java*. Addison-Wesley Longman Publishing Co., Inc., 2000.
- [141] R. Grabowski, L. E. Navarro-Serment, C. J. Paredis, and P. Khosla. Heterogeneous teams of modular robots for mapping and exploration. *Autonomous Robots*, 8(3):293–308, 2000.
- [142] J. R. Graham and K. S. Decker. Towards a distributed, environment-centered agent framework. In *Intelligent Agents VI. Agent Theories, Architectures, and Languages*, pages 290–304. Springer, 2000.
- [143] D. Griol, J. García-Herrero, and J. M. Molina. Combining heterogeneous inputs for the development of adaptive and multimodal interaction systems. *ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal*, 1(6):37–53, 2013.
- [144] BISITE Research Group. Pangea web page. <http://pangea.usal.es/>, 2013.
- [145] W. Gruver. Technologies and applications of distributed intelligent systems. *IEEE MTT-Chapter Presentation*, 2004.
- [146] J. Guerrero and G. Oliver. Multi-robot coalition formation in real-time scenarios. *Robotics and Autonomous Systems*, 60(10):1295 – 1307, 2012.
- [147] J. Gustafsson. *Analyzing execution-time of object-oriented programs using abstract interpretation*. PhD thesis, Uppsala University, 2000.
- [148] C. Gutiérrez. Analysis of bullying in cooperative multi-agent systems’ communications. *International Journal of Interactive Multimedia and Artificial Intelligence*, 2(Regular Issue), 2013.
- [149] O. Gutknecht and J. Ferber. Madkit: Organizing heterogeneity with groups in a platform for multiple multi-agent systems. *rapport n<sup>o</sup> RR97188*, 1997.
- [150] O. Gutknecht and J. Ferber. The madkit agent platform architecture. In *Infrastructure for Agents, Multi-Agent Systems, and Scalable Multi-Agent Systems*, pages 48–55. Springer, 2001.
- [151] Olivier Gutknecht and Jacques Ferber. Madkit: A generic multi-agent platform. In *Proceedings of the Fourth International Conference on Autonomous Agents*, AGENTS ’00, pages 78–79, New York, NY, USA, 2000. ACM.

- [152] G. Haddad, F. Hussain, and G. T. Leavens. The design of safejml, a specification language for scj with support for wcet specification. In *Proceedings of the 8th International Workshop on Java Technologies for Real-Time and Embedded Systems*, pages 155–163. ACM, 2010.
- [153] H. Haehnel. Remote controlled flying robot platform. In *Digital Information Management, 2008. ICDIM 2008. Third International Conference on*, pages 920–921, Nov 2008.
- [154] T. Harmon and R. Klefstad. Toward a unified standard for worst-case execution time annotations in real-time java. In *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*, pages 1–8. IEEE, 2007.
- [155] A. Helsing, M. Thome, and T. Wright. Cougaar: a scalable, distributed multi-agent architecture. In *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, volume 2, pages 1910–1917. IEEE, 2004.
- [156] L. Hernández. *Heurísticas para el control deliberativo en una arquitectura de agentes inteligentes de tiempo real*. PhD thesis, Universitat Politècnica de Valencia, 2004.
- [157] L. Hernández, V. Botti, and A. García-Fornes. A deliberative scheduling technique for a real-time agent architecture. *Engineering Applications of Artificial Intelligence*, 19(5):521–534, 2006.
- [158] B. Hirsch, T. Konnerth, and A. Heßler. Merging agents and services—the jiac agent platform. In *Multi-Agent Programming*, pages 159–185. Springer, 2009.
- [159] B. Horling and V. Lesser. Using odml to model multi-agent organizations. In *Intelligent Agent Technology, IEEE/WIC/ACM International Conference on*, pages 72–80. IEEE, 2005.
- [160] N. Howden, R. Rönnquist, A. Hodgson, and A. Lucas. Jack intelligent agents—summary of an agent infrastructure. In *5th International conference on autonomous agents*, 2001.
- [161] E. Hu, G. Bernat, and A. Wellings. Addressing dynamic dispatching issues in wcet analysis for object-oriented hard real-time systems. In *Object-Oriented Real-Time Distributed Computing, 2002.(ISORC 2002). Proceedings. Fifth IEEE International Symposium on*, pages 109–116. IEEE, 2002.
- [162] E. Hu, G. Bernat, and A. Wellings. A static timing analysis environment using java architecture for safety critical real-time systems. In *Object-Oriented Real-Time Dependable Systems, 2002.(WORDS 2002). Proceedings of the Seventh International Workshop on*, pages 77–84. IEEE, 2002.
- [163] E. Y. Hu, A. Wellings, and G. Bernat. Xrtj: An extensible distributed high-integrity real-time java environment. In *Real-Time and Embedded Computing Systems and Applications*, pages 208–228. Springer, 2004.
- [164] B. Huber and P. P. Puschner. A code policy guaranteeing fully automated path analysis. In *WCET*, pages 77–88, 2010.
- [165] J. F. Hübne, J. S. Sichman, and O. Boissier. S-moise+: A middleware for developing organised multi-agent systems. In *COIN I, volume 3913 of LNAI*, pages 64–78. Springer, 2006.
- [166] J. F. Hübner. J -moise+ programming organisational agents with moise+ & jason. Technical Fora Group at EUMAS 2007, 2007.



- [167] J. F. Hübner, R. H. Bordini, and G. Picard. Using jason and  $\text{M}^{\text{oise}}$  to develop a team of cowboys. In *Programming Multi-Agent Systems*, pages 238–242. Springer, 2009.
- [168] J. F. Hübner, J. S. Sichman, and O. Boissier. A model for the structural, functional, and deontic specification of organizations in multiagent systems. In *Advances in artificial intelligence*, pages 118–128. Springer, 2002.
- [169] J. F. Hübner, J. S. Sichman, and O. Boissier. Using the  $\text{M}^{\text{oise}}$  for a cooperative framework of mas reorganisation. In *Advances in artificial intelligence—SBIA 2004*, pages 506–515. Springer, 2004.
- [170] IBM. Mq telemetry transport. <http://mqtt.org/>, 2014.
- [171] IBM. Websphere product. <http://www-03.ibm.com/software/products/es/real-time/>, 2014.
- [172] International Business Machines Corporation (IBM). Mqtt v3.1 protocol specification. Technical report, IBAM, 2010.
- [173] Fiji Systems Inc. Fiji virtual machine. <http://fiji-systems.com/>, 2014.
- [174] V. J. Inglada, V. Botti, M. Navarro, and V. Soler. jart: A real-time multi-agent platform with rt-java. In *3rd International workshop on practical applications of agents and multiagent systems: IWPAAMS 2004*, pages 73–82. Universidad de Burgos, 2004.
- [175] P. Iñigo-Blasco, F. Diaz-del Rio, M. A. Romero-Ternero, D. Cagigas-Muñiz, and S. Vicente-Diaz. Robotics software frameworks for multi-agent robotic systems development. *Robotics and Autonomous Systems*, 60(6):803 – 821, 2012.
- [176] P. Iñigo-Blasco, F. Diaz-del Rio, M. C. Romero-Ternero, D. Cagigas-Muñiz, and S. Vicente-Diaz. Robotics software frameworks for multi-agent robotic systems development. *Robotics and Autonomous Systems*, 60(6):803–821, 2012.
- [177] B. Innocenti. *A Multi-agent Architecture with Distributed Coordination for an Autonomous Robot*. PhD thesis, Universitat de Girona, 2008.
- [178] Eurotech International Business Machines Corporation (IBM). Mqtt v3.1 protocol specification. <http://public.dhe.ibm.com/software/dw/webservices/ws-mqtt/mqtt-v3r1.html>, 1999.
- [179] Yang. J. and Z. Luo. Coalition formation mechanism in multi-agent systems based on genetic algorithms. *Applied Soft Computing*, 7(2):561 – 568, 2007.
- [180] C. Jang, S. Lee, S. Jung, B. Song, R. Kim, S. Kim, and C. Lee. Opros: A new component-based robot software platform. *ETRI journal*, 32(5), 2010.
- [181] J. Jarvis, R. Rönnquist, D. McFarlane, and L. Jain. A team-based holonic approach to robotic assembly cell control. *Journal of Network and Computer Applications*, 29(2):160–176, 2006.
- [182] The java community process. Jsrs: Java specification requests. Last Access: 2014-03-30.
- [183] N. Jennings and M. J Wooldridge. *Agent technology: foundations, applications, and markets*. Springer, 1998.
- [184] N. R. Jennings. Specification and implementation of a belief-desire-joint-intention architecture for collaborative problem solving. *International Journal of Intelligent and Cooperative Information Systems*, 2(03):289–318, 1993.

- [185] W. Jiao and H. Mei. Automated adaptations to dynamic software architectures by using autonomous agents. *Engineering Applications of Artificial Intelligence*, 17(7):749–770, 2004.
- [186] J. Johnson and M. Sugisaka. Complexity science for the design of swarm robot control systems. In *Industrial Electronics Society, 2000. IECON 2000. 26th Annual Conference of the IEEE*, volume 1, pages 695–700. IEEE, 2000.
- [187] C. Jonker, M. Klusch, and J. Treur. Design of collaborative information agents. In *Cooperative Information Agents IV-The Future of Information Agents in Cyberspace*, pages 262–283. Springer, 2000.
- [188] N. Josuttis. *SOA in Practice*. O’Reilly, 2007.
- [189] V. Julian and V. Botti. Developing real-time multi-agent systems. *Integrated Computer-Aided Engineering*, 11(2):135–149, 2004.
- [190] V. Julian, J. Soler, M. C. Moncho, and V. Botti. Real-time multi-agent system development and implementation. *Frontiers in artificial intelligence and applications, Ourense, Espana*, 113:333–340, 2004.
- [191] N. Kalra, D. Ferguson, and A. Stentz. Hoplites: A market-based framework for planned tight coordination in multirobot teams. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 1170–1177, April 2005.
- [192] C. Kalt. Internet relay chat: Architecture rfc 2811. Technical report, The Internet Society, 2000.
- [193] C. Kalt. Internet relay chat: Channel management rfc 2811. Technical report, The Internet Society, 2000.
- [194] C. Kalt. Internet relay chat: Client protocol rfc 2812. Technical report, The Internet Society, 2000.
- [195] C. Kalt. Internet relay chat: Server protocol rfc 2813. Technical report, The Internet Society, 2000.
- [196] K. Kianfar. Branch-and-bound algorithms. *Wiley Encyclopedia of Operations Research and Management Science*, 2010.
- [197] J. Kiener and O. Von Stryk. Cooperation of heterogeneous, autonomous robots: A case study of humanoid and wheeled robots. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 959–964, Oct 2007.
- [198] J. Kiener and O. Von Stryk. Towards cooperation of heterogeneous, autonomous robots: A case study of humanoid and wheeled robots. *Robotics and Autonomous Systems*, 58(7):921–929, 2010.
- [199] K. S. Kim. *Transmission scheduling with deadline and throughput constraints*. PhD thesis, Massachusetts Institute of Technology, 2013.
- [200] H. Kitano. Robocup rescue: a grand challenge for multi-agent systems. In *MultiAgent Systems, 2000. Proceedings. Fourth International Conference on*, pages 5–12, 2000.
- [201] T. Kiyofumi. Adaptive edf: Using predictive execution time. *SIGBED Rev.*, 10(4):41–44, December 2013.
- [202] T. Kiyofumi. Adaptive real-time scheduling for soft tasks with varying execution times (preprint). *IPSSJ Journal*, 55(2), feb 2014.

- [203] M. Klotzbücher, P. Soetens, and H. Bruyninckx. Orocos rtt-lua: an execution environment for building real-time robotic domain specific languages. In *International Workshop on Dynamic languages for RObotic and Sensors*, page 284289, 2010.
- [204] M. Kontitsis, R.D. Garcia, and K.P. Valavanis. Design, implementation and testing of a vision system for small unmanned vertical take off and landing vehicles with strict payload limitations. *Journal of Intelligent and Robotic Systems*, 44(2):139–159, 2005.
- [205] H. Kopetz, A. Damm, C. Koza, M. Mulazzani, W. Schwabl, C. Senft, and R. Zainlinger. Distributed fault-tolerant real-time systems: The mars approach. *Micro, IEEE*, 9(1):25–40, 1989.
- [206] J. Kramer and M. Scheutz. Development environments for autonomous mobile robots: A survey. *Autonomous Robots*, 22(2):101–132, 2007.
- [207] M. Kranz, Radu B. Rusu, A. Maldonado, M. Beetz, and A. Schmidt. A player/stage system for context-aware intelligent environments. *Proceedings of UbiSys*, 6:17–21, 2006.
- [208] J. Krone, W. F. Ogden, and M. Sitaraman. Modular verification of performance constraints. In *ACM OOPSLA Workshop on Specification and Verification of Component-Based Systems (SAVCBS)*, pages 60–67, 2003.
- [209] J. Kyparisis. On uniqueness of kuhn-tucker multipliers in nonlinear programming. *Mathematical Programming*, 32(2):242–246, 1985.
- [210] S. Labidi and W. Lejouad. De l’intelligence artificielle distribuee aux systemes multi-agents. Technical Report 1(2004), Inria Sophia University, 1993.
- [211] J. M. Lambert and J. F. Power. Platform independent timing of java virtual machine bytecode instructions. *Electronic Notes in Theoretical Computer Science*, 220(3):97–113, 2008.
- [212] E. L. Lawler and D. E. Wood. Branch-and-bound methods: A survey. *Operations research*, 14(4):699–719, 1966.
- [213] J. Lehoczky, L. Sha, and Y. Ding. The rate monotonic scheduling algorithm: Exact characterization and average case behavior. In *Real Time Systems Symposium, 1989., Proceedings.*, pages 166–171. IEEE, 1989.
- [214] Y. Lespérance, H. J. Levesque, F. Lin, D. Marcu, R. Reiter, and R. B. Scherl. Foundations of a logical approach to agent programming. In *Intelligent Agents II Agent Theories, Architectures, and Languages*, pages 331–346. Springer, 1996.
- [215] J. Y. Leung and M. L. Merrill. A note on preemptive scheduling of periodic, real-time tasks. *Information processing letters*, 11(3):115–118, 1980.
- [216] J. Y. Leung and J. Whitehead. On the complexity of fixed-priority scheduling of periodic, real-time tasks. *Performance evaluation*, 2(4):237–250, 1982.
- [217] Y. Li, S. Malik, and A. Wolfe. Efficient microarchitecture modeling and path analysis for real-time software. In *Real-Time Systems Symposium, 1995. Proceedings., 16th IEEE*, pages 298–307. IEEE, 1995.
- [218] Y. S. Li and S. Malik. Performance analysis of embedded software using implicit path enumeration. *ACM SIGPLAN Notices*, 30(11):88–98, 1995.
- [219] T. Lindholm, F. Yellin, G. Bracha, and A. Buckley. Java virtual machine specification. Last Access: 2014-03-30.

- [220] T. Lindholm, F. Yellin, G. Bracha, and A. Buckley. *The Java Virtual Machine Specification*. Addison-Wesley, 2013.
- [221] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM (JACM)*, 20(1):46–61, 1973.
- [222] J. W. Liu. *Real Time Systems*. Prentice Hall, 2000.
- [223] P. Lokuciejewski and P. Marwedel. *Worst-case execution time aware compilation techniques for real-time systems*. Springer, 2010.
- [224] F. López, M. Luck, and M. d’Inverno. A normative framework for agent-based systems. *Computational & Mathematical Organization Theory*, 12(2-3):227–250, 2006.
- [225] Agent Oriented Software Pty Ltd. Jack intelligent agents teams manual. [http://www.aosgrp.com/documentation/jack/JACK\\_Teams\\_Manual\\_WEB/](http://www.aosgrp.com/documentation/jack/JACK_Teams_Manual_WEB/), 2005.
- [226] M. Luck and P. McBurney. Computing as interaction: agent and agreement technologies. In *IEEE SMC Conference on Distributed Human-Machine Systems*, pages 1–6, 2008.
- [227] M. Luck, P. McBurney, and C. Preist. *Agent technology: Enabling next generation computing (a roadmap for agent based computing)*. AgentLink/University of Southampton, 2003.
- [228] K. S. Luckow, T. Bøgholm, B. Thomsen, and K. G. Larsen. Tetasarts: a tool for modular timing analysis of safety critical java systems. In *Proceedings of the 11th International Workshop on Java Technologies for Real-time and Embedded Systems*, pages 11–20. ACM, 2013.
- [229] M. Lützenberger, T. Küster, A. Heßler, and B. Hirsch. Unifying jiac agent development with awe. In *Multiagent System Technologies*, pages 220–225. Springer, 2009.
- [230] P. Maes. *Designing autonomous agents: Theory and practice from biology to engineering and back*. MIT press, 1990.
- [231] S. Magnenat, P. Réturnaz, M. Bonani, V. Longchamp, and F. Mondada. Aseba: a modular architecture for event-based control of complex robots. *Mechatronics, IEEE/ASME Transactions on*, 16(2):321–329, 2011.
- [232] R. Mahony, V. Kumar, and P. Corke. Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor. *Robotics Automation Magazine, IEEE*, 19(3):20–32, Sept 2012.
- [233] A. Makarenko, A. Brooks, and T. Kaupp. Orca: Components for robotics. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 163–168, 2006.
- [234] T. W. Malone and K. Crowston. The interdisciplinary study of coordination. *ACM Computing Surveys (CSUR)*, 26(1):87–119, 1994.
- [235] Q. Martín. *Investigación Operativa*. Pearson Educacion, 2003.
- [236] A. Mas. *Agentes software y sistemas multiagente: conceptos, arquitecturas y aplicaciones*. Prentice Hall, 2005.
- [237] M. A. Masmano. *Gestión de memoria dinámica en sistemas de tiempo real*. PhD thesis, Universitat Politecnica de Valencia, 2006.
- [238] I. Maza, F. Caballero, J. Capitán, J.R. Martínez-de Dios, and A. Ollero. Experimental results in multi-uav coordination for disaster management and civil security applications. *Journal of Intelligent & Robotic Systems*, 61(1-4):563–585, 2011.

- [239] F. G. McCabe and K. L. Clark. April—agent process interaction language. In *Intelligent Agents*, pages 324–340. Springer, 1995.
- [240] H. Mehrjerdi, M. Saad, and J. Ghommam. Hierarchical fuzzy cooperative control and path following for a team of mobile robots. *Mechatronics, IEEE/ASME Transactions on*, 16(5):907–917, Oct 2011.
- [241] R. Z. Mercado and J. F. Bard. Heurísticas para secuenciamiento de tareas en líneas de flujo. *Ciencia UANL*, 3(4):420–427, 2000.
- [242] L. Merino, F. Caballero, J.R. Martínez-de Dios, J. Ferruz, and A. Ollero. A cooperative perception system for multiple uavs: Application to automatic detection of forest fires. *Journal of Field Robotics*, 23(3-4):165–184, 2006.
- [243] G. Metta, P. Fitzpatrick, and L. Natale. Yarp: Yet another robot platform. *International Journal of Advanced Robotic Systems*, 3(1), 2006.
- [244] O. Michel. Webotstm: Professional mobile robot simulation. *arXiv preprint cs/0412052*, 2004.
- [245] M. Mizukawa, H. Matsuka, T. Koyama, and A. Matsumoto. Orin: Open robot interface for the network, a proposed standard. *Industrial Robot: An International Journal*, 27(5):344–350, 2000.
- [246] N. Mohamed, J. Al-Jaroodi, and I. Jawhar. Middleware for robotics: A survey. In *Robotics, Automation and Mechatronics, 2008 IEEE Conference on*, pages 736–742. IEEE, 2008.
- [247] N. Mohamed, J. Al-Jaroodi, and I. Jawhar. A review of middleware for networked robots. *International Journal of Computer Science and Network Security*, 9(5):139–148, 2009.
- [248] A. K. Mok. The design of real-time programming systems based on process models. In *Proc. of IEEE Real-Time Systems Symposium*, pages 5–17, 1984.
- [249] D. Montemerlo, N. Roy, and S. Thrun. Perspectives on standardization in mobile robot programming: The carnegie mellon navigation (carmen) toolkit. In *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 3, pages 2436–2441. IEEE, 2003.
- [250] H. J. Mueller. Negotiation principles. In *Foundations of Distributed Artificial Intelligence*. Wiley, 1996.
- [251] R. Mukul, P. Singh, D. Jayaram, D. Das, N. Sreenivasulu, K. Vinay, and A. Ramamoorthy. An adaptive bandwidth request mechanism for qos enhancement in wimax real time communication. In *Wireless and Optical Communications Networks, 2006 IFIP International Conference on*, pages 5 pp.–5, 2006.
- [252] R. Mundry and C. Sommer. Discriminant function analysis with nonindependent data: consequences and an alternative. *Animal Behaviour*, 74(4):965–976, 2007.
- [253] R. Murphy. *Introduction to AI robotics*. MIT press, 2000.
- [254] R. Murphy. *Introduction to AI robotics*. MIT press, 2000.
- [255] D. J. Musliner, E. H. Durfee, and K. G. Shin. Circa: A cooperative intelligent real-time control architecture. *Systems, Man and Cybernetics, IEEE Transactions on*, 23(6):1561–1574, 1993.

- [256] M. Namoshe, N. S. Tlale, C. M. Kumile, and G. Bright. Open middleware for robotics. In *Mechatronics and Machine Vision in Practice, 2008. M2VIP 2008. 15th International Conference on*, pages 189–194. IEEE, 2008.
- [257] O. Naseer, A. Shah, and A. A. Khan. Feedback control scheduling for crane control system. In *Engineering of Computer Based Systems (ECBS), 2013 20th IEEE International Conference and Workshops on the*, pages 187–195. IEEE, 2013.
- [258] M. Navarro. *Gestión de compromisos en sistemas multi-agente de tiempo real*. PhD thesis, Universitat Politecnica de Valencia, 2011.
- [259] P. Nebot and E. Cervera. A framework for the development of cooperative robotic applications. In *Advanced Robotics, 2005. ICAR'05. Proceedings., 12th International Conference on*, pages 901–906. IEEE, 2005.
- [260] P. Nebot and E. Cervera. Acromovi architecture: A framework for the development of multirobot applications. *Mobile Robots: Moving Intelligence*, 2006.
- [261] G. Necula. *Proof-carrying code*. Springer, 2011.
- [262] I. A. Nesnas, R. Simmons, D. Gaines, C. Kunz, A. Diaz-Calderon, T. Estlin, R. Madison, J. Guineau, M. McHenry, H. Shu, et al. Claraty: Challenges and steps toward reusable robotic software. *International Journal of Advanced Robotic Systems*, 3(1), 2006.
- [263] H. P. Nii. The blackboard model of problem solving and the evolution of blackboard architectures. *AI magazine*, 7(2):38, 1986.
- [264] K. Nilsen. Differentiating features of the perc virtual machine. Technical report, Atego, 2009.
- [265] P. Noriega and C. Sierra. Electronic institutions: Future trends and challenges. In *Cooperative information agents VI*, pages 14–17. Springer, 2002.
- [266] Aeronautical Association of Australia. First person view (fpv) policy. Technical report, Aeronautical Association of Australia, 2012.
- [267] J. Oikarinen and D. Reed. Internet relay chat protocol rfc1459. Technical report, The Internet Society, 1993.
- [268] R.J. Ojeda. *Programacion de horarios semanales de trabajadores polivalente en un centro de servicios*. PhD thesis, Universitat Politecnica de Catalunya, 2004.
- [269] A. Ollero, S. Lacroix, L. Merino, Jeremi Gancet, J. Wiklund, V. Remuss, I.V. Perez, L.G. Gutierrez, D.X. Viegas, M.A.G. Benitez, A. Mallet, R. Alami, R. Chatila, G. Hommel, F.J.C. Lechuga, B.C. Arrue, J. Ferruz, J.R. Martinez-De Dios, and F. Caballero. Multiple eyes in the skies: architecture and perception issues in the comets unmanned air vehicles project. *Robotics Automation Magazine, IEEE*, 12(2):46–57, June 2005.
- [270] A. Ollero and I. Maza. Introduction. In Aníbal Ollero and Iván Maza, editors, *Multiple Heterogeneous Unmanned Aerial Vehicles*, volume 37 of *Springer Tracts in Advanced Robotics*, pages 1–14. Springer Berlin Heidelberg, 2007.
- [271] A. Ollero and I. Maza, editors. *Multiple Heterogeneous Unmanned Aerial Vehicles*, volume 37. Springer Tracts in Advanced Robotics, 2007.
- [272] Oracle. Sun java real-time system 2.2 update 1 technical documentation. [http://docs.oracle.com/javase/realtime/rts\\_productdoc\\_2.2u1.html](http://docs.oracle.com/javase/realtime/rts_productdoc_2.2u1.html), 2009.

- [273] Oracle. Sun java real-time system 2.2 update 1. [http://docs.oracle.com/javase/realtime/doc\\_2.2u1/release/JavaRTSReleaseNotes.html](http://docs.oracle.com/javase/realtime/doc_2.2u1/release/JavaRTSReleaseNotes.html), 2010.
- [274] S. Ossowski. *Co-ordination in artificial agent societies: social structures and its implications for autonomous problem-solving agents*. Springer-Verlag, 1999.
- [275] I. M. OVACIKT and R. Uzsoy. Rolling horizon algorithms for a single-machine dynamic scheduling problem with sequence-dependent setup times. *International Journal of Production Research*, 32(6):1243–1263, 1994.
- [276] A. J. Page, T. M. Keane, and T. J. Naughton. Multi-heuristic dynamic task allocation using genetic algorithms in a heterogeneous distributed system. *Journal of parallel and distributed computing*, 70(7):758–766, 2010.
- [277] L. Parker. Intelligence, reasoning, and knowledge in multi-vehicle systems: Recent advances and current research challenges. In *1st IFAC-Symposium on Multivehicle Systems*, 2005.
- [278] L. E. Parker. Alliance: An architecture for fault tolerant multirobot cooperation. *Robotics and Automation, IEEE Transactions on*, 14(2):220–240, 1998.
- [279] H. Parunak and JamesJ. Odell. Representing social structures in uml. In MichaelJ. Wooldridge, Gerhard Weiß, and Paolo Ciancarini, editors, *Agent-Oriented Software Engineering II*, volume 2222 of *Lecture Notes in Computer Science*, pages 1–16. Springer Berlin Heidelberg, 2002.
- [280] H. E. Pattison, D. D. Corkill, and V. R. Lesser. Instantiating descriptions of organizational structures. *Distributed Artificial Intelligence*, 1:59–96, 1987.
- [281] S. Petters, D. Thomas, and O. Von Stryk. Roboframe-a modular software framework for lightweight autonomous robots. In *Proc. Workshop on Measures and Procedures for the Evaluation of Robot Architectures and Middleware of the 2007 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2007.
- [282] M. Pinedo. *Scheduling: theory, algorithms, and systems*. Springer, 2012.
- [283] L.M. Pinho. *Time-Bounded Adaptive Quality of Service Management for Cooperative Embedded Real-Time Systems*. PhD thesis, Universidade do Porto, Portugal, 2009.
- [284] A. Pokahr, L. Braubach, and W. Lamersdorf. Jadex: A bdi reasoning engine. In *Multi-agent programming*, pages 149–174. Springer, 2005.
- [285] T. Pop, P. Pop, P. Eles, and P. Zebo. Optimization of hierarchically scheduled heterogeneous embedded systems. In *Embedded and Real-Time Computing Systems and Applications, 2005. Proceedings. 11th IEEE International Conference on*, pages 67–71, Aug 2005.
- [286] J. Porubán, M. Forgáč, M. Sabo, and M. Běhálek. Annotation based parser generator. *Computer Science and Information Systems/ComSIS*, 7(2):291–307, 2010.
- [287] S. Poslad, P. Buckle, and R. Hadingham. The fipa-os agent platform: Open source for open standards. In *Proceedings of the 5th International Conference and Exhibition on the Practical Application of Intelligent Agents and Multi-Agents*, volume 355, page 368, 2000.
- [288] J. Poulsen and A. French. Discriminant function analysis (da). *San Francisco State University*, available at: <http://online.sfsu.edu/efc/classes/biol710/discrim/discrim.pdf#search1/4>, 22, 2004.

- [289] Java Community Process. Jsr 50: Distributed real-time specification. <https://jcp.org/en/jsr/detail?id=50>, 2006.
- [290] Java Community Process. Jsr 175: A metadata facility for the java programming language. <https://jcp.org/en/jsr/detail?id=175>, 2014.
- [291] Java Community Process. Jsr 308: Annotations on java types. <https://jcp.org/en/jsr/detail?id=308>, 2014.
- [292] K. Prouskas and J. Pitt. Towards a real-time architecture for time-aware agents. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*, pages 92–93. ACM, 2002.
- [293] P. P. Puschner and G. Bernat. Wcet analysis of reusable portable code. In *Real-Time Systems, 13th Euromicro Conference on, 2001.*, pages 45–52. IEEE, 2001.
- [294] P. P. Puschner and C. Koza. Calculating the maximum execution time of real-time programs. *Real-Time Systems*, 1(2):159–176, 1989.
- [295] P. P. Puschner and A. V. Schedl. Computing maximum task execution times—a graph-based approach. *Real-Time Systems*, 13(1):67–91, 1997.
- [296] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, 2009.
- [297] Talal Rahwan and Nicholas R. Jennings. An algorithm for distributing coalitional value calculations among cooperating agents. *Artificial Intelligence*, 171(8–9):535 – 567, 2007.
- [298] H. Ramli, W. Kuntjoro, and A. K. Makhtar. Advanced autonomous multirotor response system. *Applied Mechanics and Materials*, 393:299–304, 2013.
- [299] A. S Rao and M. P. Georgeff. Modeling rational agents within a bdi-architecture. *Readings in agents*, pages 317–328, 1997.
- [300] A. S. Rao, M. P. Georgeff, et al. Bdi agents: From theory to practice. In *ICMAS*, volume 95, pages 312–319, 1995.
- [301] R. Razavi, J. Perrot, and N. Guelfi. Adaptive modeling: an approach and a method for implementing adaptive agents. In *Massively Multi-Agent Systems I*, pages 136–148. Springer, 2005.
- [302] A. Renzaglia, L. Doitsidis, A. Martinelli, and E. Kosmatopoulos. Multi-robot three dimensional coverage of unknown areas. *International Journal of Robotics Research*, March 2012.
- [303] S. Rodríguez. *Adaptive Model for Virtual Organizations of Agents*. PhD thesis, University of Salamanca, 2010.
- [304] S. Rodríguez, Y. de Paz, J. Bajo, and J. M. Corchado. Social-based planning model for multiagent systems. *Expert Systems with Applications*, 38(10):13005–13023, 2011.
- [305] S. Rodríguez, Y. De Paz, J. Bajo, and J. M. Corchado. Social-based planning model for multiagent systems. *Expert Systems with Applications*, 38(10):13005 – 13023, 2011.
- [306] S. Rodríguez, V. Julián, J. Bajo, C. Carrascosa, V. Botti, and J. M. Corchado. Agent-based virtual organization architecture. *Engineering Applications of Artificial Intelligence*, 24(5):895–910, 2011.



- [307] S. Rodríguez, V. Julián, J. Bajo, C. Carrascosa, V. Botti, and J.M. Corchado. Agent-based virtual organization architecture. *Engineering Applications of Artificial Intelligence*, 24(5):895 – 910, 2011.
- [308] S. Rodríguez, V. Julián, A. L. Sánchez, C. Carrascosa, V. F. López, J. M. Corchado, and E. Corchado. Trends on the development of adaptive virtual organizations. In AndrePonce Leon F. de Carvalho, Sara Rodríguez-González, JuanF. Paz Santana, and JuanM. Corchado Rodríguez, editors, *Distributed Computing and Artificial Intelligence*, volume 79 of *Advances in Intelligent and Soft Computing*, pages 113–121. Springer Berlin Heidelberg, 2010.
- [309] S. J. Russell, P. Norvig, J. F. Canny, J. M. Malik, and D. D. Edwards. *Artificial intelligence: a modern approach*, volume 2. Prentice hall Englewood Cliffs, 1995.
- [310] A. Saabas. Jbe: Java bytecode editor. Last Access: 2014-04-03.
- [311] A. Sabelfeld and A.C. Myers. Language-based information-flow security. *Selected Areas in Communications, IEEE Journal on*, 21(1):5–19, Jan 2003.
- [312] P. Sánchez-Martín and S. López-De Haro. Programación de tareas, un reto diario en la empresa. In *Anales de mecánica y electricidad*, volume 82, pages 24–30. Asociacion de Ingenieros del ICAI, 2005.
- [313] T. W. Sandholm and V. R. Lesser. Coalitions among computationally bounded agents. *Artificial intelligence*, 94(1):99–137, 1997.
- [314] P. Sarma, L. J. Durlofsky, K. Aziz, and W. H. Chen. Efficient real-time reservoir management using adjoint-based optimal control and model updating. *Computational Geosciences*, 10(1):3–36, 2006.
- [315] E. H. Schein. *Organizational culture and leadership*, volume 356. John Wiley & Sons, 2006.
- [316] C. Schlegel and R. Worz. The software framework smartsoft for implementing sensorimotor systems. In *Intelligent Robots and Systems, 1999. IROS'99. Proceedings. 1999 IEEE/RSJ International Conference on*, volume 3, pages 1610–1616. IEEE, 1999.
- [317] D. C. Schmidt and F. Kuhns. An overview of the real-time corba specification. *Computer*, 33(6):56–63, 2000.
- [318] M. Schoeberl. *JOP: A Java Optimized Processor for Embedded Real-Time Systems*. PhD thesis, Vienna University of Technology, 2005.
- [319] M. Schoeberl. *JOP Reference Handbook: Building Embedded Systems with a Java Processor*. CreateSpace, 2009.
- [320] M. Schoeberl, W. Puffitsch, R. U. Pedersen, and B. Huber. Worst-case execution time analysis for a java processor. *Software: Practice and Experience*, 40(6):507–542, 2010.
- [321] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. 3550 a transport protocol for real-time applications (rtp). Technical report, Columbia University, Packet Design, 2013.
- [322] H. Schulzrinne, A. Rao, and R. Lanphier. Rfc 2326 real time streaming protocol (rtsp). Technical report, Columbia University, Netscape, 2013.
- [323] J. R. Searle. *Speech acts: An essay in the philosophy of language*, volume 626. Cambridge university press, 1969.

- [324] J. M. Serrano and S. Ossowski. On the impact of agent communication languages on the implementation of agent systems. In *Cooperative Information Agents VIII*, pages 92–106. Springer, 2004.
- [325] L. Sha, T. Abdelzaher, K. Årzén, A. Cervin, T. Baker, A. Burns, G. Buttazzo, M. Caccamo, J. Lehoczky, and A. K. Mok. Real time scheduling theory: A historical perspective. *Real-time systems*, 28(2-3):101–155, 2004.
- [326] L. Sha, R. Rajkumar, and J. P. Lehoczky. Priority inheritance protocols: An approach to real-time synchronization. *IEEE Transactions on computers*, 39(9):1175–1185, 1990.
- [327] A. C. Shaw. Reasoning about time in higher-level language software. *Software Engineering, IEEE Transactions on*, 15(7):875–889, 1989.
- [328] Onn Shehory and Sarit Kraus. Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 101(1–2):165 – 200, 1998.
- [329] M. Sierhuis. Modeling and simulating work practice. *Social Science and Informatics*, page 350, 2001.
- [330] M. Sierhuis, W. J. Clancey, and R. J. Van Hoof. Brahms: a multi-agent modelling environment for simulating work processes and practices. *International Journal of Simulation and Process Modelling*, 3(3):134–152, 2007.
- [331] C. Sierra, J. A. Rodríguez-Aguilar, P. Noriega, M. Esteva, and J. L. Arcos. Engineering multi-agent systems as electronic institutions. *European Journal for the Informatics Professional*, 4(4):33–39, 2004.
- [332] J. Soler, V. Julián, A. García-Fornes, and V. Botti. Real-time extensions in multi-agent communication. In *Current Topics in Artificial Intelligence*, pages 468–477. Springer, 2004.
- [333] H. Spencer. *The study of sociology*, volume 5. Henry S. King, 1873.
- [334] O. Spillum. A comparison between jack intelligent agents and jack teams applied. Technical report, Norwegian University of Science and Technology, 2008.
- [335] B. Sprunt. *Aperiodic task scheduling for real-time systems*. PhD thesis, Carnegie Mellon University, 1999.
- [336] FIPA IEEE Computer Society Standards. Fipa: Foundation for intelligent physical agents. <http://www.fipa.org/>, 2014.
- [337] J. A. Stankovic. Distributed real-time computing: The next generation. Technical Report COINS 92(01), University of Massachusetts, 1992.
- [338] J. A. Stankovic and K. Ramamritham. *Hard real-time systems*. IEEE Computer Society Press, 1988.
- [339] J. A. Stankovic and K. Ramamritham, editors. *Tutorial: Hard Real-time Systems*. IEEE Computer Society Press, Los Alamitos, CA, USA, 1989.
- [340] J. A. Stankovic, M. Spuri, M. Di Natale, and G. C. Buttazzo. Implications of classical scheduling results for real-time systems. *Computer*, 28(6):16–25, 1995.
- [341] John A. Stankovic. Real-time computing system: The next generation. Technical report, University of Massachusetts, Amherst, MA, USA, 1988.
- [342] D. C. Steere, A. Goel, J. Gruenberg, D. McNamee, C. Pu, and J. Walpole. A feedback-driven proportion allocator for real-rate scheduling. In *OSDI*, volume 99, pages 145–158, 1999.

- [343] D. C. Steere, A. Goel, J. Gruenberg, D. McNamee, C. Pu, and J. Walpole. A feedback-driven proportion allocator for real-rate scheduling. In *OSDI*, volume 99, pages 145–158, 1999.
- [344] T. Steinbach, H. D. Kenfack, F. Korf, and T. C. Schmidt. An extension of the omnet++ inet framework for simulating real-time ethernet with high accuracy. In *Proceedings of the 4th International ICST Conference on Simulation Tools and Techniques, SIMU-Tools '11*, pages 375–382. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2011.
- [345] T. Steinbach, F. Korf, and T.C. Schmidt. Comparing time-triggered ethernet with flexray: An evaluation of competing approaches to real-time for in-vehicle networks. In *Factory Communication Systems (WFCS), 2010 8th IEEE International Workshop on*, pages 199–202, May 2010.
- [346] P. Stone and M. Veloso. Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork. *Artificial Intelligence*, 110(2):241–273, 1999.
- [347] M. Stoodley, M. Fulton, M. Dawson, R. Sciampacone, and J. Kacur. Using java code to program real-time systems. <https://www.ibm.com/developerworks/library/j-rtj1/>, 2010.
- [348] J. K. Strosnider, J. P. Lehoczky, and L. Sha. The deferrable server algorithm for enhanced aperiodic responsiveness in hard real-time environments. *Computers, IEEE Transactions on*, 44(1):73–91, 1995.
- [349] D. M. Surka, M. C. Brito, and C. G. Harvey. The real-time objectagent software architecture for distributed satellite systems. In *Aerospace Conference, 2001, IEEE Proceedings.*, volume 6, pages 2731–2741. IEEE, 2001.
- [350] P. Tabuada. Event-triggered real-time scheduling of stabilizing control tasks. *Automatic Control, IEEE Transactions on*, 52(9):1680–1685, Sept 2007.
- [351] W. Taha and T. Sheard. Metaml and multi-stage programming with explicit annotations. *Theoretical Computer Science*, 248(1–2):211 – 242, 2000. PEPM'97.
- [352] A. Terrasa, A. García-Fornes, and V. J. Botti. Flexible real-time linux: A flexible hard real-time environment. *Real-Time Systems*, 22(1-2):151–173, 2002.
- [353] TimeSys. Rtsj reference implementation (ri). <http://www.timesys.com/java/>, 2008.
- [354] K. Tindell, A. Burns, and A. J. Wellings. Analysis of hard real-time communications. *Real-Time Systems*, 9(2):147–171, 1995.
- [355] R. J. Tobias. Hard real-time beam scheduler enables adaptive images in multi-probe systems. In *SPIE Medical Imaging*, pages 904010–904010. International Society for Optics and Photonics, 2014.
- [356] R. Tolksdorf and F. Zambonelli. Engineering societies in the agent world. In *First International Workshop ESAW 2000*. Springer, 2000.
- [357] V. Tretyakov and H. Surmann. Hardware architecture of a four-rotor uav for usar/wsar scenarios. In *Workshop Proceedings of SIMPAR 2008-International Conference on Simulation, Modeling and Programming for Autonomous Robots*, 2008.
- [358] L. Tribioli, M. Barbieri, R. Capata, E. Sciubba, E. Jannelli, and G. Bella. A real time energy management strategy for plug-in hybrid electric vehicles based on optimal control theory. *Energy Procedia*, 45(0):949 – 958, 2014. {ATI} 2013 - 68th Conference of the Italian Thermal Machines Engineering Association.

- [359] M. Tupia and D. Mauricio. Un algoritmo voraz para resolver el problema de la programación de tareas dependientes en máquinas diferentes. *RISI*, 1(1):9–18, 2004.
- [360] Carnegie Mellon University. The carnegie mellon robot navigation toolkit. <http://carmen.sourceforge.net>, 2008.
- [361] H. Utz, S. Sablatnog, S. Enderle, and G. Kraetzschmar. Miro-middleware for mobile robot applications. *Robotics and Automation, IEEE Transactions on*, 18(4):493–497, 2002.
- [362] H. Utz, F. Stulp, and A. Mühlendorf. Sharing belief in teams of heterogeneous robots. In Daniele Nardi, Martin Riedmiller, Claude Sammut, and José Santos-Victor, editors, *RoboCup 2004: Robot Soccer World Cup VIII*, volume 3276 of *Lecture Notes in Computer Science*, pages 508–515. Springer Berlin Heidelberg, 2005.
- [363] R.T. Vaughan, K. Stoy, G. Sukhatme, and M.J. Mataric. Lost: localization-space trails for robot teams. *Robotics and Automation, IEEE Transactions on*, 18(5):796–812, Oct 2002.
- [364] A. Viguria, I. Maza, and A. Ollero. Distributed service-based cooperation in aerial-/ground robot teams applied to fire detection and extinguishing missions. *Advanced Robotics*, 24(1-2):1–23, 2010.
- [365] J. Vitek, F. Pizlo, T. Kalibera, et al. Ovm project. <https://www.cs.purdue.edu/homes/jv/soft/ovm/index.html>, 2009.
- [366] T. Wagner and V. Lesser. Design-to-criteria scheduling: Real-time agent control. In *Infrastructure for Agents, Multi-Agent Systems, and Scalable Multi-Agent Systems*, pages 128–143. Springer, 2001.
- [367] N. Wang, Y. Yang, K. Meng, Y. Chen, and H. Ding. A task scheduling algorithm based on qos and complexity-aware optimization in cloud computing. In *Information and Communications Technology 2013, National Doctoral Academic Forum on*, pages 1–8. IET, 2013.
- [368] M. Weber. *Verteilte Systeme*. Spektrum, Akad. Verlag, 1998.
- [369] M. Weiser. Ubiquitous computing. *IEEE Computer Hot Topics*, 26(10):71–72, 1993.
- [370] G. Weiss. *Multiagent systems: a modern approach to distributed artificial intelligence*. MIT press, 1999.
- [371] A. Wellings. *Concurrent and real-time programming in Java*. John Wiley & Sons, 2005.
- [372] D. Weyns, K. Schelfhout, T. Holvoet, and O. Glorieux. A role based model for adaptive agents. In *AISB 2004 Convention*, page 75, 2004.
- [373] J. Wienie and S. Wrede. A middleware for collaborative research in experimental robotics. In *System Integration (SII), 2011 IEEE/SICE International Symposium on*, pages 1183–1190. IEEE, 2011.
- [374] R. Wilhelm, J. Engblom, A. Ermedahl, N. Holsti, S. Thesing, D. Whalley, G. Bernat, C. Ferdinand, R. Heckmann, T. Mitra, et al. The worst-case execution-time problem—overview of methods and survey of tools. *ACM Transactions on Embedded Computing Systems (TECS)*, 7(3):36, 2008.
- [375] R. Wilhelm and D. Grund. Computation takes time, but how much? *Communications of the ACM*, 57(2):94–103, 2014.

- [376] D. Windestål. Fpv starting guide. <http://rcexplorer.se/educational/2009/09/fpv-starting-guide/>, 2009.
- [377] M. Wooldridge and N. R. Jennings. Intelligent agents: Theory and practice. *The knowledge engineering review*, 10(02):115–152, 1995.
- [378] D. Wu, E. Sirin, J. Hendler, D. Nau, and B. Parsia. Automatic web services composition using shop2. Technical report, DTIC Document, 2006.
- [379] J. C. Wu, J. Hao, and C. Y. Guo. Research on real-time simulation of can bus with priority promotion algorithm. *Advanced Materials Research*, 433:5167–5171, 2012.
- [380] K.M. Wurm, C. Stachniss, and W. Burgard. Coordinated multi-robot exploration using a segmentation of the environment. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 1160–1165, Sept 2008.
- [381] P. Xiao and D. Liu. Multi-scheme co-scheduling framework for high-performance real-time applications in heterogeneous grids. *International Journal of Computational Science and Engineering*, 9(1):55–63, 2014.
- [382] J. Xu and D. L. Parnas. On satisfying timing constraints in hard-real-time systems. *IEEE Transactions on software engineering*, 19(1):70–84, 1993.
- [383] H. Yamaguchi. A cooperative hunting behavior by mobile-robot troops. *The International Journal of Robotics Research*, 18(9):931–940, 1999.
- [384] H. Yamaguchi. A cooperative hunting behavior by mobile-robot troops. *The International Journal of Robotics Research*, 18(9):931–940, 1999.
- [385] P. Yin, S. Yu, P. Wang, and Y. Wang. Task allocation for maximizing reliability of a distributed system using hybrid particle swarm optimization. *Journal of Systems and Software*, 80(5):724–735, 2007.
- [386] F. Zambonelli. Abstractions and infrastructures for the design and development of mobile agent organizations. In *Agent-Oriented Software Engineering II*, pages 245–262. Springer, 2002.
- [387] F. Zambonelli, M. P. Gleizes, M. Mamei, and R. Tolksdorf. Spray computers: frontiers of self-organisation for pervasive computing. workshop on enabling technologies: Infrastructure for collaborative enterprises. In *WETICE04*, pages 397–402. IEEE Computer Society, 2004.
- [388] F. Zambonelli, N. R. Jennings, and M. Wooldridge. Organisational abstractions for the analysis and design of multi-agent systems. In *Agent-Oriented Software Engineering*, pages 235–251. Springer, 2001.
- [389] F. Zambonelli, N. R. Jennings, and M. Wooldridge. Developing multiagent systems: The gaia methodology. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 12(3):317–370, 2003.
- [390] J. Zamorano, J. L. Redondo, C. Blanco, J. I. Tortosa, and J. A. de la Puente. Ejecutivo cíclico. *Manual de Usuario*, 1992.
- [391] C. Zato, J. F. De Paz, A. De Luis, J. Bajo, and J. M. Corchado. Model for assigning roles automatically in egovernment virtual organizations. *Expert Systems with Applications*, 39(12):10389–10401, 2012.
- [392] C. Zato, G. Villarrubia, A. Sánchez, J. Bajo, and J. M. Corchado. Pangea: A new platform for developing virtual organizations of agents. *International Journal of Artificial Intelligence<sup>TM</sup>*, 11(A13):93–102, 2013.

- 
- [393] C. Zato, G. Villarrubia, A. Sánchez, I. Barri, E. Rubión, A. Fernández, C. Rebate, J. A. Cabo, T. Álamos, J. Sanz, et al. Pangea—platform for automatic construction of organizations of intelligent agents. In *Distributed Computing and Artificial Intelligence*, pages 229–239. Springer, 2012.
- [394] M. N. Zeilinger, D. M. Raimondo, A. Domahidi, M. Morari, and C. N. Jones. On real-time robust model predictive control. *Automatica*, 2014.
- [395] J. Zhao. Analyzing control flow in java bytecode. In *Proceedings of the 16th Conference of Japan Society for Software Science and Technology*, pages 6–16, 1999.
- [396] W. Zhao, K. Ramamritham, and J. A. Stankovic. Preemptive scheduling under time and resource constraints. *Computers, IEEE Transactions on*, 100(8):949–960, 1987.
- [397] A. Zuhily. *Scheduling Analysis of Fixed Priority Hard Real-Time Systems with Multiframe Tasks*. PhD thesis, University of York, U.K., 2009.

Appendix

# A

## APPENDIX

---

This appendix is organized in two different sections. In the first one, we include a brief description of the main projects related to this dissertation. And in the second section, the main publications in the last two years of this research are presented.

### A

#### Related projects

---

Three projects are directly related with the development of this dissertation: OVAMAH, AZTECA and iHAS.

#### A.1

#### OVAMAH project

---

This project entitled in Spanish "Organizaciones Virtuales Adaptativas: Mecanismos, Arquitecturas y Herramientas." (reference TIN 2009-13839-C03-03) was funded by the Spanish Ministry of Science and Innovation.

The aim of this project is to provide a virtual organization with autonomy capabilities that allow it to have a dynamic response in view of potential changing situations by means of the adaptation or evolution of the organization. In this way, it will be able to detect situations of interest (e.g. operation errors) and to manage them maximising the adaptation flexibility and capacity. The organization adaptation implies, among other aspects, its norms, agreements, commitments and topological structure. Therefore, the adaptation of the system can be carried out at different granularity levels. At a general level, the fact that environmental changes in any type of system can give rise to the need of certain adaption or re-organization is obvious. However, to determine under which situations the agent or the organization decides to modify its

behaviour and to estimate when a partial or a complete change in the current organizational structure is necessary is extremely difficult. It must be said that the stability of a multi-agent system is a crucial aspect when developing this type of systems. However, a possible reorganization can affect notably to this stability, which makes even more crucial to detect which are the conditions that give rise to the necessity of a system re-organization even at the risk of affecting its stability. Thus, to provide the developer during the system development process with precise measures that ensure that the benefits and usefulness of adapting the system is greater than not doing so is necessary. In this way, the possibility of designing and implementing suitable mechanisms to perform these measurements can be taken into account during the system development process.

In general, it is necessary to define the standards and platforms required for the interoperability of the agents that meet these requirements. This article attempts to identify and analyze the research topics that touch on the development of open MAS systems based on virtual organizations. Additionally, it will present a study in which a high level abstract architecture was applied with the specific intent of addressing the design of open multi-agent systems and virtual organizations.

This project arises as a continuation of the THOMAS project, "Methods, techniques and tools for Open Multi-Agent Systems" (reference TIN 2006-14630-C03) project, whose main objective was to provide technology based on agents and multi-agent systems for the development of virtual organizations on open environments.

---

**A.2****AZTECA project**

---

The AZTECA project entitled in Spanish "Ambientes Inteligentes con Tecnología Accesible para el Trabajo" was supported by the Spanish CDTI. Company Cooperation Project with FEDER funds.

The effective integration of people with disabilities in the workplace is a huge challenge to society, and it presents an opportunity to make use of new technologies. The project, called AZTECA, aims to develop new tools that contribute to the employment of groups of people with visual, hearing, or motor disabilities in office environments. These different tools for the disabled people have been modelled with intelligent agents that use Web services. These agents are implemented and deployed within the PANGEA platform so PANGEA conforms the skeleton of the system and allows to develop an integral system.



The line more related with this dissertation is the "Design and implementation of a multi-agent service-oriented architecture". We try to obtain an oriented-service architecture based on MAS that provides communication and coordination mechanisms for the integration of multimodal interaction services and user services in environments where disabled people have to live. Moreover, the architecture must have advanced capabilities to customize the services depending on the needs of the users. The architecture must also include a reorganizational model based on VO, which enable to model the work environment as a virtual society, establishing special roles for disabled people and that can dynamically evolve (change roles, adapt norms, include new roles, etc.) according to the changes that occur in the work environment.

In conjunction with PANGEA, AZTECA contains an innovative solution with a high technological component that, unlike any architecture known to exist at this point, is capable of integrating adaptive interface systems, identification and localization systems, indoor guiding, and training and workplace virtualization systems using the TV and the internet for the integration of persons with physical disabilities into the workforce.

**A.3**

## iHAS project

The iHAS project entitled in Spanish "Computación Social Inteligente para Sociedades Humano-Agente" (reference TIN 2012-36586-C03-03) was funded by the Spanish Ministry of Science and Innovation.

Humans and agents have the ability to establish a series of relationships/collaborative interactions with each other, forming what might be called human-agent teams to meet their individual or collective goals within an organisation or social structure. This relationship between humans and agents can be implicit or explicit, pre-designed or emergent, static or dynamic. Considering systems of people and agents operating on a large scale offers an enormous potential and, if performed properly, it will help tackle complex social applications, which are critical to our future. This view is closely related to the concept of social computing, where systems are constructed from the interactions between entities that are part of them. Possible applications of such systems are virtual marketplaces where either agents or humans interact, simulation and training environments, the area of health and medical applications, home automation, etc..

Therefore, this work proposes the development of mechanisms, algorithms, tools and models that enable the creation of open systems where virtual agents and

humans coexist and interact transparently into a fully integrated environment. We call this type of systems as Human-Agent Societies (HAS). This project aims to provide answers to questions like: What is necessary to know and design for humans to interact with software agents? and how these interactions should be formalised and structured to obtain software products that are effective in such environments? How emergent behaviours can be included in these societies?. The objective of this project is to advance and provide solutions in these lines.

This project is proposed as an further research of the project "OVAMAH: Adaptive Virtual Organizations: Mechanisms, Architectures and Tools" (reference TIN 2009-13839-C03).

## **B** Related publications

---

In this section, the main publications related with this dissertation in the last two years are included.

### **B.1** International journals

---

1. C. Zato, G. Villarrubia, A. Sánchez, J. Bajo, and J. M. Corchado. Pangea: A new platform for developing virtual organizations of agents. *International Journal of Artificial Intelligence*, 11(A13):93–102, 2013.
2. C. Zato, A. Sánchez, G. Villarrubia, J. Bajo, and S. Rodríguez. Personalization of the workplace through a proximity detection system using user profiles. *International Journal of Distributed Sensor Networks*, 2013 (Article ID 281625):1–10, 2013.
3. C. Zato, J. F. De Paz, A. De Luis, J. Bajo, and J. M. Corchado. Model for assigning roles automatically in e-government virtual organizations. *Expert Systems with Applications*, 39(12):10389–10401, 2012.
4. C. Zato, A. De Luis, J. Bajo, J. F. De Paz, and J. M. Corchado. Dynamic model of distribution and organization of activities in multi-agent systems. *Logic Journal of IGPL*, 20(3):570–578, 2012.

5. R. González, C. Zato, R. Benito, J. Bajo, J.M. Hernández, J.F. De Paz, V. Vera, and Corchado J.M. Automatic knowledge extraction in sequencing analysis with multiagent system and grid computing. *Journal of integrative bioinformatics*, 9(3):206, 2012.
6. G. Verde, L. García-Ortiz, C. Zato, J. F. De Paz, S. Rodríguez, and M. A. Merchán. Platform image processing to study the structural properties of retinal vessel. *ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal*, 1(4):55–59, 2013.
7. D. I. Tapia, R. S. Alonso, J. F. De Paz, C. Zato, F. De la Prieta. A Tele-monitoring System for Healthcare Using Heterogeneous Wireless Sensor Networks. *International Journal of Artificial Intelligence*, 6(S11):93–102, 2011.

**B.2**

## Book chapters

1. C. Zato, S. Rodríguez, D. I. Tapia, J. M. Corchado, and J. Bajo. Virtual organizations of agents for monitoring elderly and disabled people in geriatric residences. In *Information Fusion (FUSION)*, 2013 16th International Conference on, pages 327–333. IEEE, 2013.
2. C. Zato, A. Sánchez, G. Villarrubia, J. Bajo, S. Rodríguez, and J. F. Paz. Personalization of the workplace through a proximity detection system using user’s profiles. In *7th International Conference on Knowledge Management in Organizations: Service and Cloud Computing*, volume 172 of *Advances in Intelligent Systems and Computing*, pages 505–513. Springer Berlin Heidelberg, 2013.
3. A. Sánchez, G. Villarrubia, C. Zato, S. Rodríguez, and P. Chamoso. A gateway protocol based on fipa-acl for the new agent platform pangea. In *Trends in Practical Applications of Agents and Multiagent Systems*, pages 41–51. Springer, 2013.
4. A. Sánchez, C. Zato, G. Villarrubia-González, J. Bajo, and J. F. De Paz. An integral system based on open organization of agents for improving the labour inclusion of disabled people. In *Distributed Computing and Artificial Intelligence*, pages 369–376. Springer, 2013.
5. J. F. De Paz, C. Zato, G. Villarrubia, J. Bajo, and J. M. Corchado. Distribution of roles in virtual organization of agents. In Lorna Uden, Leon S.L. Wang, Juan Manuel Corchado Rodríguez, Hsin-Chang Yang, and I-Hsien Ting, editors, *The 8th International Conference on Knowledge*

- Management in Organizations, Springer Proceedings in Complexity, pages 485–497. Springer Netherlands, 2013.
6. C. Zato, A. Sánchez, G. Villarrubia, J. Bajo, and S. Rodríguez. Integration of a proximity detection prototype into a vo developed with pangea. In *Management Intelligent Systems*, pages 197–204. Springer, 2012.
  7. C. Zato, G. Villarrubia, A. Sánchez, I. Barri, E. Rubión, A. Fernández, C. Rebate, J. A. Cabo, T. Álamos, J. Sanz, et al. Pangea—platform for automatic construction of organizations of intelligent agents. In *Distributed Computing and Artificial Intelligence*, pages 229–239. Springer, 2012.
  8. E. García, V. Gallego, S. Rodríguez, C. Zato, J. F. de Paz, and J. M. Corchado. Simulation and analysis of virtual organizations of agents. In *Trends in Practical Applications of Agents and Multiagent Systems*, pages 65–74. Springer, 2012.
  9. J. M. Corchado, G. Villarrubia, J. J. De Paz, S. Rodríguez, C Zato, and F. De la Prieta. Practical applications of virtual organizations and agent technology. In *Highlights on Practical Applications of Agents and Multi-Agent Systems, Multi-agent based Applications for Sustainable Energy Systems (MASSES)*, pages 17–23. Springer, 2013.

**B.3**

## Conferences

1. C. Zato, F. De la Prieta, S. Rodríguez, Y. Demazeau and J. M. Corchado. An adaptive multi-agent architecture for workplace integration. *ASMAS (Agentes y Sistemas Multi-Agente: de la Teoría a la Práctica (ASMas), Multiconferencia CAEPIA 2013*.
2. G. Villarrubia, J. Bajo, J. F. De Paz, C. Zato, and J. M. Corchado. Asignación de roles automática en organizaciones virtuales de agentes. In *Conferencia de la Asociación Española para la Inteligencia Artificial (CAEPIA 2013)*, pages 1523–1531, 2013.
3. C. Zato, S. Rodríguez, V. Parra, J. Bajo and J. M. Corchado. Sistema para la telemonitorización y la gestión del rastro asistencial de personas ancianas. *International Symposium on Artificial Intelligence and Assistive Technology (CEDI 2013)*, Madrid, 2013.
4. C. Zato, J. Bajo, and J. M. Corchado. A new platform for developing, management and monitoring open multiagent systems. In *Third International Workshop on Infrastructures and Tools for Multiagent Systems (ITMAS12)*, AAMAS 2012, page 27, 2012.

5. C. Zato, A. Sánchez, G. Villarrubia, S. Rodríguez, J. M. Corchado, and J. Bajo. Platform for building large-scale agent-based systems. In 2012 IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS 2012), pages 17–24, 2012.
6. Zato, C. and Sánchez, A. and Villarrubia, G. and Rodríguez, S. and Bajo, J. and Corchado, J. M. Personalization of the workplace using the PANGEA multiagent platform. In Intelligent systems for context-based information fusion (ISCIF 2012), IBERAMIA 12, 2012.





PHD. DISSERTATION

**Model for WCET Prediction, Scheduling and Task  
Allocation for Emergent Agent-behaviours in  
Real-time Scenarios**

**Author:** Davinia Carolina Zato Domínguez

University of Salamanca  
Spain  
July 2014

