A virtual laboratory for multiagent systems: Joining efficacy, learning analytics and student satisfaction

Luis Castillo
Dpt. Computer Science and Artificial Intelligence
University of Granada
18071 Granada, SPAIN
l.castillo@decsai.ugr.es

***Abstract*— This study introduces a distributed virtual laboratory for a multiagent programming course which has been very satisfactorily adopted by students, with a success rate of nearly 80%. It also aims at capturing the daily activity of students, providing the basis for data-driven assessment. Finally, it also allows for using process mining technologies to unveil successful and failed behaviors of students enabling the teacher for an early detection and intervention to improve their learning experience[1].**

***Keywords*—Learning analytics, data-driven assessment, multiagent systems**

I. INTRODUCTION

The use of virtual and remote laboratories to provide an enhanced learning experience for students is gaining interest in multiple disciplines as new communication technologies are widely adopted [5][6]. Their detailed implementation depends on the subject being taught [10] like computers networks, robotics, electronics but also psychology, biology, physics or chemistry. But they share also many common features [10] like enforcing privacy, scheduled access or support for reporting and assessment. This paper presents the distinguishing features, the results obtained and the main conclusions drawn after three academic courses of the setup of a virtual laboratory for multiagent systems programming in "Agent-based development", a 4th-year course of the degree of Computer Engineering at the University of Granada (Spain). There have been many implementations of virtual laboratories based on multiagent technology [16], but they mostly use agents as a vehicle to implement the infrastructure, not as a learning goal by itself. This paper focuses on mastering the technology and to foster a sound adoption of high-quality multiagent programming skills in distributed environments and it provides as a solid base to overcome the most important drawbacks of regular laboratories (explained in the next section). In addition to this, the implementation of a virtual laboratory grants students the access to the laboratory 24 hours a day, 7 days a week, so that they have more opportunities to improve their work. However, given that the virtual laboratory records all the interactions between the implemented agents, it ended up providing an extensive set of logged data which reflects how students have faced the practice work, day by day, and, therefore, providing a solid background for the use of several learning analytics [5]. This huge amount of information is not always easy to interpret [15], but this paper focuses on both data-driven assessment and the discovery of true patterns of successful and failed behaviors, an analysis that has shown a great potential for behavior change among the students [13][14].

II. DESIGN OF THE VIRTUAL LABORATORY

The course "Agent-based development" is strongly structured between theory and practice classrooms. Theory is devoted to general multiagent models, regardless of the programming language used to develop them, and it ranges from agents introduction, autonomy, communication and agent's societies. Practice laboratory is focused on implementing the theoretical models in Java programming language on top of Magentix 2 Agent Platform [4] on a variety of problems. Practice laboratory is organized in teams of 5-6 students to promote collaborative development and transverse competencies [1]. The main drawbacks found when students develop multiagent systems by themselves in a regular laboratory could be summarized as follows.

- Agents are situated entities, that is, their behavior depends on their environment, very often partly controllable. When student teams are left alone, the implementation of the access to and modification of the environment, the perception of agents and the interaction with other agents are full of programming shortcuts and tricks and therefore not fully satisfactory from a high-quality learning point of view. For instance, they tend to implement the environment as a shared memory object, something that is not forbidden in object-oriented programming, but it is not very recommendable in agent-based programming since every agent might execute in a different platform with different memory arrangements and accesses. It is preferable to communicate agents via message passing instead of sharing memory.
- The dialogue and message passing between agents follow strict protocols (e.g. the contract-net standard protocol designed by FIPA, the Foundation for Intelligent and Physical Agents [7]) which need to be adhered to. When implemented by the same team, these protocols could be partly ignored, but relying on external agents to validate the communication protocol requires a carefully crafted implementation of message passing.

Perhaps the most distinguishing feature of this virtual laboratory is that it has been designed to overcome these difficulties and to foster higher quality programming skills.

[1] Partly funded by TIN2015-71618-R (MINECO/FEDER)

Roughly speaking, this virtual laboratory consists of a server which runs a multiagent system who controls several virtual worlds and the students must connect to the server, with their own multiagent system, in order to solve several problems in these virtual worlds, by interacting with the agents in the server under a strict communication protocol (see Fig. 3).

A. *The problem*

The server contains several virtual worlds. Every virtual world is a square matrix that represents open spaces (in color white), obstacles (in black) and goals (in red) as it is shown in Fig. 1. Students' agents must enter into one of these virtual worlds, perceive their local surroundings, navigate through the open spaces (by using some exploratory heuristic), avoid obstacles and try to reach the goal.

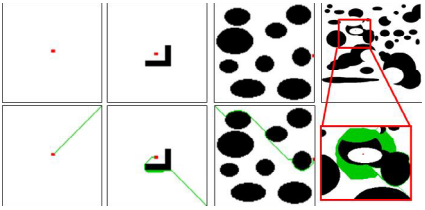


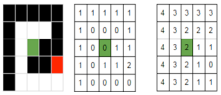
Fig. 1. Some of the virtual worlds to be solved by the students (upper pictures). Agents implemented by the students must log into one of these worlds and try to reach the goal (red-colored cells) by navigating the world and avoiding the obstacles (black-colored cells). Some of the worlds are not solvable because the goal is not reachable in order to force students' agents to reason about unsolvability. Possible successful trajectories (lower pictures) are depicted in green

Agents' perception of the environment is critical for solving these worlds. In this virtual laboratory, students can configure which of the following sensors are plugged into their agents (any combination of them):

- A GPS which tells the agents its coordinates (x,y) in the virtual world.
- A Battery sensor. Every agent is fed with a battery, with a limited capacity, and its charge decreases as long as the agent executes a movement. The charge of the battery should never be completely depleted.
- A Radar sensor which informs the agent about the type of cells that surrounds the agent in a 5x5 local perception (see Fig.2.b).
- A Scanner sensor which acts as a *goal detector* and tells the agent the distance to the goal measured from each of the 5x5 surrounding cells (see Fig. 2.c).

Based on their perception, every agent may decide to execute one of the following actions in their environment by implementing any heuristic or search procedure.

- LOGIN. Enter into any of the virtual worlds.
- MOVE. Move the agent to one of the 8 adjacent cells and expends a certain amount of battery. If the destination cell is an obstacle, or the agent runs out of battery, the agent crashes and it is logged out of the virtual world.
- REFUEL. The agent fully recharges its battery. Agents are allowed to recharge their battery as many times as they wish.



(a) (b) (c)

Fig. 2. An agent (the green cell in the middle of each figure) has a local perception of their environment: only the 5x5 surrounding cells (a) may be perceived. The Radar (b) shows the 5x5 cells that sourrounds the agent and informs whether it is an empty cell (value 0), an obstacle (value 1) or a goal (value 2). The Scanner (c) shows the distance from every surrounding cell towards the goal

The access to the virtual world and the environment of the agents is completely controlled and managed by the agents in the server. Thus, every agent implemented by students must follow the next steps, under a continuous *sense-think-act* loop:

1) Connect to the server. This is implemented through the regular Magentix 2 API and it is described in the next section.
2) Log agent(s) into a virtual world. There are two versions for each world, either with a single agent or multiple agents, all of them searching for the same goal. In the latter, agents must coordinate to avoid exploring twice the same area or to avoid crashing with each other.
3) Perceive the environment. Students should implement one of the following perception models.
   a) Synchronous perception. The agent receives a message from the server for each sensor attached to the agent, indicating the reading of that sensor, just after the execution of any action of the agent. It requires a subtle management of the incoming messages of the agent so that no reading could be lost nor to desynchronize movements and perceptions.
   b) Asynchronous model. The agent decides when to read the sensors by sending a message to the server with a request. The requesting agent will receive a response message with all the readings at that time. It is easy to manage but some readings might be lost if the requesting agent does not react fast enough.
4) Decide next action. Common choices of the students are reactive agents with memory, greedy algorithms or A* search.
5) Execute the desired action. Agents in the server receive a request from one of the students' agents, they simulate the execution of the action in the virtual world and inform back the agent of the result obtained.

III. SETTING EVERYTHING UP

The setup of this distributed virtual laboratory obeys the architecture depicted in Fig. 3. The base hardware is a dedicated

server intel Xeon running Ubuntu Linux with a VPN access restricted only to computers from inside of the campus of the University of Granada (despite of which the server received about 100 daily refused connections to *sshd* coming from unknown IPs). The multiagent platform is Magentix 2 [4] with a Java-based message broker [9]. This allows the server to open a virtual host for each team of students with an additional privacy reinforcement: every group must provide a username, a password and a virtual host when connecting to the server and every virtual host runs a different, hermetic, multiagent system to control the virtual worlds. That means that each virtual host and their associated multiagent controllers act independently of each other, avoiding name conflicts between agents and avoiding messages leaks from one group to another so that every group of students has a completely independent and safe area to effectively carry out their laboratory work.

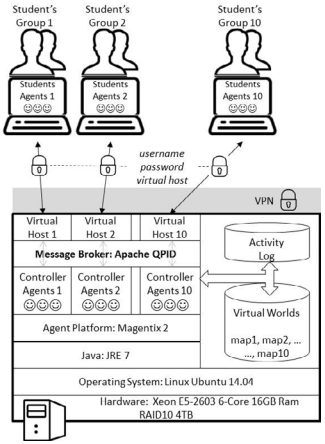


Fig. 3. Arquitecture of the virtual laboratory for distributed multiagent systems. It is supported by a Xeon server with a double security barrier. Students's groups must both connect through the University's VPN, and provide an additional username and password to connect to their own multiagent system. All multiagent systems in the server log all the transactions and have access to several virtual worlds

Students are allowed to connect to their associated controller multiagent system and interact with them to solve all virtual worlds as commented in Section II. Students' teams also have full control of their associated controller agents in their virtual host. Should any of these associated controller agent crash, students, by themselves, are allowed to reboot the controller multiagent system, so that teams are completely independent and don't need to rely on the teacher to interact, log into and play with any virtual world. Therefore, a 24x7 availability of the virtual laboratory is completely granted and the server is online all the time during the development of the course, since the last three academic years.

In addition to this, each controller multiagent system in the server logs all the activity and message exchange with the students' multiagent sytem (including sensor readings and agents' trajectories as in 0). Every transaction is stored as a JSON string [8] and it contains a timestamp, the agents involved in the transaction and the body of the message passed (also a JSON string) as it is shown in Fig. 8.

IV. RESULTS

Before entering into the detailed results of this distributed virtual laboratory, it is worth saying that, in order to provide further efficacy and autonomy of the teams of students and to promote additional transverse competencies, the whole practice work follows a SCRUM agile methodology [1] and it has provided excellent performance results during the past three years from a software engineering point of view.

This architecture allows the virtual laboratory to be an effective means from the point of view of the learning experience, due to the following reasons.

- It fosters the adoption of best multiagent programming practices among students since, in order to solve the virtual worlds, student's multiagent systems must interact with an already implemented multiagent system in the server, who controls the environment and its perception, simulate the execution of the actions in the environment, avoid the use of shared memory and enforce the adherence to strict agents communication protocols.
- It provides a safe environment with a double security barrier.
- It allows the autonomous development of the practice work for each team, avoiding interferences with other teams and does not require the presence of the teacher to reboot their virtual server in the exceptional case of crash of the controller multiagent system.

Indeed, the academic grades obtained by students in the "Agent-Based Development" course during the last three years have been very good, as it is shown in Fig. 4, with an average of 79% of success. The global satisfaction and engagement measured in official questionnaires based on Likert scales are also outstanding (shown in Fig. 5) reaching 4.86 out of 5 in the last year.

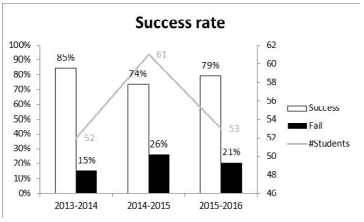


Fig. 4 Final academic result of the whole course (theory and practice) during the last three years showing a very good success rate of 79% in average

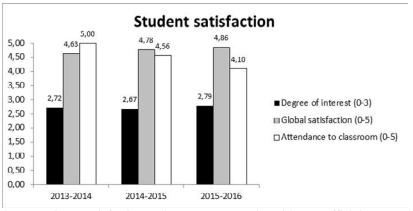


Fig. 5 Students satisfaction and engagement gathered by an official personal and anonymous survey among students

Furthermore, the logs registered by the controller multiagent systems in the server also unveil information of great value from a learning analytics point of view. Just to give an idea of the size of these logs, only in the third year, 2.2GB of transactions have been logged in disk [17], what amounts for more than 11.5 million transactions, with more than 4500 work sessions opened (login requests into a virtual world).

A. *Data driven assessment*

This section refers to the support to the evaluation of students based on valuable information extracted from the massive log of transactions like that summarized in Fig. 8. This log of transactions constitutes a footprint of students performance at any time during the course, since the server of the virtual laboratory is running non-stop, and there are several measures that could provide additional insight for the evaluation of students' performance [5] like the following items.

- Single team measures. Indicators of the performance of a single team.
  - Number of trials. Number of work sessions opened in the same world until the world has been solved for the first time.
  - Time elapsed from the first access to the world to its first solution.
  - Absolute time elapsed to the first solution of a world since the beginning of the course.
  - Number of different solutions found for the same problem. Each solution might have a different length (number of movements until the goal is reached).
- Collective measures. Competitive indicators of the performance among different teams.
  - Quickest team: the one with the lowest time elapsed or lowest number of trials.
  - Early bird: team with the lowest absolute time elapsed.
  - Most efficient team: the one with the lowest solution length.
  - Most sound team: the one with the highest number of worlds solved.

These values, which are obtained from the logged transactions in real time, provide valuable insight for evaluating the performance of each team, making visible the invisible and showing a real measure of the progress of every team as a key component of the final grades. The experience during the last three years shows that this information is extremely valuable for teachers but, since it is obtained in real time, it may be used purposely. Indeed, the key performance indicators of the progress of every team are made public every two weeks to all the teams so that all teams are aware of the progress of the remaining teams, what produces some expected changes in the behavior of the teams to improve their performance by self regulation [5][13].

B. *Discovering patterns of success and failure*

Logged transactions are very valuable from the student's evaluation point of view, but they are a very rich source of information for other purposes too. It also provides a footprint on how students have solved every world, step by step, and a sort of record of the strategy followed by each team to try to solve all the worlds. The use of process mining techniques [11] might unveil these hidden strategies and see it as the process followed by students until they reached the goal. In order to do that, raw data logs must be filtered to adapt to the simple CSV format required by a free process mining tool like Disco [2]. This filtering leaves out administrative logs and focuses particularly on the interactions among the students' agents and the server's agents and, more specifically, on the requests to perform an action which come from the students' agents. Thus, starting from a raw log like that shown in Fig. 8 of about 11.5 millions of transactions, the filtered log, like that shown in Fig. 9, reduces to a little more than 1.6 million records, filtering out useless transactions from the point of view of this investigation. This last figure shows the five more relevant items of information extracted from the raw record:

1) The Case ID. It is extracted from a random key generated at the beginning of each LOGIN operation (see Fig. 8) and it univocally distinguishes every work session of the students.
2) The Agent. It is the name of the team of students.
3) Timestamp. The date and time record of the transaction.
4) The field Activity refers to the action requested by students to be executed in the virtual world. There are regular actions allowed by the server like LOGIN, MOVE, REFUEL (as explained in Section II.A) plus an additional SUCCESS activity. This activity informs that the corresponding world has been solved and it is automatically detected when the students' agents are notified of a reading of the Radar which shows that the agent is located over a goal cell of the world (please refer to Fig. 2 for details about agent's perceptions).
5) The field Resource refers to the virtual world in which the transaction took place. There are 10 different maps named from "map1" (the leftmost empty map in Fig. 1) to "map10" (the rightmost unsolvable map is map9).

Our experience in process mining problems indicates that having a massive and clean record of activity, like that shown in Fig. 9, is not enough for obtaining good results with any process miner. The reason is that out of the 1.6 million of clean

transactions there are many mistaken cases which should be filtered out too, not to affect to the final result, that is:

- There are incomplete records due to an unexpected interruption of the case:
  - Agents which have crashed into an obstacle.
  - The Java implementation of any of the students' agents freezes and stops sending messages to the server.
  - Students decided to interrupt their agents by hand while they debug their agents.
  - Students' agents do not follow the strict protocol of communication and, therefore, they are logged out of the server, stopping the case.
  - Agents in the server receive an unreadable message (usually due to empty messages or bad JSON formatting) what causes the sender agent to be logged out of the server.
- There is just one world which never ends with SUCCESS (rightmost world in Fig. 1 which cannot be solved because it is a dead end).
- There are failed LOGIN actions into inexistent worlds or requests to execute an unknown action.
- There are synchronization problems within students' agents which do not detect on time that they have reached the goal (see Fig. 6). An early detection of this problem would warn the students of the committed mistake in order to repair it.

Once these incorrect cases are detected and filtered out, DISCO process miner can be used to detect hidden, but useful, behaviors of the students, taking into account that each team might have implemented different exploratory techniques to solve each world. The most important findings of DISCO are the following ones.

```
Case, Agent, Timestamp, Activity, Resource
zsr2f57z,Achernar,19/11/2015_12:36:03,MOVE,map1
zsr2f57z,Achernar,19/11/2015_12:36:03,SUCCESS,map1
zsr2f57z,Achernar,19/11/2015_12:36:03,MOVE,map1
zsr2f57z,Achernar,19/11/2015_12:36:03,SUCCESS,map1
zsr2f57z,Achernar,19/11/2015_12:36:03,MOVE,map1
zsr2f57z,Achernar,19/11/2015_12:36:04,SUCCESS,map1
zsr2f57z,Achernar,19/11/2015_12:36:04,MOVE,map1
```

Fig. 6. Synchronization problem between perceptions of the agent and its requested activities which illustrates a deviated behavior of agents that are not aware of having reached the goal and continue moving unnecesarilly

The most frequent process mined at the beginning of the course is shown in Fig. 7. It means that the most common activity requested to the server is LOGIN into map1. Even many groups only execute that action and then close the session immediately after. This is because, at the beginning, almost none of the groups have implemented any exploratory heuristic yet and just log into the simplest world (map1) to test the communication and the JSON encoding of the messages. Soon after that, they start moving in map1 until they solve it. Some groups recharge the battery at the beginning but others don't because it is not needed to solve this tiny world. There are some synchronization problems to solve yet (like unnecessary transitions from SUCCESS back to MOVE) as explained above. Then they move into map2 (second map from the left in Fig. 1) and just start moving like in map1 but do not solve it yet (since it requires a more complex exploratory heuristic with obstacle avoidance).

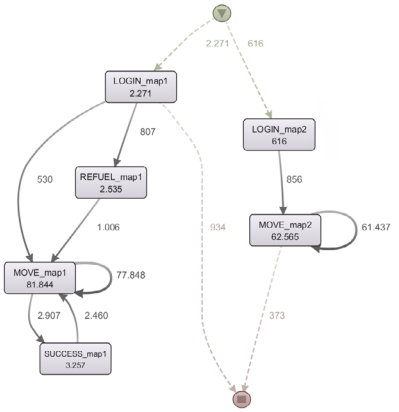


Fig. 7. General process mined at the beginning of the course. Boxes are logged activities and arc are transitions from one activity to another. Numbers close to arcs and boxes indicate the frequency that this transition has been recorded. The start and end of sessions is marked with circles at the top and bottom of the process

These results would seem quite obvious but they are not. If one segments the data logged depending on the final grade of students one might find interesting differences. For example, the process mined taking into account only the data coming from the team with the highest grade (honor) shows a sort of systematic behavior: first solve map1, then solve map2 then directly move to the most difficult one (map9). On the opposite side, the process mined for the team with the lowest grade shows that they solve the easiest world (map1) and then they successively wander into other worlds without any success, giving up the search very soon. These mined processes unveil the different progress of teams, but they also highlight the differences among teams' performance. These differences might be used for the teacher to help students to perform better based on the segmentation of the data logged.

V. CONCLUSIONS

This study has shown a learning experience consisting of a distributed virtual laboratory that fosters the acquisition of high quality skills and habits in multiagent programming. The grades obtained by students and their satisfaction are very high, as shown by anonymous surveys, since the experience was setup, three years ago. In addition to this, it has shown to be an excellent platform for capturing the daily activity of students in order to use standard process mining tools. These tools unveil hidden behaviors of students, which could be used to improve

their learning experience. In the future, these mined processes could also be used to feed an automatic virtual assistant which could guide students on how to succeed in the laboratory work by understanding what they have done and what there is left to do. But, instead of basing these suggestions in standard learning routes created by the teacher [3] these suggestions would be based on what their laboratory companions are really doing or did to succeed.

REFERENCES

[1] L. Castillo, "The use of SCRUM for laboratory sessions monitoring and evaluation in a university course. Enforcing transverse competencies", SIIE 2014.

[2] DISCO Process miner. https://fluxicon.com/disco/. 2016

[3] L. Castillo, L. Morales, A. González-Ferrer, J. Fdez-Olivares, D. Borrajo, E. Onaindía. Automatic generation of temporal planning domains for e-learning problems. Journal of Scheduling (2010), vol. 13, n. 4, p. 347-362.

[4] J. M. Such, A. García-Fornes, A. Espinosa and J. Bellver. Magentix2: A privacy-enhancing Agent Platform. Engineering Applications of Artificial Intelligence (2013) vol. 26, n.1, p. 96–109. http://www.gti-ia.upv.es/sma/tools/magentix2/index.php

[5] New Media Consortium. NMC Horizon Report > 2014 Higher Education Edition. 2014

[6] B. Balamuralithara, P. C. Woods. Virtual Laboratories in Engineering Education: The Simulation Lab and Remote Lab. Comput Appl Eng Educ 17:108, p118, 2009.

[7] FIPA, Contract Net Interaction Protocol Specification. http://www.fipa.org/specs/fipa00029/SC00029H.pdf, 2002.

[8] JavaScript Object Notation JSON, https://en.wikipedia.org/wiki/JSON, 2016.

[9] Apache Foundation. QPID Broker. https://qpid.apache.org/

[10] S.Rigby, M. Dark. Designing a Flexible, Multipurpose Remote Lab for the IT Curriculum. SIGITE 2006, p. 161-164.

[11] W. van der Aalst. Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer, 2011.

[12] Z. Nedic, J. Machotkd, A. Najhlsk. Remote Laboratories Versus Virtual And Real Laboratories. ASEE/IEEE Frontiers in Education Conference. 2003.

[13] P. Long, G. Siemens. Penetrating the fog: analytics in learning and education. EDUCAUSE. Sept-Oct 2011, p. 31-40.

[14] K. Verbert, E. Duval, J. Klerkx, S. Govaerts, J.L. Santos. Learning Analytics Dashboard Applications. American Behavioral Scientist (2013), p. 1-10.

[15] G. Siemens. Learning Analytics 2011: Reflections. http://www.elearnspace.org/blog/2011/03/11/learning-analytics-2011-reflections/

[16] Norman, T. J., & Jennings, N. R. (2002). Constructing a virtual training laboratory using intelligent agents. International Journal of Continuing Engineering Education and Life Long Learning, 12(1-4), 201-213.

[17] L. Castillo. Takeaways of this paper: Java source code of server agents, raw and filtered logs. http://decsai.ugr.es/~lcv/SIIE2016


```
{"date":"19/11/2015_12:42:27",
  "value":{"agent":"Achernar", "key":"r2bc7snv", "content":{"status":"Subscribing radar sensing to AgenteDirectorPSP"}}}
{"date":"19/11/2015_12:42:27","value":{"agent":"Achernar", "key":"r2bc7snv", "content":{"status":"Waiting for action"}}}
{"date":"19/11/2015_12:42:27","value":{"agent":"Achernar_satellite", "key":"r2bc7snv", "content":{"status":"Sending ACLM",
  "receiver":"AgenteDirectorPSP", "body":{"radar":[1,1,1,1,1,1,1,1,1,1,1,0,0,2,1,1,0,0,0,1,1,0,0,0,1,1]}}}}
{"date":"19/11/2015_12:42:27",
  "value":{"agent":"Achernar_satellite", "key":"r2bc7snv", "content":{"status":"Waiting for message"}}}
{"date":"19/11/2015_12:42:41",
  "value":{"agent":"Achernar", "key":"r2bc7snv", "content":{"status":"Received ACLM", "sender":"AgenteRobotPSP",
  "body":{"command":"moveSW","key":"r2bc7snv"}}}}
{"date":"19/11/2015_12:42:41", "value":{"agent":"Achernar", "key":"r2bc7snv", "content":{"status":"OK moveSW"}}}
```


Fig. 8. Sample log of activity (raw format in JSON). Every transaction has a timestamp, the controller agent that records the transaction, and the remaining parameters like internal states, messages received and sent, perceptions, requests to execute an action, etc. In this case all the records shown belong to the same work session identified as `"key":"r2bc7snv"`. It may also be seen that during this work session there is a Radar reading that contains a value '2' just in the center of the reading of the sensor indicating that the corresponding agent has just reached the goal and therefore, the problem has been solved. Only in the third year of the course, this raw record of transactions stores 2.2GB of data, that amounts for more than 11.5 millions of transactions


```
Case,           Agent,          Timestamp,              Activity,       Resource
v170xpay,       Achernar,       16/10/2015_10:33:00,    LOGIN,          map1
v170xpay,       Achernar,       16/10/2015_10:33:00,    MOVE,           map1
9sgmylxl,       Achernar,       19/11/2015_12:49:01,    MOVE,           map1
9sgmylxl,       Achernar,       19/11/2015_12:49:01,    SUCCESS,        map1
```


Fig. 9. Sample log of activity (clean format for Disco process miner [2]). The cleaning of a raw record like that shown in Fig. 8 throws a total of more than 1.6 millions of clean records