



VNiVERSIDAD
D SALAMANCA



Estratégias para o ensino e a aprendizagem da iniciação à programação no ensino superior

José Alberto Quitério Figueiredo

Tesis Doctoral
Programa de Doctorado Ingeniería Informática (R.D.99/2011)
Departamento de Informática y Automática
Universidad de Salamanca (<https://ror.org/02f40zc51>)
Salamanca, Espanha

Tutor: **Francisco José García-Peñalvo, Ph.D.**

Abril 2024

Francisco José García-Peñalvo, Doctor, Profesor Titular del Departamento de Informática y Automática de la Universidad de Salamanca (España), como director de la tesis doctoral titulada "Estratégias para o ensino e a aprendizagem da iniciação à programação no ensino superior" desarrollada por José Alberto Quitério Figueiredo.

Por la presente declara que

Esta tesis doctoral, desarrollada en el contexto del Programa de Doctorado de Informática del Departamento de Informática y Automática de la Universidad de Salamanca (<https://ror.org/02f40zc51>), presenta méritos suficientes (teóricos y prácticos) evaluados mediante la oportuna evaluación, publicaciones y propuestas originales para ser presentada y defendida públicamente.

Salamanca, Abril 2024

GARCIA	Firmado digitalmente
PEÑALVO	por GARCIA PEÑALVO
FRANCISCO JOSE	FRANCISCO JOSE -
- 07862615L	07862615L
	Fecha: 2024.04.30
	12:33:55 +02'00'

Francisco José García-Peñalvo, Ph.D.
Universidad de Salamanca

Estratégias para o ensino e a aprendizagem da iniciação à programação no ensino superior

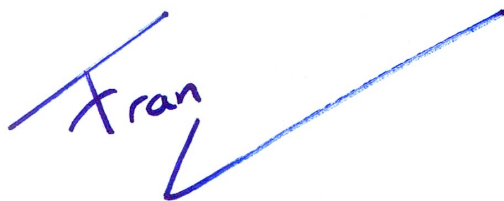
Tesis Doctoral

Programa de Doctorado Ingeniería Informática (R.D.99/2011)

Departamento de Informática y Automática

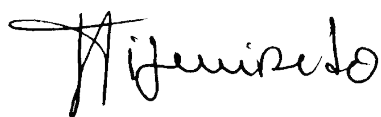
Universidad de Salamanca (<https://ror.org/02f40zc51>)

Salamanca, Espanha



Tutor

Francisco José García-Peñalvo, Ph.D.



Candidato

José Alberto Quitério Figueiredo

Abril 2024

Agradecimentos

Gostaria de agradecer ao meu orientador Professor Doutor Francisco José García-Peñalvo, pela sua visão, apoio e partilha de conhecimentos que tornaram esta tese possível.

Gostaria de agradecer aos meus filhos Francisco, Ricardo, Susana e à minha esposa Diana pela compreensão, encorajamento e carinho ao longo de todos estes anos, por terem estado sempre ao meu lado nos bons e maus momentos e sem os quais este projeto não seria possível. Gostaria também de agradecer aos meus Pais pelo apoio e incentivo ao longo de todos estes anos.

Aos colegas do Instituto Politécnico da Guarda, em especial aos vizinhos do gabinete 20, pela amizade e que das mais diversas formas contribuíram para a realização deste trabalho.

Por fim, os meus agradecimentos a todos os alunos que participaram nos mais diversos estudos, sempre com especial vontade em colaborar e de contribuir para a melhoria do seu próprio processo de ensino e aprendizagem da programação.

A todos vós, obrigado.

Resumo

A aprendizagem da programação permite adquirir competências técnicas para o desenvolvimento de software. Mas também, o desenvolvimento de competências cognitivas como a resolução de problemas, o raciocínio lógico, a criatividade, ou o pensamento crítico. Competências essas muito apreciadas e vistas como fundamentais para todos os que enfrentam o mercado de trabalho de hoje e do futuro.

A tecnologia tem um forte impacto na sociedade, ao ensinar programação capacitamos os alunos a entender melhor a tecnologia, a serem utilizadores responsáveis, mais informados e conscientes.

No entanto, o ensino e a aprendizagem da programação são uma tarefa complexa. As dificuldades no ensino e na aprendizagem da programação tem sido um tema recorrente ao longo dos tempos. Muitos alunos enfrentam dificuldades na fase inicial da aprendizagem da programação. O ensino e a aprendizagem da programação requerem muito trabalho, dedicação e empenho de todos os envolvidos neste processo.

Esta tese de doutoramento enfrenta o desafio de melhorar o processo de ensino e da aprendizagem da iniciação à programação, num contexto muito específico, na unidade curricular de introdução à programação, do curso de engenharia informática, do Instituto Politécnico da Guarda.

O nosso grupo de alunos tem características muito particulares. A maioria dos alunos do nosso curso tem baixos índices de motivação e fracos conhecimentos gerais. Uma grande percentagem dos alunos não adquire as competências básicas em programação. Este problema é de extrema importância, os fracos conhecimentos em programação serão sempre um elemento bastante perturbador à persecução dos estudos no curso de engenharia informática.

Com o principal objetivo de minimizar este problema, investigamos, experimentamos e implementamos um conjunto de atividades e metodologias de ensino que nos permitiram melhorar as competências básicas em programação. Para a concretização deste objetivo e, atendendo às características do nosso grupo de estudo, usamos a metodologia de investigação científica, com o desenvolvimento de um artefacto, designada por *Design Science Research Methodology*.

Desta investigação resultou uma estratégia para o ensino e a aprendizagem da iniciação à programação que permitiu melhorar significativamente as competências dos nossos alunos.

Para esta melhoria contribuíram claramente as atividades do pensamento computacional usadas como introdução e um conjunto de atividades de escolha múltipla e *parson problems*, com *feedback*

imediatamente, colocadas em prática para os conceitos iniciais da aprendizagem da programação. Mas também, o grande número de exercícios de codificação, também com *feedback* imediato, disponíveis para os alunos praticar e consolidar os diferentes conceitos de aprendizagem da iniciação à programação.

Nesta estratégia, incluímos um modelo preditivo do insucesso do aluno, baseado numa rede neuronal, com o objetivo de detetar precocemente as dificuldades do aluno e agir rapidamente em seu auxílio.

Toda a estratégia envolvida no processo de ensino e de aprendizagem na iniciação à programação, não seria possível de concretizar sem o auxílio de uma ferramenta tecnológica de gestão. Apresentamos neste trabalho, o desenvolvimento de uma ferramenta tecnológica de ajuda ao processo de ensino e de aprendizagem.

Os resultados obtidos na avaliação da unidade curricular de introdução à programação, em função da estratégia e ferramenta adotada, mostram que melhoramos significativamente o nosso processo de ensino e de aprendizagem. Indicando que estamos no bom caminho e os esforços realizados para que um maior número de alunos adquira as competências básicas em programação, foi concretizado.

O nosso trabalho tem uma origem e características muito particulares, no entanto estamos convictos de que as medidas sugeridas contribuirão com melhorias significativas em qualquer sistema de ensino e de aprendizagem da iniciação à programação.

Concluimos este trabalho com uma abordagem sobre o que poderá ser a utilização e o impacto de sistemas inteligentes, como o *ChatGPT*, no ensino e na aprendizagem da iniciação à programação.

Palavras Chave

Introdução à Programação; CS1; Programação; Codificação; Ensino Iniciação Programação; Aprendizagem Iniciação Programação;

Abstract

Learning to program allows you to acquire technical skills for software development. But also, the development of cognitive skills such as problem solving, logical reasoning, creativity, and critical thinking. These skills are highly valued and seen as fundamental for anyone facing the job market today and in the future.

Technology has a strong impact on society and by teaching programming we enable students to better understand technology, to be responsible, more informed, and aware users.

However, teaching and learning of programming is a complex task. Difficulties in teaching and learning programming have been a recurring theme throughout the ages. Many students face difficulties in the initial stages of learning programming. Teaching and learning programming requires a lot of work, dedication and commitment from everyone involved in this process.

This doctoral thesis faces the challenge of improving the teaching and learning process of beginning programming, in a very specific context, in the introduction to programming curricular unit of the computer science course at the Instituto Politécnico da Guarda.

Our group of students has very particular characteristics. Most of the students on our course have low levels of motivation and poor general knowledge. A large percentage of students do not acquire basic programming skills. This problem is extremely important, as poor programming skills will always be a major obstacle to the pursuit of studies in computer science course.

With the main objective of minimizing this problem, we investigated, experimented, and implemented a set of activities and teaching methodologies that allowed us to improve basic programming skills. To achieve this goal, and given the characteristics of our study group, we used the scientific research methodology, with the development of an artifact, called *Design Science Research Methodology*.

This research has resulted in a strategy for teaching and learning initial programming that has significantly improved our students' skills.

The computational thinking activities used as an introduction and a set of multiple-choice activities and *parson problems*, with immediate *feedback*, put into practice for the initial concepts of learning programming, clearly contributed to this improvement. But also, the large number of coding exercises, also with immediate feedback, available for students to practice and consolidate the different learning concepts of initial programming.

In this strategy, we have included a predictive model of student failure, based on a neural network, with the aim of detecting the student's difficulties early on and acting quickly to help them.

The entire strategy involved in the teaching and learning process in the initial programming would not be possible without the help of a technological management tool. In this work, we present the development of a technological tool to aid the teaching and learning process.

The results obtained in the evaluation of the Introduction to Programming course, depending on the strategy and tool adopted, show that we have significantly improved our teaching and learning process. This indicates that we are on the right track and that the efforts made to ensure that a greater number of students acquire basic programming skills have been achieved.

Our work has a very particular origin and characteristics, but we are convinced that the measures suggested will contribute to significant improvements in any system for teaching and learning initial programming.

We conclude this work with an approach to what could be the use and impact of intelligent systems, such as *ChatGPT*, in teaching and learning initial programming.

Keywords

Programming; Learning Initial Programming; Teaching Initial Programming; CS0; CS1

Resumen

Aprender a programar permite adquirir competencias técnicas para el desarrollo de software. Pero también el desarrollo de competencias cognitivas como la resolución de problemas, el razonamiento lógico, la creatividad y el pensamiento crítico. Estas competencias son muy valoradas y se consideran fundamentales para cualquier persona que se enfrente al mercado laboral de hoy y del futuro.

La tecnología tiene un fuerte impacto en la sociedad y enseñando programación permitimos a los estudiantes comprender mejor la tecnología, ser usuarios responsables, más informados y conscientes.

Sin embargo, enseñar y aprender a programar es una tarea compleja. Las dificultades en la enseñanza y el aprendizaje de la programación han sido un tema recurrente a lo largo de los tiempos. Muchos estudiantes se enfrentan a dificultades en las fases iniciales del aprendizaje de la programación. Enseñar y aprender programación requiere mucho trabajo, dedicación y compromiso por parte de todos los implicados en este proceso.

Esta tesis doctoral aborda el reto de mejorar el proceso de enseñanza y el aprendizaje de la iniciación de la programación, en un contexto muy concreto, en la unidad curricular de Introducción a la Programación de la carrera de Ingeniería Informática del Instituto Politécnico de Guarda.

Nuestro grupo de estudiantes tiene características muy particulares. La mayoría del estudiantado de nuestro curso tienen bajos niveles de motivación y escasos conocimientos generales. Un gran porcentaje de estudiantes no adquiere conocimientos básicos de programación. Este problema es extremadamente importante, ya que unos conocimientos de programación deficientes siempre serán un obstáculo importante para cursar estudios de ingeniería informática.

Con el objetivo principal de paliar este problema, investigamos, experimentamos e implementamos un conjunto de actividades y metodologías didácticas que nos permitieron mejorar las competencias básicas de programación. Para conseguir este objetivo, y dadas las características de nuestro grupo de estudio, utilizamos metodología de investigación científica, con el desarrollo de un artefacto, denominado *Design Science Research Methodology*.

Esta investigación ha dado lugar a una estrategia para la enseñanza y el aprendizaje de la iniciación de la programación que ha mejorado notablemente las competencias de nuestros alumnos.

Las actividades de pensamiento computacional utilizadas como introducción y un conjunto de actividades de elección múltiple y *problemas de parson*, con realimentación inmediato, puestas en práctica

para los conceptos iniciales del aprendizaje de la programación, contribuyeron claramente a esta mejora. Pero también, el gran número de ejercicios de codificación, también con realimentación inmediato, puestos a disposición del alumnado para practicar y consolidar los diferentes conceptos de aprendizaje de la programación inicial.

En esta estrategia, hemos incluido un modelo predictivo del fracaso escolar, basado en una red neuronal, con el objetivo de detectar precozmente las dificultades del alumnado y actuar con rápidamente para ayudarles.

Toda la estrategia implicada en el proceso de enseñanza y aprendizaje de la programación para principiantes no sería posible sin la ayuda de una herramienta tecnológica de gestión. En este trabajo presentamos el desarrollo de una herramienta tecnológica de ayuda al proceso de enseñanza y aprendizaje.

Los resultados obtenidos en la evaluación del curso de Introducción a la Programación, en función de la estrategia y la herramienta adoptadas, muestran que hemos mejorado significativamente nuestro proceso de enseñanza y aprendizaje. Esto indica que vamos por el buen camino, y que los esfuerzos realizados para que un mayor número de alumnos adquieran las competencias básicas de programación, se han hecho realidad.

Nuestro trabajo tiene un origen y unas características muy particulares, pero estamos convencidos de que las medidas sugeridas supondrán mejoras significativamente en cualquier sistema de la enseñanza y el aprendizaje de la iniciación a la programación.

Concluimos este trabajo con una breve aproximación a lo que podría ser el uso e impacto de sistemas inteligentes, como *ChatGPT*, en la enseñanza y el aprendizaje de la iniciación a la programación.

Palabras Clave

Programación; Codificación; Enseñanza iniciación programación; Aprendizaje iniciación programación;

Conteúdo

1	Introdução	1
1.1	Problema	3
1.1.1	Caraterização do grupo de estudo	5
1.1.2	Causas para o problema	6
1.1.3	Questões de investigação	8
1.2	Objetivos	9
1.3	Metodologia	12
1.4	Contexto, estrutura e organização	13
2	Estado da arte	15
2.1	Pensamento Computacional	18
2.2	Conceções erradas e dificuldades	20
2.3	Aptidão para a programação	21
2.4	Aprendizagem autoregulada	22
2.5	Fatores que promovem o sucesso	23
2.6	Competências na resolução de problemas	24
2.7	Live-Coding no ensino de introdução à programação	25
2.8	Erros de codificação	26
2.9	Parson Problems	28
2.10	Gamificação	29
2.11	Conclusão	30
3	Aplicação Design Science Research Methodology na Introdução à Programação	31
3.1	Identificação e motivação do problema	33
3.2	Definição dos objetivos para uma solução	37
3.3	Design e desenvolvimento	40
3.3.1	Prólogo ao ensino e aprendizagem da programação	40
3.3.2	Aprendizagem da programação em C	43
3.3.3	Construção da Aplicação HTProgramming	47

3.3.3.A	Módulo de administração	47
3.3.3.B	Módulo do aluno	50
3.3.3.C	Modelo preditivo do insucesso do aluno	53
3.4	Demonstração	54
3.5	Avaliação	58
3.5.1	Avaliação pelos alunos	58
3.5.2	Avaliação do modelo preditivo de insucesso do aluno	59
3.5.3	Avaliação pelos resultados finais	61
3.6	Comunicação	62
3.7	Conclusão	64
4	Discussão	67
4.1	Perfil do aluno	69
4.2	Linguagem de programação adotada	72
4.3	Tipo de Atividades	74
4.4	Modelo preditivo	77
4.5	Ferramenta automática	77
4.6	Conclusão	78
5	Conclusão	81
5.1	Principais contributos	84
5.2	Trabalho Futuro	86
5.3	Publicações	89
	Referências	93
A	ChatGPT no ensino e aprendizagem inicial da programação	109
A.1	ChatGPT na avaliação de identificadores	109
A.2	ChatGPT na avaliação de instruções	115
A.3	ChatGPT na correção de programas	116
A.4	ChatGPT na construção de programas	119
A.5	ChatGPT na ajuda aos Parson Problems	121

Lista de Figuras

3.1	Exemplo de atividade de visualização espacial, um exemplo de dobrar e perfurar papel, adaptado de [1].	42
3.2	Exemplo de atividade de seguir e dar instruções.	42
3.3	Esquema geral da aplicação HTProgramming.	48
3.4	Módulo de administração	48
3.5	Módulo de administração, visualização pormenorizada das atividades realizadas, tentativas e resultados.	49
3.6	Opção Basics Concepts, visualização de atividades.	50
3.7	Opção Basics Concepts, configuração de uma nova atividades.	50
3.8	Opção Coding Activity, visualização e configuração de atividades de codificação.	50
3.9	Opção Coding Activity, configuração de casos de teste para a atividade de codificação.	50
3.10	Opção Presences, registo de presenças dos alunos.	51
3.11	Opção Presences, visualização por aula do número de registo de alunos.	51
3.12	Dados de caracterização do aluno.	51
3.13	Opção <i>Parson Problems</i> , visualização de atividades <i>Parson Problems</i>	52
3.14	Opção <i>Parson Problems</i> , criação de atividades <i>Parson Problems</i>	52
3.15	Matriz de dados totais dos alunos, dados a utilizar no modelo preditivo de insucesso do aluno.	52
3.16	Opção Gamification, sorteio e registo de resultados.	52
3.17	Módulo do aluno, validação de entrada.	53
3.18	Módulo do aluno, resultados e atividades realizadas.	53
3.19	Modelo preditivo do insucesso do aluno.	54
3.20	Matriz de confusão. VP-Verdadeiros Positivos; FN-Falsos Negativos; FP-Falsos Positivos; VN-Verdadeiros Negativos	59
A.1	Exemplo de programa, atividade do tipo <i>Parson Problem</i> , programa original vs programa com instruções desordenadas.	121

A.2 Exemplo de programa, atividade do tipo <i>Parson Problem</i> , programa original vs programa com instruções desordenadas.	122
---	-----

Lista de Tabelas

1.1	Notas de acesso e número da opção ao ensino superior dos alunos do curso de engenharia informática do Instituto Politécnico da Guarda (IPG), nos últimos 5 anos.	5
1.2	Total de alunos inscritos na unidade curricular de introdução à programação e tipo de acesso, nos últimos 5 anos.	6
3.1	Avaliação de conhecimentos em programação. Resultados dos alunos do IPG, anos letivos de 2021 e 2022.	34
3.2	Resultados do estudo do <i>McCracken Group</i> , realizado em [2].	34
3.3	Resultados finais da avaliação, dos últimos 6 anos, da unidade curricular de introdução à programação, do curso de engenharia informática, do IPG.	35
3.4	Exemplo de atividade do tipo Multiple Choice Question (MCQ), sobre o tópico " <i>Nomes e variáveis</i> ".	43
3.5	Exemplo de atividade do tipo MCQ, sobre o tópico " <i>Tipo de dados e constantes</i> ".	44
3.6	Exemplo de atividade do tipo MCQ, sobre o tópico " <i>Input/Output, formatação e constantes</i> ".	44
3.7	Exemplo de atividade do tipo MCQ, sobre o tópico " <i>Avaliação de instruções de um programa</i> ".	45
3.8	Exemplo de atividade do tipo Parson Problems (PP), sobre o tópico " <i>Divisão inteira e resto da divisão inteira</i> ".	46
3.9	Perfil do aluno, conjunto de características dos alunos.	55
3.10	Resultados do inquérito de opinião sobre a utilização da aplicação Help To Programming (HTProgramming) e sua contribuição na aprendizagem dos alunos. (1) - Sem importância; (2) - Muito pouco importante; (3) - Pouco importante; (4) - Importante; (5) - Muito importante.	58
3.11	Matriz de confusão - Fase de treino e avaliação, dividida por fases de aprendizagem. . .	61
3.12	Matriz de confusão - Aplicação de rede gerada para resultados do ano letivo 2022/2023. .	61

3.13 Resultados da avaliação, dos últimos 6 anos, da unidade curricular de introdução à programação, do curso de engenharia informática, do IPG.	62
3.14 Total de alunos que usaram HTProgramming nos anos letivos 2020, 2021, 2022. Número e percentagem de aprovados que usaram HTProgramming.	62
4.1 Presenças às aulas, anos letivos de 2019 e 2022.	72
4.2 Identificação de linguagem de programação, por artigo, no ensino e aprendizagem da programação (pesquisa Association for Computing Machinery (ACM) Digital Library) . . .	73
4.3 Linguagem de programação em cursos universitários em Portugal.	74
4.4 Linguagem de programação em cursos politécnicos em Portugal.	75

Acrónimos

ACM	Association for Computing Machinery
BASIC	Beginner's All-purpose Symbolic Instruction Code
CT	Computational Thinking
DSRM	Design Science Research Methodology
GCC	GNU Compiler Collection
HTProgramming	Help To Programming
IDE	Integrated Development Environment
IPG	Instituto Politécnico da Guarda
ML	Machine Learning
MOODLE	Modular Object-Oriented Dynamic Learning Environment
MBP	Multiple Back-Propagation
MCQ	Multiple Choice Question
MFF	Multiple Feed-Forward
PALOP	Países Africanos de Língua Oficial Portuguesa
PP	Parson Problems
PreSS	Predict Student Success
SRL	Self-Regulation of Learning
SVM	Support Vector Machine

1

Introdução

Conteúdo

1.1 Problema	3
1.2 Objetivos	9
1.3 Metodologia	12
1.4 Contexto, estrutura e organização	13

As competências em programação são fundamentais para todos os alunos. O conhecimento básico de informática, não é suficiente para qualquer aluno que queira ser competitivo no mercado de trabalho atual [3]. Tornou-se essencial que os alunos desenvolvam conhecimentos mais profundos em informática. O desenvolvimento de competências como a criatividade, a resolução de problemas, a persistência, a colaboração, a comunicação e a capacidade de pensamento crítico, são cruciais para o sucesso de um aluno, face a uma tecnologia em constante evolução [4,5]. Os cursos de programação são geralmente considerados eficazes para promover e melhorar essas competências. Em consequência, muitos dos cursos do ensino superior incluem no seu plano curricular a disciplina de introdução à programação. Muitos alunos universitários estão a aprender programação, mas, infelizmente, têm dificuldades. Estas dificuldades e problemas têm um impacto mais significativo quando a programação é uma das áreas fundamentais do curso. Como resultado verificam-se altas taxas de insucesso, de desmotivação e de abandono dos cursos.

O ensino e aprendizagem inicial da programação é uma tarefa difícil que requer muito trabalho, dedicação e prática. Com o objetivo principal de ensinar e aprender os princípios básicos da programação, nos últimos anos, tem existido uma intensa atividade de investigação sobre as dificuldades no ensino e nos problemas dos alunos em adquirir essas competências. São apontadas várias razões e causas para o problema, sugeridas diversas estratégias, metodologias e técnicas para ajudar a minimizar. No entanto, na maioria dos casos, os problemas persistem e a falta de empenho, de motivação e de trabalho conduzem ao abandono e insucesso na aquisição de competências na aprendizagem inicial da programação.

Sob o tema geral das dificuldades no ensino e aprendizagem inicial da programação apresentamos neste capítulo o problema, identificamos algumas das razões que o fundamentam, caracterizamos o nosso grupo de estudo e elaboramos as nossas questões de investigação. Definimos o objetivo a alcançar e a metodologia a aplicar neste trabalho. Por fim, apresentamos a estrutura da dissertação.

1.1 Problema

Ao longo dos anos dedicados ao ensino inicial da programação, no ensino superior, temos observado e experimentado várias abordagens e métodos. E, quase invariavelmente o resultado é idêntico, altas taxas de insucesso, de abandono e de desmotivação. Reduzir significativamente estes resultados é a nossa principal motivação e razão de desenvolvimento desta investigação. O ensino inicial da programação é o objetivo da unidade curricular de introdução à programação, do primeiro ano, do curso de engenharia informática, da Escola Superior de Tecnologia e Gestão, do Instituto Politécnico da Guarda (IPG). Com esta unidade curricular, esperamos que os alunos adquiram as competências fundamentais para a construção de programas de computador, capazes de resolver problemas. Não

alcançando este objetivo, interpretado pelas altas taxas de insucesso e de abandono. Identificamos, claramente, o nosso problema: ***os alunos não adquirem as competências básicas em programação para a construção de programas de computador.***

Este problema é a consequência de um conjunto de outros fatores, como: a fraca motivação dos alunos para o curso; a fraca formação geral, em especial na matemática; a ausência às aulas; os problemas na construção e aplicação das diferentes estruturas da linguagem de programação; os problemas na capacidade de resolução de problemas; e, ainda, os problemas na transformação da estratégia de resolução num programa de computador [4, 6–8].

Os desafios inerentes ao ensino e aprendizagem inicial da programação é um tema frequente desde o aparecimento das linguagens de programação [8–12]. No exigente e competitivo mercado de trabalho atual, é fundamental que os alunos adquiram e desenvolvam competências transversais, como a criatividade, a resolução de problemas, a persistência, a colaboração, a comunicação e o pensamento crítico. A programação é vista como uma excelente disciplina para o desenvolvimento dessas características e competências [13–16]. Esta é uma das razões pelas quais muitos cursos na atualidade, nas mais diversas áreas, incluem conteúdos de pensamento computacional e codificação [17].

No entanto, a programação tem uma conotação negativa. O ensino e aprendizagem da programação é difícil. O ensino e aprendizagem inicial da programação é considerada como um dos grandes desafios na área do ensino da computação. Existem muitos trabalhos que descrevem e argumentam essas dificuldades, como por exemplo os trabalhos desenvolvidos em [7, 18, 19]. Nestes trabalhos, é claramente identificado e do conhecimento de toda comunidade científica, em especial na área das ciências da computação, que os alunos têm grandes dificuldades na programação.

Com o objetivo de investigar a razão de todas estas dificuldades, importa analisar o que é a programação. A Programação é um processo de transformação de um plano de termos correntes para termos compatíveis com um computador [20]. Ou seja, a programação é um processo de criação de um conjunto organizado de instruções que permitem que um computador execute uma tarefa. Mas para a criação e concretização deste processo, é necessário a elaboração um plano que inclui um conjunto de passos de execução que permitirão alcançar o objetivo pretendido. No primeiro passo é necessário que o aluno construa um plano, uma estratégia que permita resolver uma tarefa [6]. No segundo passo, o aluno deve conhecer e saber aplicar a sintaxe e a semântica de uma linguagem de programação. Depois, é necessário que essa estratégia de resolução esteja de acordo com os recursos disponíveis pela linguagem de programação. Por tudo isto, o ensino inicial da programação deve ter como principal objetivo que os alunos adquiram as competências necessárias na criação de estratégias para a resolução de problemas, e ainda, no conhecimento da sintaxe e semântica de uma linguagem de programação, tudo num curto espaço de tempo [7, 21, 22]. Atendendo a estes factos, é difícil encontrar um método de ensino eficaz e adequado a todos os alunos. Por conseguinte, é importante que o professor seja

experiente, conhecedor de um conjunto de estratégias e exemplos com bons resultados, para que todos consigam alcançar os seus objetivos.

De um modo geral, a literatura na área descreve casos de problemas na aprendizagem inicial da programação por todo o mundo, independentemente da região, da área do conhecimento ou da linguagem de programação. Os problemas e dificuldades no ensino e aprendizagem da programação são um fenómeno universal.

Desde que surgiram as primeiras linguagens de programação que este problema tem sido alvo de vários estudos e reflexões. São vários os grupos de trabalhos que se constituíram desde então com o sentido de analisar aspetos gerais do ensino e aprendizagem da programação [23, 24]. Outros, concentraram a sua ação em aspetos mais específicos das dificuldades do ensino e aprendizagem da programação [5, 7, 25–27]. No entanto, em todos eles existe a constatação das dificuldades no ensino e aprendizagem da programação independentemente da sua localização geográfica ou curso.

Na secção seguinte, analisamos as características muito particulares do nosso grupo de estudo, talvez o fator mais importante de todo o problema do ensino e aprendizagem inicial da programação.

1.1.1 Caraterização do grupo de estudo

O nosso grupo de estudo envolve alunos com características muito particulares. Os alunos frequentam a unidade curricular de introdução à programação, do primeiro ano, do curso de engenharia informática, do IPG. Os objetivos da unidade curricular visam a aquisição e aplicação dos conceitos básicos da programação. É usada a linguagem de programação C.

O curso de engenharia informática, do IPG, na generalidade, não é a primeira opção de escolha dos alunos na candidatura ao ensino superior, como pode ser comprovado pela tabela 1.1. Isto pode revelar alguma falta de motivação pelo curso. A média de candidatura dos alunos situa-se entre os valores de 11.45 e 13.48, já o valor da média a matemática posiciona-se entre os valores de 10.62 e 11.96. Valores estes reveladores de alguns problemas na formação geral dos alunos que constituem o nosso grupo de estudo, em especial na matemática. Nos últimos anos, temos recebido alunos oriundos de Países Africanos de Língua Oficial Portuguesa (PALOP) que ingressam no ensino superior a partir de acordos de cooperação, com graves problemas na sua formação geral.

Tabela 1.1: Notas de acesso e número da opção ao ensino superior dos alunos do curso de engenharia informática do IPG, nos últimos 5 anos.

Ano letivo	Número de alunos colocados	Média de candidatura	Média em matemática	1ª Opção	2ª Opção	3ª Opção
2018	20	11.45	10.62	8 (40%)	5	7
2019	20	12.28	11.02	7 (35%)	3	10
2020	36	12.64	11.72	6 (17%)	6	24
2021	11	13.48	11.96	7 (64%)	0	4
2022	45	13.37	11.12	13 (29%)	18	14

Apresentamos na tabela 1.2, o número de alunos inscritos na unidade curricular de introdução à programação por ano letivo, nos últimos cinco anos. O total de alunos inscritos é calculado pela soma do valor do número de alunos do concurso nacional de acesso ao ensino superior e os alunos que têm acesso por concurso especial, e, ainda, o número de alunos repetentes. No geral, podemos constatar que o número de alunos que ingressam no curso pela via do concurso nacional de acesso ao ensino superior é inferior a 30%, com a exceção do último ano letivo que foi de 45%. Verificamos, também, que na sua maioria, os alunos que frequentam a unidade curricular de introdução à programação, são alunos que entraram no curso pela via dos concursos especiais e os alunos repetentes. O número de alunos matriculados à unidade curricular de introdução à programação, é em média de 96 alunos. No entanto, unicamente 50% a 60% dos alunos frequentam assiduamente as aulas.

Tabela 1.2: Total de alunos inscritos na unidade curricular de introdução à programação e tipo de acesso, nos últimos 5 anos.

Ano letivo	Total de alunos inscritos	Número de alunos via concurso nacional	Número de alunos via concursos especiais	Alunos repetentes
2018	85	20 (24%)	14 (16%)	51 (60%)
2019	119	20 (17%)	50 (42%)	49 (41%)
2020	112	36 (32%)	27 (24%)	49 (44%)
2021	64	11 (17%)	30 (47%)	23 (36%)
2022	99	45 (45%)	30 (30%)	24 (24%)

1.1.2 Causas para o problema

Quais os motivos que provocam todos os estudos e preocupações sobre o tema do ensino e aprendizagem da programação? Para responder a esta pergunta, é importante analisar, um pouco mais em detalhe, o que é a programação. Como referido, podemos definir a programação como um processo de criação de um conjunto organizado de instruções que permite a resolução de problemas através da sua execução por um computador. Tendo como foco principal o ensino e aprendizagem inicial da programação os professores devem conseguir transmitir os conhecimentos necessários para que os alunos adquiram as competências essenciais à concretização deste processo. Numa primeira abordagem, é importante que o aluno interprete bem o problema, identifique claramente os dados fornecidos e qual o resultado pretendido. De seguida, os alunos devem aplicar um conjunto de estratégias que permitam resolver um problema [6]. Além disso, os alunos devem conhecer e saber aplicar a sintaxe e semântica de uma linguagem de programação. Depois, é necessário coordenar a estratégia de resolução com os recursos disponíveis pela linguagem de programação. Por fim, é necessário verificar e analisar os resultados e, eventualmente, corrigir erros de sintaxe ou semântica, ou rever todo o processo. Com este conjunto complexo de tarefas é difícil encontrar um método de ensino eficaz e adequado a todos os alunos [28]. Por outro lado, muitos dos alunos não estão motivados ou dispostos a fazer todo este esforço cognitivo.

Neste processo de criação de um programa de computador, é necessário a aprendizagem de uma nova linguagem. Esta linguagem é diferente das linguagens escritas e faladas comuns. É uma linguagem de programação, caracterizada por um conjunto de conceitos abstratos e com regras muito específicas. A linguagem de programação é normalmente usada com o recurso a um ambiente integrado de desenvolvimento, Integrated Development Environment (IDE). Na maioria das vezes este ambiente de desenvolvimento é uma ferramenta útil para o programador. Esta aplicação inclui no seu ambiente recursos sobre a linguagem de programação e compilador e/ou interpretador. Oferece também informação adicional sobre erros de código e localização dos mesmos. No entanto, por vezes este ambiente funciona mais como um obstáculo à aprendizagem inicial da programação, do que propriamente uma ferramenta de ajuda.

Uma das melhores competências para todos, nos dias de hoje e para o futuro, é a capacidade de resolução de problemas e o pensamento computacional (*Computational Thinking* (CT)) [29, 30]. O pensamento computacional, ou a falta de capacidade de aplicar e utilizar o pensamento computacional, é um dos temas mais abordado e identificado como uma das dificuldades na aprendizagem da programação. O Computational Thinking (CT) é um dos caminhos para o desenvolvimento da capacidade de resolução de problemas. Podemos definir o CT como um conjunto de competências para a resolução de problemas baseadas em técnicas de computação [31]. Para muitos problemas, nas mais diversas áreas, é uma boa estratégia fazer um plano para a sua resolução baseado em algumas técnicas de computação, como: decomposição de um problema complexo em partes mais simples; procurar semelhanças em outras resoluções e experiências; foco no que é realmente importante e eliminação de partes específicas para uma solução mais abrangente; e ainda, o desenvolvimento de uma solução passo-a-passo. Ou seja, os quatros principais fatores do pensamento computacional: a decomposição, o reconhecimento de padrões, a abstração e a construção de algoritmos. Em muitos dos trabalhos realizados, constatamos que os alunos nunca tiveram a oportunidade de praticar ou utilizar estratégias e conhecimento de resolução de problemas, ou seja, nunca tiveram a oportunidade de desenvolver as suas capacidades em pensamento computacional.

A aprendizagem inicial da programação é normalmente uma unidade curricular do primeiro ano, de um curso do ensino superior. Para a maioria dos alunos, o primeiro ano é um período difícil. É a adaptação a um novo sistema de ensino, é viver longe da família, é fazer novos amigos, é todo um novo ambiente que é necessário adaptar-se e gerir. A programação adquiriu a reputação de que é difícil [21]. Esta visão é passada dos alunos mais velhos para os alunos mais novos. Por outro lado, existe uma imagem social do "programador" como sendo um indivíduo socialmente alienado, vivendo detrás de um computador e alimentando-se a pizza e cafeína. Não é uma imagem muito agradável para servir de referência ou de motivação. E, alunos que não estão motivados não terão sucesso.

Em síntese, identificamos como as principais causas para o nosso problema a falta de capacidade

dos alunos na resolução de problemas. Neste ponto, constatamos que a maioria dos alunos nunca ouviu falar em pensamento computacional, ou nunca praticou técnicas de resolução de problemas. Outro dos problemas identificados mais significativo, é a transformação da estratégia de resolução em código, de acordo com as regras sintaxe e semântica de uma linguagem de programação. Para além destes problemas, também identificados pelos mais diversos estudos [7, 8, 19, 32, 33], devemos incluir fatores específicos do nosso grupo de estudo, como a fraca formação geral e falta de motivação para o curso ou instituição de ensino e o abandono, identificado pela ausência às aulas e avaliação. Acrescente-se que, a aprendizagem da programação exige um considerável nível de esforço e de trabalho árduo. Muitos dos alunos não estão dispostos ou motivados a fazer esse esforço. Pelo que todo o processo de ensino e aprendizagem fica comprometido.

1.1.3 Questões de investigação

Depois de identificar o problema, de caracterizar nosso grupo de estudo e de caracterizar algumas das possíveis causas para o problema, devemos elaborar as nossas questões de investigação que possibilitarão a definição dos nossos objetivos.

Como descrito anteriormente, a programação é um processo de criação de um conjunto organizado de instruções que permitem que um computador execute uma tarefa. Mas, para que este processo seja concretizado, é necessária uma estratégia que a resolva, conhecer e aplicar a sintaxe e semântica de uma linguagem de programação. Por fim, é necessário que a estratégia de resolução esteja de acordo com os recursos disponíveis pela linguagem. Como referido, este processo é de difícil concretização para um grande número de alunos. Com este problema em mente, podemos formular a nossa primeira questão de investigação.

Q11 - Que conjunto de atividades devem ser usadas para melhorar as competências na aprendizagem inicial da programação?

Com esta questão, pretendemos investigar que conjunto de atividades devem ser usadas no nosso processo de ensino e aprendizagem inicial da programação, de modo a minimizar os nossos problemas e melhorar significativamente as competências básicas dos alunos na resolução de problemas e na construção de programas de computador. Tratando-se da aprendizagem inicial da programação, o tipo de atividades a desenvolver durante o processo de ensino é de especial importância. Por um lado, devemos adotar medidas que permitam motivar e cativar o aluno para a aprendizagem da programação e, o tipo de atividades da preferência dos alunos é fulcral. Por outro lado, devemos tentar aliar as preferências dos alunos com modelos e técnicas clássicas de computação para a interiorização e aquisição de conceitos e práticas de resolução de problemas e na construção dos respetivos programas de computador.

A questão de investigação seguinte, está diretamente relacionada com a identificação das dificuldades e padrão de comportamento que conduzem os alunos ao insucesso. A pesquisa na capacidade de previsão do desempenho do aluno, é um tema que tem sido pesquisado desde que surgiram as linguagens de programação, e investigado mais intensamente com o uso de técnicas e modelos de previsão baseados em aprendizagem automática, em inglês *Machine Learning* (ML) [34–36].

Q12 - Como podemos prever o insucesso dos alunos?

Com esta questão pretendemos investigar sobre quais as características que mais contribuem para o insucesso do aluno. Características essas identificadas a partir do conjunto de ações e atividades desenvolvidas pelos alunos no processo de ensino e aprendizagem da programação. Com esta informação pretendemos detetar problemas precocemente e agir rapidamente no diagnóstico e minimização dos problemas.

O ensino e aprendizagem da programação envolve um grande volume de informação num curto espaço de tempo. Por outro lado, os alunos têm diferentes ritmos de aprendizagem, de conhecimento e motivação. Para gerir o processo é necessária uma ferramenta tecnológica que permita responder às necessidades dos alunos e professor. A questão de investigação seguinte, permitirá avaliar a necessidade e importância de uma ferramenta tecnológica no processo de ensino e aprendizagem inicial da programação.

Q13 - Que contribuição a utilização de uma ferramenta tecnológica poderá trazer para o processo de ensino e aprendizagem inicial da programação?

A proposta de uma ferramenta tecnológica de auxílio ao processo de ensino e aprendizagem inicial da programação parece evidente, em face das questões de investigação propostas e problemas enunciados. No entanto, com esta questão, ambicionamos avaliar a importância e necessidade da sua utilização na gestão de todo o processo de ensino e aprendizagem. E, ainda, determinar o valor da sua utilização no resultado dos alunos.

Formuladas as questões de investigação, na secção seguinte, definimos o nosso objetivo geral e objetivos específicos que permitirão desenvolver esta investigação, em busca de conhecimento necessário à sua concretização.

1.2 Objetivos

O objetivo geral de qualquer projeto será a definição do que se pretende atingir. O objetivo geral deve demonstrar a mudança na realidade que se pretende atingir, ou seja, visa dar resposta ao principal problema apontado. De acordo com o enquadramento descrito neste trabalho, e especificidade do

mesmo, o principal objetivo do nosso trabalho é **que os alunos adquiram as competências básicas em programação para a construção de programas de computador.**

Identificado o problema principal, apontadas as causas para o problema e definidas as questões de investigação, devemos explicitar os detalhes ou etapas do trabalho para alcançar o objetivo geral. Estas etapas do trabalho ou objetivos específicos devem corresponder a resultados intermédios que devemos atingir para alcançar o objetivo geral.

Formulada a questão de investigação *Q11 - Que conjunto de atividades devem ser usadas para melhorar as competências na aprendizagem inicial da programação?*, pretendemos investigar sobre um conjunto de objetivos específicos, que atividades devem ser usadas no nosso processo de ensino e aprendizagem inicial da programação, de modo a minimizar os nossos problemas e melhorar significativamente as competências básicas dos alunos na construção e resolução de problemas de computador.

No desenvolvimento desta investigação, é importante que consigamos de alguma forma dar resposta, ou de alguma forma compensar a fraca motivação dos alunos, a sua fraca formação geral, nomeadamente em matemática, a ausência às aulas, as dificuldades na resolução de problemas e, conseqüente, a transformação da sua estratégia de resolução num programa de computador. Neste sentido, definimos como objetivo a utilização de atividades que permitam praticar, de uma forma lúdica, conceitos mais sérios das ciências da computação, que possibilitarão aumentar as capacidades cognitivas dos alunos na resolução de problemas.

Ao longo dos anos dedicados ao ensino da programação, temos menosprezando os conceitos iniciais da linguagem de programação, como identificadores e variáveis, tipos de dados, formatação de dados input e output, instruções e expressões aritméticas simples, expressões booleanas, blocos de instruções, entre outras. Por outro lado, valorizamos o saber fazer, resolver e codificar pequenos problemas. No entanto, constatamos que muitos dos problemas existentes na codificação são derivados de pequenos erros e dificuldades na aplicação desses conceitos iniciais, específicos de uma linguagem de programação. Com este pressuposto, é nosso objetivo implementar um conjunto de atividades, que permitam ao aluno praticar e consolidar os seus conhecimentos nesta fase inicial de aprendizagem de uma linguagem de programação. Para que o aluno se sinta motivado e desenvolva as atividades ao seu ritmo é importante que as atividades tenham *feedback* imediato. Desta forma, o aluno pode avaliar a evolução da sua aprendizagem e constatar que as suas ações têm uma respostas, mantendo a sua confiança e motivação na resolução de problemas. Pelo exposto, definimos como objetivo a implementação de um sistema de *feedback* imediato em todas as atividades a desenvolver.

O ensino e aprendizagem inicial da programação exige de todos os envolvidos no processo muito trabalho e dedicação. Ao professor exige uma busca constante dos melhores métodos e dos melhores exemplos de atividades para apresentar aos alunos. No entanto, constatamos todos os anos que o grupo de alunos é bastante heterogéneo, com ritmos de aprendizagem, níveis de conhecimento, com-

portamento e de motivação bastante diferentes. Pelo que, é um desafio constante e bastante exigente para o professor. Igualmente para o aluno é um trabalho exigente, que requer muita dedicação, empenho e trabalho, muito trabalho. Como tal, é importante uma grande variedade de atividades orientadas, com *feedback* imediato, para que o aluno possa trabalhar e praticar a utilização dos diferentes conceitos e técnicas de uma linguagem de programação. Mediante o exposto, é importante definir como objetivo a realização de um conjunto diversificado e organizado de atividades, para que o aluno possa praticar e consolidar os seus conhecimentos. Do conjunto de atividades apresentadas importa, também, analisar e avaliar as atividades de maior interesse e preferência dos alunos, mas também as que mais contribuem para a aprendizagem e a consolidação dos conceitos essenciais da aprendizagem.

Na formulação da questão de investigação *Q12 - Como podemos prever o insucesso dos alunos?*, é nosso propósito investigar sobre a existência de um padrão ou comportamento do aluno que permita prever o seu insucesso. A previsão do insucesso do aluno seria uma ajuda relevante, pois permitiria agir e trabalhar as dificuldades do aluno de forma a contrariar essa previsão. Por conseguinte, estabelecemos como objetivo a investigação de que variáveis devemos analisar e incluir, e se será factível um modelo de previsão do insucesso do aluno.

Por último, de acordo com a questão de investigação *Q13 - Que contribuição a utilização de uma ferramenta tecnológica poderá trazer para o processo de ensino e aprendizagem inicial da programação?* enunciada, parece inevitável a utilização de uma ferramenta tecnológica que permita gerir todo o processo descrito. A proposta de atividades ao aluno com *feedback* imediato, não será possível sem uma ferramenta automática que permita gerir este processo. Por outro lado, o modelo de previsão não será executável sem esta ferramenta automática para o tratamento e análise de dados. Além de que, de outra forma o acompanhamento do professor da atividade e evolução de cada aluno seria impraticável ou quase impossível, mesmo para o nosso reduzido número de alunos. Consequentemente, o objetivo de construção de uma aplicação para auxiliar alunos e professor no processo de ensino e aprendizagem é evidente e essencial.

Com a definição do objetivo geral e objetivos específicos estabelecemos as etapas no desenvolvimento deste trabalho. Com este planeamento esperamos ser capazes de sugerir uma estratégia de ensino e aprendizagem, baseada num conjunto de atividades e com ajuda tecnológica, que permitirá auxiliar alunos e professores neste difícil e complexo processo de ensino e aprendizagem inicial da programação, e ainda, eliminar ou reduzir o problema. Esperamos, também, que este trabalho seja uma boa contribuição para todos aqueles que se identificam com a área e enfrentam dificuldades semelhantes.

1.3 Metodologia

Os trabalhos desenvolvidos nesta área do conhecimento, normalmente, seguem um padrão de investigação que consiste na definição do problema, na revisão da literatura, no desenvolvimento de hipóteses de investigação, na recolha e análise de dados e na discussão de resultados. Neste trabalho, optamos por aplicar *Design Science Research Methodology (DSRM)*, metodologia de investigação científica com desenvolvimento de artefacto [37–39]. Atendendo às características do nosso trabalho e grupo de estudo consideramos ser a melhor opção para o seu desenvolvimento e concretização dos nossos objetivos.

Nos últimos anos temos assistido a um crescente interesse pela metodologia de investigação de DSRM. Esta metodologia de investigação tem como foco principal a criação de artefactos de utilização prática [38, 39]. Alguns dos trabalhos desenvolvidos sugerem que o artefacto deve resolver um problema importante [40]. Outros trabalhos propõem que a investigação em DSRM deve abordar um problema de uma forma única e inovadora, ou de uma forma mais efetiva [41]. No trabalho desenvolvido em [37], os seus autores propõem que DSRM deve incluir seis etapas: identificação e motivação do problema, definição dos objetivos para uma solução, design e desenvolvimento, demonstração, avaliação e comunicação.

Na primeira etapa, identificação do problema e motivação, deve ser definido o problema e justificar a importância de uma solução. Enquadrar a importância do problema e identificar pontos fracos das soluções atuais. Na etapa seguinte, ou passo 2, consiste em definir os objetivos de uma solução. Devem ser definidos objetivos gerais como a viabilidade e desempenho, e objetivos específicos que a solução para o problema deve incluir. Devem, ainda, ser considerados os métodos, as teorias e as tecnologias que podem ajudar na definição dos objetivos. O passo 3 consiste na criação de um artefacto. Ou seja, construir um modelo, aplicar uma teoria, um método ou tecnologia que resolva o problema. No passo 4, demonstração, consiste em provar que o artefacto funciona, com a resolução de uma ou mais instâncias do problema. A etapa 5 consiste na avaliação. Observar e medir os resultados da aplicação do artefacto na solução do problema. Devemos avaliar os resultados e, se necessário, repetir o passo 2 e/ou o passo 3. O passo 6, e último, consiste na comunicação. Comunicar e apresentar o problema, a sua solução e utilidade. Divulgar os resultados e eficácia da solução para a comunidade. Neste último passo devemos também avaliar a solução a sua utilidade e se necessário repetir o passo 2 e/ou 3.

Esta metodologia de DSRM é uma boa opção a adotar no desenvolvimento do nosso trabalho. No nosso estudo, normalmente existem dois pequenos grupos de alunos a iniciar a aprendizagem da programação. Não nos parece adequado a utilização de grupo de teste e grupo de controlo. Não queremos adotar uma estratégia em que apenas o grupo experimental utiliza conceitos, métodos e meios tecnológicos para ajudar o aluno na motivação e aquisição de conhecimentos.

1.4 Contexto, estrutura e organização

Este trabalho de investigação foi realizado no âmbito do programa de doutoramento em Engenharia Informática, da Universidade de Salamanca. Formalmente, com o plano de estudos: D015 - Ingeniería Informática (R.D. 99/2011), na Escuela de Doctorado “Studii Salamantini”, na Universidad de Salamanca, com início no ano letivo de 2016 – 2017. O trabalho desenvolvido teve a participação e colaboração do grupo de investigação GRupo de Investigación en InterAcción y eLearning (GRIAL). O estudo e aplicação prática foi realizada na Escola Superior de Tecnologia e Gestão, do Instituto Politécnico da Guarda, no âmbito da unidade curricular de Introdução à Programação, do 1º ano, do curso de Engenharia Informática.

A experiência profissional com mais de 32 anos de docência, sempre como professor da unidade curricular de Introdução à Programação, tornam relevante o interesse por este tema, e simultaneamente de pertinência científica, no desenvolvimento de estudos para melhorar o processo de ensino e aprendizagem inicial da programação.

Este trabalho foi dividido de modo a estabelecer uma ligação coerente entre as diferentes etapas de um processo de investigação. Deste modo, o presente documento encontra-se estruturado em cinco capítulos. O primeiro, *Introdução*, tem como objetivo, contextualizar a área em que se desenvolveu o trabalho, identificar o problema e formular as questões de investigação, definir objetivos gerais e específicos e apresentar a metodologia a utilizar neste trabalho. O capítulo 2, *Estado da arte* apresenta o estado da arte na área do ensino e aprendizagem inicial da programação. Apresentamos uma revisão da literatura pertinente, procuramos identificar os temas mais abordados, quais os assuntos identificados como problemas, quais as soluções propostas, as metodologias de ensino e aprendizagem utilizados, e ainda, a identificação de trabalhos caracterizados como boas práticas na área. No capítulo 3, *Aplicação Design Science Research Methodology na Introdução à Programação*, apresentamos DSRM como metodologia de investigação e descrevemos pormenorizadamente as seis atividades que a constituem: a identificação e motivação do problema; a definição da solução e objetivos; a criação do artefacto; a demonstração; a avaliação; a comunicação e apresentação de resultados. No capítulo 4, *Discussão*, apresentamos a análise sobre os resultados obtidos e suas implicações. Por fim, o último capítulo, *Conclusão*, inclui uma síntese das principais ideias desenvolvidas e trabalho futuro. São efetuadas as conclusões gerais e considerações sobre o trabalho apresentado, bem como apresentadas sugestões de trabalho futuro.

2

Estado da arte

Conteúdo

2.1	Pensamento Computacional	18
2.2	Conceções erradas e dificuldades	20
2.3	Aptidão para a programação	21
2.4	Aprendizagem autoregulada	22
2.5	Fatores que promovem o sucesso	23
2.6	Competências na resolução de problemas	24
2.7	Live-Coding no ensino de introdução à programação	25
2.8	Erros de codificação	26
2.9	Parson Problems	28
2.10	Gamificação	29
2.11	Conclusão	30

O ensino inicial da programação tem como propósito principal habilitar os alunos com os conhecimentos necessários, para a construção de programas de computador capazes de resolver problemas. Este processo requer a aquisição de um conjunto grande de conhecimentos num curto espaço de tempo. O presente capítulo apresenta o estado da arte na área do ensino e aprendizagem inicial da programação. Por meio de uma revisão da literatura, procuramos identificar os temas mais abordados, quais os assuntos identificados como problemas, quais as soluções propostas, as metodologias de ensino e aprendizagem utilizados, e ainda, a identificação de trabalhos caracterizados como boas práticas na área.

A ação de transmitir conhecimentos e competências na área da programação é uma área relativamente recente, quando comparada com outras áreas do conhecimento, como por exemplo a área da matemática ou física. O ensino da programação requer diferentes metodologias.

Nenhum método ou ferramenta terá qualquer efeito sem motivação e muito trabalho de todos os envolvidos no processo de ensino e aprendizagem da programação. No entanto, a falta de empenho, de dedicação, de persistência, de esforço e de trabalho, muito trabalho, por parte do aluno pode comprometer o sucesso no curso. Métodos e ferramentas são apenas meios que podem auxiliar e incentivar o processo. Todos os grandes músicos ou atletas, o talento ajuda muito, mas é necessário trabalhar muito. É necessário praticar e trabalhar muito todos os dias. É necessário analisar e avaliar o feedback do nosso trabalho para poder melhorar.

Neste capítulo abordamos o estado da arte na área do ensino e aprendizagem da programação, analisamos alguns dos temas mais relevantes e dos quais consideramos mais pertinentes para uma possível aplicação no nosso trabalho. Abordamos dez temas divididos pelo mesmo número de secções. Iniciamos com a apresentação do pensamento computacional e na sua contribuição para o desenvolvimento de técnicas para a resolução de problemas. Na secção dois, desenvolvemos o tema da concepções erradas e dificuldades preconcebidas dos alunos na aprendizagem inicial da programação, seguida pela necessidade de identificar os alunos com a aptidão para a programação. Na quarta secção analisamos trabalhos na área da aprendizagem autoregulada. Na secção cinco, são analisados os principais fatores que promovem o sucesso dos alunos no estudo da programação. Na sexta secção descrevemos o trabalho realizado com o intuito de auxiliar os alunos na aquisição de competências na resolução de problemas. O Live-coding no ensino de introdução à programação é explorado na secção sete. Na secção oito são descritos os erros mais comuns praticados pelos principiantes na programação. Na secção nove, descremos a utilização e resultados da aplicação de atividades Parson Problems no ensino inicial da programação. Por fim, a última secção abordamos a gamificação e o seu importante papel no ensino e aprendizagem da programação.

2.1 Pensamento Computacional

Uma das melhores competências para todos é a capacidade de resolução de problemas. O pensamento computacional é um processo que ajuda a desenvolver essas competências [29]. O pensamento computacional, (CT em inglês), pode ser definido como um conjunto de competências para a resolução de problemas baseados em técnicas de computação [17, 42–45]. Para muitos problemas é boa ideia criar um plano com recurso a técnicas das ciências da computação, como: decompor um problema complexo em componentes mais simples e de resolução independente, a decomposição; procurar semelhanças entre problemas e de outras experiências, o reconhecimento de padrões; focar na informação que é realmente importante, e eliminar diferenças específicas de modo que uma solução inclua o maior número de problemas, a abstração; o desenvolvimento de uma solução que nos permita realizar uma tarefa através de uma sequência de passos simples, o algoritmo. O pensamento computacional ensina-nos a pensar, a encontrar formas de resolver um problema, a organizar e planejar a resolução de uma tarefa, e ensina-nos os métodos e as técnicas para resolver problemas complexos. Muitos dos alunos que chegam ao ensino universitário, nunca tiveram a oportunidade de trabalhar e desenvolver as suas competências no pensamento computacional. O dos motivos apontados é a falta de conhecimento e formação dos professores na área. Esta é provavelmente a razão pela qual, nos últimos anos, temos assistido à proliferação de projetos patrocinados por organizações governamentais e não governamentais, com o objetivo específico de incentivar o pensamento computacional e o estudo da programação, especialmente nos anos pré-universitários [43, 46, 47]. Muitas organizações trabalham arduamente para preparar os jovens para o sucesso no mundo digital. No trabalho apresentado em [43], foi feita uma pesquisa exaustiva e respetiva avaliação das ferramentas existentes para o ensino e aprendizagem da programação para estudos pré-universitários.

Existem diferentes abordagens que podem ser utilizadas para introduzir a prática do pensamento computacional em sala de aula [43]. Destacamos as que utilizam algum tipo de programação baseado em blocos, como o Scratch; as abordagens que usam a robótica e, ainda, os que utilizam conjuntos de kits ou objetos físicos que permitem a manipulação. A ideia principal destas abordagens ao pensamento computacional não é uma alternativa à aprendizagem da codificação, mas uma forma de introduzir e complementar conceitos no ensino da codificação. O objetivo é que os alunos possam desenvolver modelos mentais e competências que melhorem as suas competências na aprendizagem da programação [43].

Um bom exemplo da abordagem utilizada é o projeto CS Unplugged (<http://csunplugged.org/>), do CS Education Research Group da Universidade de Canterbury, na Nova Zelândia. Onde o principal objetivo é promover e estimular o estudo das ciências da computação para todos os jovens. Neste sentido, o projeto CS Unplugged, distingue-se dos demais projetos pelo seu conjunto de abordagens, como atividades que não necessitam a utilização de computadores, a utilização de conceitos fundamentais e

reais das ciências da computação, a interação e cooperação entre alunos fisicamente para a resolução de problemas, entre outras.

Outro exemplo é o Scratch, projeto do Grupo Lifelong Kindergarten do Massachusetts Institute of Technology (MIT) Media Lab, desenvolvido pela Scratch Foundation. O Scratch (<https://scratch.mit.edu/>) promove o pensamento computacional através de uma programação por blocos, como uma interface muito simples e visual. O Scratch é também uma comunidade online de programação para crianças, onde procura estimular a criatividade, o aprender a pensar, a raciocinar e a trabalhar de uma forma colaborativa.

A utilização da robótica em sala de aula, com o objetivo de estimular o desenvolvimento do pensamento computacional e programação, é outro dos excelentes exemplos de sucesso utilizado nos últimos anos e explorado em diversos projetos [48–51]. Em Portugal, a Direção-Geral da Educação iniciou em 2015, um conjunto de iniciativas para a implementação de atividades de programação e robótica no ensino básico. Esta iniciativa, atividades de “Programação e Robótica no Ensino Básico”, é dirigida aos alunos do 1º ao 9º ano de escolaridade, visa incentivar os diversos estabelecimentos de ensino a incluir estas atividades na sua oferta educativa [52].

Outras abordagens e estratégias, mas com o mesmo objetivo de estimular e incentivar o pensamento computacional, foram explorados com sucesso. Os trabalhos realizados em [53, 54], onde atividades do tipo *Seguir e Dar Instruções*, o *desenho de Mapas* e a *Dobragem de Papel*, são utilizadas com o objetivo de desenvolver capacidades cognitivas e de visualização espacial, fortemente associadas a características fundamentais para a programação, mas também com o objetivo de prever as competências cognitivas dos alunos para a programação.

Nesta secção, procuramos explorar e apresentar alguns trabalhos realizados com o objetivo de desenvolver e aumentar as capacidades dos alunos no pensamento computacional. No entanto, estas ferramentas ou atividades são mais indicadas para alunos da faixa etária dos 3 aos 18 anos, ou seja, para alunos em idade pré-universitária. Como referido anteriormente, a maioria dos alunos nunca teve a oportunidade de trabalhar e praticar as suas capacidades em atividades do pensamento computacional. Por outro lado, só muito recentemente que o ensino em Portugal adotou medidas para o estudo e desenvolvimento deste tipo de características. Talvez num futuro muito próximo, a maioria dos alunos que chegam ao ensino superior já tenham estas competências do pensamento computacional mais consistentes. No entanto, alguns dos trabalhos descritos, como [53, 54], sugerem a utilização de cursos de pré-programação ou em paralelo com as unidades curriculares de programação ou codificação.

2.2 Concepções erradas e dificuldades

As concepções erradas ou ideias preconcebidas dos alunos na aprendizagem inicial da programação referem-se a ideias erradas, a interpretações incorretas ou modelos mentais que os alunos desenvolvem na aprendizagem da programação. O estudo de concepções erradas tem início nos anos 80, em várias áreas da ciência, e introduzido na área do ensino e aprendizagem da programação pelos trabalhos realizados por [55], ou [56], entre outros.

Essas concepções erradas podem surgir de interpretações erradas de conceitos chave ou ideias preconcebidas de que os alunos trazem consigo e, que ao longo do tempo, construíram um modelo mental de ideias incorretas e que podem dificultar o progresso na aprendizagem inicial da programação.

No trabalho desenvolvido em [55], é descrito que um grupo de alunos aprenderam *Beginner's All-purpose Symbolic Instruction Code (BASIC)* através de um estudo autónomo e no final foram inquiridos sobre a sua interpretação para a execução de algumas instruções. O estudo mostrou que os iniciantes em programação criaram um conjunto de ideias erradas sobre as instruções que tinham aprendido. As principais concepções erradas verificadas neste trabalho referem-se à dificuldade em perceber onde e como ficam guardados os valores em memória e, como são usados e atualizados ao longo da execução de um programa.

Identificar e corrigir essas concepções é uma parte importante do processo de ensino e aprendizagem inicial da programação. O trabalho desenvolvido em [57] foram analisadas as várias fontes de concepções erradas e outras dificuldades dos alunos. Segundo os seus autores, a origem de muitos problemas, como as dificuldades na sintaxe da linguagem, os conhecimentos matemáticos, os modelos mentais incorretos e estratégias inadequadas, está relacionada com os conhecimentos existentes ou com a experiência anterior dos alunos. Ou seja, o conhecimento prévio dos alunos é uma das principais fontes das concepções erradas e de outras dificuldades na aprendizagem da programação. Para compreender as dificuldades dos alunos e suas causas, devemos prestar atenção aos seus conhecimentos prévios.

Alguns dos problemas estão relacionados com os conhecimentos dos professores. Embora o fraco conhecimento ou experiência dos professores possa levar a problemas, o elevado conhecimento também pode causar problemas, na medida em que o professor pode ter dificuldades em compreender as dificuldades dos alunos.

Ainda no trabalho desenvolvido por [57], os seus autores analisaram várias estratégias e ferramentas para lidar com as concepções erradas dos alunos e outras dificuldades. Apresentam, ainda, um conjunto de exemplos de dificuldades em diferentes tipos de conhecimentos de programação, os seus principais fatores relacionados e potenciais estratégias e ferramentas para os resolver. Relativamente aos conhecimentos sintáticos são apontadas como as principais dificuldades detetadas a colocação de parêntesis e a incorreta utilização do operador de comparação. Como principais fatores deste problema

são identificados o aumento de complexidade da tarefa. Com o aumento da complexidade, aumentam as necessidades cognitivas do aluno, fazendo esquecer a sintaxe básica, e ainda, o ambiente de programação usado não identificar os erros de sintaxe. Como estratégias ou ferramentas é sugerido a utilização de editores avançados que assinalam os erros de sintaxe, e ajudam na construção sintática das diferentes estruturas.

Relativamente à compreensão conceptual, as concepções erradas e dificuldades, são apontadas à compreensão das diferentes estruturas da programação e o seu funcionamento no computador, especialmente no que diz respeito às variáveis e atribuição, expressões condicionais, os ciclos e, ainda a execução sequencial do código. São apontados como principais fatores o conhecimento prévio do aluno na linguagem natural e os conhecimentos em matemática, que pode interferir com os conceitos na programação. Os conhecimentos dos professores e a sua formação, e alguns dos elementos da linguagem de programação, como o sinal de atribuição ou o sinal de adição e concatenação, podem contribuir para as dificuldades e concepções erradas dos alunos na aprendizagem da programação. Como ferramentas ou potenciais estratégias para a resolução destes problemas conceptuais são apontadas as boas escolhas de exemplos de programas, aplicações que permitam a visualização gráfica da execução dos programas, a análise do conhecimento prévio e das concepções erradas dos alunos e, ainda, a ajuda entre pares.

Por fim, ao nível do conhecimento estratégico, foram identificados dificuldades e concepções erradas relativamente à aplicação dos elementos de programação na resolução de problemas, nomeadamente a escolha do ciclo mais adequados e as dificuldades no teste e *debug* de um programa. Como causas para o problema é apontada o conhecimento fragmentado, e o conjunto de estratégias e padrões inadequados à resolução de problemas. A leitura e traçagem de programas exemplos e ensinar os alunos a fazer *debug*, são as estratégias sugeridas para a resolução destes problemas e dificuldades.

2.3 Aptidão para a programação

O interesse na identificação de pessoas com talento para a programação remonta ao início dos anos 60 [58]. Na altura, com o surgir da nova ocupação, apelidada de programação de computadores, era importante seleccionar pessoas com qualificações para essa atividade. Um pouco mais tarde surgem estudos sobre a identificação das características que parecem influenciar a aptidão para a programação. A utilização de testes como o *IBM Programmer Aptitude Test* e a avaliação dos conhecimentos em matemática, como no trabalho desenvolvido em [59]. Ou ainda, como referido no trabalho em [60], a análise de um conjunto de variáveis, como: perfis demográficos; resultados do ensino secundário; formação e experiência prévia em informática; estilos cognitivos e capacidade de resolução de problemas. De salientar ainda, neste último trabalho [60], o facto de nenhuma destas variáveis ser res-

ponsável por um aumento significativo no modelo de regressão linear, de R-quadrado, usado para prever a aptidão informática. Em vez disso, os resultados sugerem que vários fatores, como: demográficos, comportamentais, cognitivos ou de resolução de problemas; podem ser úteis na previsão da aptidão informática.

Nos últimos anos temos assistido a uma forte tendência dos trabalhos na área do ensino e aprendizagem inicial da programação, na pesquisa da capacidade de prever o desempenho do aluno. Ao identificar essas competências e aptidão cria oportunidades para os seus resultados de aprendizagem [34, 61]. A pesquisa na previsão do desempenho do aluno procura identificar os recursos que podem ser usados para fazer essas previsões e os algoritmos que podem melhorar e quantificar aspectos de desempenho do aluno. Procura, também, determinar as características inter-relacionadas e identificar as razões subjacentes pelas quais umas características funcionam melhor do que outras.

Ser capaz de identificar os fatores que influenciam, ou podem influenciar, a aptidão para a programação é de especial importância para a melhoria da conceção dos cursos, nos recursos a utilizar e no apoio a dar aos alunos. E, dessa forma, minimizar os problemas e dificuldades do ensino e aprendizagem inicial da programação.

2.4 Aprendizagem autoregulada

Estudos recentes na área da autorregulação da aprendizagem, em inglês Self-Regulation of Learning (SRL), aplicado ao ensino e aprendizagem da programação [62], mostram que os alunos que fazem uso de estratégias cognitivas capazes de reconhecer suas dificuldades e procurar formas de superá-las tendem a ter mais sucesso academicamente [62, 63]. Na atualidade não basta aprender os conteúdos formais, é importante ser crítico, agir com autonomia e, principalmente, saber gerir a grande quantidade de informação disponível. Pelo que, é importante que o processo de ensino e aprendizagem promova a autorregulação do aluno para aprender [64].

A aprendizagem regulada pelo próprio aluno resulta da sua capacidade de monitorização, controlo e avaliação dos seus próprios comportamentos e atividades cognitivas [65]. Os professores devem ser capazes de estimular os seus alunos para uma utilização mais competente, eficaz e motivada dos processos de aprendizagem e dos meios tecnológicos e culturais a que tem acesso [65].

Segundo as autoras do trabalho desenvolvido em [65], a autorregulação da aprendizagem desenvolve-se em quatro fases. A primeira fase consiste na antevisão e definição de objetivos. A segunda fase, diz respeito à elaboração de um plano estratégico, ou seja, na capacidade do aluno em organizar estratégias e métodos para atingir os seus objetivos. Nesta fase, o aluno deve recolher e organizar a informação necessária, compreender a exigência da tarefa e selecionar as estratégias adequadas para a execução da tarefa. Durante a terceira fase, a de monitorização e execução, o aluno coloca em

ação as estratégias para a concretização do plano delineado. Nesta fase o aluno deve ter consciência das ações que está a realizar e dos resultados obtidos. Ou seja, as ações de testar-se, verificar e rever assumem um papel importante. Por último, a fase de autorreflexão e auto-reação. Nesta fase o aluno deve fazer uma autoavaliação sobre os resultados alcançados. Essa avaliação pressupõe uma comparação com os seus pares ou com um padrão por forma a medir a sua eficácia e refletir sobre as causas do sucesso ou fracasso.

2.5 Fatores que promovem o sucesso

São vários os estudos que procuram determinar quais os fatores que mais contribuem para o sucesso dos alunos, em especial nos cursos de programação. Um desses estudos é o apresentado em [66], onde os seus autores elaboraram um modelo baseado em vários fatores, como: as competências em matemática; a atribuição de fatores como a sorte; esforço; dificuldade e habilidade para o sucesso/in-sucesso; o nível de conforto no curso; a experiência prévia em programação; a experiência prévia em informática; entre outros. O estudo revelou que os fatores do nível de conforto e a experiência em matemática tiveram uma influência positiva no sucesso e atribuição do fator sorte uma influência negativa para sucesso. O estudo revelou ainda, que ao nível da experiência prévia com computadores os fatores preditivos do sucesso são as aulas formais em programação e jogos. A aprendizagem formal da programação com uma influência positiva e os jogos com uma influência negativa.

O estudo apresentado em [67], descreve o trabalho efetuado na revalidação e atualização do modelo de previsão computacional do sucesso dos alunos nos cursos iniciais de programação denominado por Predict Student Success (PreSS). Este modelo computacional capaz de prever o sucesso do aluno com uma precisão de 77.5%, foi usado com bastante sucesso. No entanto, o modelo exigia a recolha de dados em papel e processamento manual, pelo que levou ao desenvolvimento de um modelo atualizado e totalmente automatizado, denominado de *Press#*. O modelo PreSS original usa três fatores para prever o sucesso do aluno, a autoeficácia em programação (do inglês self-efficacy), o resultado do exame de matemática no final ensino secundário e no número de horas por semana usadas em jogos de computador. Por autoeficácia, entenda-se como a atitude e confiança na capacidade de cada aluno para organizar e executar as ações necessárias para produzir os resultados desejados [68]. No novo modelo [67], foram incluídos novos fatores, como: a idade dos alunos; o resultado esperado pelo aluno; o tipo de instituição escolar; a maturidade do aluno (se maiores de 23 anos) e o tempo gasto nas redes sociais. A combinação com melhores resultados foi a que combinava a idade do aluno, o resultado esperado, a competência em matemática e a autoeficácia do aluno. No trabalho realizado é salientado que foi detetada uma correlação entre o tempo gasto em redes sociais e as dificuldades no curso, pelo que em trabalho futuro deve ser explorado.

2.6 Competências na resolução de problemas

Um dos principais problemas nas dificuldades no ensino e aprendizagem inicial da programação é a resolução de problemas. No entanto, a maioria dos cursos de aprendizagem da programação não ensina explicitamente as competências de resolução de problemas necessários para o sucesso na implementação de soluções de programação.

No trabalho apresentado em [69] é referida a importância do ensino da resolução de problemas na generalidade, com o principal objetivo da sua implementação num programa de computador. O autor do referido trabalho, David Gries em [69], de modo a auxiliar os alunos na resolução de problemas recomenda o seguinte conjunto de processos:

- Compreender o problema.
- Elaborar um plano.
- Executar o plano.
- Rever todo o processo.

No trabalho descrito em [70], é apresentada uma abordagem para transmitir as competências na resolução de problemas, que consiste em: 1) ensino explícito sobre a resolução de problemas de programação, que enquadra a codificação como um processo de tradução de representações mentais em soluções em código, 2) um método de visualização e monitorização da progressão da resolução de problemas, dividido em seis fases, 3) sugestão explícitas para que os alunos reflitam sobre as suas estratégias quando pedem ajuda, e 4) ajuda sensível ao contexto integrada num editor de código que reforça a aquisição de conhecimentos de resolução de problemas.

Esta abordagem baseia-se na ideia de que a programação não tem apenas a ver com a sintaxe e a semântica da linguagem de programação, mas fundamentalmente com o processo iterativo de aperfeiçoar representações mentais de problemas e soluções computacionais e de exprimir essas representações sob a forma de código.

Para pesquisar e avaliar o impacto das intervenções metacognitivas, os seus autores no trabalho apresentado em [70], realizaram uma experiência com o objetivo de verificar se com as suas intervenções melhoraram a capacidade dos alunos para descrever o seu progresso na resolução de problemas, se reforçaram a sua auto-eficácia, ou seja, o seu grau de confiança na realização de tarefas, se promoveram a mentalidade de crescimento, se as teorias são aprendidas ou inatas e, por último, se produzem maior quantidade de código funcional.

As seis fases de resolução de problemas propostas, são:

- Interpretar e clarificar o problema.

- Procura de problemas análogos.
- Procura de soluções.
- Avaliar uma potencial solução.
- Implementar uma solução.
- Avaliar a solução implementada.

Os resultados obtidos provam que o ensino da resolução de problemas para a programação é praticável e os alunos melhoram consideravelmente o seu grau de confiança na realização de tarefas, reforçam a mentalidade de crescimento e melhoram as suas capacidades de produtividade.

2.7 Live-Coding no ensino de introdução à programação

Um dos objetivos mais importantes no ensino de introdução à programação é que os alunos aprendam de uma forma metódica e coerente o desenvolvimento de programas de computador [71]. Live-coding no ensino de introdução à programação consiste em escrever código de diferentes problemas em frente dos alunos em sala de aula [72].

Revelar o processo de implementação de um programa de computador é uma parte importante para o processo de aprendizagem inicial da programação. Os meios e materiais didáticos tradicionalmente usados, como livros e apresentações de diapositivos, são insuficientes. Para além de serem insuficientes para a apresentação de um processo de desenvolvimento, normalmente são utilizados para a apresentação de uma solução ideal, que é o resultado de um processo de desenvolvimento não linear. A apresentação de um produto final sem revelar o seu processo de desenvolvimento dará inevitavelmente aos alunos uma falsa sensação que existe um processo linear e direto de solução do problema. Os alunos ao constatarem com estas apresentações simples, pensam que devem também desenvolver este tipo de soluções de uma forma semelhante. Pelo que, pode causar desmotivação e a perda da sua auto-confiança na aprendizagem inicial da programação [71].

Ainda segundo os autores do trabalho apresentado em [71], a programação ao vivo em sala de aula, utilizando o computador e o projetor, com a capacidade de executar e testar o programa, a capacidade adicional de mostrar as ferramentas de desenvolvimento, e ainda, comentar o desenvolvimento das diferentes instruções e técnicas de programação aplicadas, é uma abordagem mais próxima do processo de desenvolvimento real. No trabalho desenvolvido em [71] é descrita uma experiência com sucesso de utilização deste tipo metodologia para o ensino inicial da programação. No entanto, revelam que existem algumas desvantagens pelo facto de o tempo em sala de aula ser limitado e, de alguma forma limita a complexidade dos exemplos que são apresentados, e também, a apresentação desaparece

à medida que ocorre. Para colmatar esta desvantagem é sugerido a gravação de todo o processo e disponibilizar aos alunos para reverem as vezes que desejarem.

A preocupação com o processo de ensino e resolução de problemas não é nova. Em [69] o seu autor faz uma analogia bastante interessante e que clarifica o ponto de vista de utilização de codificação em sala de aula. O seu autor, David Gries, descreve o que pensariam os alunos de um curso de marcenaria onde o professor mostra e deixa usar durante alguns minutos uma serra, uma plaina, um martelo e outras ferramentas. De seguida, mostra um móvel com um excelente acabamento e pede aos alunos para reproduzirem esse móvel. Os alunos pensariam que seria impossível, ou muito difícil, de fazer esse trabalho com a informação transmitida pelo professor. É um pouco o que se passa nos tradicionais cursos de introdução à programação.

São vários os trabalhos [69, 71, 72] que revelam o sucesso de utilização de live-coding em sala de aula no ensino da introdução à programação. No entanto, este processo tem um elevado custo. É necessária muita experiência do professor e muito trabalho na preparação dos exemplos a apresentar, para não fazer má figura perante os alunos. Apesar de cometer erros, testar, refazer e reavaliar fazerem parte do processo de resolução de problemas e construção de programas de computador.

2.8 Erros de codificação

De modo a compreender o processo de aprendizagem dos alunos em programação, existem vários trabalhos de pesquisa, que concentram os seus esforços, nos erros de codificação que os alunos cometem, bem como, no seu tempo para os corrigir [73, 74].

No trabalho desenvolvido em [75, 76], os seus autores analisaram o comportamento dos alunos no ciclo de edição-compilação. Este indicador usado para identificar os alunos em risco académico. Segundo eles, este ciclo de edição-compilação é o reflexo típico dos alunos em colocar os seus programas sintaticamente corretos. Embora o objetivo de identificar os alunos em risco académico não tenha sido atingido, o indicador ajuda na identificação dos alunos com necessidade de atenção especial.

Identificar e corrigir erros de sintaxe é um desafio que todos os iniciantes em programação têm de enfrentar. Em [77] é analisado o tempo que os alunos demoram a corrigir esses erros. Este é um fator indicativo das capacidades de sucesso do aluno. Os alunos com menor tempo gasto na resolução dos erros é previsivelmente um aluno com maior probabilidade de sucesso. Do ponto de vista do professor, conhecer os erros tipicamente realizados pelos alunos e como são solucionados, permite melhorar o processo de ensino. Ainda, segundo [28], a capacidade de correção dos erros aumenta a confiança do aluno. Pelo que, se deve valorizar mais o ensino das competências na depuração do código na aprendizagem inicial da programação.

Determinar a origem dos erros é um problema que muitos dos principiantes na programação enfren-

tam. Essa dificuldade é na sua maioria devida à mensagem apresentada pelo compilador. No trabalho realizado em [78], é desenvolvido um detetor dos erros produzidos pelos alunos e transformado numa mensagem de erro mais informativa. No trabalho apresentado nada é referido sobre o comportamento e resultados dos alunos. No entanto, pensamos que será um fator positivo no processo de aprendizagem dos alunos.

Praticar o mesmo erro de compilação repetidamente é um forte indicador de que o aluno tem dificuldades [79]. Neste artigo, é introduzida uma métrica para quantificar erros repetidos, denominada *repeated error density* (RED). Segundo os seus autores, esta métrica é bastante independente do contexto, aspeto muito importante para a generalização dos resultados e uso efetivo.

Conhecer os erros mais comuns praticados pelos estudantes ajuda o professor a melhorar o seu processo de ensino. O professor deve alertar os alunos para as diferentes situações de erros frequentes. Apresentamos, em seguida, uma compilação dos erros mais frequentes registados pela literatura na área [74, 79, 80].

Erros de sintaxe mais comuns

- Confusão entre o operador de atribuição "=" e o operador de comparação "==".
- Utilização do operador "==" para a comparação de string's.
- Uso indevido ou contagem incorreta de { }, [], (), " ", e ' '.
- Confusão entre os operadores lógicos && e ||, e os operadores bit-a-bit & e |.
- Utilização incorreta do fim de instrução ";". No final da estrutura condicional if (); Ou nas estruturas de repetição, como: for (); while ();
- Na sintaxe da estrutura de repetição for, utilização de vírgula "," em detrimento de ";" na separação dos itens de inicialização, teste e incremento.
- Utilização da estrutura condicional if com chavetas "if { }" em vez de parêntesis "if ()"
- Utilização de palavras reservadas como nomes de identificadores.
- Chamada de métodos/funções com parâmetros incorretos.
- Não utilização de parêntesis em invocar de métodos/funções.
- Utilização de ponto e vírgula no final do cabeçalho do método/função.
- Utilização incorreta do sinal de igual nos operadores relacionais, como =<, ou =>

Erros lógicos

- Atribuição entre tipos de dados não adequada.
- Chamada de método/função não nulo em que a instrução requer um valor.
- Falta da instrução return num método/função não nulo.
- Confusão entre passar parâmetros, declarar e identificar na definição do método/função.

- Incompatibilidade entre o tipo devolvido por um método/função e o tipo que deve ser usado na sua invocação.

2.9 Parson Problems

No trabalho desenvolvido em [81], conhecemos um novo tipo de atividade aplicada à aprendizagem inicial da programação, designada por Parson Programming Puzzles. No trabalho desenvolvido, os seus autores descrevem a experiência de aplicação deste tipo de atividade no ensino e aprendizagem inicial da programação, com excelentes resultados. Estes tipos de atividades consistem no preenchimento de espaços em programas, a partir de um conjunto de opções disponíveis. Ou, um conjunto de instruções desordenadas, onde o aluno deve colocar as instruções na sua ordem lógica com a finalidade de formar um programa. Com o objetivo de tornar a aprendizagem inicial da programação mais divertida, os autores do trabalho em [81], desenvolveram uma ferramenta para gerir este tipo de atividades, com o conceito de interação de *drag-and-drop*.

As atividades foram desenvolvidas segundo os princípios seguintes:

- Maximização do empenho, ou compromisso, pelo facto de ser um jogo é muito mais apelativo para os alunos.
- Foco na lógica, o código apresentado é correto, o aluno apenas se deve concentrar na lógica do programa.
- Modelos de código. Apresentar atividades com boas práticas de utilização de código, conceitos e padrões frequentemente utilizados em programação.
- Feedback imediato, o sistema deve dar resposta imediata as ações do aluno.

Os autores sugerem ainda pequenas variações aos tipos de atividades. Sugerem a introdução de instruções de "distração", ou seja, instruções que não estão corretas, adicionadas ao conjunto de instruções disponíveis para completar o puzzle ou formar um programa.

Outros trabalhos se seguiram na utilização de este tipo de atividade Parson Problems (PP). No trabalho apresentado em [82], os seus autores investigam sobre a utilização de outro tipo de atividades de avaliação que não a escrita de código e a traçagem de um programa. Os seus autores propõem a utilização de atividades do tipo PP com esta finalidade. No trabalho apresentado foram encontradas fortes correlações entre os resultados da escrita de código e a realização das atividades de PP. No trabalho realizado foram usadas as variantes de parson problems de conjunto de instruções a ordenar em número exato e com maior número de instruções e, ainda, o preenchimento de espaços. Em conclusão os seus autores sugerem que a utilização das atividades de Parson Problems são uma boa alternativa às questões tradicionais de escrita de código.

2.10 Gamificação

A motivação é um fator preponderante em todo este processo de ensino e aprendizagem. Os jogos têm o forte poder motivacional. Os jogos utilizam um conjunto de técnicas para incentivar as pessoas a se envolverem com eles apenas pelo prazer de jogar e pela possibilidade de ganhar algo [83]. O uso de toda a mecânica dos jogos em contextos do mundo real, quais como na educação, designamos de gamificação [84]. O uso da gamificação na educação é uma das tendências mais emergentes. Nos últimos anos, a gamificação tem atraído a atenção de diversos investigadores na área de ensino e aprendizagem da programação, com o objetivo de motivar e aumentar o interesse dos alunos [85–89].

Ainda segundo [90], os professores necessitam de resolver imensos desafios, como os problemas específicos dos alunos, as suas preferências, características e motivações. Em face, desta diversidade de solicitações nem sempre é possível responder e acompanhar todos esses pedidos. Pelo que, um número considerável de alunos perde o interesse e motivação. É neste contexto, que a gamificação surge como uma ferramenta auxiliar para manter e aumentar os níveis de motivação e, consequentemente, proporcionar o sucesso dos alunos.

A gamificação oferece aos professores inúmeras técnicas para animar o processo de ensino e aprendizagem. A gamificação incentiva a participação em algo, adicionando elementos semelhantes aos de um jogo a uma tarefa. Os elementos de gamificação mais utilizadas no ensino são: tabela de classificação, ou quadro de honra (*leaderboard*), pontos, emblemas, desafios, feedback imediato, caixa surpresa e torneios de eliminação. Estes elementos permitem o reconhecimento e uma atitude positiva em relação ao trabalho desenvolvido. Estes elementos, aproveitam ainda o desejo de competir e socializar, e ainda, de avaliar a progressão do aluno, proporcionando uma competição contra si mesmos [90–92].

Os pontos, os emblemas e as tabelas de classificação são uma forma de dar feedback e acompanhar o progresso do aluno. A atribuição de pontos e emblemas é uma forma de recompensar os alunos pelo desenvolvimento de alguma atividade. As tabelas de classificação, ou quadros de honra, categorizam os alunos. Este é um conceito popular para os alunos que gostam da emoção de uma competição. Todos gostam de ver a sua equipa no topo de uma tabela de classificação. No entanto, alguma cautela deve ser tida em consideração relativamente ao uso com os resultados dos alunos. A exposição dos resultados dos alunos com maiores dificuldades e resultados baixos, pode afetar negativamente a motivação e auto-estima.

São inúmeros os estudos de utilização de gamificação no ensino superior [91, 93–95], alguns dos quais relacionados diretamente como a aprendizagem inicial da programação [96, 97], onde são relatadas melhorias significativas nos resultados da aprendizagem e um impacto muito positivo no envolvimento e na motivação dos alunos em todo o processo de ensino e aprendizagem.

As atividades idealizadas com recurso a técnicas de gamificação tem uma grande aceitação pelos

alunos em sala de aula. Sendo a falta de motivação um dos principais fatores para o abandono e insucesso dos alunos, a gamificação pode ser um dos caminhos no combate a este problema. Numa larga maioria dos trabalhos de investigação, os resultados sugerem uma influência positiva das estratégias de gamificação. No entanto, a longo prazo essa motivação possa diminuir [98]. Os autores do referido trabalho [98]., identificam um efeito de novidade e recompensas extrínsecas na motivação, seguido de uma diminuição do interesse e motivação com uma exposição a longo prazo à gamificação.

2.11 Conclusão

Neste capítulo apresentamos o estado da arte na área do ensino e aprendizagem inicial da programação. Procuramos identificar os temas mais abordados, quais os assuntos identificados como problemas, quais as soluções propostas, as metodologias de ensino e aprendizagem utilizados, e identificamos os trabalhos como boas práticas na área. Dos trabalhos analisados, destacamos aqueles que ao longo destes anos mais trabalhamos e tentamos reproduzir junto do nosso particular grupo de alunos. Destacamos claramente, as secções 2.1, a identificação do talento e tipo de atividades para a aptidão da programação, na secção 2.3, e ainda, o método de live-coding no ensino de introdução à programação, na secção 2.7. Igualmente importantes, salientamos a abordagem a um dos principais problemas no ensino e aprendizagem inicial da programação, a competências necessárias à resolução de problemas, secção 2.6. E ainda, um tipo de atividades aplicadas à aprendizagem inicial da programação, designadas por Parson Problems, secção 2.9. Como referimos, nenhum método ou ferramenta terá qualquer efeito sem motivação e muito trabalho de todos os envolvidos no processo de ensino e aprendizagem da programação. A falta de empenho, de dedicação, de persistência, de esforço e muito trabalho, por parte do aluno pode comprometer o sucesso no curso. Métodos e ferramentas são apenas meios que podem auxiliar e incentivar o processo. Com este propósito é importante conhecer e aplicar métodos e práticas testados em outros trabalhos e contexto. É importante o professor ter um conjunto de ferramentas, para as aplicar e trabalhar de acordo com as suas necessidades e dos alunos.

3

Aplicação Design Science Research Methodology na Introdução à Programação

Conteúdo

3.1	Identificação e motivação do problema	33
3.2	Definição dos objetivos para uma solução	37
3.3	Design e desenvolvimento	40
3.4	Demonstração	54
3.5	Avaliação	58
3.6	Comunicação	62
3.7	Conclusão	64

Neste capítulo, apresentamos o conjunto de operações sistemáticas e racionais a executar com a finalidade de alcançar o nosso objetivo, traçar o caminho necessário, detetar e corrigir erros no processo de ensino e aprendizagem inicial da programação. Descrevemos a aplicação da nossa metodologia de investigação, Design Science Research Methodology, no desenvolvimento da nossa proposta de ensino e aprendizagem inicial da programação, tendo em conta os objetivos definidos, reforçados pelas características muito particulares do nosso grupo de alunos, dos problemas e dificuldades identificadas, os diversos aspetos da literatura analisada e a experiência vivida ao longo dos anos de docência em introdução à programação.

Como referido na secção 1.3 - Metodologia, optamos pela DSRM como metodologia de investigação. Esta metodologia consiste numa sequência de seis atividades: na identificação e motivação do problema; na definição da solução e objetivos; na criação do artefacto; na demonstração; na avaliação; na comunicação e apresentação de resultados. Pelas razões e características do nosso grupo de estudo, enumeradas anteriormente, consideramos ser a melhor opção para o desenvolvimento e concretização dos nossos objetivos.

Identificada a metodologia a adotar, descrevemos os seis passos que a caracterizam, em cada uma das secções que formam este capítulo. Descrevemos pormenorizadamente a sua aplicação a fim de produzir informação confiável e verificável para a concretização dos nossos objetivos, e consequentemente minimizar o nosso problema.

3.1 Identificação e motivação do problema

Nesta primeira etapa, deve ser definido o problema e justificar a importância de uma solução. Enquadrar a importância do problema e identificar pontos fracos das soluções atuais.

No capítulo 1 - Introdução, identificamos como principal problema os alunos não adquirem as competências básicas em introdução à programação. E, apontamos como as principais causas para o problema a fraca motivação dos alunos para o curso, a fraca formação geral, em especial na matemática, e as dificuldades ao nível do pensamento computacional. Em consequência, estas causas evidenciam-se nas altas taxas de ausência às aulas, nas dificuldades de interpretação dos problemas, na incapacidade de construção de uma estratégia de resolução do problema e na transformação da estratégia num programa de computador. Acresce-se ainda, que uma percentagem significativa dos nossos alunos são oriundos dos PALOP, com graves problemas na sua formação geral, dificuldades na sua adaptação à sociedade, mas também de integração ao meio académico.

Apontamos, também, algumas das possíveis razões para o problema. Nesta secção complementamos a informação com a apresentação de dados que corroboraram as nossas afirmações.

Identificamos como uma das causas para o problema a fraca motivação e formação geral dos alunos,

em particular na matemática. Os resultados apresentados na tabela 1.1, em 1.1.1 - Caraterização do grupo de estudo, são demonstrativos dos problemas mencionados. Os resultados das candidaturas ao ensino superior entre os anos 2018 e 2022, revelam que o curso de engenharia informática do IPG, não é a primeira opção de escolha entre os alunos, em média apenas 37% dos alunos escolheram o curso como primeira opção. O que pode indicar alguma falta de motivação para o curso. Por outro lado, na referida tabela, podemos ainda constatar que a média de candidatura se situa entre os 11.45 e 13.48 valores, e a média para a prova de ingresso exigida, a matemática, é da ordem dos 10.62 e os 11.96 valores. Estes valores são reveladores de uma fraca formação geral dos alunos. Para além destes indicadores, é de salientar a existência de um elevado número de alunos que frequentam o curso, oriundos dos concursos especiais, na ordem dos 31%, ver tabela 1.2 - Total de alunos inscritos na unidade curricular de introdução à programação e tipo de acesso, nos últimos 5 anos., destacando os alunos dos PALOP, com graves deficiências na sua formação geral e dificuldades na sua adaptação cultural.

Nesta fase de identificação dos problemas, é importante aferir dos conhecimentos e competências específicas adquiridas ao longo do processo de aprendizagem. Para o efeito, usamos o estudo desenvolvido pelo grupo de trabalho ITICSE 2004 [2] (designado por *McCracken Group*), testado em vários países, com o objetivo de avaliar os alunos em relação aos seus conhecimentos de programação. Neste estudo, os alunos foram avaliados sobre a sua capacidade em analisar uma porção de código e prever o seu resultado e, ainda, completar código a partir de um pequeno conjunto de possibilidades.

No nosso estudo, foram utilizados os exercícios do referido estudo, com a devida adaptação para a linguagem C, conteúdos lecionados e traduzidos para Português, que resultou num conjunto de 10 exercícios, no trabalho original são 12. A tabela 3.1 mostra os resultados dos alunos divididos em quartil.

Tabela 3.1: Avaliação de conhecimentos em programação. Resultados dos alunos do IPG, anos letivos de 2021 e 2022.

Quartil	Intervalo	Número de estudantes	Percentagem de estudantes
Nível 1 (Top)	9 - 10	7	10%
Nível 2	6 - 8	18	27%
Nível 3	3 - 5	20	29%
Nível 4 (Fraco)	0 - 2	24	34%

Os resultados do estudo efetuado em *McCracken Group* [2], é apresentado na tabela 3.2.

Tabela 3.2: Resultados do estudo do *McCracken Group*, realizado em [2].

Quartil	Intervalo	Número de estudantes	Percentagem de estudantes
Nível 1 (Top)	10 - 12	46	20%
Nível 2	8 - 9	56	25%
Nível 3	5 - 7	60	26%
Nível 4 (Fraco)	0 - 4	66	29%

Analisando os resultados entre os dois estudos, no geral os resultados obtidos no nosso estudo são mais fracos que os obtidos no estudo realizado pelo *McCracken Group*. Verificamos que as maiores diferenças dizem respeito aos extremos da escala, no quartil top uma diferença de 10% para 20%, e no quartil mais inferior (fraco) uma diferença de 34% para 29%, respetivamente entre o nosso estudo (tabela 3.1) e o estudo do *McCracken Group* (tabela 3.2). Os resultados obtidos para os níveis intermédios têm diferenças menos significativas, para o nível 2 uma diferença de 2 valores (de 27% para 25%) e para o nível 3 uma diferença de 3 valores percentuais (de 29% para 26%). De assinalar as diferenças no número de alunos envolvidos no estudo (N) e no número de questões (Q), para o estudo realizado no IPG N = 68, Q = 10 e N = 228, Q = 12 para o estudo apresentado pelo *McCracken Group*. É ainda de salientar, que as diferenças aqui assinaladas entre os dois estudos é semelhante às diferenças encontradas entre os resultados obtidos pelo *McCracken Group* com a participação de todas as instituições e os resultados obtidos quando retirada a instituição com maior número participações.

Em apreciação aos resultados obtidos por este estudo, podemos afirmar que os resultados obtidos pelos nossos alunos são mais fracos que os resultados expressos pelo estudo do *McCracken Group*, salvaguardando as claras diferenças em número de inquiridos entre os estudos. Apesar dos resultados dos níveis intermédios serem bastantes semelhantes, os resultados dos extremos são 10% inferiores no quartil 1, os melhores alunos, e 5% superiores no quartil 4, dos alunos com resultados mais fracos.

Neste processo de identificação do problema e motivação, apresentamos na tabela 3.3, os resultados finais da avaliação, na unidade curricular de introdução à programação, do curso de engenharia informática, do IPG, nos últimos 6 anos. Na referida tabela, podemos verificar que a percentagem de alunos aprovados por total inscritos é na ordem dos 30%. Outro dos valores preocupantes, é o número de alunos avaliados. Este valor é da ordem dos 60% dos alunos inscritos, valor muito semelhante ao do número de presenças às aulas.

Tabela 3.3: Resultados finais da avaliação, dos últimos 6 anos, da unidade curricular de introdução à programação, do curso de engenharia informática, do IPG.

Ano letivo	Número de alunos inscritos	Número de alunos avaliados	Número de alunos aprovados	Percentagem de alunos aprovados por total de inscritos	Percentagem de alunos aprovados por total de avaliados
2017 - 2018	88	70	16	18.2%	22.9%
2018 - 2019	85	66	30	35.3%	45.5%
2019 - 2020	119	57	20	16.8%	35.1%
2020 - 2021	112	72	61	54.5%	84.7%
2021 - 2022	64	30	16	25.0%	53.3%
2022 - 2023	99	43	33	33.3%	76.7%

Na nossa instituição de ensino politécnico, IPG, é usado um sistema de ensino presencial, com metodologias de ensino e aprendizagem centradas no aluno. A unidade curricular de introdução à programação decorre no primeiro semestre do primeiro ano do curso de engenharia informática, e está estruturada em aulas teórico-práticas, aulas práticas e tutoriais, com blocos de duas horas por cada tipo de aulas. Os conteúdos programáticos definidos consistem na aprendizagem dos conceitos funda-

mentais de uma linguagem de programação para a construção de programas de computador, no caso é utilizada a linguagem de programação C. As aulas teórico-práticas são lecionadas de forma expositiva e dialogada, consistindo na apresentação dos conceitos e regras da linguagem de programação e na construção de programas de computador para a resolução de problemas. Em geral, existe uma introdução ao tema, apresentação da sintaxe, exemplificação de uso e construção de exemplos de aplicação prática, onde o professor resolve exercícios na sala de aula, comentado e questionando o desenvolvimento. As aulas práticas têm como objetivo a consolidação dos conhecimentos, despertar o interesse e curiosidade dos alunos e desenvolver a capacidade de resolução de problemas. Nas aulas práticas os alunos são confrontados com um conjunto de atividades que permitem ao aluno a pesquisa, a aplicação e experimentação dos conceitos ensinados. Por fim, as aulas tutoriais têm como objetivo o reforço ou suporte individualizado dos problemas e dificuldades dos alunos. No entanto, verificamos que este tipo de aula funciona mais como uma aula prática, do que propriamente uma aula de acompanhamento e orientação do aluno.

No modelo de ensino usado, verificamos que é difícil acompanhar o trabalho dos alunos, avaliar a sua evolução na aprendizagem, e resolver com rapidez os seus problemas e dificuldades. Existem várias razões para este facto, como por exemplo, os alunos têm dificuldades em verbalizar as suas dificuldades, ou são tímidos, ou sentem vergonha em expor os seus problemas em presença dos outros colegas, ou por considerarem as suas dúvidas insignificantes e irão expor-se ao ridículo perante a turma. Ou, ainda, devido ao número de alunos em sala de aula o professor não consegue dar a atenção e a ajuda necessária a todos.

Ao longo dos anos de experiência no ensino inicial da programação, verificamos que alguns dos problemas se devem à dificuldade que o aluno tem em interpretar a exposição do problema e mais dificuldades ainda em perceber o que está a ser pedido como resultado. Nestes casos, a dificuldade não está nos conceitos inerentes à aprendizagem da programação, mas na interpretação da língua portuguesa. A capacidade de leitura e interpretação são fundamentais para a resolução de problemas.

Constatamos ainda, que a identificação e aplicação dos conceitos mais básicos de construção de programas, como a indentação do código, a definição do tipo de dados a usar, a escolha de nomes dos identificadores, a formatação de instruções de input/output, ou ainda, a função das variáveis num programa, são um grande obstáculo para quem inicia a aprendizagem de uma linguagem de programação. Aprender a programar requer dedicação, empenho e trabalhar arduamente desde o primeiro dia. Os conteúdos de aprendizagem da programação estão fortemente interligados, a deficiência num destes conteúdos pode comprometer todo o processo de aprendizagem. Neste sentido, um bom *feedback*, contínuo e imediato entre professor e aluno, é de grande importância. O *feedback* é importante para o aluno avaliar os seus conhecimentos e manter os seus níveis de motivação altos. Por outro lado, permite ao professor conhecer objetivamente e individualmente a evolução de cada aluno no processo

de ensino e aprendizagem.

Em termos de conclusão desta etapa, identificamos o problema e justificamos o problema com a apresentação dos resultados dos últimos anos da unidade curricular de introdução à programação. Estes resultados tem uma importância enorme no desenvolvimento geral do aluno no curso de engenharia informática. Não só pela importância das competências em programação, mas também, pela forte desmotivação que pode causar aos alunos na sua formação como engenheiros informáticos. Tudo isto aliado ao conjunto de fatores que constituem a nova vida de um estudante no ensino superior.

Identificamos como pontos fracos do nosso processo de ensino e aprendizagem da programação, o fraco nível de conhecimento geral dos nossos alunos, a dificuldade do professor em acompanhar todo o processo de aprendizagem do aluno, a fraca interação dos alunos com o professor, os alunos não verbalizam as suas dificuldades e uma grande percentagem de alunos não vai às aulas. Resultando tudo isto em altas taxas de insucesso na avaliação de competências em programação.

A principal motivação para o desenvolvimento desta investigação é minimizar todos estes problemas. Fazer cada vez melhor em prol da aprendizagem do aluno em programação. Conseguir que a cada ano um maior número de alunos atinja os seus objetivos.

3.2 Definição dos objetivos para uma solução

A segunda etapa da nossa metodologia de investigação, consiste na definição dos objetivos que conduzem a uma solução. Nesta fase devem ser definidos objetivos gerais como a viabilidade e desempenho, e objetivos mais específicos que a solução para o problema deve incluir. Devem, ainda, ser considerados os métodos, as teorias e as tecnologias que podem ajudar na definição dos objetivos.

Numa primeira abordagem, no capítulo 1, na secção 1.2 *Objetivos*, enunciamos os objetivos específicos que nos permitem traçar o caminho para a concretização do objetivo geral. Nesta etapa da nossa metodologia de investigação, descrevemos e clarificamos pormenorizadamente os objetivos específicos enunciados, que tipo de atividades e como devem ser implementadas tendo em vista a sua utilização por uma ferramenta automática.

No primeiro passo da nossa metodologia de investigação identificamos como um dos problemas, os fracos conhecimentos gerais dos alunos que pertencem ao curso de engenharia informática, do IPG. Neste ponto, pouco podemos fazer. No entanto, podemos melhorar os nossos padrões de ensino, formando profissionais competentes e altamente qualificados para o desempenho da sua profissão de engenheiro informático. Com isso, ganhamos a confiança e o crédito das empresas, a reputação de formação de bons profissionais, e a procura de cada vez mais alunos pelo nosso curso. E, naturalmente, a melhoria da qualidade dos alunos que procura o curso de engenharia informática, do IPG.

Um acontecimento de realce aos dias de hoje, pode favorecer a procura pelo nosso curso, o elevado

custo de vida dos grandes centros, onde estão localizadas muitas instituições de ensino. Este facto, tem levado os candidatos ao ensino superior, a procurar instituições de ensino superior fora destes grandes centros populacionais.

Com o objetivo de solucionar o problema dos fracos conhecimentos gerais dos alunos, propomos como um dos primeiros objetivos um prólogo ao ensino inicial da programação. Este prólogo, ou pré-programação, consiste num conjunto de atividades que pretendem colmatar algumas das lacunas dos nossos alunos, em especial atividades que privilegiem o desenvolvimento do pensamento computacional. Por outro lado, este primeiro contato com os alunos pode ser usado para promover um ambiente agradável em sala de aula, criando alguma empatia entre alunos e professor e restantes colegas de curso. Desta forma, estamos também a favorecer a resolução da dificuldade dos alunos em expor e verbalizar as suas dificuldades.

Nesta etapa de definição dos objetivos para uma solução, a aquisição das competências básicas para a construção de programas de computador, é o objetivo primordial. No exercício de identificação do problema, constatamos que existem dificuldades na aplicação dos conceitos básicos de construção de programas. Estes conceitos simples tornam-se num elemento perturbador ao desenvolvimento das suas competências na programação. Verificamos, ainda, que estes conceitos provocam frequentemente grandes quantidades de erros de sintaxe e, conseqüentemente, desmotivação no aluno. Para fazer face a este problema, consideramos importante definir como objetivo a produção de um grande conjunto de atividades, com *feedback* imediato, para que o aluno possa trabalhar e praticar estes conceitos e, assim, consolidar estes conceitos introdutórios da linguagem de programação. Para apresentar este tipo de atividades aos alunos, optamos por atividades do tipo questionários de escolha múltipla, ou em inglês *Multiple Choice Question (MCQ)*. Atividades estas bastante apelativas e do gosto dos alunos.

A construção de um plano para a resolução de problemas aplicando as estruturas de uma linguagem de programação, é talvez o problema mais comum. Constatamos que os alunos conseguem apresentar uma solução, mas não construir um programa de computador. Aliado a este problema, ouvimos com frequência a expressão de “*Como iniciar ...*”. Na tentativa de minimizar este problema, definimos como objetivo a utilização de atividades do tipo *Parson Problems (PP)*. *Parson Problems* são atividades cujo objetivo consiste em colocar na ordem correta um conjunto de instruções desordenadas de um programa. Com este tipo de atividade, procuramos apresentar um conjunto de programas tipo, ou exercícios modelo, que servem de base a resolução de outros problemas. Estas atividades PP, são bastante apreciadas pelos alunos, uma vez que não cometem erros de sintaxe e aprendem conceitos e técnicas de resolução de problemas, aplicando diretamente a estruturas disponíveis na linguagem de programação.

Mitigar o problema da construção de programas de computador, sem atividades de codificação é muito difícil ou quase impossível. Estamos convictos que a melhor forma de aprender a programar é

criar e escrever programas. Pelo que, definimos como objetivo a criação de atividades de codificação. É essencial que o aluno tenha *feedback* imediato da atividade realizada. Como tal, um sistema de casos de teste deve ser implementado para avaliar a atividade realizada.

Para a concretização dos nossos objetivos, consideramos essencial que as atividades de codificação iniciem com a apresentação das diferentes estruturas da linguagem de uma forma isolada. Desta forma o aluno pratica e ganha as competências necessárias à sua utilização e aplicação. Depois de alguma prática e conhecimento nas diferentes estruturas, o aluno está mais apto à sua aplicação e junção de diferentes tipos de estruturas e técnicas de programação. Neste sentido, a organização de atividades com diferentes estruturas e níveis de dificuldade permite uma evolução lógica e consistente na aquisição de conhecimento.

Nas atividades de codificação um dos problemas manifestados pelos alunos é o *feedback* da sua resolução. É essencial que o aluno tenha *feedback* imediato da atividade realizada. Como tal, devemos considerar como objetivo a implementação de um sistema de casos de teste para avaliar a atividade.

Com o conjunto de objetivos descritos, identificamos como necessário a existência de um artefacto tecnológico no auxílio a todo o processo de ensino e aprendizagem inicial da programação. É com este propósito, que achamos importante definir como objetivo a criação de uma aplicação que centralize as melhores estratégias de ensino e aprendizagem por nós identificadas. Este artefacto deve incluir um conjunto de atividades identificadas e orientadas pelo professor, abordando as diferentes fases da aprendizagem inicial da programação em C. Este conjunto de atividades deve permitir ao aluno praticar sem receios, ao seu próprio ritmo, e com *feedback* imediato. Com este processo pretendemos que o aluno se sinta acompanhado e motivado para a aprendizagem da programação.

Com o volume de informação gerado pelos alunos na realização das diferentes atividades, traçamos igualmente como objetivo a utilização de um sistema preditivo de insucesso do aluno, baseado num modelo de rede neuronal. Este modelo tem como objetivo determinar os alunos com maior probabilidade de insucesso na aprendizagem da programação. Estes resultados são determinados em função do conhecimento adquirido ao longo dos anos e dos resultados das atividades realizadas pelos alunos. Com estes valores o professor tem a perceção dos alunos com maiores necessidades de ajuda e intervir rapidamente na resolução das suas dificuldades.

Nesta etapa definimos os objetivos para uma solução. Depois de identificados os nossos problemas e definidas as nossas questões de investigação, delineamos os principais objetivos para dar resposta a essas questões e, consequentemente, minimizar os problemas no processo de ensino e aprendizagem inicial da programação, no curso de engenharia informática, do IPG. Iniciamos com o objetivo de criar um prólogo em programação como o intuito de desenvolver alguns conceitos do pensamento computacional, e de um modo mais informal, despertar o gosto e interesse pela programação. De seguida, traçamos objetivos que permitirão o desenvolvimento das competências na aplicação das diferentes

estruturas que constituem uma linguagem de programação e, conseqüentemente, a construção de programas de computador para a resolução de problemas. Nesta fase, identificamos de grande importância a utilização de uma grande conjunto de atividades de MCQ, de PP, e de codificação, para os alunos praticarem e consolidarem os seus conhecimentos. Com o grande volume de informação gerado pelos alunos na resolução das diferentes atividades, definimos como objetivo construir um meio tecnológico capaz de ajudar alunos e professores na gestão de todas as atividades do processo de ensino e aprendizagem. Por outro lado, consideramos que os dados produzidos poderiam ser usados na previsão do insucesso do aluno e, com este propósito estabelecemos um novo objetivo.

Na etapa seguinte descrevemos o desenvolvimento dos objetivos definidos.

3.3 Design e desenvolvimento

Nesta etapa descrevemos o desenvolvimento e a concretização dos objetivos definidos. Fazemos a descrição pormenorizada da implementação dos objetivos propostos para a construção da nossa metodologia de ensino e aprendizagem inicial da programação.

Orientamos o nosso trabalho em função das características muito particulares do nosso grupo de alunos, dos diversos exemplos da literatura analisada e da experiência vivida ao longo dos anos de docência em introdução à programação, com o objetivo de melhorar o processo de ensino e aprendizagem inicial da programação. E, conseqüentemente, melhorar as competências básicas dos alunos na construção de programas de computador.

Os professores são os principais impulsionadores e motivadores deste processo de ensino e aprendizagem. Na definição dos nossos objetivos, a intervenção rápida do professor sobre as ações do aluno tem um papel importante. Os professores devem acompanhar o trabalho do aluno e responder de uma forma imediata e eficaz às dificuldades, mesmo que não solicitadas. Nesta etapa, descrevemos, também, a importância da construção de um artefacto tecnológico que permitirá a fiabilidade de todo o processo.

3.3.1 Prólogo ao ensino e aprendizagem da programação

Nesta secção, descrevemos a concretização do objetivo de implementação de um prólogo ao ensino e aprendizagem da programação.

O início do ano letivo é um processo bastante atribulado. Como referido no capítulo 1 - Introdução, na tabela 1.2, um número relevante de alunos que frequentam a unidade curricular de introdução à programação, é proveniente dos concursos especiais e alunos repetentes. Pelo que, normalmente nas duas a três primeiras semanas de aulas o número de alunos não está estabilizado, quer pelas burocracias do processo dos concursos especiais, quer pelos alunos repetentes não comparecem às

aulas, ou ainda pelo número de alunos que são colocados na segunda e terceira fase dos concursos nacionais de acesso ao ensino superior. Neste período, aproveitamos com alguma informalidade, para além de abordar os conceitos introdutórios de tecnologias de informação, a construção de um ambiente agradável, conhecer os novos alunos e realizar atividades do pensamento computacional.

O ambiente de uma sala de aula deve ser agradável, alunos e professores devem sentir-se bem. Deve ser um espaço descontraído, não demasiado formal nem demasiado relaxado. Com regras bem definidas e uma organização visível. A motivação deve ser o principal fator a promover. A abordagem dos diversos conteúdos deve ser variada, de modo a cativar a atenção dos alunos. Estes aspetos são tão importantes como a criação de empatia entre alunos e professor e restantes colegas de curso. Só assim o aluno terá a confiança necessária para verbalizar e expor as suas dificuldades. O conhecimento que o professor tem sobre os alunos permitirá adaptar as estratégias de ensino, os tipos de atividade e preferências de aprendizagem. Só deste modo, o professor poderá ter um papel preponderante e motivador na redução da taxa de absentismo, de abandono e de insucesso dos alunos na aprendizagem. A construção de um ambiente agradável é potenciada pelo facto de o número de alunos ser reduzido e existir apenas um professor na leção da unidade curricular de introdução à programação.

Idealmente este prólogo ao ensino e aprendizagem da programação, deveria funcionar como uma atividade extracurricular e complementar ao curriculum. Não sendo possível, optamos por este período inicial para propor algumas atividades que promovem a melhoria das competências cognitivas dos alunos para a programação.

Neste prólogo, criamos um conjunto de atividades que não envolvem dispositivos eletrónicos, inspirados pelo *Computer Science Unplugged* (<https://www.csunplugged.org/>). Numa época onde os jovens vivem rodeados de tecnologia e não conhecem a vida sem *smartphones* e computadores, é nossa intenção que os alunos realizem atividades sem o uso dessas tecnologias. Pretendemos que os nossos alunos resolvam um conjunto de atividades através da manipulação, explorando com prazer e sem receios de cometer erros. De este conjunto de atividades, destacamos as atividades de *Visualização espacial* e de *Seguir e Dar instruções*. Salientamos, também, a prática de *Origami*, arte secular Japonesa difundida por todo o mundo, conhecida pelo desenvolvimento de características, como: perceção visual e espacial, coordenação motora fina, memória, paciência e persistência, autoconfiança, raciocínio lógico, atenção, concentração e alívio de stress e tensão.

As atividades de *Visualização espacial* têm sido alvo de pesquisa e investigação, na sua relação entre as competências dos alunos e o sucesso na aprendizagem da programação, são exemplo os trabalhos realizados em [54, 99–101]. No nosso caso, adaptamos um conjunto de exercícios sugeridos por [1, 102] que combinam a dobragem de papel e a visualização espacial. Na figura 3.1 apresentamos um exemplo da atividade a desenvolver pelos alunos. Neste tipo de atividade, o aluno tem de escolher a opção correta, identificada com as letras de A a E, depois de o papel dobrado, perfurado e desdobrado,

como mostram as duas primeiras posições à direita da imagem. No exemplo apresentado a resposta correta é a opção C.

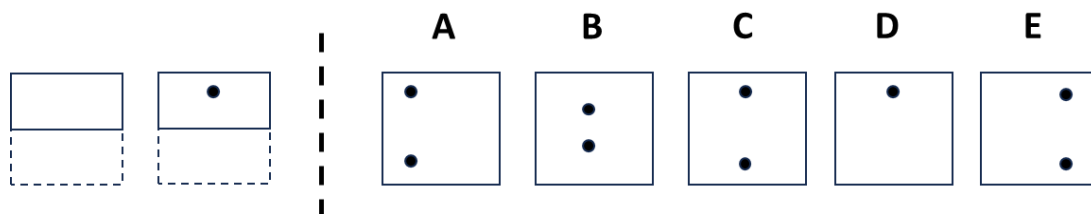


Figura 3.1: Exemplo de atividade de visualização espacial, um exemplo de dobrar e perfurar papel, adaptado de [1].

As atividades de *Seguir e Dar instruções*, também relacionada com o desenvolvimento de capacidades cognitivas para aprendizagem da programação [103], consistem num conjunto de exercícios onde os alunos devem desenhar numa folha de papel as instruções dadas por um colega ou professor. Na figura 3.2 podemos ver um dos exemplos sugeridos.

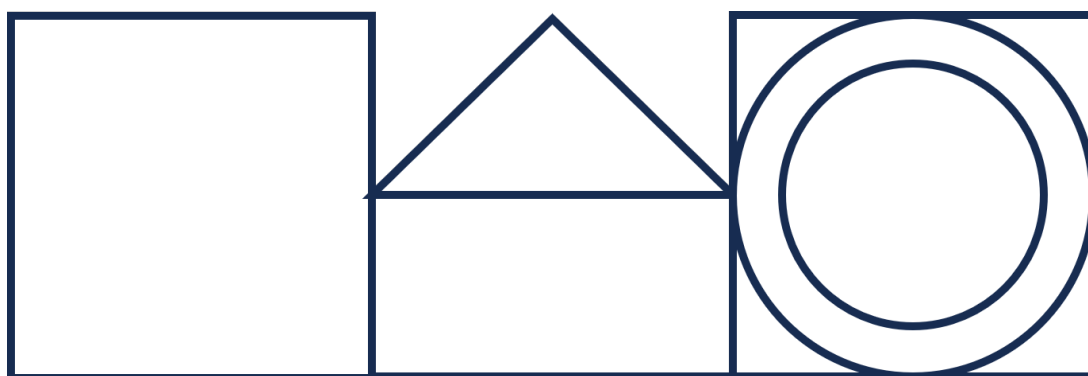


Figura 3.2: Exemplo de atividade de seguir e dar instruções.

Incluímos, também, neste grupo de atividades, exercícios que consistem em desenhar mapas para uma viagem, ou pequeno percurso, entre um ponto A e um ponto B. Ou ainda, perante um mapa, um ponto de partida e um conjunto de instruções, identificar o ponto de chegada. Outra das atividades propostas para este prólogo ao ensino da programação, é a prática de *Origami*. Sobre este tema existem muitos modelos disponíveis, com diferentes níveis de dificuldade.

Neste espaço de atividades introduzimos, também, problemas mais académicos e diretamente relacionados com as ciências da computação, como o problema *Dos Missionários e Canibais*, *As Torres de Hanói*, a *Pesquisa Binária* ou ainda a *Técnicas de Ordenação*, entre outros.

Em conclusão, o prólogo ao ensino e aprendizagem da programação, deve incluir um conjunto diversificado de atividades com o propósito de praticar e melhorar as capacidades cognitivas de cada aluno, para a aprendizagem inicial da programação. O ensinar a pensar e a aplicação de técnicas do

pensamento computacional como processo estratégico para a resolução de problemas, através de um conjunto de atividades de manipulação prática e da preferência dos alunos, devem ser consideradas como fundamentais para atingir os objetivos deste prólogo ao ensino da programação.

3.3.2 Aprendizagem da programação em C

Nesta etapa descrevemos os passos para a concretização dos nossos objetivos, relativamente aos problemas na aquisição de competências na linguagem de programação C, e a sua aplicação na construção de programas de computador para a resolução de problemas.

Na identificação do problema, constatamos que existem dificuldades na identificação e aplicação dos conceitos básicos da linguagem de programação. Verificamos que os alunos cometem erros em conceitos iniciais da linguagem de programação, como por exemplo na definição de tipos de dados, na definição de identificadores, ou na formatação de instruções de input e output. Para enfrentar este problema, definimos como objetivo implementar um grande conjunto de atividades, com *feedback* imediato, do tipo MCQ.

Nas tabelas 3.4 a 3.7, apresentamos exemplos de atividades do tipo MCQ sobre os temas iniciais da introdução à programação.

Tabela 3.4: Exemplo de atividade do tipo MCQ, sobre o tópico "*Nomes e variáveis*".

1. Assinale os identificadores corretos.	número	mesDia	Xpto
	numero1	1ºMes	Xpt0
	numeroA	ultimoMes	#inteiro
	letra2	12Meses	1valor
for	repeatUntil	soma	
doWhile	whiledo	média	
ifelse	mainprincipal	real	
while	double	int	

Na tabela 3.4, apresentamos um exemplo da atividade proposta aos alunos sobre a identificação de nomes e variáveis. No exemplo, os alunos devem identificar os nomes corretos. No entanto, usamos outras variantes da mesma atividade, como o pedido de identificação dos nomes incorretos ou, ainda, com o pedido de justificação para o nome correto ou incorreto.

Tabela 3.5: Exemplo de atividade do tipo MCQ, sobre o tópico *"Tipo de dados e constantes"*.

1. Assinale as instruções corretas. var int i = 0; double x = 1; char letra = A; int i, j, k = 1;	2. Assinale as instruções incorretas. int valor1, numero1, num1; double valor, Valor, VALOR = 0.54; char letra = '1' char letras = 'aeiou';
3. Assinale as instruções corretas. float idade = 12.5; float iva = 23%; double iva = 13%; char x123 = '+';	4. Assinale as instruções corretas. int idade = 20, anos = 31; float x = 0.001; char anos = 7; int anoletivo = 2019 float xpto = 1230;

Tabela 3.6: Exemplo de atividade do tipo MCQ, sobre o tópico *"Input/Output, formatação e constantes"*.

1. Assinale o output das instruções: int x = 7, y = 8; printf("%d : %d", x, y);	7 / 8 x = 7 : y = 8 7 : 8 x = 7 , y = 8
2. Assinale o output das instruções: float x = -5, y = 10; printf("%f %f", x, y);	x = -5.000000 y = 10.000000 -5.000000 10.000000 -5 10 -5.000000 , 10.000000
3. Assinale o output das instruções: char c = 'A'; printf("%c", c);	A a 65 97
4. Assinale o output das instruções: char c = 'A'; printf("%d", c);	A a 65 97
5. Qual o input correto para a instrução: scanf("%d", &x);	5.0 5 '5' "5"

Tabela 3.7: Exemplo de atividade do tipo MCQ, sobre o tópico "Avaliação de instruções de um programa".

1. Assinale as instruções do programa com erros: <pre>main(){ print("Vamos praticar muito. . ."); int x = 987; int soma = x + 789; print("soma = %d\n", soma); }</pre>
2. Assinale as instruções do programa com erros: <pre>main(){ int a; float soma, b; printf("A? "); scanf("%d", &a); printf("B? "); scanf("%d", &b); soma = a + b; printf("Soma = %d\n", soma); }</pre>
3. Assinale as instruções do programa com erros: <pre>main(){ int a, b; printf("Enter two numbers ? "); scanf("%d", &a, &b); printf("Sum = %d\n", a + b); }</pre>
4. Assinale as instruções do programa com erros: <pre>main(){ float a, b; int soma = 0; printf("Enter two numbers ? "); scanf("%f%f", &a, &b); soma = a + b; printf("Sum = %d\n", soma); }</pre>

Nesta etapa de descrição do processo para a concretização do objetivo que permita o desenvolvimento das competências na aplicação das diferentes estruturas da linguagem C. Classificamos de grande importância a utilização de atividades do tipo PP, para os alunos praticarem e consolidarem os seus conhecimentos.

As atividades *Parson Problems*, são exercícios em que o aluno deve ordenar as instruções de um programa. De acordo com [82, 104–106], a aprendizagem e prática com recurso a atividades de *Parson Problems* são consideradas um bom exercício para todos os que iniciam a aprendizagem da programação. Este tipo de atividade desperta grande interesse e curiosidade nos alunos. Os alunos sentem-se motivados na prática deste tipo de atividade, uma vez que não cometem erros de sintaxe

e não exige grande esforço cognitivo para definir a estratégia de resolução do problema. Do ponto de vista estratégico para o processo de ensino e aprendizagem, aproveitamos o entusiasmo na resolução destas atividades, e incluímos um conjunto de operações e técnicas clássicas de programação, de forma ao aluno se familiarizar com estas técnicas e práticas comuns.

As atividades PP são usadas preferencialmente na fase inicial de cada tópico. Com este conjunto de atividades o aluno complementa a informação e o seu conhecimento sobre o assunto em estudo. São propostas atividades com práticas comuns, algoritmos e exemplos clássicos de aplicação do tópico.

Nos exemplos seguintes, apresentamos algumas das atividades propostas do tipo PP. Na tabela 3.8, podemos visualizar na coluna da esquerda exemplos de programas, e na coluna da direita uma das configurações possíveis da atividade apresentada ao aluno. Nos exemplos apresentados, as atividades propostas sugerem a utilização de operações aritméticas de divisão inteira e resto da divisão inteira.

Tabela 3.8: Exemplo de atividade do tipo PP, sobre o tópico *"Divisão inteira e resto da divisão inteira"*.

Programa original	Parson Problem
<pre>main(){ int numero, digito; printf("Numero ? "); scanf("%d", &numero); digito = numero % 10; printf("Ultimo digito = %d \n", digito); }</pre>	<pre>printf("Ultimo digito = %d \n ", digito); } scanf("%d", &numero); main(){ printf("Numero ? "); digito = numero % 10; int numero, digito;</pre>
<pre>main(){ int valor, horas, minutos, segundos; printf("Introduza um valor em segundos? "); scanf("%d", &valor); horas = valor / 3600; valor = valor % 3600; minutos = valor / 60; segundos = valor % 60; printf("\t%02d:%02d:%02d\n", horas, minutos, segundos); }</pre>	<pre>scanf("%d", &valor); valor = valor % 3600; int valor, horas, minutos, segundos; } segundos = valor % 60; printf("Introduza um valor em segundos? "); horas = valor / 3600; printf("\t%02d:%02d:%02d\n", horas, minutos, segundos); main(){ minutos = valor / 60;</pre>
<pre>int main(int argc, char *argv[]) { int numero, digito, numInv = 0; printf("Introduza um número [10, 99]? "); scanf("%d", &numero); digito = numero % 10; numInv = numInv * 10 + digito; numero = numero / 10; digito = numero % 10; numInv = numInv * 10 + digito; printf("%d \n", numInv); return 0; }</pre>	<pre>numInv = numInv * 10 + digito; return 0; scanf("%d", &numero); } digito = numero % 10; printf("%d \n", numInv); int numero, digito, numInv = 0; numero = numero / 10; { digito = numero % 10; numInv = numInv * 10 + digito; printf("Introduza um número [10, 99]? "); int main(int argc, char *argv[])</pre>

Com o propósito de concretização do objetivo relativo aos problemas na aquisição das competências em programação e sua aplicação na resolução de problemas, propomos a existência de um conjunto de atividades que designamos por atividades de codificação.

As atividades de codificação estão agrupadas em 5 grupos. O primeiro grupo inclui os conceitos básicos da programação, como programas com instruções sequenciais simples. Onde são usadas operações aritméticas, input e output de diferentes tipos de dados e a formatação do output. No segundo grupo reunimos os problemas com as estruturas condicionais e, conseqüentemente, expressões com operadores relacionais e booleanos. As estruturas iterativas são agrupadas no terceiro grupo. Os *arrays* estão inseridos num quarto grupo. O quinto e último grupo, o qual designamos de "avançado", inclui todas as atividades de codificação com duas ou mais estruturas diferentes de programação e como maior complexidade. Com esta organização o aluno pode praticar o grupo de atividades onde sente mais dificuldades, consolidar e aperfeiçoar os seus conhecimentos em grupos específicos e, avançar ao seu próprio ritmo.

3.3.3 Construção da Aplicação HTProgramming

Para a concretização dos nossos objetivos é essencial a existência de um artefacto tecnológico que permita auxiliar alunos e professores em todo o processo de ensino e aprendizagem inicial da programação. Numa primeira fase, todas as atividades de MCQ e PP foram implementadas com recurso às ferramentas disponibilizadas pelas plataformas *Modular Object-Oriented Dynamic Learning Environment (MOODLE)* e *Google forms*. No entanto, rapidamente sentimos a necessidade de inclusão de novos elementos de avaliação e *feedback*, de gestão dados e resultados, que as ferramentas não permitiam. Nesta secção descrevemos o desenvolvimento do nosso artefacto tecnológico de ajuda ao ensino e aprendizagem inicial da programação, designado por *Help To Programming - HTProgramming*.

O esquema geral da aplicação Help To Programming (HTProgramming) é apresentado na figura 3.3. A aplicação está dividida no módulo de administração e no módulo do aluno. A aplicação foi desenvolvida na linguagem Java, com o auxílio ao IDE Netbeans [107]. É uma aplicação local, com uma ligação a uma base de dados *MySQL* remota, alojada num serviço *Hosting Smart Linux*. Para a utilização pelos alunos é necessário a instalação do compilador da linguagem C, por exemplo GNU Compiler Collection (GCC).

3.3.3.A Módulo de administração

O módulo de administração é usado pelo professor para gerir todo o processo de ensino e aprendizagem. É neste módulo que o professor cria e configura as atividades a apresentar aos alunos, pode consultar os resultados das atividades realizadas pelos alunos, bem como a sua resolução e tentativas de resolução. Os resultados obtidos pelos alunos nas diferentes atividades, são usados no modelo

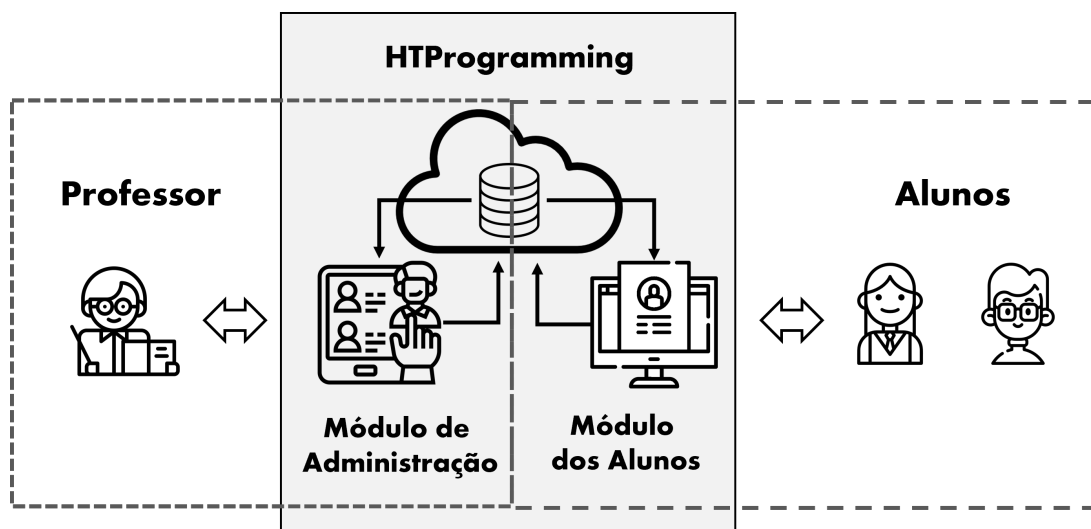


Figura 3.3: Esquema geral da aplicação HTProgramming.

de rede neuronal preditivo do insucesso do aluno (caraterizado em 3.3.3.C), com o objetivo de indicar quais os alunos com maior necessidade de atenção e apoio.

Na figura 3.4, apresentamos o aspeto geral da aplicação HTProgramming no módulo de administração. Descrevemos de seguida as opções disponíveis no módulo de administração.

ID	Nome	Pontuação
1704696	Carlos da Silva Esteves	272
1704890	Maria Jorge Mendes Almeida	267
1704729	João Paulo Santos	266
1705007	João Almeida	235
1704479	João Almeida	233
1703638	João Almeida	219
1704730	João Almeida	217
1703762	João Almeida	216
1705182	João Almeida	194
1701480	João Almeida	193
1705064	João Almeida	193
1704228	João Almeida	191

Na base da tabela, há um botão 'New Student'.

Figura 3.4: Módulo de administração

- 1 - **Students**. Permite ao professor ver a pontuação de cada aluno e introduzir novos alunos. Ao seleccionar um aluno podemos visualizar pormenorizadamente as atividades realizadas, o número de tentativas, a data da sua realização e a pontuação obtida, ver figura 3.5.

- **2 - Definitions.** Definições e configurações da aplicação. Como por exemplo: definição do ano letivo, ativar/desativar alunos na aplicação e, ainda, a configuração das atividades a visualizar pelos alunos.

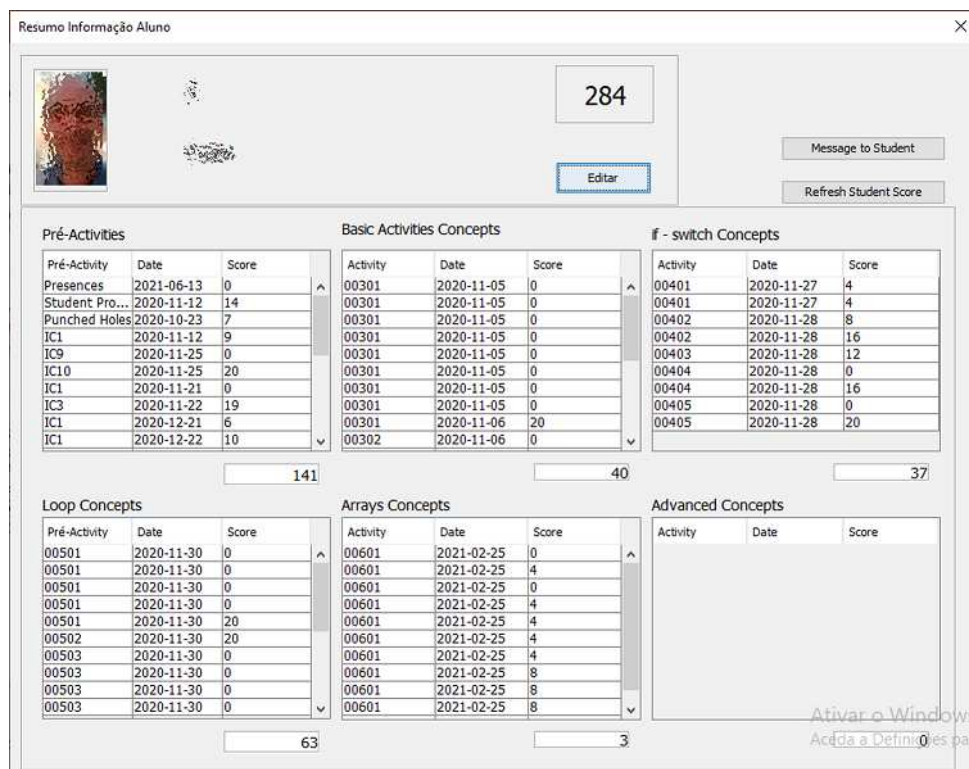


Figura 3.5: Módulo de administração, visualização pormenorizada das atividades realizadas, tentativas e resultados.

- **3 - Basic Concepts.** Esta opção permite ao professor/administrador visualizar as atividades do tipo MCQ, figura 3.6. Possibilita ainda, a configuração de novas atividades, figura 3.7.
- **4 - Coding Activity.** Visualização e configuração de atividades de codificação, representadas pela figura 3.8. Na figura 3.9 é apresentado um exemplo de configuração de um caso de teste.
- **5 - Presences.** Permite o registo da presença dos alunos, ver figura 3.10. Nesta opção é também possível visualizar todos os registos de presenças dos alunos, ver figura 3.11.
- **6 - Student Characterization.** Visualização da informação de caracterização do aluno, figura 3.12. Informação como: o número de matrículas na unidade curricular de introdução à programação, a área de estudos anteriores, a número de ordem e média de candidatura ao acesso ao ensino superior, ou ainda, a linguagem ou linguagens de programação que conhece.
- **7 - Parson Problems.** Visualização e criação de atividades do tipo *Parson Problems*, figura 3.13 e figura 3.14 respetivamente.

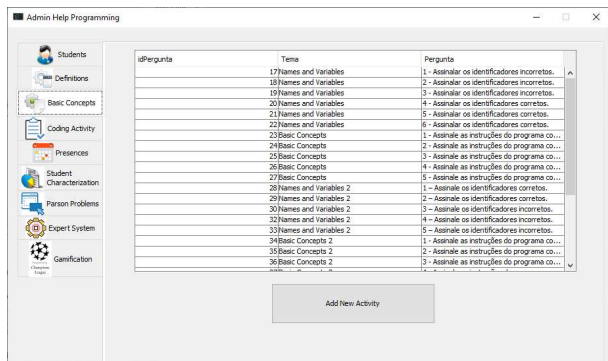


Figura 3.6: Opção Basics Concepts, visualização de atividades.

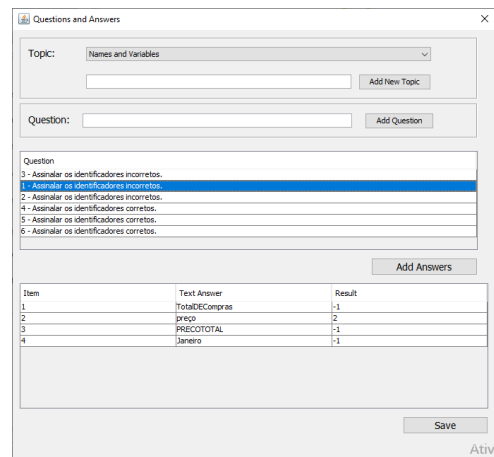


Figura 3.7: Opção Basics Concepts, configuração de uma nova atividades.

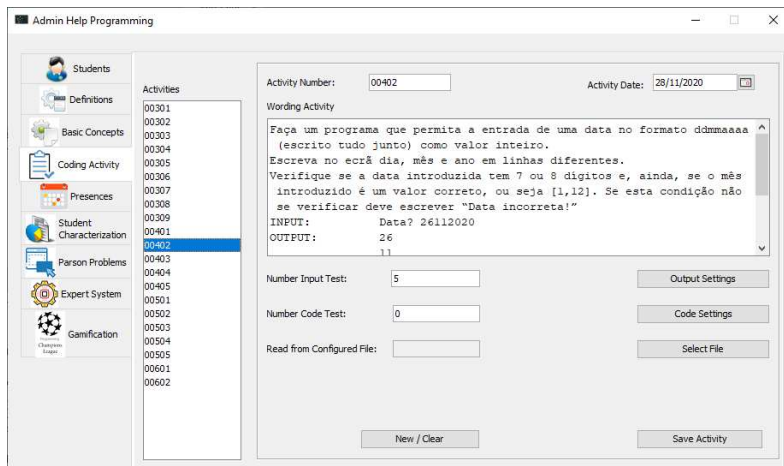


Figura 3.8: Opção Coding Activity, visualização e configuração de atividades de codificação.

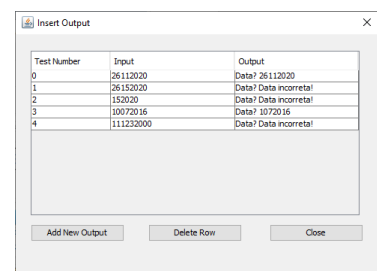


Figura 3.9: Opção Coding Activity, configuração de casos de teste para a atividade de codificação.

- **8 - Expert System.** Matriz de dados para utilização no modelo preditivo de insucesso do aluno figura 3.15.
- **9 - Gamification.** Champions League sorteio e registo de resultados, ver figura 3.16. Possibilidade de sorteio aleatório ou sorteio condicionado à pontuação de cada aluno. Para além do sorteio dos alunos a participar no jogo é, também, sorteada a atividade de codificação a resolver.

3.3.3.B Módulo do aluno

O módulo do aluno é usado por todos os alunos inscritos na unidade curricular de introdução à programação, do curso de engenharia informática, do IPG. Ou, por todos os alunos que manifestem interesse

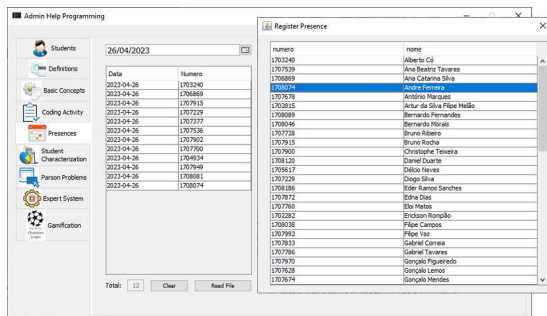


Figura 3.10: Opção Presences, registo de presenças dos alunos.

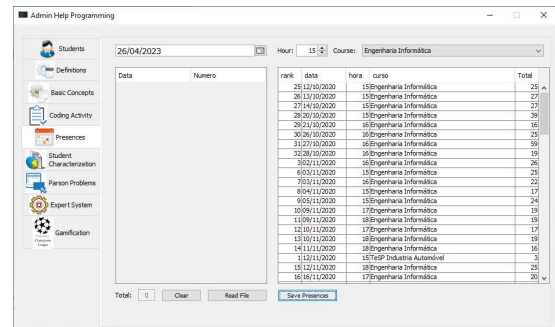


Figura 3.11: Opção Presences, visualização por aula do número de registo de alunos.

Numero	Data	NRepete	CursoAn...	OpcaoEn...	MediaCa...	Computa...	LigacaoI...	NivelInfo...	NivelPro...	Language...	Resultado
1012385	15/11/2021	1	3#Ciência...	1	12	✓	✓	3	2	C# Java	10
1012541	09/12/2020	4	2#	4	12	✓	✓	3	2	C Java	8
1701041	15/11/2021	1	2#ciências...	1	12	✓	✓	3	3	C Java	11
1701041	02/12/2020	1	3#ciências...	1	12	✓	✓	3	3	C	9
1701058	03/12/2020	1	4#Ensino ...	1	14	✓	✓	4	2	C++ Java	12
1701429	03/12/2020	1	1#CiberSe...	1	12	✓	✓	3	3	C Java	14
1701443	03/12/2020	1	1#CiberSe...	1	14	✓	✓	4	3	C C++ Java	17
1701472	03/12/2020	1	1#Ciberse...	1	10	✓	✓	4	3	Pascal C J...	16
1701480	03/12/2020	1	1#Ciberse...	1	15	✓	✓	4	4	Pascal C C...	18
1702282	02/12/2020	2	3#Ciência...	2	12	✓	✓	3	3	C Java	10
1702363	16/11/2021	1	5#engenher...	1	12	✓	✓	2	2	C++ Java	13
1702470	03/12/2020	3	4#12.ano	3	14	✓	✓	2	3	C Java	10
1702622	03/12/2020	2	3#Ciência...	2	14	✓	✓	2	2	C++ Java	10
1702770	02/12/2020	1	5#todas a...	1	512	✓	✓	5	1	C Java	13
1702815	02/12/2020	1	4#Ciência...	1	12	✓	✓	3	3	C	11
1702815	15/11/2021	1	3#Ciência...	1	12	✓	✓	3	2	C++ Java	8
1703076	12/11/2020	2	4#Curso 12	2	12	✓	✓	2	2	C	10
1703341	15/11/2021	1	1#CiberSe...	1	18	✓	✓	5	5	Pascal C+...	20
1703353	15/11/2021	1	1#Desenv...	1	13	✓	✓	3	3	Java Visua...	14
1703638	03/12/2020	2	4#Artes	2	13	✓	✓	3	3	C Java	10
1703665	16/11/2021	1	1#Desenv...	1	15	✓	✓	4	3	C# Java	16
1703762	03/12/2020	3	3#ciências...	3	12	✓	✓	3	3	C C++ Java	10
1703826	15/11/2021	1	3#Ciência...	1	13	✓	✓	0	3	C Java	9
1703918	12/11/2020	1	1#Técnico...	1	14	✓	✓	4	3	C Java Vis...	17

Figura 3.12: Dados de caraterização do aluno.

em utilizar junto do professor. Na figura 3.17, podemos observar o ecrã de entrada para o módulo do aluno. Cada aluno está obrigado a passar por um processo de identificação perante a aplicação HTProgramming. Depois da entrada validada, o aluno pode iniciar a sua utilização. O aluno tem à sua disposição um conjunto organizado de atividades para complementar o seu processo de aprendizagem. O aluno pode assim praticar, testar e avaliar os seus conhecimentos nas diferentes áreas do conhecimento que compõem a aprendizagem inicial da programação. A aluno pode realizar as diferentes atividades ao seu próprio ritmo e sem receios de errar. Por cada atividade realizada o aluno recebe de imediato a sua avaliação. De acordo com o resultado obtido, pode recebe sugestões de leitura ou revisão, ou sugestões de novas atividades. Na figura 3.18, podemos visualizar a área do aluno com os resultados e atividades realizadas.

O artefacto desenvolvido, de apoio ao ensino e aprendizagem inicial da programação, inclui o con-

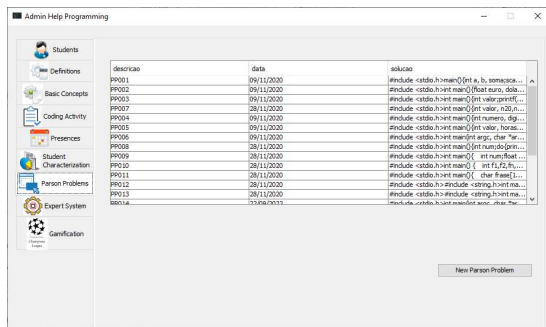


Figura 3.13: Opção *Parson Problems*, visualização de atividades *Parson Problems*.

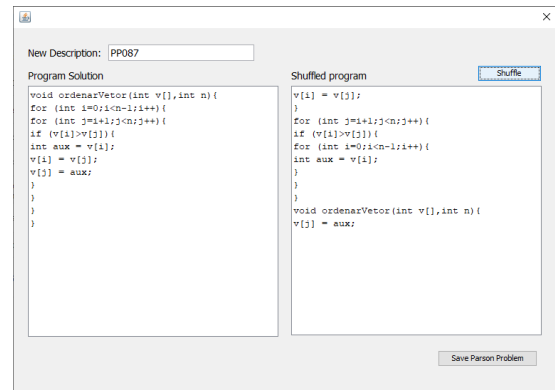


Figura 3.14: Opção *Parson Problems*, criação de atividades *Parson Problems*.

Number	Presences	SP	PH	IC	PP	Basic	if/switch	Loops	Arrays	Advanced	Expected
1707855	0	0	0	0	0	0	0	0	0	0.0	0.0
1707872	0	0	0	0	0	0	0	0	0	0.0	0.0
1707882	0	0	0	0	0	0	0	0	0	0.0	0.0
1707900	0	0	0	0	0	0	0	0	0	0.0	0.0
1707962	1	8	17	47	3	0	0	0	0	0.0	0.0
1707973	1	0	0	0	0	0	0	0	0	0.0	0.0
1707949	1	13	4	6	6	0	0	0	0	0.0	0.0
1707952	0	0	0	0	0	0	0	0	0	0.0	0.0
1707960	0	18	16	0	43	0	0	0	0	0.0	0.0
1707961	0	0	0	0	0	0	0	0	0	0.0	0.0
1707970	0	11	11	0	83	0	0	0	0	0.0	0.0
1707992	0	9	12	65	20	0	0	0	0	0.0	0.0
1708000	0	0	0	0	0	0	0	0	0	0.0	0.0
1708011	0	0	0	0	0	0	0	0	0	0.0	0.0
1708038	0	0	0	0	0	0	0	0	0	0.0	0.0
1708046	0	0	0	0	0	0	0	0	0	0.0	0.0
1708074	1	0	0	0	0	0	0	0	0	0.0	0.0
1708081	1	0	0	0	0	0	0	0	0	0.0	0.0
1708089	0	0	0	0	0	0	0	0	0	0.0	0.0
1708094	0	14	0	0	120	0	0	0	0	0.0	0.0
1708107	0	0	0	0	0	0	0	0	0	0.0	0.0
1708120	0	0	0	0	0	0	0	0	0	0.0	0.0
1708146	0	13	0	0	0	0	0	0	0	0.0	0.0
1708189	0	0	0	0	0	0	0	0	0	0.0	0.0

Figura 3.15: Matriz de dados totais dos alunos, dados a utilizar no modelo preditivo de insucesso do aluno.

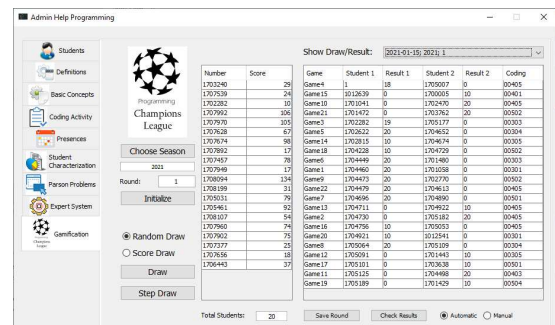


Figura 3.16: Opção Gamification, sorteio e registro de resultados.

junto de atividades identificadas nos objetivos, e que ao longo dos anos, melhor representam as necessidades, as dificuldades e preferências dos alunos. O conjunto de atividades está diretamente relacionadas com a aprendizagem inicial da programação, em linguagem C. Deste modo, organizamos as atividades em três grandes áreas. A primeira, em atividades que constituem conceitos introdutórios como nomes e identificadores, tipos de dados, identificação da estrutura de um programa e identificação de erros. A segunda área em atividades *Parson Problems*. E por fim, a área de codificação, esta encontra-se dividida de acordo com os objetivos definidos anteriormente.

O módulo do aluno, como acabamos de descrever, contém um conjunto de áreas de atividades e informação. O aluno pode praticar, testar e avaliar os seus conhecimentos e evolução no seu processo de aprendizagem. O aluno realiza as atividades ao seu próprio ritmo, recebendo automaticamente resultados do seu trabalho, sugestões de leitura ou revisão, ou ainda sugestões de novas atividades. No caso em que o aluno é sinalizado pelo sistema preditivo de insucesso, será alvo de um acompanhamento especial e personalizado pelo professor.

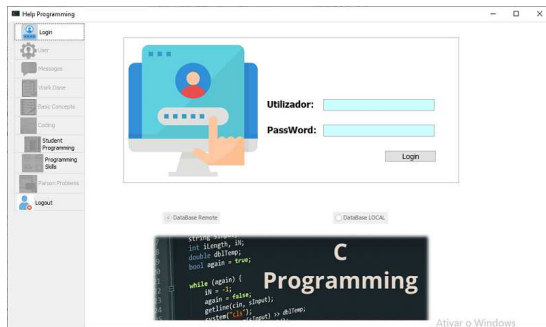


Figura 3.17: Módulo do aluno, validação de entrada.

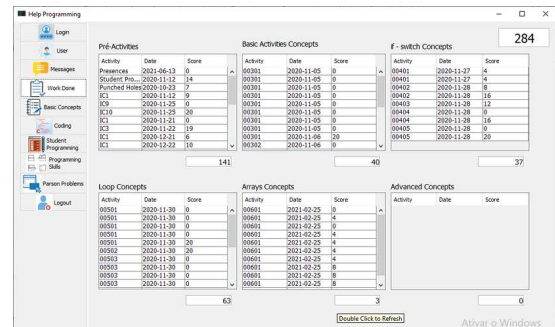


Figura 3.18: Módulo do aluno, resultados e atividades realizadas.

3.3.3.C Modelo preditivo do insucesso do aluno

Nesta secção, abordamos a descrição e o desenvolvimento do nosso modelo preditivo do insucesso do aluno.

Vários trabalhos têm concentrado os seus objetivos em prever o sucesso dos alunos. Na sua maioria a análise é baseada na interação do aluno com o compilador da linguagem de programação, como: a análise de erros dos iniciantes à aprendizagem da programação [74]; ou, na análise do comportamento do aluno no ciclo de edição e compilação [75]; ou ainda, na análise da ação dos alunos às mensagens de erro do compilador [108]. Outros estudos centram os seus estudos em comparar e testar um conjunto de preditores de desempenho tradicionais e novos preditores baseados em dados [109]. Vários outros trabalhos analisam as competências em matemáticas com o desempenho nos cursos de programação [66, 110, 111]. Existem também estudos onde foi analisada a importância da experiência prévia em programação [112, 113]. Para além de todos os fatores apresentados, o papel dos fatores cognitivos, como a resolução de problemas, a abstração, a habilidade lógica ou estilo cognitivo são também usados para prever o desempenho do aluno na aprendizagem inicial da programação [114, 115].

Nos últimos tempos, métodos de *Machine Learning (ML)* têm sido usados para prever o sucesso do aluno em cursos de introdução à programação. Em [35], é usado um modelo baseado num classificador binário de *Support Vector Machine (SVM)* para prever os resultados finais dos alunos. Em [36] é proposto um modelo de previsão, denominado de PreSS, baseado em algoritmos de ML, para prever o sucesso dos alunos em cursos de introdução à programação. O estudo analisa e compara o desempenho de vários métodos de ML.

No nosso trabalho, o modelo de previsão apresentado e desenvolvido em [116], é usado um algoritmo de *Multiple Back-Propagation (MBP)*, proposto em [117] para o treino de redes *Multiple Feed-Forward (MFF)*. Com o objetivo de construir o nosso modelo preditivo de insucesso do aluno, é criado o perfil do aluno baseado num conjunto de dados resultantes das atividades efetuadas pelos alunos e

dos dados fornecidos pelo professor. A ideia de construir o perfil com as competências do aluno em programação é baseada no conceito dos videojogos, onde as personagens podem melhorar as suas competências em áreas específicas, de forma a completar tarefas ou mudar de nível. Por exemplo, um jogador do FIFA, pode treinar a marcação de penáltis, o drible, a marcação de livres e cruzamentos, ou muitas outras ações de forma a melhorar as suas competências durante o jogo. De modo semelhante, queremos que o aluno possa melhorar e aprofundar as suas competências em programação através da realização de um conjunto de atividades adequados às suas dificuldades. Os perfis dos alunos, construídos ao longo das aulas de introdução à programação, através da monitorização e avaliação contínua das suas atividades, são usados para construir o modelo preditivo de insucesso do aluno. Ao longo do tempo, este modelo identifica os alunos com mais dificuldades e com a necessidade de maior atenção por parte do professor. Na figura 3.19, apresentamos o nosso modelo preditivo de insucesso do aluno, onde a recolha de dados é dividida num conjunto de dados de treino e de teste. O conjunto de dados de treino é usado para construir o modelo, enquanto o conjunto de dados de teste é usado para validar o modelo.

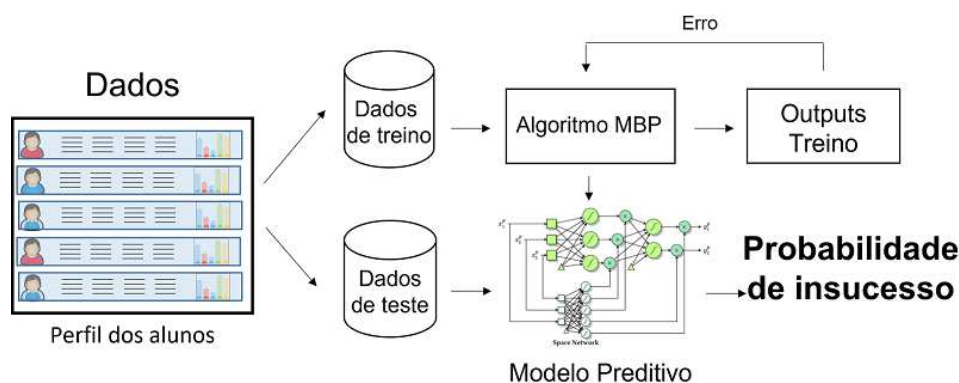


Figura 3.19: Modelo preditivo do insucesso do aluno.

Na tabela 3.9 descrevemos o conjunto de dados de características, ou variáveis, recolhidas com o objetivo de construir o perfil do aluno.

3.4 Demonstração

O objetivo desta secção é o de demonstrar que o uso das estratégias e implementação dos objetivos propostos, permitem resolver uma ou mais instâncias do problema. Ou seja, o conjunto de medidas adotadas para a melhoria das competências básicas dos alunos para a construção de programas de computador produz resultados significativos.

Na generalidade os conteúdos são apresentados, em sala de aula, com recurso a projeção de diapositivos e quadro, onde o professor escreve e interage com os alunos, formando exemplos, aler-

Tabela 3.9: Perfil do aluno, conjunto de características dos alunos.

Variáveis	Descrição
Opinião do professor	Opinião do professor, com base na participação, curiosidade, iniciativa, dedicação e empenho
Assiduidade	Número de presenças em sala de aula
Perfil do aluno	Formação anterior, média, nota de candidatura, conhecimentos em programação
Visualização espacial	Resultado das atividades de raciocínio cognitivo e visualização espacial
Seguir e dar instruções	Capacidade em dar e seguir instruções
Nomes e variáveis	Resultado das atividades de identificação de nomes e variáveis
Identificação de erros	Resultado das atividades de identificação de erros em instruções e programas
Input / Output	Resultado das atividades de definição de diferentes tipos de dados de input e identificação e formatação de output
Basics	Capacidade em definir e estruturar programas simples
Parson Problems	Resultado das atividades de <i>Parson Problems</i>
if / switch	Capacidade na utilização das estruturas condicionais if e switch
for	Capacidade no uso da estrutura iterativa for
do-while	Capacidade no uso da estrutura iterativa do-while
while	Capacidade no uso da estrutura iterativa while
Atividade_01 Atividade_02 Atividade_03 Atividade_04 Atividade_05	Atividades que avaliam as competências do aluno na utilização de diferentes estruturas de programação em linguagem C

tando para os erros mais comuns e respondendo a dúvidas. Como descrito, para a consolidação dos conceitos iniciais de aprendizagem da linguagem de programação C, implementamos a resolução de atividades do tipo MCQ, com o intuito de consolidar as competências dos alunos, mas também, que o aluno ganhe gosto e interesse pela programação. Cada atividade inclui um conjunto de questões sobre o assunto a desenvolver. Ao terminar a atividade é apresentado de imediato a sua avaliação e o conjunto de questões com a informação de resposta correta ou incorreta. Se o resultado for inferior a um limiar, definido pelo professor, é enviada automaticamente a informação para o aluno de que deve rever os conteúdos em questão. Essa informação é também enviada ao professor. O professor pode acompanhar o percurso do aluno através da visualização dos resultados das atividades e, se achar pertinente, intervir e auxiliar o aluno. Os resultados das atividades são, também, usados para quantificar as variáveis a usar no modelo preditivo de insucesso do aluno. O incentivo à realização das atividades propostas na aplicação é impulsionado pela melhoria do valor do perfil do aluno. O perfil do aluno é calculado em função dos resultados obtidos nas atividades. Este tipo de atividades envolvendo este conjunto de operações e funcionalidades só é possível se integrado numa aplicação. Pelo que, a construção de uma ferramenta tecnológica que permita implementar todas estas funcionalidades e gerir toda a informação é evidente e essencial. Este conjunto de atividades e a sua integração com a aplicação HTProgramming tornou o ensino e aprendizagem bastante acessível e participativo. Permite um maior controlo do estado de evolução da aprendizagem de cada aluno.

Definimos como objetivo a implementação de atividades do tipo PP. Com este tipo de atividades, pretendemos que os alunos complementem os seus conhecimentos na aplicação e prática de conceitos modelo para a resolução de problemas de computador. Para a utilização massiva dos alu-

nos, é necessário que seja apresentada de uma forma apelativa e que responda às necessidades dos seus utilizadores. Pelo que, é necessário que esteja integrado um artefacto tecnológico com usabilidade. A avaliação de este tipo de atividade com *feedback* imediato, é um exemplo da necessidade desta integração. Procuramos constantemente incentivar os alunos para a utilização da aplicação HTProgramming e a resolução das suas atividades. Uma das formas de estimular a sua utilização é divulgar a “*Leader Board*” do perfil dos alunos.

A prática constante de atividades de codificação é de especial relevância na aprendizagem da programação. Em especial, um conjunto de atividades orientadas de acordo com os objetivos definidos para o curso. Por outro lado, é importante a existência de *feedback* para o aluno, de modo que o aluno consiga avaliar os seus conhecimentos e manter os seus níveis de motivação elevados. Para o professor é igualmente importante o *feedback*, para acompanhar e avaliar o desenvolvimento do processo de ensino e aprendizagem da programação. A existência de um conjunto variado e organizado de atividades de codificação com *feedback* imediato, e integradas na aplicação HTProgramming, é uma das estratégias usadas para a concretização do nosso objetivo de aquisição das competências básicas para resolução de problemas de computador. Sem o recurso da ferramenta automática para auxiliar na avaliação das tarefas de codificação, será muito difícil ao professor acompanhar o processo.

Nesta etapa de demonstrar que o uso das estratégias e implementação dos objetivos propostos, permitem resolver uma ou mais instâncias do problema, destacamos a associação efetuada entre as atividades PP e de codificação. Como referido, as atividades PP devem servir de introdução e apresentação de técnicas de resolução de problemas e, em seguida, a resolução de atividades de codificação por forma a aplicar e consolidar essas competências. Apresentamos alguns exemplos de atividades de codificação associados à prática dos exercícios PP, referidos em 3.8.

- Faça um programa que calcule e escreva, em linhas diferentes, o dígito das centenas, o dígito das dezenas e o dígito das unidades, de um valor inteiro introduzido pelo utilizador.
- Faça um programa que permita converter um valor introduzido em segundos, para horas, minutos e segundos. Escreva o resultado como aspeto de hh:mm:ss.
- Faça um programa que calcule e escreva a hora de chegada, formato hh:mm:ss, dada uma hora de partida e o tempo de viagem, em horas, minutos e segundos.
- Faça um programa que determine e escreva o número de notas de 500, 200, 100, 50, 20, 10, 5 e o número de moedas de 2 e 1, existentes num valor inteiro introduzido pelo utilizador.
- Faça um programa que determine e escreva o número de notas de 500, 200, 100, 50, 20, 10, 5 e o número de moedas de 2, 1, 0.5, 0.2, 0.1, 0.05, 0.02, 0.01, existentes num valor real introduzido pelo utilizador.

- Faça um programa que permita inverter um número inteiro introduzido, no intervalo [0, 999]. Escreva o número inteiro calculado no ecrã.

No decorrer da unidade curricular de introdução à programação, procuramos efetuar este tipo de associação entre as atividades de PP e as atividades de codificação propostas. O mesmo tipo de associação e coordenação fazemos com a introdução de novos conceitos da linguagem de programação. Partindo de problemas resolvidos e conceitos já assimilados pelos alunos, introduzimos o novo elemento que queremos trabalhar. Desta forma, eliminamos a dificuldade de iniciar o novo programa. Por outro lado, justificamos a divisão das nossas atividades de codificação por grupos de estruturas da linguagem de programação. Este tipo de coordenação entre atividades, incentivam a resolução de novos problemas e à prática constante.

Utilizando como exemplo as atividades anteriores, sugerimos as seguintes atividades para a introdução das estruturas condicionais:

- Faça um programa que escreva no ecrã par ou ímpar, de acordo com o número inteiro introduzido.
- Faça um programa que determine se um valor inteiro introduzido pertence ao intervalo [0, 100]. Escreva a mensagem “Dentro do intervalo” ou “Fora do intervalo” de acordo com a situação.
- Os números atuais de telemóvel são formados com 9 dígitos. Faça um programa que determine se o número de telemóvel introduzido é formado por 9 dígitos. Escreva a mensagem de “Número correto” ou “Número incorreto”.
- Faça um programa que determine a empresa de comunicação a que pertence um número de telemóvel introduzido. Escreva no ecrã “NOS” se inicia por 93, “Vodafone” se inicia por 91, “MEO” se inicia por 96 e “Não identificada” se inicia por qualquer outro valor.
- Faça um programa que calcule e escreva a hora de chegada, formato hh:mm:ss, dada uma hora de partida e o tempo de viagem, em horas, minutos e segundos. Deve ainda considerar a possibilidade em escrever uma mensagem indicando o dia seguinte, se a hora de chegada ultrapassar as 24h.
- Faça um programa que determine e escreva o número de notas de 500, 200, 100, 50, 20, 10, 5 e o número de moedas de 2 e 1, existentes num valor inteiro introduzido pelo utilizador. Deve ter em conta que exista apenas o seguinte número de notas e moedas: notas de 500 = 1; notas de 200 = 1; notas de 100 = 1; notas de 50 = 2; notas de 20 = 2; notas de 10 = 2; notas de 5 = 2; moedas de 2 = 1; moedas de 1 = 100000 (número grande para completar o valor).
- Faça um programa que escreva se um número inteiro introduzido, no intervalo [100, 999], é uma capicua. No caso do número introduzido não pertencer ao intervalo deve escrever a mensagem de “Número fora do Intervalo”.

Por fim, os recursos necessários à implementação do modelo preditivo de insucesso do aluno, não seria possível sem o recurso à aplicação HTProgramming na recolha e preservação de dados.

3.5 Avaliação

Nesta etapa de aplicação do DSRM na aprendizagem inicial da programação, devemos medir e avaliar a estratégia adotada, bem como, o quanto o artefacto tecnológico suporta uma solução para o problema.

Abordamos este processo de avaliação em três áreas: a avaliação pelos alunos na utilização da aplicação HTProgramming, a avaliação do modelo preditivo de insucesso do aluno e, por fim, a avaliação com base nos resultados finais na unidade curricular de introdução à programação, do curso de engenharia informática, do IPG.

3.5.1 Avaliação pelos alunos

De modo a conhecer a opinião dos alunos da utilização da aplicação HTProgramming, bem como, a contribuição das suas atividades no seu processo de aprendizagem, elaboramos um inquérito de opinião. No inquérito, foi solicitado aos alunos que avaliassem numa escala de 1 (sem importância) a 5 (muito importante), a importância de cada atividade no seu processo de aprendizagem inicial da programação. Os resultados podem ser analisados na tabela 3.10. Os alunos foram inquiridos nos anos letivos de 2021 e 2022, com o número de total de participantes no inquérito de 53 alunos, respetivamente 24 e 29 alunos para os anos letivos mencionados.

Tabela 3.10: Resultados do inquérito de opinião sobre a utilização da aplicação HTProgramming e sua contribuição na aprendizagem dos alunos. (1) - Sem importância; (2) - Muito pouco importante; (3) - Pouco importante; (4) - Importante; (5) - Muito importante.

Atividades	(1)	(2)	(3)	Total [1, 3]	Percentagem [1, 3]	(4)	(5)	Total [4, 5]	Percentagem [4, 5]
Basic Concepts	1	3	7	11	20.8%	32	10	42	79.2%
Parson Problems	1	0	10	11	20.8%	17	25	42	79.2%
Coding	1	1	15	17	32.1%	17	19	36	67.9%
HTProgramming	0	1	16	17	32.1%	22	14	36	67.9%

Analisando os resultados produzidos pelos alunos no inquérito de opinião sobre a utilização da aplicação HTProgramming, constatamos que 79.2% dos alunos avaliaram como *"Importante"* ou *"Muito Importante"* a utilização da aplicação HTProgramming em atividades como *Basic Concepts* e *Parson problems*. No que diz respeito às atividades de *Coding*, 67.9% dos alunos classificaram como *"Importante"* ou *"Muito Importante"* a utilização da aplicação no seu processo de aprendizagem da programação. Em resposta à importância da aplicação, no seu geral, e sua contribuição para o seu processo de ensino inicial da programação, 67.9% dos alunos classificaram como *"Importante"* ou

"Muito importante", e 32.1% dos alunos avaliaram como "Muito pouco importante" ou "Pouco importante". Nenhum dos alunos respondeu com o valor de "Sem importância" à utilização da aplicação HTProgramming.

3.5.2 Avaliação do modelo preditivo de insucesso do aluno

Os métodos de análise preditiva são tipicamente constituídos por duas fases [118], pela fase de treino do modelo e pela fase de teste ou predição, como é o caso do nosso modelo preditivo de insucesso do aluno. Na fase de treino, usam-se os dados para os quais são conhecidos os valores de todas as variáveis, quer os dados de entrada quer os dados de saída, com o objetivo de criar um modelo. Como já referido na secção 3.3.3.C - Modelo preditivo do insucesso do aluno, o processo de construção do modelo é baseado numa rede neuronal. Na fase seguinte, a fase de predição, é usado o modelo construído para prever os valores da variável de saída. Quando se utiliza um modelo para realizar previsões existem erros, pelo que é importante avaliar o modelo. Para a avaliação do modelo preditivo uma das medidas mais usadas é a percentagem de valores corretamente classificados, ou seja, a precisão da classificação (em inglês *accuracy*) [118–120]. Que representa o rácio entre o número de valores classificados corretamente pelo modelo sobre o número total de valores classificados. No nosso caso existem apenas dois resultados o insucesso e o sucesso. Pelo que, para calcular e avaliar o desempenho do modelo de classificação podemos usar uma matriz de confusão, ou também designada de matriz de erro, ver figura 3.20. A matriz de confusão é uma matriz com duas linhas e duas colunas, onde são expressos os valores de verdadeiros positivos (VP), falsos positivos (FP), verdadeiros negativos (VN) e falsos negativos (FN). Das métricas usadas, as mais comuns são a precisão de classificação, a precisão, a sensibilidade, a especificidade e o *F-score*.

Valores Reais	Valores Previstos	
	Insucesso	Sucesso
Insucesso	VP	FN
Sucesso	FP	VN

Figura 3.20: Matriz de confusão. VP-Verdadeiros Positivos; FN-Falsos Negativos; FP-Falsos Positivos; VN-Verdadeiros Negativos

Como já referido, a precisão da classificação é o número de valores classificados corretamente pelo modelo sobre o número total de valores classificados, ou seja, segundo a representação da figura Figure 3.20 é o resultado da expressão $(VP+VN)/(VP+FP+FN+VN)$. Este valor é usado para classificar o desempenho do modelo. A precisão é a medida de verdadeiros positivos sobre o total de positivos previstos pelo modelo. Ou seja, o resultado da expressão $VP/(VP+FP)$, que representa a taxa em as

previsões positivas são realmente positivas. No nosso contexto, representa o valor de todos os alunos classificados com insucesso tiveram realmente insucesso. A sensibilidade, em inglês *recall*, é calculada pela expressão $VP/(VP+FN)$. Com este valor podemos avaliar a capacidade do nosso modelo identificar o resultado real, ou seja, de todos os alunos que realmente tiveram insucesso foram classificados com insucesso. A especificidade, ou taxa de verdadeiro negativo, é calculada pela expressão $VN/(VN+FP)$. Com o significado de quantos alunos foram identificados com sucesso, tiveram um resultado realmente de sucesso. O valor de F, em inglês *F-score*, é a média harmónica calculada com base na precisão e sensibilidade, ou seja, o valor calculado pela expressão: $2 \times (precisao \times sensibilidade)/(precisao + sensibilidade)$. O *F-score* é frequentemente usado para avaliar o desempenho de um classificador.

Na tabela 3.11, apresentamos a matriz de confusão com os resultados do nosso modelo preditivo de insucesso do aluno, com os dados dos anos letivos de 2020, 2021, 2022. Na tabela estão representadas 4 matrizes de confusão, uma para cada uma das diferentes fases de aplicação do modelo preditivo ao longo do semestre, que correspondem às diferentes fases de aprendizagem do aluno. Das características usadas no nosso modelo preditivo, descritas na tabela 3.9, a fase 1 inclui as 8 primeiras características: opinião do professor, assiduidade, perfil do aluno, visualização espacial, seguir e dar instruções, nomes e variáveis, identificação de erros e input/output. A fase 2, inclui 14 características de perfil do aluno, é constituída pelas características da fase 1 mais as características: *basics*; *Parson Problems*; *if/switch*; *for*; *do-while* e *while*. A fase 3, é formada pelas características das fases anteriores mais 3 atividades. Finalmente, a fase final inclui todas as características descritas na tabela 3.9.

Como pode ser verificado na tabela 3.11, o modelo preditivo de insucesso do aluno usado tem uma avaliação de desempenho relativamente à sua precisão de classificação superior a 70%, em todas as fases. Com exemplo, na fase 1, dos 19 alunos em que o resultado foi de insucesso, o modelo classificou corretamente 14 (VP). E, dos 16 alunos com sucesso, foram classificados pelo modelo 13 (VN). Nesta análise, o valor mais problemático é o número de alunos classificados com sucesso e na realidade tiveram insucesso, valor expresso pelo resultado dos falsos negativos (FN) 5. Ou seja, uma vez que estes alunos não foram assinalados pelo modelo preditivo, o professor não teria o alerta de necessidade de atenção especial para os referidos alunos. Na fase final, o modelo apresenta valores acima dos 81% em todas as métricas usadas na avaliação de desempenho do modelo, destacando o valor de *F-score* com um resultado de 84.2%.

Da aplicação do modelo preditivo do insucesso do aluno, com os resultados constantes na tabela 3.11, resultaram 4 redes neuronais, uma para cada fase, que vão ser aplicadas nos conjuntos de teste do último ano letivo 2022/2023. Na tabela 3.12 são apresentados os resultados obtidos. A precisão da classificação aumentou na ordem do 23%, ou seja, passou de um valor da ordem dos 75% para um valor de 92%. Nos resultados da fase 1, podemos verificar que dos 23 alunos que tiveram insucesso, foram identificados pelo modelo 21 (VP). Foram identificados pelo modelo 25 alunos com sucesso

Tabela 3.11: Matriz de confusão - Fase de treino e avaliação, dividida por fases de aprendizagem.

Valores Reais	Valores Previstos							
	Fase 1		Fase 2		Fase 3		Final	
	Insucesso	Sucesso	Insucesso	Sucesso	Insucesso	Sucesso	Insucesso	Sucesso
Insucesso	VP 14 (40,0%)	FN 5 (14,3%)	VP 15 (42,9%)	FN 4 (11,4%)	VP 13 (37,1%)	FN 6 (17,1%)	VP 16 (45,7%)	FN 3 (8,6%)
Sucesso	FP 3 (8,6%)	VN 13 (37,1%)	FP 4 (11,4%)	VN 12 (34,3%)	FP 4 (11,4%)	VN 12 (34,3%)	FP 3 (8,6%)	VN 13 (37,1%)
P. Classificação		77,1%	P. Classificação		77,1%	P. Classificação		71,4%
Precisão		82,4%	Precisão		78,9%	Precisão		76,5%
Sensibilidade		73,7%	Sensibilidade		78,9%	Sensibilidade		68,4%
Especificidade		81,3%	Especificidade		75,0%	Especificidade		75,0%
F-score		77,8%	F-score		78,9%	F-score		72,2%
P. Classificação		82,9%	P. Classificação		82,9%	P. Classificação		82,9%
Precisão		84,2%	Precisão		84,2%	Precisão		84,2%
Sensibilidade		84,2%	Sensibilidade		84,2%	Sensibilidade		84,2%
Especificidade		81,3%	Especificidade		81,3%	Especificidade		81,3%
F-score		84,2%	F-score		84,2%	F-score		84,2%

(VN), do total de 27 alunos. Apenas 2 alunos foram identificados com sucesso e na realidade tiveram insucesso (FN). Na fase final, foram identificados todos os 23 alunos que tiveram insucesso e, apenas 2 não foram identificados com sucesso, de um total de 27 alunos. Os resultados obtidos pelas métricas utilizadas são muito bons. Podemos destacar os valores obtidos para a precisão de classificação do modelo com 96%, a sensibilidade com 100% e, finalmente, o valor de *F-score* com 95,8%.

De acordo com os resultados obtidos, podemos afirmar que o modelo preditivo de insucesso do aluno, é uma boa ferramenta de ajuda ao professor na identificação de alunos com dificuldades no processo de ensino e aprendizagem inicial da programação.

Tabela 3.12: Matriz de confusão - Aplicação de rede gerada para resultados do ano letivo 2022/2023.

Valores Reais	Valores Previstos							
	Fase 1		Fase 2		Fase 3		Final	
	Insucesso	Sucesso	Insucesso	Sucesso	Insucesso	Sucesso	Insucesso	Sucesso
Insucesso	VP 21 (42,0%)	FN 2 (4,0%)	VP 21 (42,0%)	FN 2 (4,0%)	VP 21 (42,0%)	FN 2 (4,0%)	VP 23 (46,0%)	FN 0 (0,0%)
Sucesso	FP 2 (4,0%)	VN 25 (50,0%)	FP 2 (4,0%)	VN 25 (50,0%)	FP 1 (2,0%)	VN 26 (52,0%)	FP 2 (4,0%)	VN 25 (50,0%)
P. Classificação		92,0%	P. Classificação		92,0%	P. Classificação		94,0%
Precisão		91,3%	Precisão		91,3%	Precisão		95,5%
Sensibilidade		91,3%	Sensibilidade		91,3%	Sensibilidade		91,3%
Especificidade		92,6%	Especificidade		92,6%	Especificidade		96,3%
F-score		91,3%	F-score		91,3%	F-score		93,3%
P. Classificação		96,0%	P. Classificação		96,0%	P. Classificação		96,0%
Precisão		92,0%	Precisão		92,0%	Precisão		92,0%
Sensibilidade		100,0%	Sensibilidade		100,0%	Sensibilidade		100,0%
Especificidade		92,6%	Especificidade		92,6%	Especificidade		92,6%
F-score		95,8%	F-score		95,8%	F-score		95,8%

3.5.3 Avaliação pelos resultados finais

Os resultados da unidade curricular de introdução à programação, ver tabela 3.13, são o indicador bastante relevante. Para além, do número de alunos inscritos, avaliados e aprovados podemos também analisar a percentagem de alunos aprovados por total de inscritos e a percentagem de alunos aprovados por total de avaliados. Os valores obtidos nos anos letivos de 2019 e 2020, anos de pandemia, refletem algo do que passamos. No primeiro ano de pandemia, todos perante a adaptação a um novo sistema

de ensino e avaliação, com resultados bastante baixos. No segundo ano de pandemia, valores bastante altos, talvez pela maioria do processo de ensino e aprendizagem ter passado para o online e, todas as consequências e facilidades que isso possa ter causado. Verificamos que a percentagem de alunos aprovados por total de alunos inscritos com um valor de 25.0%, em 2021, passou para um valor de 33.3%, em 2022, ou seja, um aumento de 32%. Já o resultado da percentagem de alunos aprovados por total de avaliados em 2021, de 53.3%, passou para 76.6%, em 2022, o que corresponde a um aumento de 45%. Destacamos os anos letivos de 2021 e 2022, pelo facto de considerarmos que a ferramenta de ajuda à aprendizagem inicial da programação desenvolvida, a aplicação HTProgramming, estar numa fase de desenvolvimento mais avançada.

Tabela 3.13: Resultados da avaliação, dos últimos 6 anos, da unidade curricular de introdução à programação, do curso de engenharia informática, do IPG.

Ano letivo	Número de alunos inscritos	Número de alunos avaliados	Número de alunos aprovados	Percentagem de alunos aprovados por total de inscritos	Percentagem de alunos aprovados por total de avaliados
2017 - 2018	88	70	16	18.2%	22.9%
2018 - 2019	85	66	30	35.3%	45.5%
2019 - 2020	119	57	20	16.8%	35.1%
2020 - 2021	112	72	61	54.5%	84.7%
2021 - 2022	64	30	16	25.0%	53.3%
2022 - 2023	99	43	33	33.3%	76.7%

Na tabela 3.14, são apresentados os valores relativamente à utilização da aplicação HTProgramming. Apesar da sua utilização ser reduzida, os alunos que utilizaram, entre 68% e 70% dos alunos, obtiveram aprovação.

Apesar de existirem valores a melhorar, como o número de alunos avaliados e o número de alunos aprovados, os resultados de sucesso dos alunos têm vindo a melhorar. Pelo que, a promoção de utilização da ferramenta pelos alunos tem de ser mais clara e precisa. A comunicação destes resultados perante os alunos pode funcionar como um bom veículo de divulgação.

Tabela 3.14: Total de alunos que usaram HTProgramming nos anos letivos 2020, 2021, 2022. Número e percentagem de aprovados que usaram HTProgramming.

Ano letivo	Usaram HTProgramming	Aprovados Usaram HTProgramming	Percentagem Aprovados Usaram HTProgramming
2020-2021	44	30	49.2%
2021-2022	26	11	68.8%
2022-2023	32	23	69.7%

3.6 Comunicação

A última etapa do DSRM é a comunicação, a divulgação e apresentação do problema, as suas soluções e utilidade do trabalho desenvolvido. Este trabalho, é o resultado de vários anos de trabalho e muitos mais anos experiência vivida no ensino inicial da programação.

A minha dedicação ao ensino da programação teve início em 1990, no IPG, com a linguagem de programação Pascal. Outras se seguiram em vários cursos como a linguagem *Fortran 77*, *Basic*, *Visual Basic*, *C* e *Java*. Este trabalho evoluiu ao longo dos últimos anos, e só foi possível a sua concretização com a ferramenta de recolha de dados e auxílio ao ensino e aprendizagem construída para o efeito, que apelidamos de *HTProgramming*.

Iniciamos este percurso em 2016, com o trabalho desenvolvido e sistematizado em [121], como o título *"Ne-course for learning programming"*, onde fizemos uma análise sobre a utilização de um conjunto de atividades sem o recurso a meios eletrónicos, *ne-course (no electronic course)*, como um meio de desenvolvimento da aptidão dos alunos no pensamento computacional. Os resultados obtidos no trabalho desenvolvido em [122], *Improving computational thinking using follow and give instructions*, indicaram que a atividade de "seguir e dar instruções" era um bom caminho para o desenvolvimento cognitivo dos alunos em pensamento computacional e, conseqüentemente, a melhoria dos seus resultados na aprendizagem inicial da programação.

Numa apreciação e reflexão sobre o trabalho desenvolvido, até ao momento, e das experiências sentidas em sala de aula, aliar o mundo dos jogos ao processo de ensino e aprendizagem inicial da programação, parece ser uma boa tentativa de minimizar os problemas e dificuldades dos alunos. Com este objetivo, iniciamos a recolha de dados produzidos pelos alunos. Procurando fazer uma analogia com o fantástico mundo dos videojogos, onde um jogador pode construir e melhorar as suas características e competências, através de um conjunto de atividades, ou desafios, que são colocados ao longo do jogo. Desta experiência surgiu o trabalho descrito em [123], *"Building skills in introductory programming"*. Baseado nesta experiência, surge também, um pouco mais tarde, o trabalho descrito em [124], sob título *"Increasing student motivation in computer programming with gamification"*. Neste trabalho, é descrita a nossa experiência de elementos e conceitos dos jogos em sala de aula, ou seja, gamificação. Neste último trabalho, verificamos que apesar de técnicas e conceitos bastante interessantes e apelativos, dependendo do contexto e do grupo de estudo, as técnicas aplicadas de gamificação podem não ter o efeito desejado. Por exemplo, alguns alunos manifestaram o seu desagrado pela divulgação da *leader board*. Segundo eles não gostavam de ver o seu nome e, sob o olhar de todos, no final da tabela. Outra atividade, apenas para citar como exemplo, é o desafio entre pares, ao que apelidamos de *"Champions league"*. Neste tipo de atividade era sorteado um desafio, resolução de um problema de codificação, entre dois alunos, também sorteados. A atividade era bastante divertida em sala de aula, com bastante animação e entusiasmo dos alunos. No entanto, quando os alunos foram questionados sobre a atividade, alguns alunos manifestaram o seu desagrado. Pois, os alunos com mais dificuldades ao competir com os mais fortes, ou "tubarões", como apelidaram, não tinham qualquer hipótese de sucesso. Para colmatar este problema, optamos por outro sistema de sorteio, de acordo com o seu resultado de perfil.

Com o desenvolvimento do trabalho *"Building Skills in Introductory Programming"* [123], surge também a ideia de prever o insucesso do aluno. O estudo realizado em [116], com o título *"Predicting Student Failure in an Introductory Programming Course with Multiple Back-Propagation"*, descreve todo o trabalho realizado para testar a utilização de um modelo preditivo de *machine learning*. Este modelo utiliza os resultados das diversas atividades realizadas pelos alunos, no treino de uma rede neuronal, com o objetivo de tentar prever antecipadamente o insucesso do aluno. E, desta forma, lançar um alerta com o propósito de agir rapidamente na tentativa de contrariar esta previsão. Por tudo o que é descrito, é inevitável a existência de uma ferramenta de recolha de dados. Com este objetivo iniciamos a construção de um artefacto tecnológico que cumpra os objetivos pretendidos. Iniciamos o desenvolvimento da aplicação HTPProgramming (Help To Programming). O trabalho desenvolvido é descrito em [125] e [126], sob o título *"A tool help for introductory programming courses"* e *"HTPProgramming: a helping tool for teach and learn programming"*, respetivamente.

Relativamente à solução e à sua utilidade, como demonstrado no passo 5, todos os resultados indicam que estamos no caminho certo. Os alunos gostam de utilizar e consideram que a ferramenta é muito útil e complementa o seu processo de aprendizagem de programação. Naturalmente, a aplicação precisa de ser revista e melhorada no seu funcionamento. De acordo com a metodologia DSRM, é necessário rever o passo 2 e/ou o passo 3. Apontamos como alguns desses problemas o tempo inicial de acesso à ferramenta e, melhorar a forma como avaliamos as atividades de codificação. Por outro lado, a alteração de toda a interface da aplicação pode ser melhorada, com a intervenção de um especialista. E, ainda, a conversão da aplicação para uma aplicação web.

Os resultados obtidos para o nosso modelo de rede neuronal preditiva do insucesso dos alunos evidenciam bons resultados. No entanto, com uma maior quantidade de dados e com um conjunto mais estável de variáveis/atividades, o nosso modelo preditivo de insucesso do aluno baseado no perfil do aluno, deverá produzir melhorias significativas nas diferentes fases de aprendizagem.

3.7 Conclusão

Em suma, apresentamos neste capítulo aplicação da nossa metodologia de investigação *Design Science Research Methodology* (DSRM). Com esta metodologia de investigação, procuramos concretizar os objetivos definidos, traçando o caminho, identificando e corrigindo os nossos erros no processo de ensino e aprendizagem. Descrevemos a construção da nossa proposta de ensino e aprendizagem inicial da programação, com o intuito de reduzir os nossos problemas. Especificamos a construção de um artefacto tecnológico, que designamos por HTPProgramming, e avaliamos a sua influência no processo de ensino e aprendizagem. Procuramos, também, demonstrar que o uso das estratégias e implementação dos objetivos propostos, como o uso de atividades PP e de codificação, com *feed-*

back imediato, permitem minimizar o problema. Apresentamos um modelo preditivo de insucesso do aluno usado no nosso trabalho, como o propósito de detetar precocemente o insucesso do aluno, e intervir rapidamente. Os resultados obtidos pelo modelo preditivo de insucesso do aluno, são promissores. Mostramos, também, que os resultados finais da avaliação da unidade curricular com a utilização da aplicação *HTProgramming* melhoram significativamente. E, por fim, descrevemos a última etapa da nossa metodologia de investigação *DSRM*, a comunicação. Nesta última secção, descrevemos o percurso traçado ao longo destes últimos anos, dos estudos e artigos realizados, das sugestões e comentários obtidos, na tentativa de melhorar e aperfeiçoar o nosso processo de ensino e aprendizagem inicial da programação. E, conseqüentemente, a concretização do nosso objetivo, que os alunos consigam adquirir as competências básicas para a construção de programas de computador.

4

Discussão

Conteúdo

4.1	Perfil do aluno	69
4.2	Linguagem de programação adotada	72
4.3	Tipo de Atividades	74
4.4	Modelo preditivo	77
4.5	Ferramenta automática	77
4.6	Conclusão	78

Neste capítulo serão apresentados e discutidos os resultados deste trabalho de investigação. Tendo em conta a revisão bibliográfica e os resultados obtidos, nomeadamente pelas atividades realizadas pelos alunos, os inquéritos realizados e por observação, procuramos interpretar os resultados obtidos e suas implicações no processo de ensino e aprendizagem inicial da programação.

Optamos por dividir esta apresentação e discussão de resultados em quatro tópicos. Iniciamos com a análise às características dos alunos que constituem o nosso grupo de estudo, a que designamos por “Perfil do aluno”. De seguida, na secção 2, abrimos a discussão e análise à linguagem de programação adotada no nosso curso e a sua influência no processo de ensino e aprendizagem da programação. Na secção três, analisamos e discutimos os resultados da aplicação do tipo de atividades usadas neste trabalho, por forma a minimizar o problema. Na secção quatro, discutimos a importância de utilização e implementação de um modelo preditivo do insucesso do aluno na concretização do nosso objetivo. Por fim, analisamos a importância da utilização de uma ferramenta automática de ajuda ao processo de ensino e aprendizagem inicial da programação.

4.1 Perfil do aluno

Este trabalho de investigação surge da necessidade de aumentar o nível de competências dos alunos nos conceitos básicos de programação na construção de programas de computador. E, consequentemente, melhorar todo o nosso processo de ensino e aprendizagem inicial da programação, reduzir as nossas taxas de insucesso, de abandono e de desmotivação dos alunos. Nesta secção, abrimos a discussão analisando o perfil e motivação dos nossos alunos.

Aprender a programar é difícil. A aprendizagem da programação exige um importante nível de esforço cognitivo e de trabalho árduo. Muitos dos alunos não estão disponíveis nem motivados para este nível de esforço. E, consequentemente, problemas graves no processo de ensino e aprendizagem inicial da programação. Este é um problema universal, não é específico de um curso, de uma escola ou de uma região ou país. Como abordado nos capítulos 1 e 2, são vários os trabalhos de investigação onde o foco principal são as dificuldades no ensino e aprendizagem da programação. Infelizmente, a nossa escola não é exceção. Os problemas são semelhantes aos das outras instituições. Os resultados obtidos no estudo sobre a avaliação de conhecimentos em programação, apresentado na secção 3.1 - Identificação e motivação do problema, onde foi efetuada uma comparação com os resultados obtidos na nossa escola e os resultados obtidos no estudo efetuado em [2], comprova essas semelhanças nos problemas e dificuldades.

Analisando o conjunto de características ou competências do aluno, ou seja, o perfil do aluno, a motivação é talvez o fator mais preponderante para qualquer atividade. Se gostamos, se estamos motivados, se temos prazer na atividade que estamos a desenvolver tudo fica mais fácil. Na caracterização

do nosso grupo de estudo abordamos este tema através da análise dos resultados das candidaturas ao ensino superior, tabela 1.1. Como podemos observar, a escolha do curso como primeira opção na candidatura de acesso ao ensino superior é inferior a 40%, nos 5 últimos anos. A exceção é o ano letivo de 2021, onde o número de alunos colocados foi muito baixo (11) e o número de alunos que escolheram como primeira opção o curso de engenharia de informática, no IPG, foi de 7, que corresponde a uma percentagem de aproximadamente 64%. Estes valores são reveladores da falta de motivação para o curso.

Encontramos na nossa pesquisa que os conhecimentos em matemática têm uma forte influência na aptidão para a programação. São vários os trabalhos que mostraram previamente esta relação, como o trabalho realizado em [59], [66] e [67]. Ainda, da análise dos valores do acesso ao ensino superior, apresentados na tabela 1.1, verificamos que as competências em matemática são fracas, reveladores de dificuldades na área. A nota de candidatura ao ensino superior em matemática, situa-se entre os valores de 10.62 e 11.96.

Para além desta importante constatação sobre os conhecimentos em matemática, de acordo com os resultados apresentados na tabela 1.2, podemos comprovar que os alunos que frequentam a unidade curricular de introdução à programação, resultantes do concurso nacional de acesso ao ensino superior é, em média, inferior a 30%. A percentagem de alunos no nosso curso, com acesso via concursos especiais é na ordem dos 31%. Desses alunos, na sua maioria são alunos com origem nos PALOP. Estes alunos têm graves problemas na sua formação geral e, de um modo geral, problemas e dificuldades na sua vida pessoal. Os problemas começam logo na obtenção de visto de autorização à permanência no nosso país. Constatamos que estes alunos, na sua generalidade, não se apresentam nos primeiros dias de início do curso, vão surgindo ao longo do semestre devido à dificuldade na obtenção do visto. Com as dificuldades inerentes à aprendizagem inicial da programação, o não acompanhamento das primeiras aulas, traduz-se em graves problemas e dificuldades para os alunos no acompanhamento dos objetivos e competências definidas para cada momento do processo de aprendizagem. Tudo isto, aliado às dificuldades de adaptação a um novo sistema de ensino, viver num novo país e muito distante da família, com hábitos e culturas diferentes.

Nesta discussão sobre o perfil do aluno, analisamos de seguida as capacidades dos alunos na resolução de problemas. Como abordado no nosso trabalho, o pensamento computacional é uma das melhores competências para todos. Definimos o CT como um conjunto de competências para a resolução de problemas usando técnicas das ciências da computação. Para a resolução de muitos problemas, é importante definir um plano para a sua resolução usando técnicas de computação. Abordado no capítulo 2, a falta destas competências, em aplicar e utilizar o pensamento computacional, é um dos temas mais investigados e associado à causa dos problemas na aprendizagem da programação.

No prólogo de aprendizagem à programação, através do conjunto de atividades propostas, cons-

tatamos, por observação, que grande parte dos alunos tem grandes dificuldades em descrever uma figura, em indicar o caminho para um lugar, e em interpretar muitas das instruções na prática de Origami, para referir apenas alguns exemplos. Apesar de uma amostragem reduzida, podemos comprovar que a maioria dos alunos nunca teve a oportunidade de praticar e desenvolver este tipo de atividades. Verificamos, também, que os alunos depois de instruídos sobre algumas simples técnicas a utilizar, melhoram significativamente a qualidade das suas instruções e resolução das atividades. Destacamos neste prólogo, a motivação extra e entusiasmo nos alunos quando adicionamos alguma competição. Como exemplo, na prática de origami a escolha da figura é muito relevante. A escolha da figura da *“rã saltadora”* destaca-se claramente na preferência dos alunos. Para além de ser de um nível de execução fácil, no final podemos efetuar uma competição do maior salto entre as rãs saltadoras. Outro dos exemplos bastante apreciado pelos alunos são as figuras de aviões. Onde, como no exemplo anterior, podemos efetuar uma competição entre os diferentes tipos de aviões, o que realiza o voo mais longo.

Analisar e interpretar o resultado e a importância deste tipo de atividades é um dos objetivos desta discussão. Como descrevemos, este tipo de atividades é do interesse da maioria dos alunos, existindo também alunos que mostram indiferença. No entanto, verificamos que maioria dos alunos mostra grande interesse e curiosidade, quando explicada a ligação e enquadramento com a aprendizagem da programação. Observamos melhorias significativas no desenvolvimento das primeiras atividades para as atividades seguintes. Por outro lado, depois de alertados para a necessidade de um plano onde são aplicadas técnicas como a decomposição, o reconhecimento de padrões, a abstração e o algoritmo. Ou ainda, como abordado na secção 2.6 - *Competências na resolução de problemas*, compreender o problema, elaborar um plano, executar o plano e rever todo o processo, constatamos que os alunos começam a adotar estas estratégias na resolução dos problemas apresentados.

Em análise à implementação do prólogo à aprendizagem da programação, consideramos os resultados bastante positivos. Verificamos diferenças significativas na motivação e conhecimentos dos alunos, no antes e depois da prática das atividades implementadas. No entanto, como referido é aplicado num curto espaço de tempo. A nossa sugestão é a criação de uma atividade extracurriculares com o tema do pensamento computacional, a realizar ao longo do semestre. Mas a sua implementação é de difícil concretização. A falta de tempo referida pelos alunos, a falta de disponibilidade dos professores, o facto de ser facultativa os alunos não participam, ou mesmo, por simples problemas administrativos, são algumas das razões que justificam a dificuldade de implementação.

Incluimos, ainda, nesta secção de discussão e análise ao perfil do aluno, os resultados do absentismo escolar. Os valores relativos ao absentismo às avaliações, já apresentados na tabela 3.3 - Resultados finais da avaliação, dos últimos 6 anos, da unidade curricular de introdução à programação, do curso de engenharia informática, do IPG., na coluna do *“Número de alunos avaliados”*, confirmam

este problema. Os alunos que se submetem à avaliação são em média 52.7%, dos alunos inscritos na unidade curricular de introdução à programação. Introduzimos, também, os valores relativos ao número médio de alunos presentes às aulas, ver tabela 4.1 - Presenças às aulas, anos letivos de 2019 e 2022.. Como podemos constatar o número médio de alunos presentes nas aulas é de 39 alunos, divididos por duas turmas, o que corresponde a uma percentagem média de alunos por alunos inscritos de 40.5%. Estes valores expressam o abandono e desmotivação dos alunos perante as dificuldades encontradas no processo de ensino e aprendizagem inicial da programação. O número de presenças às aulas diminui gradualmente ao longo do semestre, com uma redução significativamente no último mês de aulas. Apontamos como razão deste fenómeno, o aumento das dificuldades e exigências dos conceitos mais avançados de programação. O somatório de dificuldades e problemas vão se agravando e culminam no abandono da unidade curricular. Como já foi salientado, os conteúdos de aprendizagem da programação estão fortemente interligados, a deficiência num destes conteúdos pode comprometer todo o processo de aprendizagem. Relativamente à diferença entre a percentagem média de alunos presentes nas aulas (40.5%) e a percentagem média de alunos avaliados (52.7%), apontamos como justificação o número de alunos repetentes que não frequentam as aulas, mas apresentam-se à avaliação.

Com o nosso perfil de alunos e adicionado o absentismo escolar estão reunidas as condições para o insucesso escolar.

Tabela 4.1: Presenças às aulas, anos letivos de 2019 e 2022.

Ano letivo	Número médio de alunos por aula	Número de alunos inscritos	Percentagem de presenças por total de alunos inscritos
2019 - 2020	43	119	36.1%
2020 - 2021	37	112	33.0%
2021 - 2022	29	64	45.3%
2022 - 2023	47	99	47.5%

Em suma, os valores reportados revelam que os alunos desistem da unidade curricular de introdução à programação, não frequentam as aulas, nem se submetem à avaliação. Estamos convictos, que este problema tem origem nas dificuldades inerentes ao tema deste trabalho, as dificuldades de ensino e aprendizagem da programação. O sucesso na concretização dos nossos objetivos vai minimizar este problema.

4.2 Linguagem de programação adotada

Discutimos nesta secção, a adoção da linguagem de programação C. Podemos questionar se os problemas e dificuldades na aprendizagem inicial da programação serão da utilização da própria linguagem C. Neste ponto, podemos iniciar a discussão com o porquê da utilização da linguagem C, e não de outra. Em primeiro lugar, é uma decisão da unidade técnica e científica de informática, e não de uma

pessoa em particular. No entanto, pensamos que para alunos de um curso de engenharia informática é uma linguagem que cumpre com os requisitos e exigências necessárias ao desenvolvimento do curso.

Procuramos, também, saber o que fazem os outros. Em consequência, efetuamos uma pesquisa em Association for Computing Machinery (ACM) Digital Library, com a seguinte *string* de pesquisa: *[[Title: "programming"] OR [Title: "coding"]]* AND *[Title: teach* learn* problem* difficult*]* AND *[[Abstract: "programming"] OR [Abstract: "coding"]]* AND *[Abstract: teach* learn* problem* difficult*]* AND *[Keywords: programming teaching learning problems difficulties cs0 cs1]* AND NOT *[Keywords: object]* AND *[E-Publication Date: (01/01/2013 TO *)]*

Obtivemos 813 resultados, mas apenas 156 diretamente relacionados com o ensino e aprendizagem inicial da programação. Destes, conseguimos identificar 83 relacionados com o ensino superior, que é o foco do nosso trabalho. Na tabela 4.2 - Identificação de linguagem de programação, por artigo, no ensino e aprendizagem da programação (pesquisa ACM Digital Library), podemos visualizar a linguagem, ou linguagens, de programação e o número de trabalhos que a referem. Apesar de existirem em maior número os artigos que referem a linguagem C, achamos não ser uma diferença significativa. Este valor pode ter duas interpretações, uma existem mais problemas com a utilização da linguagem C, ou é a linguagem mais usada e alvo de mais estudos. No entanto, podemos considerar que independentemente da linguagem de programação usada existem problemas e dificuldades no ensino e aprendizagem da programação.

Tabela 4.2: Identificação de linguagem de programação, por artigo, no ensino e aprendizagem da programação (pesquisa ACM Digital Library)

Linguagem de programação	Número de artigos
C	13
Java	11
Python	10
Python/Java/C/C++	6
C++	5
Python/Java	3
Outras	35

Analizamos, também, o que fazem as outras instituições do ensino superior em Portugal. Dividimos a nossa pesquisa em dois grupos, universidades e politécnicos. A pesquisa é baseada na informação disponibilizada nos sites das respetivas instituições, em julho de 2023. Na tabela 4.3 - Linguagem de programação em cursos universitários em Portugal., podemos visualizar as linguagens de programação usadas no primeiro ano dos cursos de engenharia informática, em universidades. Constatamos, que na sua maioria existe uma primeira unidade curricular que usa a linguagem *Phyton*. No entanto, também existe uma segunda linguagem nesse primeiro ano que é a linguagem C. No que diz respeito aos politécnicos, ver tabela 4.4 - Linguagem de programação em cursos politécnicos em Portugal., a linguagem que predomina é a linguagem *Java*.

Nesta discussão da linguagem de programação e sua aprendizagem inicial, constatamos que em

Tabela 4.3: Linguagem de programação em cursos universitários em Portugal.

Instituição de Ensino	Curso	Unidade Curricular	Linguagem de Programação	Ano / Semestre
Universidade Aveiro	Licenciatura em Engenharia Informática	Fundamentos de Programação	Python	1 / 1
		Programação Orientada aos Objectos	Java	1 / 2
Universidade Coimbra Faculdade de Ciências e Tecnologia	Licenciatura em Engenharia Informática	Introdução à Programação e Resolução de Problemas	Python	1 / 1
		Princípios de Programação Procedimental	C	1 / 2
Universidade de Lisboa Instituto Superior Técnico	Engenharia Informática e de Computadores	Fundamentos da Programação	Python	1 / 1
		Introdução aos Algoritmos e Estruturas de Dados	C	1 / 2
Universidade do Minho	Engenharia Informática	Laboratórios de Informática I	Haskell	1 / 1
		Laboratórios de Informática II	C	1 / 2
Universidade do Porto Faculdade de Engenharia	Engenharia Informática e Computação	Fundamentos da Programação	Python	1 / 1
		Programação	C/C++	1 / 2

todas as linguagens existem problemas. A utilização da linguagem C não está desfasada da realidade. Podemos refletir sobre a utilização de outra linguagem de programação, mas a maioria dos problemas vão persistir. Os problemas da falta de motivação dos alunos para o curso, a fraca formação geral, a ausência às aulas, os problemas na capacidade de resolução de problemas e, ainda, os problemas na transformação da estratégia de resolução num programa de computador, não se resolvem apenas com a utilização de outra linguagem de programação.

4.3 Tipo de Atividades

Como referido anteriormente, os conteúdos de aprendizagem da programação estão fortemente interligados, a deficiência num destes conteúdos pode comprometer todo o processo de aprendizagem. Com esta perspetiva, é importante que o aluno inicie a aprendizagem da programação motivado, com empenho e dedicação para aprendizagem. Os problemas nos conceitos iniciais, como: indentação do código; a definição e identificação dos tipos de dados a utilizar; a escolha de nomes para os identificadores; e, a aplicação da sintaxe correta de instruções de *input* e *output*; são de grande importância no impacto geral da aprendizagem da programação. As atividades relativas a este tipo de conceitos, são bastante simples e é dedicado pouco tempo para a sua apresentação e prática em sala de aula. Pelo que, como observado os alunos tem problemas na abordagem a este tipo de conceitos. Em sala de aula, são apresentados os tipos de dados, as regras de construção de nomes e identificadores e a sintaxe para as instruções de *input* e *output*, complementados com alguns exemplos. Reconhecemos que não observamos, atentando aos pormenores, e dificuldades individuais dos alunos. Pelo que, os problemas revelam-se mais tarde. A agravar a situação, a maioria dos alunos não manifestam as suas dificuldades e problemas, pelo que é difícil corrigir essa série de acontecimentos. Ou seja, não existe um *feedback* efetivo entre alunos e professor. O professor não consegue acompanhar a resolução de

Tabela 4.4: Linguagem de programação em cursos politécnicos em Portugal.

Instituição de Ensino	Curso	Unidade Curricular	Linguagem de Programação	Ano / Semestre
Politécnico de Beja	Engenharia Informática	Introdução à Programação	Java	1 / 1
		Programação Orientada aos Objectos	Java	1 / 2
Politécnico de Bragança	Engenharia Informática	Laboratório de Programação	C	1 / 1
		Programação Orientada por Objetos	C++	1 / 2
Politécnico de Coimbra	Engenharia Informática	Introdução à Programação	C	1 / 1
		Programação	C	1 / 2
Politécnico da Guarda	Engenharia Informática	Introdução à Programação	C	1 / 1
		Algoritmos e Estruturas de Dados	Python	1 / 1
Politécnico de Santarém	Informática	Fundamentos da Programação	Python	1 / 1
		Programação Orientada aos Objectos	Java	1 / 2
Politécnico de Setúbal	Engenharia Informática	Introdução à Programação Orientada por Objetos	Java	1 / 1
		Programação Orientada por Objetos	Java	1 / 2
Politécnico de Tomar	Engenharia Informática	Introdução à Programação e à Resolução de Problemas	Python	1 / 1
		Programação Orientada por Objectos	Java	1 / 2
Politécnico de Viseu	Engenharia Informática e Telecomunicações	Fundamentos da Programação	Python	1 / 1
		Complementos de Programação	Java	1 / 2

todos os alunos, não consegue seguir as atividades realizadas pelos alunos com incorreções. Pelo que, o problema propaga-se pelo processo de aprendizagem do aluno e quando detetado é tarde.

A utilização de atividades do tipo escolha múltipla, MCQ, relacionadas com este tipo de dificuldades iniciais, comprovou-se serem de grande utilidade e eficácia. Nas atividades propostas, foram concebidos exercícios especificamente para este tipo de atividade, de modo a criar a dúvida, a hesitação e a incerteza na resposta entre os alunos. Deste modo, os alunos praticam com um conjunto de exercícios muito específico para consolidar os seus conhecimentos.

Para os conceitos subsequentes na aprendizagem da linguagem de programação usamos atividades do tipo *Parson Problems*, com resultados muito bons. De um modo geral, os alunos conhecem a sintaxe e sabem as diferenças entre estruturas condicionais e iterativas. Os alunos sabem exemplos de aplicação muito simples. As maiores dificuldades surgem na transformação do seu plano de resolução, de uma linguagem informal, para uma linguagem de programação. Neste tipo de atividade o aluno tem tudo o que necessita para a concretização da atividade, não comete erros de sintaxe, apenas analisar o conjunto de instruções existentes e colocar na ordem correta. Como referido anteriormente, os alunos realizam este tipo de atividades com agrado. Neste tipo de atividades, aproveitamos para introduzir alguns exemplos clássicos e conjunto de instruções típicas da aprendizagem de uma linguagem de programação. Antes de iniciar as atividades de codificação, o aluno já praticou e conhece um conjunto de instruções típicas de resolução de problemas mais comuns.

As atividades anteriormente identificadas como de codificação, referem-se a atividades onde o aluno

codifica uma solução para um problema proposto. Este conjunto de atividades tem como principal objetivo a motivação do aluno para a resolução de vários problemas, o trabalho árduo e prática constante para a aquisição das competências básicas na linguagem C. Com a realização destas atividades o aluno pode avaliar a sua evolução no processo de aprendizagem, e o professor tem conhecimento das atividades desenvolvidas, dificuldades e problemas de cada aluno.

Destacamos a implementação de *feedback* imediato às atividades de codificação. É referido com frequência pelos alunos, que “*Não sabem se o programa está correto!*”. Esta dúvida leva à desmotivação dos alunos, uma vez que não têm uma forma de aferir sobre a qualidade do trabalho realizado. A criação de uma ferramenta que permita dar resposta a esta necessidade, justifica a implementação na aplicação HTProgramming. Para a sua concretização implementamos para cada atividade de codificação, um conjunto de casos de teste para a avaliação do código realizado pelo aluno. Neste ponto, podemos discutir sobre a utilização de ferramentas existentes para este tipo de operações, como os sistemas de gestão de concursos de programação na Web, ou plataformas de programação competitiva, como por exemplo: *Mooshak 2* (<https://mooshak2.dcc.fc.up.pt/>), ou ainda, *Project Euler* (<https://projecteuler.net/>), *Coderbyte* (<https://coderbyte.com/developers>), *HackerRank* (<https://www.hackerrank.com/>) ou *CodeChef* (<https://www.codechef.com/>). Sendo estas plataformas ótimas ferramentas para desenvolver e praticar as competências na resolução de problemas e programação, o nível de desafios é elevado e exigente para o nosso grupo de alunos. Mas esta possibilidade não deve ser descorada, a existência de desafios categorizados por níveis de dificuldade, a possibilidade de escrever código e testar diretamente, e normalmente associada a este tipo de plataformas existe uma grande comunidade de programadores que contribuem ativamente em fóruns, são excelentes ferramentas para a aprendizagem e prática da programação. Podem ainda ser, uma fonte de motivação extra para determinados grupos de alunos. Ao longo destes anos de experiência de ensino da programação com o nosso enquadramento e perfil de alunos, este tipo de ferramentas exige um nível bastante mais elevado de conhecimentos e empenho, que de um modo geral, os nossos alunos não estão preparados para o desafio. Por outro lado, não se enquadra no nosso objetivo de aprendizagem das competências básicas de programação.

Em resumo, as atividades implementadas apresentam bons resultados na concretização dos nossos objetivos. O conjunto de atividade do tipo MCQ, permite ao aluno praticar um variado número de conceitos simples, de uma forma bastante ativa e motivadora. As atividades do tipo PP, permitem ao aluno adquirir um conjunto de conceitos e práticas comuns da aprendizagem inicial da programação, de uma forma bastante apelativa e do interesse dos alunos. Por fim, atividades de codificação orientadas pelo professor, divididas por níveis e temas, de *feedback* imediato, permitem ao aluno a aplicar e consolidar os seus conhecimentos na aprendizagem da programação.

4.4 Modelo preditivo

A implementação e utilização de um modelo preditivo de insucesso do aluno, como descrito, surge com o intuito de motivar os alunos através do incrível mundo dos videojogos, e correlacionar o volume de dados gerado com os resultados da unidade curricular de introdução à programação. Com base do trabalho desenvolvido e nos resultados obtidos no modelo preditivo de insucesso do aluno, devemos valorizar esta ferramenta como um auxiliar importante no processo de ensino e aprendizagem inicial da programação.

Nesta análise, atenção especial deve ser dada ao conjunto de características dos alunos usadas na construção do modelo preditivo do insucesso do aluno. Como por exemplo a variável *Opinião do professor*, obtida através do conhecimento do professor com base na participação, curiosidade, iniciativa, dedicação e empenho do aluno. Esta variável é de difícil apreciação e valorização. Como descrito, um dos problemas é o acompanhamento pelo professor do trabalho e da evolução dos alunos. Como tal, este valor é de difícil apreciação. Por outro lado, se o professor avalia o aluno neste conjunto de parâmetros é com toda a certeza um bom indicador.

A variável *Seguir e dar instruções* é também de discussão e análise. Em estudos anteriores, como o trabalho realizado em [122], constatamos que os resultados desta variável tem grande relevância nos resultados finais dos alunos. O estabelecimento de um valor para esta variável é de complexa automatização. Este tipo de atividade é normalmente usado nas primeiras semanas e em sala de aula, pelo que facilita o processo de avaliação.

O restante conjunto de variáveis são os resultados das atividades realizadas pelos alunos. Estes valores são de simples automatização e recolha, uma vez que fazem parte do conjunto de atividades implementadas na aplicação *HTProgramming*.

O nosso melhor conjunto de características dos alunos experimentado na construção do modelo preditivo do insucesso do aluno é o apresentado na tabela 3.9 - Perfil do aluno, conjunto de características dos alunos., com os resultados da avaliação do modelo divulgados na tabela 3.12 - Matriz de confusão - Aplicação de rede gerada para resultados do ano letivo 2022/2023.. No entanto outros conjuntos de características foram testados e avaliados com bons resultados, como os exemplos dos trabalhos realizados em [116, 125–130].

4.5 Ferramenta automática

Nesta secção, analisamos a importância da existência de uma ferramenta automática de gestão do processo de ensino e aprendizagem da programação, designada por *HTProgramming (Help To Programming)*. Como descrito, a aplicação surge da necessidade de uma gestão muito personalizada das atividades de escolha múltipla, *MCQ*, e *Parson Problems*. Naturalmente, ao longo da investigação deste

trabalho, foram surgindo propostas e necessidades de várias ferramentas que constituem a aplicação HTProgramming. Com o propósito de cumprir o objetivo de fornecer *feedback* imediato às atividades realizadas pelos alunos, a construção de uma ferramenta automática é imprescindível. Por outro lado, o registo das atividades desenvolvidas ao longo do processo de ensino e aprendizagem, é de especial importância para o professor. Uma vez que, permite avaliar o progresso de cada aluno, aferir das dificuldades e propor atividades de acordo com as necessidades individuais e coletivas. Em apreciação final, a aplicação foi desenvolvida à medida das necessidades e concretização de ideias, e como consequências não tem uma estrutura bem definida e organizada. Necessita, também, de um novo design mais apelativo e usabilidade. No entanto, satisfaz as nossas necessidades de proporcionar um conjunto de atividades com *feedback* imediato, registar as pontuações obtidas, calcular o perfil do aluno, e ainda, permitir a utilização do modelo preditivo de insucesso do aluno.

4.6 Conclusão

Neste capítulo apresentamos e discutimos as principais conclusões do nosso trabalho. Em resumo, iniciamos o nosso trabalho com a constatação de que os nossos alunos têm fracos níveis de motivação, fracos conhecimentos na sua formação e dificuldades na resolução de problemas. Atendendo a estas características, desenvolvemos um conjunto de atividades por forma a minimizar estes problemas, que incluímos num contexto de que designamos por prólogo ao ensino e aprendizagem inicial da programação. Os resultados obtidos pela implementação prólogo são significativos e muito importantes para prossecução do curso. Para além, de conceitos e técnicas de que a maioria dos alunos nunca teve a oportunidade de praticar, permite criar um ambiente agradável em sala de aula, de entreajuda e estimular o gosto e a paixão pela aprendizagem da programação. De acordo com os resultados obtidos consideramos ter alcançado o nosso objetivo específico de trabalhar atividades que permitam melhorar as competências dos alunos na resolução de problemas.

A segunda secção deste capítulo, analisamos a linguagem de programação adotada. Abordamos este tema devido ao problema inicial dos nossos alunos, relativamente aos conceitos iniciais da linguagem de programação, como identificadores e variáveis, atribuição, tipos de dados, formatação de dados input e output, instruções e expressões aritméticas simples, expressões booleanas, blocos de instruções, entre outras. Por outro lado, na divulgação do nosso trabalho, ouvimos alguns comentários sobre a utilização da linguagem C, que sugeriam a utilização de outra linguagem de programação, como por exemplo Java ou Python, para a minimização destes problemas. No entanto, consideramos que os problemas são mais abrangentes, e não se devem unicamente à linguagem de programação adotada. Apresentamos, também, resultados de outros trabalhos na área e o que outras instituições de ensino português adotam nos seus cursos, onde podemos concluir que a linguagem C adotada não

está desfasada da realidade.

Apresentamos de seguida os resultados sobre a utilização dos diferentes tipos de atividades adotadas para ajudar os nossos alunos na resolução dos seus problemas. Abordados e enquadrámos as atividades de escolha múltipla, *MCQ*, adotadas para a prática e consolidação dos conceitos iniciais enunciados. Apresentamos, descrevemos e avaliamos a importância de utilização das atividades *Parson Problems* no nosso contexto de ensino e aprendizagem inicial da programação. Concluindo que são uma ótima ferramenta para a prática inicial das diferentes estruturas que constituem uma linguagem de programação e, também, para transmissão de técnicas clássicas de resolução de problemas na construção de programas de computador.

Na aprendizagem inicial da programação a prática constante e trabalho árduo do aluno são importantes. Mas para que tal possa suceder, é necessária a motivação dos alunos e a existência de um grande número de atividades de codificação, com *feedback* imediato. Em face de este conjunto de atividades é imprescindível a existência de uma ferramenta automática capaz de gerir toda a informação do processo de ensino e aprendizagem. Justificando a construção da aplicação apresentada, *HTProgramming*.

Analisamos, também, a importância da implementação de um modelo preditivo do insucesso do aluno. Com os resultados obtidos, consideramos importante a utilização desta ferramenta no nosso processo de ensino e aprendizagem inicial da programação. O modelo preditivo permite determinar os alunos com maior probabilidade de insucesso e agir rapidamente em auxílio desses alunos.

Por fim, fundamentamos a utilização e construção de uma ferramenta automática, *HTProgramming*, de ajuda a todo o processo de ensino e aprendizagem inicial da programação.

5

Conclusão

Conteúdo

5.1 Principais contributos	84
5.2 Trabalho Futuro	86
5.3 Publicações	89

Neste último capítulo, produzimos as nossas conclusões sobre o trabalho realizado. Destacamos os principais contributos do nosso trabalho, através da resposta às questões de investigação. Abordamos, também, as possibilidades de desenvolvimento e de trabalho futuro, com ênfase no envolvimento do ChatGPT no ensino e aprendizagem inicial da programação. Por fim, apresentamos as publicações realizadas com o objetivo de divulgação, apresentação de resultados e oportunidade de discussão com a comunidade académica sobre o trabalho realizado.

Iniciamos este trabalho enfatizado a importância das competências básicas em programação. Destacando a importância da programação e de todas as competências adicionais que o seu estudo pode proporcionar. Competências essas como a criatividade, a capacidade de resolução de problemas, a persistência, a colaboração, a comunicação e o pensamento crítico. De vital importância para os todos os que enfrentam o mercado de trabalho atual e futuro.

Salientamos a importância de uma base sólida de conhecimentos em programação, em especial para os alunos dos cursos onde a programação é uma área fundamental, como o curso de engenharia informática. Um aluno com dificuldades em programação irá sofrer desse estigma ao longo do curso.

Por outro lado, o ensino e aprendizagem da programação é uma área relativamente recente e em constante evolução e transformação. Felizmente, verificamos com agrado, que existe muito trabalho e preocupação em melhorar todo o processo de ensino e aprendizagem da programação. No nosso trabalho, centramos a nossa reflexão nas dificuldades no ensino e aprendizagem inicial da programação no ensino superior, e num caso muito particular, na unidade curricular de introdução à programação, do primeiro ano, do curso de engenharia informática do IPG. No entanto, muitos outros trabalhos concentraram as suas análises e apreciações em outros níveis de ensino. Uma dessas evidências é a existência de atividades de enriquecimento curricular, no primeiro ciclo do ensino básico, em domínios científico e tecnológico, entre outros. Onde a generalização e divulgação de conceitos básicos do pensamento computacional, de robótica e de programação, são praticados desde a tenra idade. Por outro lado, são bastante frequentes as disciplinas e cursos profissionais, ao nível do ensino secundário, onde a oferta formativa inclui a programação. Com toda a certeza, num futuro muito próximo, algumas das dificuldades e problemas do ensino e aprendizagem da programação que enfrentamos hoje, não o serão amanhã.

Em todo este processo de ensino o professor tem evidentemente um papel fulcral. O professor deve gostar do que faz, deve ter experiência no ensino inicial da programação e deve ser capaz de compreender e respeitar todos os ritmos de aprendizagem. Ao longo dos muitos anos de ensino da programação, temos constatado que a quantidade certa de empatia com os alunos, facilita bastante todo o processo. Especialmente, quando queremos motivar os alunos, incentivar o gosto pela programação, e que verbalizem as suas dificuldades.

5.1 Principais contributos

O objetivo principal deste trabalho foi o de melhorar todo o processo de ensino e aprendizagem da programação, com o desígnio de os alunos adquirirem as competências básicas em programação para a construção de programas de computador. E, conseqüentemente, um maior número de alunos ter sucesso na unidade curricular de introdução à programação. Salientando a importância de uma boa base de conhecimentos em introdução à programação, para um bom desempenho no curso de engenharia informática.

Objetivamente, em face à complexidade do tema, o nosso trabalho consistiu em investigar que soluções podemos incluir no nosso processo para minimizar os problemas no ensino e aprendizagem inicial da programação. Este trabalho, tem uma origem e um destino muito particular associado ao nosso perfil de alunos. É, também, o resultado dos vários anos de docência e de pesquisa. Esperamos, ainda, que o nosso trabalho seja um bom contributo para todos aqueles que enfrentam problemas no ensino e aprendizagem da programação.

De seguida, apresentamos os principais contributos deste trabalho, que nos permitem responder às questões de investigação.

Relativamente à primeira questão de investigação: ***Q11 - Que conjunto de atividades devem ser usadas para melhorar as competências na aprendizagem inicial da programação?*** O nosso objetivo foi o de investigar que conjunto de atividades devem ser usadas no nosso processo de ensino e aprendizagem inicial da programação, de modo a minimizar os nossos problemas e melhorar significativamente as competências básicas dos alunos na resolução de problemas e construção de programas de computador.

Em resposta à primeira questão de investigação formulada e o principal contributo do nosso trabalho, identificamos o aumento de mais de 100%, mais precisamente o valor de 107.2%, para o sucesso dos alunos. Este valor é obtido, com os valores da tabela 3.13 - Resultados da avaliação, dos últimos 6 anos, da unidade curricular de introdução à programação, do curso de engenharia informática, do IPG., calculando a média da percentagem de alunos aprovados por total de avaliados para os anos letivos de 2017 a 2019 (34.5%) e a os anos letivos de 2020 a 2022 (71.5%). Esta é a principal evidência do nosso trabalho. Demonstrou-se que através do conjunto de medidas adotadas, as competências na aprendizagem inicial da programação melhoraram significativamente. E conseqüentemente, o sucesso dos alunos da unidade curricular de introdução à programação.

Para este resultado, contribuíram de modo significativo o conjunto de atividades adotadas no processo de ensino. A proposta de atividades de escolha múltipla, designadas ao longo do nosso trabalho com a expressão em inglês *Multiple Choice Question* (MCQ), o conjunto de atividade do tipo *Parson Problems* (PP) e, ainda, as atividades de codificação, todas com *feedback* imediato, revelaram-se de especial importância para os resultados obtidos. Em resposta à questão de investigação enunciada,

destacamos claramente estas atividades como as que mais contribuíram para a melhoria das competências em programação.

As atividades de escolha múltipla, permite ao aluno praticar e consolidar a sua aprendizagem num conjunto de conceitos iniciais da linguagem de programação, e que como constatamos têm uma forte influência na evolução dessa aprendizagem.

As atividades do tipo *Parson Problems*, têm um papel de destaque neste processo de ensino e aprendizagem. Para além dos alunos manifestarem um interesse especial na resolução dessas atividades, permitem transmitir um conjunto de conceitos modelo de resolução de problemas e de construção de programas, que facilitam a aquisição e consolidação das competências dos alunos na aplicação das diferentes estruturas da programação na construção de programas de computador.

As atividades de codificação têm um papel importante em todo o processo de ensino e aprendizagem inicial da programação. Estamos convictos que prática constante e trabalho árduo na criação de programas é a melhor forma de aprender. A resolução uma grande variedade de exercícios de codificação, com *feedback* imediato, permite ao aluno adquirir experiência na utilização de um conjunto de técnicas e padrões de resolução. Este conjunto de técnicas e padrões de resolução ajudam no combate à dificuldade dos alunos na aplicação das diferentes estruturas da linguagem de programação, à dificuldade na resolução de problemas, e ainda, à dificuldade em transformar a estratégia de resolução num programa de computador.

Devemos salientar o prólogo ao ensino e aprendizagem da programação. Como referido, pouco podemos fazer, ou controlar, no que respeita aos conhecimentos prévios dos nossos alunos. Ou, na motivação dos alunos na escolha do nosso curso e instituição de ensino. No entanto, podemos proporcionar um bom ambiente de aprendizagem, fortalecer e estimular a camaradagem, a comunicação e amizade entre os alunos e professor, através do conjunto de atividades e jogos propostos neste prólogo. Por outro lado, são propostas atividades que permitem ao aluno conhecer, praticar e desenvolver competências do pensamento computacional. E, talvez o mais importante, estimular o gosto pela programação.

Relativamente à segunda questão de investigação: ***Q12 – Como podemos prever o insucesso dos alunos?*** Foi nosso objetivo identificar algum tipo de comportamento, conjunto de características ou perfil do aluno, que pudesse indicar antecipadamente o insucesso. E, com essa informação, atuar rapidamente em seu auxílio.

Como resposta a esta questão, implementamos um modelo de previsão de insucesso do aluno, baseado numa rede neuronal. Foram avaliados vários conjuntos de dados dos alunos, com o objetivo de construir o melhor conjunto de características para a concretização do nosso objetivo. Em face dos resultados obtidos, na avaliação do modelo de previsão de insucesso do aluno, com o conjunto as características selecionadas, podemos concluir que é um ótimo aliado ao nosso processo de ensino e

aprendizagem. Com este modelo será possível prever os alunos com maior probabilidade de insucesso, e intervir rapidamente em seu auxílio, minimizando os seus problemas e dificuldades, no sentido de combater a previsão.

Por último, argumentamos relativamente à terceira questão de investigação formulada: ***Q13 - Que contribuição a utilização de uma ferramenta tecnológica poderá trazer para o processo de ensino e aprendizagem inicial da programação?***

Ao longo do desenvolvimento deste trabalho, mostramos que a utilização de uma ferramenta tecnológica que concentre em si, um grande conjunto de atividades, que permita avaliar e dar *feedback* imediato ao aluno, e professor, é de grande importância e essencial para a concretização do nosso objetivo. No nosso trabalho, construímos e descrevemos uma plataforma, *HTProgramming*, com o intuito de colmatar as nossas necessidades e problemas enunciados. *HTProgramming* integra diversos tipos de atividades, desenvolvidas especificamente pelo professor, com propósitos bastante específicos e em função das características muito particulares no nosso grupo de trabalho. *HTProgramming* é uma ferramenta de grande importância na recolha de dados para utilização no sistema preditivo de insucesso do aluno.

Com esta ferramenta, *HTProgramming*, contribuímos para a resolução de alguns problemas apontados, como a dificuldade no acompanhamento das atividades realizadas pelos alunos e na respetiva evolução do seu processo de aprendizagem. Em especial, aqueles alunos que não divulgam as suas ações e não verbalizam as suas dificuldades. *HTProgramming* permite também, colmatar a falta de *feedback* imediato sentida por muitos alunos, motivar e incentivar o trabalho autónomo. Com esta ferramenta, o aluno pode testar e avaliar os seus conhecimentos e, também, praticar as atividades ao seu ritmo e repetir o número de vezes que necessitar.

Do ponto de vista do professor, esta ferramenta ajuda no acompanhamento de todo o processo de ensino e aprendizagem. O professor tem uma visão geral dos resultados e atividades realizadas pelos alunos. Com estes indicadores o professor pode intervir em auxílio do aluno, quer com o envio de mensagens, quer por um contacto mais individualizado.

Em suma, a concretização e utilização da ferramenta tecnológica é preponderante em todo o processo de ensino e aprendizagem proposto neste trabalho.

5.2 Trabalho Futuro

O Ensino e aprendizagem da programação é uma área de investigação muito vasta e complexa. Muitas abordagens e linhas de investigação podem ser seguidas e testadas, dependendo das características dos alunos e sistema de ensino, umas abordagens funcionam melhor do que outras. No entanto, é importante que o professor tenha a consciência de que existem problemas, e esteja propenso a novas

ideias e alterações ao sistema de ensino da programação, em face das dificuldades e problemas dos alunos.

Um debate profícuo deve ser realizado entre todos os envolvidos no sistema de ensino e aprendizagem inicial da programação, não só pelo que a disciplina representa e sua importância no seio de um curso em informática, mas também pela pouca importância que na grande maioria dos casos se atribui à disciplina aquando da distribuição de professor. As disciplinas de introdução à programação deviam ser atribuídas aos professores com mais experiência e não aos professores mais novos sem grande experiência na área, e em início de carreira profissional.

Relativamente à nossa proposta, melhorias significativas devem ser realizadas na aplicação desenvolvida para auxiliar todo o processo de ensino e aprendizagem inicial da programação, *HTProgramming*. Nomeadamente, a nível de usabilidade e design da aplicação. Por outro lado, a transformação numa aplicação web, melhoraria significativamente a velocidade de acesso e resposta da aplicação.

Os resultados obtidos através do sistema preditivo do insucesso do aluno são bons. Contudo, maior volume de dados, o nosso modelo de *machine-learning*, baseado numa rede neuronal, deverá produzir melhorias significativas. O conjunto de características deve, também, ser alvo de atenção especial. Por um lado, devemos seleccionar um conjunto de variáveis que melhor possam caracterizar o aluno, mas também que sejam fáceis de obter por um processo automático.

Na sequência do aparecimento do ChatGPT, não podemos deixar de refletir e analisar a sua possível utilização no processo de ensino e aprendizagem inicial da programação. Em finais de 2022, começamos a ouvir falar com frequência num programa considerado como um grande passo evolutivo da inteligência artificial (IA), o ChatGPT. O ChatGPT é um modelo avançado de linguagem, desenvolvido pela *OpenAI*, baseado numa arquitetura de *Generative Pre-trained Transformer* (GPT), com a capacidade de compreensão e criação de respostas de texto semelhantes às capacidades humanas. Importa compreender de que se trata a IA Generativa, pois ajuda a gerir as expectativas sobre o que ela pode realizar. Segundo [131], a IA Generativa é a produção de novos conteúdos, de qualquer tipo para apoiar uma tarefa, através de uma modelação generativa, ou seja, partindo de recursos existentes. É importante compreender esta técnica da IA generativa não é uma solução mágica, mas sim um procedimento de treino de um modelo de dados de input.

Basicamente, o ChatGPT é um chat com uma interface bastante simples, que responde a perguntas por escrito, sobre os mais diversos temas. As suas fontes de dados são a informação existente na *Web* [132–134].

Estamos a viver um momento único, com reações de muito entusiasmo e curiosidade por uns, e com muitos receios e dúvidas por outros [135]. Especialmente na área do ensino, onde uns veem oportunidades, como por exemplo a criação de conteúdos para facilitar a aprendizagem, a facilidade de acesso à informação pretendida, um assistente virtual na aprendizagem disponível 24 horas por dia, 7

dias por semana. Outros veem problemas, como o uso desonesto da ferramenta, uma aprendizagem rápida e superficial, respostas incorretas, possibilidade de substituir os professores e dependência tecnológica [133, 136]. Neste momento, é importante que todos os elementos do ensino e aprendizagem conheçam e utilizem a tecnologia existente e tirar partido dela nas suas atividades.

Em [137], é apresentada uma reflexão efetuada por professores da área das ciências da computação, sobre a integração do ChatGPT e respetivas mudanças nos métodos de ensino. Segundo a sua autora, os professores do estudo não questionam a integração do ChatGPT no ensino, mas como deve ser integrado. Abordaram o ChatGPT como uma ferramenta capaz de melhorar as competências dos alunos, através da capacidade em formular questões com precisão e da capacidade de analisar da qualidade e fiabilidade das suas respostas.

Ao longo dos últimos anos, o nosso trabalho tem como foco principal as dificuldades de ensino e aprendizagem inicial da programação e como minimizar esses problemas. No trabalho desenvolvido, do qual este documento é o seu reflexo, colocamos em prática um conjunto de atividades, que consideramos as mais importantes e que melhor caracterizam os problemas e dificuldades dos alunos no seu percurso de aprendizagem inicial da programação. E, delineamos uma estratégia que permite o acompanhamento contínuo e efetivo dos alunos em todo o seu processo de aprendizagem. Em face do acontecimento recente, o aparecimento do ChatGPT, somos obrigados a utilizar e experimentar, de forma a conhecer as suas potencialidades ou debilidades, na resolução das atividades propostas. Se não o fizermos, os alunos vão fazer com toda a certeza.

O ChatGPT é uma novidade na sua forma de utilização simples e nível de conversação. No entanto, no que diz respeito a assistentes no auxílio à programação, que utilizam AI, existem ferramentas disponíveis há algum tempo [138]. Os assistentes de IA de codificação, são ferramentas que ajudam o programador a codificar de uma forma rápida e eficiente. Podem, ainda, criar e sugerir código, fazer revisão de código e sugestões de manuais. Algumas dessas ferramentas são: *OpenAI Codex*, *GitHub Copilot*, *Tabnine*, *AlphaCode*, *CodeT5*, *Polycoder* e *DeepCode*. Talvez as mais conhecidas, no entanto existem outros assistentes de IA de codificação [139, 140]. No trabalho descrito em [141], são apresentados os resultados sobre o desempenho do *Open AI Codex* na resolução de exame de CS2, estruturas de dados e algoritmos. Ficou demonstrado que os resultados do código gerado pelo *Codex*, supera o código criado pela maioria dos alunos.

No artigo apresentado em [142], são analisados os novos desafios e oportunidades proporcionados pela IA generativa para o ensino e aprendizagem, na área das ciências da computação. Salientando a necessidade de atualização das estratégias pedagógicas que se concentrem em novos conjuntos de competências. E ainda, evidenciando as novas abordagens pedagógicas para ensinar aos alunos como aproveitar a IA generativa, destacando a decomposição estratégica de problemas e a importância da precisão ao especificar tarefas de programação para sistemas de IA. De salientar ainda, segundo [142],

que os modelos de IA generativa são muito eficazes em gerar soluções para os problemas típicos do ensino inicial da programação, o que levanta graves preocupações sobre o excesso de confiança e uso indevido dos alunos. Por outro lado, as ferramentas baseadas em IA permitem gerar de uma forma muito eficiente recursos educacionais, como exercícios de programação.

Quando a tecnologia começa a fazer melhor que um humano, não adianta concorrer com ela. O importante passa a ser saber usar a tecnologia e tirar o maior proveito da sua utilização. Neste contexto, é importante saber perguntar. Para saber perguntar temos de perceber minimamente do assunto e pensar de uma forma crítica. Pelo que, num contexto de ensino e aprendizagem inicial da programação, deve ter uma utilização cuidada. Uma vez que o conhecimento do aluno e maturidade na área é ainda muito frágil, podem existir erros, faltas de rigor e dependência da tecnologia, que adquiridos nesta fase inicial de aprendizagem, podem tornar-se em graves problemas no futuro.

Não podemos concluir este trabalho sem utilizar o ChatGPT como ferramenta de auxílio ao ensino e aprendizagem inicial da programação, e ponderar a sua utilização num futuro próximo. Neste contexto, com a utilização do ChatGPT resolvemos algumas das atividades sugeridas no nosso plano de atividades, do curso de introdução à programação. Analisamos as respostas, enumeramos um conjunto de pontos forte e fracos da utilização do ChatGPT, como ferramenta de ensino e aprendizagem inicial da programação. Apresentamos no anexo A o detalhe da utilização do ChatGPT na resolução das atividades propostas.

Em conclusão, podemos afirmar que o ChatGPT é uma boa ajuda para o ensino e aprendizagem inicial da programação. E, deve ser incluído no processo de ensino o mais rapidamente possível. Pelo facto de que os alunos já o utilizam sem critério, apenas com o intuito de facilitar o seu trabalho. O ChatGPT pode ajudar numa ampla gama de exercícios frequentemente utilizados no ensino inicial de programação, tais como: na identificação e correção de nomes e variáveis, tipos de dados, instruções entrada/saída, identificação de erros de um programa e também na construção de programas partindo de um enunciado. No entanto, o espírito crítico e a análise cuidadosa das respostas devem estar sempre presentes, pois verificamos que existem algumas alucinações por parte do ChatGPT. Nas atividades de *Parson Problems*, as respostas produzidas contêm diversas imprecisões, o que compromete sua utilização. No que diz respeito, à solicitação de atividades e exemplos de aplicação de conhecimentos sobre os mais diversos tópicos de aprendizagem da iniciação à programação, o ChatGPT, produz excelentes resultados.

5.3 Publicações

Ao longo dos anos de desenvolvimento deste trabalho foram efetuadas publicações com o objetivo de divulgação do nosso trabalho, apresentação de resultados e oportunidade de discussão com a comu-

nidade académica sobre o trabalho realizado. As publicações realizadas no âmbito deste trabalho são as seguintes:

Publicações em Revistas

- Figueiredo, J. and García-Peñalvo, F.J. 2022. Design science research applied to difficulties of teaching and learning initial programming. *Universal Access in the Information Society*. (2022). DOI:<https://doi.org/10.1007/s10209-022-00941-4>.

Factor de impacto: 2.4 (2022)

Capítulos de Livros

- Figueiredo, J., García-Peñalvo, F.J. (2023). The Way for Learning Initial Programming: Activities, Feedback, and Predict Model. In: García-Peñalvo, F.J., García-Holgado, A. (eds) *Proceedings TEEM 2022: Tenth International Conference on Technological Ecosystems for Enhancing Multiculturality*. TEEM 2022. *Lecture Notes in Educational Technology*. Springer, Singapore. https://doi.org/10.1007/978-981-99-0942-1_3

Publicações em conferências

- José Figueiredo, Natália Gomes, and Francisco José García-Peñalvo. 2016. Ne-course for learning programming. In *Proceedings of the Fourth International Conference on Technological Ecosystems for Enhancing Multiculturality (TEEM '16)*. Association for Computing Machinery, New York, NY, USA, 549–553. <https://doi.org/10.1145/3012430.3012572>
- José Figueiredo and Francisco J. García-Peñalvo. 2017. Improving Computational Thinking Using Follow and Give Instructions. In *Proceedings of the 5th International Conference on Technological Ecosystems for Enhancing Multiculturality (TEEM 2017)*. Association for Computing Machinery, New York, NY, USA, Article 3, 1–7. <https://doi.org/10.1145/3144826.3145351>
- J. Quitério Figueiredo, “How to improve computational thinking: a case study,” in *Education in The Knowledge Society (EKS)*, ser. 4, vol. 18, no. ISSN: 2444-8729, 2017.
- José Figueiredo and Francisco José García-Peñalvo. 2018. Building Skills in Introductory Programming. In *Proceedings of the Sixth International Conference on Technological Ecosystems for Enhancing Multiculturality (TEEM'18)*. Association for Computing Machinery, New York, NY, USA, 46–50. <https://doi.org/10.1145/3284179.3284190>
- José Figueiredo and Francisco José García-Peñalvo. 2019. Teaching and learning strategies of programming for university courses. In *Proceedings of the Seventh International Conference on*

Technological Ecosystems for Enhancing Multiculturality (TEEM'19). Association for Computing Machinery, New York, NY, USA, 1020–1027. <https://doi.org/10.1145/3362789.3362926>

- José Figueiredo, Noel Lopes, and Francisco José García-Peñalvo. 2019. Predicting Student Failure in an Introductory Programming Course with Multiple Back-Propagation. In Proceedings of the Seventh International Conference on Technological Ecosystems for Enhancing Multiculturality (TEEM'19). Association for Computing Machinery, New York, NY, USA, 44–49. <https://doi.org/10.1145/3362789.3362925>
- J. Figueiredo and F. J. García-Peñalvo, "Increasing student motivation in computer programming with gamification," 2020 IEEE Global Engineering Education Conference (EDUCON), Porto, Portugal, 2020, pp. 997-1000, doi: 10.1109/EDUCON45650.2020.9125283.
- José Figueiredo and Francisco José García-Peñalvo. 2021. Intelligent Tutoring Systems approach to Introductory Programming Courses. In Eighth International Conference on Technological Ecosystems for Enhancing Multiculturality (TEEM'20). Association for Computing Machinery, New York, NY, USA, 34–39. <https://doi.org/10.1145/3434780.3436614>
- J. Figueiredo and F. García-Peñalvo, "Teaching and Learning Tools for Introductory Programming in University Courses," 2021 International Symposium on Computers in Education (SIIE), Malaga, Spain, 2021, pp. 1-6, doi: 10.1109/SIIE53363.2021.9583623.
- José Figueiredo and Francisco García-Peñalvo. 2021. A Tool Help for Introductory Programming Courses. In Ninth International Conference on Technological Ecosystems for Enhancing Multiculturality (TEEM'21) (TEEM'21). Association for Computing Machinery, New York, NY, USA, 18–24. <https://doi.org/10.1145/3486011.3486413>
- José Figueiredo and Francisco García-Peñalvo. 2021. Teaching and Learning Strategies for Introductory Programming in University Courses. In Ninth International Conference on Technological Ecosystems for Enhancing Multiculturality (TEEM'21) (TEEM'21). Association for Computing Machinery, New York, NY, USA, 746–751. <https://doi.org/10.1145/3486011.3486540>
- Figueiredo, J., García-Peñalvo, F.J. (2022). HProgramming: a Helping Tool for Teach and Learn Programming. In: Primo, T.T., Gomes, A.S., Moreira, F., Collazos, C. (eds) Advanced Virtual Environments and Education. WAVE 2021. Communications in Computer and Information Science, vol 1425. Springer, Cham. https://doi.org/10.1007/978-3-031-07018-1_6
- J. Figueiredo and F. J. García-Peñalvo, "Strategies to increase success in learning programming," 2022 International Symposium on Computers in Education (SIIE), Coimbra, Portugal, 2022, pp. 1-6, doi: 10.1109/SIIE56031.2022.9982358.

- José Figueiredo and Francisco García-Peñalvo. 2023. ChatGPT in the teaching and learning initial programming: a case study. Conference on Technological Ecosystems for Enhancing Multiculturality (TEEM'23). Association for Computing Machinery, New York, NY, USA. pp-pp.

Bibliografia

- [1] A. J. Jaeger, “What does the punched holes task measure?” Ph.D. dissertation, University of Illinois at Chicago, 2015.
- [2] R. Lister, E. S. Adams, S. Fitzgerald, W. Fone, J. Hamer, M. Lindholm, R. McCartney, J. E. Moström, K. Sanders, O. Seppälä, B. Simon, and L. Thomas, “A multi-national study of reading and tracing skills in novice programmers,” *SIGCSE Bull.*, vol. 36, no. 4, p. 119–150, jun 2004. [Online]. Available: <https://doi.org/10.1145/1041624.1041673>
- [3] S. Dekhane, X. Xu, and M. Tsoi, “Mobile app development to increase student engagement and problem solving skills,” *Journal of Information Systems Education*, vol. 24, pp. 299–308, 01 2013.
- [4] M. McCracken, V. Almstrum, D. Diaz, M. Guzdial, D. Hagan, Y. B.-D. Kolikant, C. Laxer, L. Thomas, I. Utting, and T. Wilusz, “A multi-national, multi-institutional study of assessment of programming skills of first-year cs students,” *SIGCSE Bull.*, vol. 33, no. 4, p. 125–180, dec 2001. [Online]. Available: <https://doi.org/10.1145/572139.572181>
- [5] E. Lahtinen, K. Ala-Mutka, and H.-M. Järvinen, “A study of the difficulties of novice programmers,” *SIGCSE Bull.*, vol. 37, no. 3, p. 14–18, jun 2005. [Online]. Available: <https://doi.org/10.1145/1151954.1067453>
- [6] D. Sleeman, “The challenges of teaching computer programming,” *Commun. ACM*, vol. 29, no. 9, p. 840–841, sep 1986. [Online]. Available: <https://doi.org/10.1145/6592.214913>
- [7] A. Robins, J. Rountree, and N. Rountree, “Learning and teaching programming: A review and discussion,” *Computer Science Education*, vol. 13, no. 2, pp. 137–172, jun 2003. [Online]. Available: <https://doi.org/10.1076/csed.13.2.137.14200>
- [8] U. Omer, S. Farooq, and A. Abid, “Introductory programming course: review and future implications,” *PeerJ. Computer science*, vol. 7, p. e647, 07 2021.
- [9] H. Zhao, M. Li, T. Lin, R. Wang, and Z. Wu, “Prolog2vec: Detecting novices’ difficulty in programming using deep learning,” *IEEE Access*, vol. 9, pp. 53 243–53 254, 2021.

- [10] J. McBroom, B. Paassen, B. Jeffries, I. Koprinska, and K. Yacef, "Progress networks as a tool for analysing student programming difficulties," in *Proceedings of the 23rd Australasian Computing Education Conference*, ser. ACE '21. New York, NY, USA: Association for Computing Machinery, feb 2021, p. 158–167. [Online]. Available: <https://doi.org/10.1145/3441636.3442366>
- [11] S. López-Pernas, M. Saqr, and O. Viberg, "Putting it all together: Combining learning analytics methods and data sources to understand students' approaches to learning programming," *Sustainability*, vol. 13, no. 9, 2021. [Online]. Available: <https://www.mdpi.com/2071-1050/13/9/4825>
- [12] D. Song, H. Hong, and E. Y. Oh, "Applying computational analysis of novice learners' computer programming patterns to reveal self-regulated learning, computational thinking, and learning performance," *Computers in Human Behavior*, vol. 120, p. 106746, 2021. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0747563221000686>
- [13] F. J. García Peñalvo, "Entrepreneurial and problem solving skills in software engineers," *Journal of Information Technology Research* 8, 3, iv-vi., 2015.
- [14] F. Llorens-Largo, F. J. García-Peñalvo, X. Molero Prieto, and E. Vendrell Vidal, "La enseñanza de la informática, la programación y el pensamiento computacional en los estudios preuniversitarios," *Education in the Knowledge Society (EKS)*, vol. 18, no. 2, pp. 7–17, aug 2017. [Online]. Available: <https://revistas.usal.es/index.php/eks/article/view/eks2017182717>
- [15] B. J. Hirsch, "Wanted: Soft skills for today's jobs," *Phi Delta Kappan*, vol. 98, no. 5, pp. 12–17, 2017. [Online]. Available: <https://doi.org/10.1177/0031721717690359>
- [16] F. Michavila, J. Martínez, M. Martín-González, F. García-Peñalvo, and J. Cruz-Benito, "Empleabilidad de los titulados universitarios en España. Proyecto OEEU," *Education in the Knowledge Society (EKS)*, vol. 19, p. 21, 2018.
- [17] F. J. García-Peñalvo and A. J. Mendes, "Exploring the computational thinking effects in pre-university education," *Computers in Human Behavior*, vol. 80, pp. 407–411, Mar. 2018.
- [18] J. Bennedsen, "Teaching and learning introductory programming : a model-based approach," phdthesis, University of Oslo, 2008. [Online]. Available: <http://hdl.handle.net/10852/9962>
- [19] A. d. J. Gomes, "Dificuldades de aprendizagem de programação de computadores : contributos para a sua compreensão e resolução," phdthesis, Faculdade de Ciências e Tecnologia da Universidade de Coimbra, 2010. [Online]. Available: <https://estudogeral.sib.uc.pt/jspui/handle/10316/14586>

- [20] J.-M. Hoc and A. Nguyen-Xuan, "Chapter 2.3 - language semantics, mental models and analogy," in *Psychology of Programming*, J.-M. Hoc, T. Green, R. Samurçay, and D. Gilmore, Eds. London: Academic Press, 1990, pp. 139–156. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/B9780123507723500148>
- [21] T. Jenkins, "On the difficulty of learning to program," in *Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences*, vol. 4, no. 2002. Citeseer, 2002, pp. 53–58.
- [22] A. Gomes and A. Mendes, "A teacher's view about introductory programming teaching and learning: Difficulties, strategies and motivations," in *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*, 2014, pp. 1–8.
- [23] C. Watson and F. W. Li, "Failure rates in introductory programming revisited," in *Proceedings of the 2014 conference on Innovation & technology in computer science education*, 2014, pp. 39–44.
- [24] A. Luxton-Reilly, Simon, I. Alblawi, B. A. Becker, M. Giannakos, A. N. Kumar, L. Ott, J. Paterson, M. J. Scott, J. Sheard, and C. Szabo, "Introductory programming: A systematic literature review," in *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*, ser. ITiCSE 2018 Companion. New York, NY, USA: Association for Computing Machinery, jul 2018, p. 55–106. [Online]. Available: <https://doi.org/10.1145/3293881.3295779>
- [25] S. Fincher, B. Baker, I. Box, Q. Cutts, M. de Raadt, P. Haden, J. Hamer, M. Hamilton, R. Lister, and M. Petre, "Programmed to succeed?: A multi-national, multi-institutional study of introductory programming courses," University of Kent, Canterbury, United Kingdom, Technical report, April 2005. [Online]. Available: <https://kar.kent.ac.uk/14335/>
- [26] S. Shuhidan, M. Hamilton, and D. D'Souza, "A taxonomic study of novice programming summative assessment," *Conferences in Research and Practice in Information Technology Series*, vol. 95, pp. 147–156, 2009.
- [27] J. Bennedsen and M. E. Caspersen, "Failure rates in introductory programming: 12 years later," *ACM Inroads*, vol. 10, no. 2, p. 30–36, apr 2019. [Online]. Available: <https://doi.org/10.1145/3324888>
- [28] M. Ahmadzadeh, D. Elliman, and C. Higgins, "An analysis of patterns of debugging among novice computer science students," *SIGCSE Bull.*, vol. 37, no. 3, p. 84–88, jun 2005. [Online]. Available: <https://doi.org/10.1145/1151954.1067472>
- [29] J. Wing, "Computational thinking," *Communications of the ACM*, vol. 49, pp. 33–35, 03 2006.

- [30] —, “Computational thinking and thinking about computing,” *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, vol. 366, no. 1881, pp. 3717–25, 11 2008.
- [31] F. J. García-Peñalvo and J. Cruz-Benito, “Computational thinking in pre-university education,” in *Proceedings of the Fourth International Conference on Technological Ecosystems for Enhancing Multiculturality*, ser. TEEM '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 13–17. [Online]. Available: <https://doi.org/10.1145/3012430.3012490>
- [32] A. Gomes and A. Mendes, “Learning to program - difficulties and solutions,” in *International Conference on Engineering Education – ICEE 2007*, 01 2007, pp. 283–287.
- [33] A. Luxton-Reilly, “Learning to program is easy,” in *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*, ser. ITiCSE '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 284–289. [Online]. Available: <https://doi.org/10.1145/2899415.2899432>
- [34] A. Hellas, P. Ihanola, A. Petersen, V. V. Ajanovski, M. Gutica, T. Hynninen, A. Knutas, J. Leinonen, C. Messom, and S. N. Liao, “Predicting academic performance: A systematic literature review,” in *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*, ser. ITiCSE 2018 Companion. New York, NY, USA: Association for Computing Machinery, 2018, p. 175–199. [Online]. Available: <https://doi.org/10.1145/3293881.3295783>
- [35] S. N. Liao, D. Zingaro, K. Thai, C. Alvarado, W. G. Griswold, and L. Porter, “A robust machine learning technique to predict low-performing students,” *ACM Trans. Comput. Educ.*, vol. 19, no. 3, jan 2019. [Online]. Available: <https://doi.org/10.1145/3277569>
- [36] K. Quille and S. Bergin, “Cs1: how will they do? how can we help? a decade of research and practice,” *Computer Science Education*, vol. 29, no. 2-3, pp. 254–282, 2019. [Online]. Available: <https://doi.org/10.1080/08993408.2019.1612679>
- [37] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, “A design science research methodology for information systems research,” *Journal of Management Information Systems*, vol. 24, no. 3, pp. 45–77, 2007. [Online]. Available: <https://doi.org/10.2753/MIS0742-1222240302>
- [38] J. Choi, D. L. Nazareth, and H. K. Jain, “Implementing service-oriented architecture in organizations,” *Journal of Management Information Systems*, vol. 26, no. 4, pp. 253–286, apr 2010. [Online]. Available: <https://doi.org/10.2753/MIS0742-1222260409>
- [39] G. L. Geerts, “A design science research methodology and its application to accounting information systems research,” *International Journal of Accounting Information Systems*, vol. 12,

- no. 2, pp. 142–151, jun 2011, special Issue on Methodologies in AIS Research. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1467089511000200>
- [40] S. T. March and G. F. Smith, “Design and natural science research on information technology,” *Decision Support Systems*, vol. 15, no. 4, pp. 251–266, 1995. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/0167923694000412>
- [41] A. R. Hevner, S. T. March, J. Park, and S. Ram, “Design science in information systems research,” *MIS Q.*, vol. 28, pp. 75–105, 2004.
- [42] V. Allan, V. Barr, D. Brylow, and S. Hambrusch, “Computational thinking in high school courses,” Mar. 2010.
- [43] F. J. García-Peñalvo, D. Reimann, M. Tuul, A. Rees, and I. Jormanainen, “An overview of the most relevant literature on coding and computational thinking with emphasis on the relevant issues for teachers,” 2016.
- [44] V. J. Shute, C. Sun, and J. Asbell-Clarke, “Demystifying computational thinking,” *Educational Research Review*, vol. 22, pp. 142–158, Nov. 2017.
- [45] P. J. Denning, “Remaining trouble spots with computational thinking,” *Communications of the ACM*, vol. 60, no. 6, pp. 33–39, May 2017.
- [46] J. Quitério Figueiredo, “How to improve computational thinking: a case study,” in *Education in The Knowledge Society (EKS)*, ser. 4, vol. 18, no. ISSN: 2444-8729, 2017.
- [47] J. Fraillon, M. Rožman, D. Duckworth, S. Dexter, J. Bundsgaard, and Schulz, *IEA international computer and information literacy study 2023 - Assessment Framework*, J. Fraillon and M. Rožman, Eds., 2023.
- [48] L. Major, T. Kyriacou, and O. Brereton, “Systematic literature review: teaching novices programming using robots,” *IET Software*, vol. 6, no. 6, p. 502, 2012.
- [49] S. Y. Lye and J. H. L. Koh, “Review on teaching and learning of computational thinking through programming: What is next for k-12?” *Computers in Human Behavior*, vol. 41, pp. 51–61, Dec. 2014. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0747563214004634>
- [50] A. S. Liu, J. Newsom, C. D. Schunn, and R. Shoop, “Students learn programming faster through robotic simulation.” *Tech Directions*, vol. 72, pp. 16–19, 2013. [Online]. Available: <https://api.semanticscholar.org/CorpusID:61357214>

- [51] E. B. Witherspoon, R. M. Higashi, C. D. Schunn, E. C. Baehr, and R. Shoop, “Developing computational thinking through a virtual robotics programming curriculum,” *ACM Trans. Comput. Educ.*, vol. 18, no. 1, pp. 1–20, oct 2017. [Online]. Available: <https://doi.org/10.1145/3104982>
- [52] E. de Recursos e Tecnologias Educativas (ERTE) é uma equipa multidisciplinar da Direção-Geral da Educação, “Programação e robótica no ensino básico,” <https://erte.dge.mec.pt/programacao-e-robotica-no-ensino-basico-0>, accessed: 2024-01-14.
- [53] S. Fincher, A. Robins, B. Baker, B. Ilona, Q. Cutts, M. de Raadt, P. Haden, J. Hamer, M. Hamilton, R. Lister, M. Petre, K. Sutton, D. Tolhurst, and J. Tutty, “Predictors of success in a first programming course,” *Proceedings of the 8th Australasian Computing Education Conference (ACE 2006)*, no. pp. 189-196, p. 8, 2006. [Online]. Available: <http://crpit.com/confpapers/CRPITV52Simon2.pdf>
- [54] S. Cooper, K. Wang, M. Israni, and S. Sorby, “Spatial skills training in introductory computing,” in *Proceedings of the Eleventh Annual International Conference on International Computing Education Research*, ser. ICER ’15. New York, NY, USA: Association for Computing Machinery, 2015, p. 13–20. [Online]. Available: <https://doi.org/10.1145/2787622.2787728>
- [55] P. Bayman and R. E. Mayer, “A diagnosis of beginning programmers’ misconceptions of basic programming statements,” *Commun. ACM*, vol. 26, no. 9, p. 677–679, sep 1983. [Online]. Available: <https://doi.org/10.1145/358172.358408>
- [56] R. D. Pea, “Language-independent conceptual “bugs” in novice programming,” *Journal of Educational Computing Research*, vol. 2, no. 1, pp. 25–36, 1986. [Online]. Available: <https://doi.org/10.2190/689T-1R2A-X4W4-29J2>
- [57] Y. Qian and J. Lehman, “Students’ misconceptions and other difficulties in introductory programming: A literature review,” *ACM Trans. Comput. Educ.*, vol. 18, no. 1, oct 2017. [Online]. Available: <https://doi.org/10.1145/3077618>
- [58] W. J. MCNAMARA and J. L. HUGHES, “A REVIEW OF RESEARCH ON THE SELECTION OF COMPUTER PROGRAMMERS,” *Personnel Psychology*, vol. 14, no. 1, pp. 39–51, mar 1961.
- [59] C. A. Alsbaugh, “Identification of some components of computer programming aptitude,” *Journal for Research in Mathematics Education*, vol. 3, no. 2, pp. 89–98, mar 1972.
- [60] G. E. Evans and M. G. Simkin, “What best predicts computer proficiency?” *Commun. ACM*, vol. 32, no. 11, p. 1322–1327, nov 1989. [Online]. Available: <https://doi.org/10.1145/68814.68817>
- [61] S. Hooshangi, M. Ellis, and S. H. Edwards, “Factors influencing student performance and persistence in cs2,” in *Proceedings of the 53rd ACM Technical Symposium on Computer Science*

- Education - Volume 1*, ser. SIGCSE 2022. New York, NY, USA: Association for Computing Machinery, 2022, p. 286–292. [Online]. Available: <https://doi.org/10.1145/3478431.3499272>
- [62] J. Prather, B. A. Becker, M. Craig, P. Denny, D. Loksa, and L. Margulieux, “What do we think we think we are doing? metacognition and self-regulation in programming,” in *Proceedings of the 2020 ACM Conference on International Computing Education Research*, ser. ICER '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 2–13. [Online]. Available: <https://doi.org/10.1145/3372782.3406263>
- [63] L. Silva, A. Mendes, A. Gomes, G. Fortes, C. T. Lam, and C. Chan, “Exploring the association between self-regulation of learning and programming learning: A multinational investigation,” in *2021 IEEE Frontiers in Education Conference (FIE)*, 2021, pp. 1–8.
- [64] L. G. L. Freire, “Auto-regulação da aprendizagem,” *Ciências & Cognição*, vol. 14, no. 2, 2009.
- [65] A. Silva, A. M. Veiga Simão, and I. Sá, “A auto-regulação da aprendizagem: estudos teóricos e empíricos,” *InterMeio*, vol. 10, pp. 58–74, 01 2004.
- [66] B. C. Wilson and S. Shrock, “Contributing to success in an introductory computer science course: A study of twelve factors,” *SIGCSE Bull.*, vol. 33, no. 1, p. 184–188, feb 2001. [Online]. Available: <https://doi.org/10.1145/366413.364581>
- [67] K. Quille and S. Bergin, “Programming: Predicting student success early in cs1. a re-validation and replication study,” in *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*, ser. ITiCSE 2018. New York, NY, USA: Association for Computing Machinery, 2018, p. 15–20. [Online]. Available: <https://doi.org/10.1145/3197091.3197101>
- [68] A. Bandura, W. H. Freeman, and R. Lightsey, “Self-efficacy: The exercise of control,” 1999.
- [69] D. Gries, “What should we teach in an introductory programming course?” *SIGCSE Bull.*, vol. 6, no. 1, p. 81–89, jan 1974. [Online]. Available: <https://doi.org/10.1145/953057.810447>
- [70] D. Loksa, A. J. Ko, W. Jernigan, A. Oleson, C. J. Mendez, and M. M. Burnett, “Programming, problem solving, and self-awareness: Effects of explicit guidance,” in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, ser. CHI '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 1449–1461. [Online]. Available: <https://doi.org/10.1145/2858036.2858252>
- [71] J. Bennedsen and M. E. Caspersen, “Revealing the programming process,” *SIGCSE Bull.*, vol. 37, no. 1, p. 186–190, feb 2005. [Online]. Available: <https://doi.org/10.1145/1047124.1047413>

- [72] M. J. Rubin, "The effectiveness of live-coding to teach introductory programming," in *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 651–656. [Online]. Available: <https://doi.org/10.1145/2445196.2445388>
- [73] E. Soloway, "Learning to program = learning to construct mechanisms and explanations," *Commun. ACM*, vol. 29, no. 9, p. 850–858, sep 1986. [Online]. Available: <https://doi.org/10.1145/6592.6594>
- [74] A. Altadmri and N. C. Brown, "37 million compilations: Investigating novice programming mistakes in large-scale student data," in *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '15. New York, NY, USA: Association for Computing Machinery, feb 2015, p. 522–527. [Online]. Available: <https://doi.org/10.1145/2676723.2677258>
- [75] M. C. Jadud, "Methods and tools for exploring novice compilation behaviour," in *Proceedings of the Second International Workshop on Computing Education Research*, ser. ICER '06. New York, NY, USA: Association for Computing Machinery, 2006, p. 73–84. [Online]. Available: <https://doi.org/10.1145/1151588.1151600>
- [76] E. S. Tabanao, M. M. T. Rodrigo, and M. C. Jadud, "Predicting at-risk novice java programmers through the analysis of online protocols," in *Proceedings of the Seventh International Workshop on Computing Education Research*, ser. ICER '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 85–92. [Online]. Available: <https://doi.org/10.1145/2016911.2016930>
- [77] P. Denny, A. Luxton-Reilly, and E. Tempero, "All syntax errors are not equal," in *Proceedings of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education*, ser. ITiCSE '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 75–80. [Online]. Available: <https://doi.org/10.1145/2325296.2325318>
- [78] T. Dy and M. M. Rodrigo, "A detector for non-literal java errors," in *Proceedings of the 10th Koli Calling International Conference on Computing Education Research*, ser. Koli Calling '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 118–122. [Online]. Available: <https://doi.org/10.1145/1930464.1930485>
- [79] B. A. Becker, "A new metric to quantify repeated compiler errors for novice programmers," in *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*, ser. ITiCSE '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 296–301. [Online]. Available: <https://doi.org/10.1145/2899415.2899463>

- [80] M. Hristova, A. Misra, M. Rutter, and R. Mercuri, "Identifying and correcting java programming errors for introductory computer science students," *SIGCSE Bull.*, vol. 35, no. 1, p. 153–156, jan 2003. [Online]. Available: <https://doi.org/10.1145/792548.611956>
- [81] D. Parsons and P. Haden, "Parson's programming puzzles: A fun and effective learning tool for first programming courses," in *Proceedings of the 8th Australasian Conference on Computing Education - Volume 52*, ser. ACE '06. AUS: Australian Computer Society, Inc., 2006, p. 157–163.
- [82] P. Denny, A. Luxton-Reilly, and B. Simon, "Evaluating a new exam question: Parsons problems," in *Proceedings of the Fourth International Workshop on Computing Education Research*, ser. ICER '08. New York, NY, USA: Association for Computing Machinery, 2008, p. 113–124. [Online]. Available: <https://doi.org/10.1145/1404520.1404532>
- [83] S. Deterding, "Gamification: Designing for motivation," *Interactions*, vol. 19, no. 4, p. 14–17, jul 2012. [Online]. Available: <https://doi.org/10.1145/2212877.2212883>
- [84] S. Deterding, D. Dixon, R. Khaled, and L. Nacke, "From game design elements to gamefulness: Defining "gamification"," in *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments*, ser. MindTrek '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 9–15. [Online]. Available: <https://doi.org/10.1145/2181037.2181040>
- [85] S. Deterding, M. Sicart, L. Nacke, K. O'Hara, and D. Dixon, "Gamification. using game-design elements in non-gaming contexts," in *CHI '11 Extended Abstracts on Human Factors in Computing Systems*, ser. CHI EA '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 2425–2428. [Online]. Available: <https://doi.org/10.1145/1979742.1979575>
- [86] M.-B. Ibáñez, A. Di-Serio, and C. Delgado-Kloos, "Gamification for engaging computer science students in learning activities: A case study," *IEEE Transactions on Learning Technologies*, vol. 7, no. 3, pp. 291–301, 2014.
- [87] R. Elshiekh and L. Butgerit, "Using Gamification to Teach Students Programming Concepts," *OA-Lib*, vol. 04, no. 08, pp. 1–7, 2017.
- [88] M. Piteira, C. J. Costa, and M. Aparicio, "Computer Programming Learning: How to Apply Gamification on Online Courses?" *Journal of Information Systems Engineering & Management*, vol. 3, no. 2, 2018.
- [89] R. Queirós, "Proud—a gamification framework based on programming exercises usage data," *Information*, vol. 10, no. 2, 2019. [Online]. Available: <https://www.mdpi.com/2078-2489/10/2/54>

- [90] M. Cunha, “Aplicação de técnicas de gamificação em sistemas educacionais,” 2021. [Online]. Available: <https://hdl.handle.net/1822/81327>
- [91] B. E. Wiggins, “An overview and study on the use of games, simulations, and gamification in higher education,” *International Journal of Game-Based Learning*, vol. 6, no. 1, pp. 18–29, Jan. 2016.
- [92] R. S. Alsawaier, “The effect of gamification on motivation and engagement,” *The International Journal of Information and Learning Technology*, vol. 35, no. 1, pp. 56–79, 2018.
- [93] H. Fischer, M. Heinz, L. Schlenker, and F. Follert, “Gamifying higher education. beyond badges, points and leaderboards,” in *Workshop Gemeinschaften in Neuen Medien (GeNeMe) 2016*. Dresden: TUDpress, 2016, pp. 93–104. [Online]. Available: <https://dl.gi.de/handle/20.500.12116/35002>
- [94] M. E. O. Rojas, K. Chiluiza, and M. Valcke, “Gamification in computer programming : effects on learning, engagement, self-efficacy and intrinsic motivation,” 2017. [Online]. Available: <https://api.semanticscholar.org/CorpusID:198599250>
- [95] R. Queirós, “Proud—a gamification framework based on programming exercises usage data,” *Information*, vol. 10, no. 2, 2019. [Online]. Available: <https://www.mdpi.com/2078-2489/10/2/54>
- [96] M. Ortiz-Rojas, K. Chiluiza, and M. Valcke, “Gamification through leaderboards: An empirical study in engineering education,” *Computer Applications in Engineering Education*, vol. 27, pp. 777 – 788, 2019. [Online]. Available: <https://api.semanticscholar.org/CorpusID:164983044>
- [97] M. I. Venter, “Gamification in stem programming courses: State of the art,” *2020 IEEE Global Engineering Education Conference (EDUCON)*, pp. 859–866, 2020. [Online]. Available: <https://api.semanticscholar.org/CorpusID:220261032>
- [98] E. Ratinho and C. Martins, “The role of gamified learning strategies in student’s motivation in high school and higher education: A systematic review,” *Heliyon*, vol. 9, no. 8, p. e19033, Aug. 2023.
- [99] J. Parkinson and Q. Cutts, “Investigating the relationship between spatial skills and computer science,” in *Proceedings of the 2018 ACM Conference on International Computing Education Research*, ser. ICER ’18. ACM, 08 2018, pp. 106–114.
- [100] R. Bockmon, S. Cooper, W. Koperski, J. Gratch, S. Sorby, and M. Dorodchi, “A cs1 spatial skills intervention and the impact on introductory programming abilities,” in *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, ser. SIGCSE ’20. New York, NY, USA: Association for Computing Machinery, 2020, p. 766–772. [Online]. Available: <https://doi.org/10.1145/3328778.3366829>

- [101] R. Bockmon, S. Cooper, J. Gratch, J. Zhang, and M. Dorodchi, "Can students' spatial skills predict their programming abilities?" in *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*, ser. ITiCSE '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 446–451. [Online]. Available: <https://doi.org/10.1145/3341525.3387380>
- [102] Z. Falomir, "Towards a qualitative descriptor for paper folding reasoning," in *In Proc. of the 29th International Workshop on Qualitative Reasoning.*, 2016. [Online]. Available: <https://api.semanticscholar.org/CorpusID:26140499>
- [103] A. Mühling, A. Ruf, and P. Hubwieser, "Design and first results of a psychometric test for measuring basic programming abilities," in *Proceedings of the Workshop in Primary and Secondary Computing Education*, ser. WiPSCE '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 2–10. [Online]. Available: <https://doi.org/10.1145/2818314.2818320>
- [104] B. J. Ericson, "Adaptive parsons problems with discourse rules," in *Proceedings of the Tenth Annual Conference on International Computing Education Research*, ser. ICER '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 145–146. [Online]. Available: <https://doi.org/10.1145/2632320.2632324>
- [105] B. B. Morrison, L. E. Margulieux, B. Ericson, and M. Guzdial, "Subgoals help students solve parsons problems," in *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, ser. SIGCSE '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 42–47. [Online]. Available: <https://doi.org/10.1145/2839509.2844617>
- [106] Y. Du, A. Luxton-Reilly, and P. Denny, "A review of research on parsons problems," in *Proceedings of the Twenty-Second Australasian Computing Education Conference*, ser. ACE'20. New York, NY, USA: Association for Computing Machinery, 2020, p. 195–202. [Online]. Available: <https://doi.org/10.1145/3373165.3373187>
- [107] J. Figueiredo, "HTProgramming - Help To Programming," <https://github.com/jfig21/HTProgramming.git>, 2022.
- [108] J. P. Munson and E. A. Schilling, "Analyzing novice programmers' response to compiler error messages," *J. Comput. Sci. Coll.*, vol. 31, no. 3, p. 53–61, jan 2016.
- [109] C. Watson, F. W. Li, and J. L. Godwin, "No tests required: Comparing traditional and dynamic predictors of programming success," in *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 469–474. [Online]. Available: <https://doi.org/10.1145/2538862.2538930>

- [110] S. P. Patil and A. C. Goje, "The effect of developments in student attributes on success in programming of management students," in *2009 International Conference on Education Technology and Computer*, 2009, pp. 191–193.
- [111] L. H. Werth, "Predicting student performance in a beginning computer science class," *SIGCSE Bull.*, vol. 18, no. 1, p. 138–143, feb 1986. [Online]. Available: <https://doi.org/10.1145/953055.5701>
- [112] D. Hagan and S. Markham, "Does it help to have some programming experience before beginning a computing degree program?" *SIGCSE Bull.*, vol. 32, no. 3, p. 25–28, jul 2000. [Online]. Available: <https://doi.org/10.1145/353519.343063>
- [113] E. Holden and E. Weeden, "The impact of prior experience in an information technology programming course sequence," in *Proceedings of the 4th Conference on Information Technology Curriculum*, ser. CITC4 '03. New York, NY, USA: Association for Computing Machinery, 2003, p. 41–46. [Online]. Available: <https://doi.org/10.1145/947121.947131>
- [114] T. R. Hostetler, "Predicting student success in an introductory programming course," *SIGCSE Bull.*, vol. 15, no. 3, p. 40–43, sep 1983. [Online]. Available: <https://doi.org/10.1145/382188.382571>
- [115] R. H. Nowaczyk, "Cognitive skills needed in computer programming," 1983.
- [116] J. Figueiredo, N. Lopes, and F. J. García-Peñalvo, "Predicting student failure in an introductory programming course with multiple back-propagation," in *Proceedings of the Seventh International Conference on Technological Ecosystems for Enhancing Multiculturality*, ser. TEEM'19. New York, NY, USA: Association for Computing Machinery, 2019, p. 44–49. [Online]. Available: <https://doi.org/10.1145/3362789.3362925>
- [117] N. Lopes and B. Ribeiro, "Hybrid learning in a multi-neural network architecture," in *IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No.01CH37222)*, vol. 4, 2001, pp. 2788–2793 vol.4.
- [118] "Encyclopedia of machine learning," 2010.
- [119] D. Chicco and G. Jurman, "The advantages of the matthews correlation coefficient (MCC) over f1 score and accuracy in binary classification evaluation," *BMC Genomics*, vol. 21, no. 1, jan 2020.
- [120] A. Tharwat, "Classification assessment methods," *Applied Computing and Informatics*, vol. 17, no. 1, pp. 168–192, jul 2020.
- [121] J. Figueiredo, N. Gomes, and F. J. García-Peñalvo, "Ne-course for learning programming," in *Proceedings of the Fourth International Conference on Technological Ecosystems for Enhancing*

- Multiculturalism*, ser. TEEM '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 549–553. [Online]. Available: <https://doi.org/10.1145/3012430.3012572>
- [122] J. Figueiredo and F. J. García-Peñalvo, “Improving computational thinking using follow and give instructions,” in *Proceedings of the 5th International Conference on Technological Ecosystems for Enhancing Multiculturalism*, ser. TEEM 2017. New York, NY, USA: Association for Computing Machinery, 2017. [Online]. Available: <https://doi.org/10.1145/3144826.3145351>
- [123] J. Figueiredo and F. García-Peñalvo, “Building skills in introductory programming,” in *Proceedings of the Sixth International Conference on Technological Ecosystems for Enhancing Multiculturalism*, ser. TEEM'18. New York, NY, USA: Association for Computing Machinery, 2018, p. 46–50. [Online]. Available: <https://doi.org/10.1145/3284179.3284190>
- [124] J. Figueiredo and F. J. García-Peñalvo, “Increasing student motivation in computer programming with gamification,” in *2020 IEEE Global Engineering Education Conference (EDUCON)*, 2020, pp. 997–1000.
- [125] J. Figueiredo and F. García-Peñalvo, “A tool help for introductory programming courses,” in *Ninth International Conference on Technological Ecosystems for Enhancing Multiculturalism (TEEM'21)*, ser. TEEM'21. New York, NY, USA: Association for Computing Machinery, oct 2021, p. 18–24. [Online]. Available: <https://doi.org/10.1145/3486011.3486413>
- [126] J. Figueiredo and F. J. García-Peñalvo, “Hprogramming: a helping tool for teach and learn programming,” in *Advanced Virtual Environments and Education*, T. T. Primo, A. S. Gomes, F. Moreira, and C. Collazos, Eds. Cham: Springer International Publishing, 2022, pp. 65–76.
- [127] J. Figueiredo and F. García-Peñalvo, “Teaching and learning strategies for introductory programming in university courses,” in *Ninth International Conference on Technological Ecosystems for Enhancing Multiculturalism (TEEM'21)*, ser. TEEM'21. New York, NY, USA: Association for Computing Machinery, 2021, p. 746–751. [Online]. Available: <https://doi.org/10.1145/3486011.3486540>
- [128] J. Figueiredo and F. García-Peñalvo, “Teaching and learning tools for introductory programming in university courses,” in *2021 International Symposium on Computers in Education (SIIE)*, 2021, pp. 1–6.
- [129] J. Figueiredo and F. J. García-Peñalvo, “Strategies to increase success in learning programming,” in *2022 International Symposium on Computers in Education (SIIE)*, 2022, pp. 1–6.
- [130] J. Figueiredo and F. J. García-Peñalvo, “Design science research applied to difficulties of teaching and learning initial programming,” *Universal Access in the Information Society*, nov 2022.

- [131] F. García-Peñalvo and A. Vázquez-Ingelmo, "What do we mean by genai? a systematic mapping of the evolution, trends, and techniques involved in generative ai," *International Journal of Interactive Multimedia and Artificial Intelligence*, vol. 8, no. 4, p. 7, 2023.
- [132] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," 2020.
- [133] F. J. García-Peñalvo, "La integración de la inteligencia artificial generativa en la práctica docente," Apr. 2023. [Online]. Available: <https://doi.org/10.5281/zenodo.7853091>
- [134] X. Zhai, "Chatgpt for next generation science learning," *XRDS*, vol. 29, no. 3, p. 42–46, apr 2023. [Online]. Available: <https://doi.org/10.1145/3589649>
- [135] B. Meyer, "What do chatgpt and ai-based automatic program generation mean for the future of software." [Online]. Available: <https://bit.ly/3LyAJLj>
- [136] A. Bozkurt, J. Xiao, S. Lambert, A. Pazurek, H. Crompton, S. Koseoglu, R. Farrow, M. Bond, C. Nerantzi, S. Honeychurch, M. Bali, J. Dron, K. Mir, B. Stewart, E. Costello, J. Mason, C. M. Stracke, E. Romero-Hall, A. Koutropoulos, C. M. Toquero, L. Singh, A. Tlili, K. Lee, M. Nichols, E. Ossiannilsson, M. Brown, V. Irvine, J. E. Raffaghelli, G. Santos-Hermosa, O. Farrell, T. Adam, Y. L. Thong, S. Sani-Bozkurt, R. C. Sharma, S. Hrastinski, and P. Jandrić, "Speculative futures on chatgpt and generative artificial intelligence (ai): A collective reflection from the educational landscape," 2023.
- [137] O. Hazzan, "Chatgpt in computer science education." [Online]. Available: <https://bit.ly/3WYTxxpv>
- [138] M. Kazemitabaar, J. Chow, C. K. T. Ma, B. J. Ericson, D. Weintrop, and T. Grossman, "Studying the effect of ai code generators on supporting novice learners in introductory programming," in *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, ser. CHI '23. New York, NY, USA: Association for Computing Machinery, 2023. [Online]. Available: <https://doi.org/10.1145/3544548.3580919>
- [139] T. Phillips, "Ai code tools: The ultimate guide in 2023," url = <https://codesubmit.io/blog/ai-code-tools/>, accessed: 2023-05-24.
- [140] SourceForge, "Ai coding assistants," url = <https://sourceforge.net/software/ai-coding-assistants/>, accessed: 2023-05-24.

- [141] J. Finnie-Ansley, P. Denny, A. Luxton-Reilly, E. A. Santos, J. Prather, and B. A. Becker, "My ai wants to know if this will be on the exam: Testing openai's codex on cs2 programming exercises," in *Proceedings of the 25th Australasian Computing Education Conference*, ser. ACE '23. New York, NY, USA: Association for Computing Machinery, Jan. 2023, p. 97–104. [Online]. Available: <https://doi.org/10.1145/3576123.3576134>
- [142] P. Denny, J. Prather, B. A. Becker, J. Finnie-Ansley, A. Hellas, J. Leinonen, A. Luxton-Reilly, B. N. Reeves, E. A. Santos, and S. Sarsa, "Computing education in the era of generative ai," *Commun. ACM*, vol. 67, no. 2, p. 56–67, jan 2024. [Online]. Available: <https://doi.org/10.1145/3624720>



ChatGPT no ensino e aprendizagem inicial da programação

Neste anexo, apresentamos e analisamos a utilização do ChatGPT como ferramenta de auxílio ao ensino e aprendizagem inicial da programação. Neste contexto, questionamos o ChatGPT na resolução das atividades sugeridas no nosso plano de atividades, do curso de introdução à programação.

A.1 ChatGPT na avaliação de identificadores

O principal objetivo deste trabalho é analisar de que forma podemos usar o ChatGPT, no auxílio ao ensino e aprendizagem inicial da programação. Pretendemos analisar do ponto de vista dos alunos e, naturalmente, do ponto de vista do professor. Com referido anteriormente, este trabalho insere-se numa investigação mais abrangente de análise de estratégias e metodologias de ensino e aprendizagem inicial da programação. Como minimizar os seus problemas e dificuldades, num contexto muito particular, com alunos de características peculiares, que é a unidade curricular de introdução à programação,

do primeiro ano e primeiro semestre, do curso de engenharia informática, do Instituto Politécnico da Guarda. Neste curso de introdução aos conceitos da programação, é usada a linguagem C. É aplicado um sistema de ensino e aprendizagem com aulas presenciais, e aulas do tipo prático, teórico-prático e tutorial, num total de 6 horas semanais, ao longo das quinze semanas do semestre.

De que forma podemos usar o ChatGPT, no auxílio ao ensino e aprendizagem inicial da programação, é o principal objetivo. Pretendemos analisar do ponto de vista dos alunos e, naturalmente, do ponto de vista do professor.

No nosso modelo de ensino, constatamos que é difícil acompanhar o trabalho e progresso de cada aluno. Seja pela dificuldade do aluno em verbalizar as suas dificuldades, ou pela vergonha em expor os seus problemas perante os seus colegas, ou por qualquer outra razão, o professor não consegue dar a atenção ou ajuda suficiente ao aluno para resolver as suas dificuldades. O que pode resultar em altas taxas de abandono, desmotivação e insucesso na aprendizagem dos conceitos fundamentais da programação. Com este objetivo em mente, elaboramos um conjunto de atividades para ajudar os alunos na aquisição dos conhecimentos necessários e, também, ajudar o professor na perceção sobre a aprendizagem de cada aluno, através da análise de resultados das atividades. O conjunto de atividades realizadas para a primeira abordagem à introdução à programação, em linguagem C, consiste em atividades do tipo "Escolha múltipla". Neste tipo de atividades é pedido ao aluno para responder correto/incorrecto, sobre um conjunto de afirmações abordando conceitos sobre nomes e identificadores, tipos de dados e instruções de atribuição, instruções de input e output e respetiva formatação e, ainda, a identificação de erros de sintaxe. Este tipo de atividade é de fácil utilização, por todos, e com um elevado nível eficiência no ChatGPT. Basta, pedir ao ChatGPT para ajudar na identificação de identificadores correto ou incorretos na linguagem C, como mostra o seguinte texto em destaque.

Questionar ChatGPT

Podes ajudar a identificar quais os identificadores corretos e incorretos, em C, e porquê?

número numero1 numeroA letra2 mesDia 1ºMes ultimoMes 12Meses Xpto Xpt0 #inteiro 1valor for doWhile ifelse while repeatUntil whiledo mainprincipal double

Apresentamos de seguida a resposta do ChatGPT.

Resposta ChatGPT

Certamente! Vou listar os identificadores corretos e incorretos em C e explicar o motivo de cada um:

Corretos:

- **número:** Correto, pois pode conter letras, números e sublinhados (.) e não começa com um número.
- **numero1:** Correto, pois pode conter letras, números e sublinhados (.) e não começa com um número.
- **numeroA:** Correto, pois pode conter letras, números e sublinhados (.) e não começa com um número.
- **letra2:** Correto, pois pode conter letras, números e sublinhados (.) e não começa com um número.
- **mesDia:** Correto, pois pode conter letras, números e sublinhados (.) e não começa com um número.
- **ultimoMes:** Correto, pois pode conter letras, números e sublinhados (.) e não começa com um número.
- **12Meses:** Correto, pois pode conter letras, números e sublinhados (.) e não começa com um número.
- **Xpto:** Correto, pois pode conter letras, números e sublinhados (.) e não começa com um número.
- **Xpt0:** Correto, pois pode conter letras, números e sublinhados (.) e não começa com um número.
- **mainprincipal:** Correto, pois pode conter letras, números e sublinhados (.) e não começa com um número.
- **double:** Correto, pois é uma palavra-chave reservada da linguagem C.

Incorretos:

- **1ºMes:** Incorreto, pois começa com um número, o que não é permitido para identificadores em C. Os identificadores devem começar com uma letra ou sublinhado.
- **#inteiro:** Incorreto, pois contém o caractere especial "#" que não é permitido em identificadores em C. Os identificadores só podem conter letras, números e sublinhados (.).
- **1valor:** Incorreto, pois começa com um número, o que não é permitido para identificadores em C. Os identificadores devem começar com uma letra ou sublinhado.
- **for:** Incorreto, pois "for" é uma palavra-chave reservada da linguagem C e não pode ser usada como um identificador.
- **doWhile:** Incorreto, pois "doWhile" é uma palavra-chave reservada da linguagem C e não pode ser usada como um identificador.
- **ifelse:** Incorreto, pois "ifelse" é uma palavra-chave reservada da linguagem C e não pode ser usada como um identificador.
- **while:** Incorreto, pois "while" é uma palavra-chave reservada da linguagem C e não pode ser usada como um identificador.
- **repeatUntil:** Incorreto, pois "repeatUntil" não é uma construção válida em C. O loop semelhante é chamado de "do-while" em C.
- **whiledo:** Incorreto, pois "whiledo" não é uma construção válida em C. O loop semelhante é chamado de "do-while" em C.

É importante lembrar que em C, os identificadores são sensíveis a maiúsculas e minúsculas. Portanto, "numero" e "Numero" seriam identificadores diferentes.

Na resposta gerada, verificamos que está dividida em identificadores corretos e incorretos. Cada identificador é classificado como correto ou incorreto, seguido de uma justificação. Analisando em pormenor, verificamos que várias respostas estão incorretas, como:

- **número**: classificado como correto. No entanto contém caractere especial, pelo que deve ser classificada como incorreto.
- **12Meses**: classificado como correto, mas o primeiro caractere é um número, deve ser classificada como incorreto.
- **double**: classificado como correto e com a justificação de que é uma palavra reservada da linguagem C. A palavra reservada não pode ser identificador, logo deve ser classificada como incorreta.
- **doWhile**: identificador classificado como incorreto, com a justificação de que é uma palavra reservada da linguagem C e não pode ser usada como um identificador. O identificador "doWhile", não é palavra reservada. Foi propositadamente escrita desta forma e devia ser classificada como correta.
- **ifelse, repeatUntil, whiledo**: identificadores também classificados como incorretos, pois segundo o ChatGPT, são palavras reservadas da linguagem C e não pode ser usada como um identificador. Tal como exemplo anterior deviam ser classificada como identificadores corretos.

Perante a resposta, questionamos novamente o ChatGPT para corrigir a resposta. A resposta é idêntica à anterior, apenas com a diferença de indicar os identificadores corretos sem justificação. Repetimos a pergunta, mas indicando os identificadores incorretos, como a seguir descrevemos.

Questionar ChatGPT

Tens a certeza relativamente aos identificadores: número, double, doWhile, ifelse, repeatUntil, whiledo, for, 12Meses ?

Apresentamos a nova resposta gerada pelo ChatGPT. Constatamos de que a resposta continua a ser semelhante, apenas com a correção do identificador **12Meses**.

Resposta ChatGPT

Agradeço por trazer o erro à minha atenção, e sinto muito por qualquer confusão causada.

Peço desculpas novamente pela resposta anterior. Houve mais uma confusão em relação aos identificadores corretos e incorretos em C. Vamos corrigir a lista:

Corretos:

numero, numero1, numeroA, letra2, mesDia, ultimoMes, Xpto, Xpt0, mainprincipal, double

Incorretos:

1ºMes: Incorreto, pois começa com um número. Os identificadores em C devem começar com uma letra ou sublinhado.

#inteiro: Incorreto, pois contém o caractere especial "#", que não é permitido em identificadores em C. Os identificadores em C podem conter apenas letras, números e sublinhados.

1valor: Incorreto, pois começa com um número. Os identificadores em C devem começar com uma letra ou sublinhado.

for: Incorreto, pois "for" é uma palavra-chave reservada da linguagem C e não pode ser usado como um identificador.

doWhile: Incorreto, pois "doWhile" é uma palavra-chave reservada da linguagem C e não pode ser usado como um identificador.

ifelse: Incorreto, pois "ifelse" é uma palavra-chave reservada da linguagem C e não pode ser usado como um identificador.

while: Incorreto, pois "while" é uma palavra-chave reservada da linguagem C e não pode ser usado como um identificador.

repeatUntil: Incorreto, pois "repeatUntil" não é uma construção válida em C. O loop semelhante é chamado de "do-while" em C.

whiledo: Incorreto, pois "whiledo" não é uma construção válida em C. O loop semelhante é chamado de "do-while" em C.

12Meses: Incorreto, pois começa com um número. Os identificadores em C devem começar com uma letra ou sublinhado.

Peço desculpas novamente pelo erro anterior e agradeço por trazer isso à minha atenção.

Como demonstrado, a resposta continua incorreta, pelo que questionamos novamente o ChatGPT, desta vez fazendo a afirmação: **"double é identificado como palavra reservada. Como pode ser um identificador correto?"** e, ainda **"doWhile, ifelse, repeatUntil não são palavras reservadas. Deviam ser classificadas como corretas?"**. A resposta produzida pelo ChatGPT, finalmente, é correta.

Como conclusão desta primeira abordagem ao ChatGPT, identificação de nomes e variáveis corretos e incorretos, constatamos que existem muitas incorreções nas respostas apresentadas. Pelo que se sugere uma análise atenta e crítica a todas as respostas para despistar possíveis erros. O que para os alunos, com poucos conhecimentos e experiência, se possa traduzir numa tarefa bastante difícil.

Do ponto de vista do professor, é importante ter o conhecimento destes resultados e a possibilidade de "alucinações" por parte do ChatGPT. Com o conhecimento destes factos e possível advertir os alunos para este tipo de ocorrência.

No que diz respeito ao pedido de exemplos de exercícios do mesmo tipo, com respostas corretas e incorretas, o ChatGPT é um excelente aliado. De seguida, apresentamos um exemplo produzido pelo ChatGPT.

Resposta ChatGPT

Claro! Aqui estão mais alguns exemplos de identificadores corretos e incorretos em C:

Corretos:

nomeCompleto, saldo_conta, Numero123, _variavel, preco_unitario, valorTotal, PI, velocidadeMaxima, primeiraLetra, numero_de_telefone

Incorretos:

123numero (começa com um número)

@simbolo (contém um caractere especial não permitido)

while (palavra-chave reservada)

main-function (contém um caractere especial não permitido)

if-else (contém um caractere especial não permitido)

3.14 (começa com um número e contém um caractere especial não permitido)

double (palavra-chave reservada)

#define (contém um caractere especial não permitido)

primeiro-nome (contém um caractere especial não permitido)

quantidade\$ (contém um caractere especial não permitido)

Lembrando que os identificadores em C devem começar com uma letra (maiúscula ou minúscula) ou com um sublinhado (_) e podem conter letras, números e sublinhados. Além disso, é importante evitar o uso de palavras-chave reservadas da linguagem C como identificadores.

A.2 ChatGPT na avaliação de instruções

No nosso conjunto de atividades de iniciação à aprendizagem da linguagem C, fazem parte questionários de escolha múltipla com afirmações sobre instruções em linguagem C, as quais os alunos devem assinalar as afirmações corretas/incorrectas. Apresentamos de seguida, alguns desses exemplos.

Questionar ChatGPT

São as afirmações corretas?

printf() função usada para output de dados.

printf("Hello World!"); instrução usada para escrever "Hello World!" no ecrã.

printf("Olá Mundo"); função usada para output de dados, mas não pode conter caracteres especiais.

printf("Ola Mundo #123!\n"); não pode conter caracteres especiais.

Resposta ChatGPT

Vamos analisar cada afirmação separadamente:

1 - *printf()* é uma função usada para output de dados. A afirmação está correta. A função *printf()* em muitas linguagens de programação, incluindo C e C++, é usada para imprimir dados na saída padrão, como o terminal ou console.

2 - *printf("Hello World!");* é uma instrução usada para escrever "Hello World!" no ecrã. A afirmação está correta. Essa instrução em linguagens como C e C++ usa a função *printf()* para imprimir a mensagem "Hello World!" na saída padrão.

3 - *printf("Olá Mundo");* é uma função usada para output de dados, mas não pode conter caracteres especiais. A afirmação está incorreta. A função *printf()* pode lidar com caracteres especiais, incluindo caracteres acentuados como "á", "é", "ç", etc. Portanto, não há restrições em relação a caracteres especiais no conteúdo passado para *printf()*.

4 - *printf("Ola Mundo #123!\n");* não pode conter caracteres especiais. A afirmação está incorreta. A função *printf()* pode lidar com caracteres especiais, incluindo símbolos como "#". Portanto, não há restrições em relação a caracteres especiais no conteúdo passado para *printf()*.

Em resumo, as afirmações 1, 2, 3 e 4 são corretas, exceto a afirmação 3 que está incorreta.

As respostas produzidas foram avaliadas corretamente. No entanto, é de salientar a conclusão produzida *"Em resumo, as afirmações 1, 2, 3 e 4 são corretas, exceto a afirmação 3 que está incorreta."*. Mais uma "alucinação" produzida pelo ChatGPT.

Apresentamos outro exemplo onde é solicitado que sejam analisadas um conjunto de instruções.

A pergunta efetuada ao ChatGPT:

São as as afirmações ou instruções corretas?

1. int, character, float, double: são palavras que representam tipos de dados

2. var int i = 0; 3. char letra = A; 4. int i, j, k = 1; 5. double valor, Valor, VALOR = 0.54; 6. char letra = '1'; 7. char letras = 'aeiou'; 8. float iva = 23%; 9. char anos = 7;

Em resposta à nossa solicitação, o ChatGPT produziu respostas corretas e com justificações adequadas. No entanto, chamamos a atenção para a resposta ao exemplo identificado como número 1. A resposta a esta afirmação é considerada como correta. A palavra "character" propositadamente assim escrita, é interpretada como "char". E, deste modo considerada como uma palavra que representa um tipo de dados em linguagem C, o que está incorreto.

Relativamente a atividades de correção de instruções simples, na generalidade o ChatGPT produz respostas corretas, complementado com comentários sobre as instruções bastante completas e elucidativas. No entanto, é importante referir que por vezes surgem pequenas alucinações, como a inclusão de novas instruções no programa, ou simplesmente, apontando correções em instruções onde não existem. O que para utilizadores menos experientes pode levar a alguma confusão.

Na ajuda ao professor à construção de atividades do mesmo tipo, o ChatGPT revela-se um aliado importante.

A.3 ChatGPT na correção de programas

Neste conjunto de atividades, são apresentados aos alunos conjuntos de instruções que formam um programa, em linguagem C. É solicitado ao aluno que corrija as instruções incorretas e/ou na falta de alguma instrução indicar qual.

Pergunta ao ChatGPT: *"Assinale as instruções do programa com erros. Ou, no caso de instrução em falta, assinale a linha/instrução onde deveria existir essa instrução."*

```
1 #include <stdio.h>
2 {
3     printf("Ola Mundo !");
4 }
```

Resposta do ChatGPT: *No código fornecido, a linha 2 está incorreta. Ela deveria conter a assinatura da função 'main()', que é o ponto de entrada do programa em C. Além disso, a linha 4 possui um espaço extra antes do ponto de exclamação. A correção seria a seguinte:*

```
1 #include <stdio.h>
2     int main()
3     {
4         printf("Ola Mundo!");
5         return 0;
6     }
```

Nessa versão corrigida do programa, a função 'main()' é declarada na linha 2 e envolve o código principal do programa, delimitado pelas chaves ". A instrução 'printf' na linha 4 imprime a mensagem "Olá Mundo!" na saída padrão. A linha 5 é opcional e usada para indicar que o programa foi executado com sucesso, retornando o valor 0.

A resposta produzida é correta. No entanto, verificamos que a afirmação *"Além disso, a linha 4 possui um espaço extra antes do ponto de exclamação. não faz sentido, ou podemos considerar como uma alucinação do ChatGPT.*

Apresentamos de seguida outro exemplo.

Pergunta ao ChatGPT: *"Identifique as instruções do programa com erros."*

```
1 #include <stdio.h>
2     main(){
3     int a, b;
4     printf("Enter two numbers ? ");
5     scanf("%d", &a, &b);
6     printf("Sum = %d\n", a + b);
7 }
```

No trecho de código fornecido, há um erro na declaração da função main(). Aqui está o código corrigido:

```
1 #include <stdio.h>
2 int main(){
3     int a, b;
4     printf("Enter two numbers ? ");
5     scanf("%d", &a);
```

```

6     scanf("%d", &b);
7     printf("Sum = %d\n", a + b);
8     return 0;
9 }

```

As correções realizadas foram:

Adição do tipo de retorno int na declaração da função main() e adição do comando return 0; para indicar um encerramento bem-sucedido do programa. Separação das instruções scanf() em duas linhas distintas, uma para cada variável a e b. Isso garante que cada número seja lido corretamente em uma variável separada.

Em conclusão, verificamos que o ChatGPT produziu respostas corretas e complementou com informação sobre as operações realizadas. Pode ser discutível a inclusão de novas instruções, quando só é pedido para corrigir, como por exemplo a correção efetuada na instrução na linha 5, **scanf("%d", &a, &b);**. É importante referir que por vezes são apontados erros onde não existem e onde a correção é igual à original. O que para utilizadores menos experientes pode levar a alguma confusão.

Comprovamos mais uma vez, que quando pedido ao ChatGPT para criar exemplos de programas com erros, para os alunos corrigir, a tarefa é realizada de uma forma bastante eficiente. Como podemos comprovar pelos exemplos seguintes.

Exemplo 1

```

1  #include <stdio.h>
2  int main() {
3      int x;
4      printf("Digite um numero: ");
5      scanf("%d", x);
6      printf("O numero digitado: %d\n", x);
7      return 0;
8  }

```

Erro: Na linha `scanf("%d", x);`, está faltando o operador de endereço & antes da variável x no argumento da função `scanf()`. Os alunos devem adicionar o operador de endereço para corrigir o erro.

Exemplo 2

```

1  #include <stdio.h>
2  int main() {
3      float a, b, soma;
4      printf("Digite dois numeros: ");

```

```

5     scanf("%f %f", &a, &b);
6     soma = a + b;
7     printf("A soma = %f\n", soma)
8     return 0;
9 }

```

Erro: Na linha `printf("A soma = %f\n", soma)`, está faltando um ponto e vírgula no final da linha. Os alunos devem adicionar o ponto e vírgula para corrigir o erro.

Exemplo 3

```

1  #include <stdio.h>
2  int main() {
3      int num1, num2;
4      printf("Digite o primeiro numero: ");
5      scanf("%d", &num1);
6      printf("Digite o segundo numero: ");
7      scanf("%d", &num2);
8      int resultado = num1 + num2;
9      printf("A soma de %d e %d = %d\n", num1, num2, resultado);
10     return 0;
11 }

```

Erro: O programa está faltando uma chave de fechamento (`}`) no final da função `main()`. Os alunos devem adicionar a chave de fechamento para corrigir o erro.

A.4 ChatGPT na construção de programas

O desenvolvimento de programas em C, requer um conhecimento e compreensão de conjunto de conceitos, como variáveis, tipos de dados, estruturas de controlo e outros. Atendendo às características e desenvolvimento do ChatGPT, a construção de programas é uma das suas melhores capacidades. Em resumo, o código dos programas produzido é correto e complementado com uma explicação e, por vezes, com algoritmo. Os vários exemplos de atividades de construção de programas em C testados, foram produzidos resultados excelentes. Apresentados alguns dos exemplos solicitados e produzidos pelos ChatGPT.

Exemplo 1

Faça um programa que permita converter um valor introduzido em quilómetros para milhas. Milhas=Quilometros × 0.62137 O programa deve especificamente ler e escrever como o exemplo se-

guinte: *INPUT: Valor em quilometros? 295.45 OUTPUT: 183.58 mi*

```
1  #include <stdio.h>
2  int main() {
3      float quilometros, milhas;
4      printf("Valor em quilometros? ");
5      scanf("%f", &quilometros);
6      milhas = quilometros * 0.62137;
7      printf("%.2f mi", milhas);
8      return 0;
9  }
```

Exemplo 2

Faça um programa que calcule o volume de uma pirâmide quadrangular dado um valor inteiro para a aresta da base um valor inteiro para a altura. Onde, $\text{Volume} = 1/3 \times \text{Area da Base} \times \text{Altura}$. O programa deve especificamente ler e escrever como o exemplo seguinte:

INPUT: Aresta da base? 3 Altura? 6

OUTPUT: Volume = 18.0 cm3

```
1  #include <stdio.h>
2  int main() {
3      int arestaBase, altura;
4      float volume, areaBase;
5      printf("Aresta da base? ");
6      scanf("%d", &arestaBase);
7      printf("Altura? ");
8      scanf("%d", &altura);
9      areaBase = arestaBase * arestaBase;
10     volume = (1.0 / 3) * areaBase * altura;
11     printf("Volume = %.1f cm3", volume);
12     return 0;
13 }
```

A.5 ChatGPT na ajuda aos Parson Problems

Com o intuito de conhecer as respostas produzidas pelo ChatGPT ao conjunto de atividades do tipo *parson problems*, experimentamos alguns exemplos simples. Na A.1, apresentamos o programa original, e à direita, o programa com as suas instruções desordenadas.

<pre>main(){ int a, b, soma; scanf("%d", &a); scanf("%d", &b); soma = a + b; printf("Soma = %d\n", soma); }</pre>	<pre>printf("Soma = %d\n", soma); soma = a + b; main(){ } scanf("%d", &b); int a, b, soma; scanf("%d", &a);</pre>
---	---

Figura A.1: Exemplo de programa, atividade do tipo *Parson Problem*, programa original vs programa com instruções desordenadas.

O resultado produzido pelo ChatGPT é o seguinte:

```
1  #include <stdio.h>
2  int main() {
3      int a, b, soma;
4      printf("Enter the value of a: ");
5      scanf("%d", &a);
6      printf("Enter the value of b: ");
7      scanf("%d", &b);
8      soma = a + b;
9      printf("Soma = %d\n", soma);
10     return 0;
11 }
```

Verificamos que o programa está correto, mas novas instruções foram adicionadas, como: **#include <stdio.h>; printf("Enter the value of a: "); printf("Enter the value of b: "); return 0;** e modificou a instrução **main(){** para **int main(){**. Concordamos que o programa produzido é um programa correto que soma dois inteiros inseridos pelo usuário, assim como o problema original. No entanto, o objetivo do exercício é ordenar as instruções formando um programa semanticamente correto.

Na figura A.2, podemos visualizar outro exemplo de atividade *parson problema*, e teste de resolução

com o ChatGPT.

<pre>int main(int argc, char *argv[]) { int n, d, ni = 0, no; printf("Enter a number [100, 999]? "); scanf("%d", &n); no = n; d = n % 10; ni = ni * 10 + d; n = n / 10; d = n % 10; ni = ni * 10 + d; n = n / 10; d = n % 10; ni = ni * 10 + d; n = n / 10; if (ni == no){ printf("Capicua\n"); } else { printf("NOT capicua\n"); } return 0; }</pre>	<pre>d = n % 10; int n, d, ni = 0, no; } else { } no = n; printf("NOT capicua\n"); printf("Capicua\n"); ni = ni * 10 + d; d = n % 10; n = n / 10; n = n / 10; return 0; if (ni == no){ { ni = ni * 10 + d; int main(int argc, char *argv[]) d = n % 10; } printf("Enter a number [100, 999]? "); n = n / 10; scanf("%d", &n); ni = ni * 10 + d;</pre>
---	---

Figura A.2: Exemplo de programa, atividade do tipo *Parson Problem*, programa original vs programa com instruções desordenadas.

O resultado produzido é um programa incorreto. As instruções são:

```
1  #include <stdio.h>
2  int main(int argc, char *argv[]) {
3      int n, d, ni = 0, no;
4      printf("Enter a number [100, 999]? ");
5      scanf("%d", &n);
6      no = n;
7      d = n % 10;
8      n = n / 10;
9      d = n % 10;
10     n = n / 10;
11     ni = ni * 10 + d;
12     if (ni == no) {
13         printf("Capicua \n");
14     } else {
15         printf("NOT capicua \n");
16     }
```

```
17     return 0;  
18 }
```

Importante realçar que quando solicitamos a correção do ChatGPT, o resultado continua incorreto. Porém, quando solicitado a testar com um valor, o ChatGPT responde corretamente. Ou seja, responde se o valor introduzido é capicua ou não, não fazendo qualquer teste no programa produzido.

Estes são apenas alguns exemplos dos testes realizados com este tipo de atividade. Os resultados são semelhantes, as respostas produzidas estão incorretas. Nas respostas produzidas são inseridas novas instruções, e/ou faltam instruções, em suma o programa gerado não produz o resultado esperado.

Estratégias para o ensino e a aprendizagem da iniciação à programação no ensino superior

Estrategias para la enseñanza y el aprendizaje de la iniciación a la
programación en la universidad

José Alberto Quitério Figueiredo

Tesis Doctoral

**Programa de Doctorado Ingeniería Informática
Universidad de Salamanca**

Orientador: Prof. Francisco José García-Peñalvo

Abril 2024

Este texto es un resumen, escrito en español, de la tesis doctoral
*"Estratégias para o ensino e a aprendizagem da iniciação à programação no
ensino superior"*, integrada en el programa de Doctorado en Ingeniería
Informática de la Universidad de Salamanca.

Resumen

Aprender a programar permite adquirir competencias técnicas para el desarrollo de software. Pero también el desarrollo de competencias cognitivas como la resolución de problemas, el razonamiento lógico, la creatividad y el pensamiento crítico. Estas competencias son muy valoradas y se consideran fundamentales para cualquier persona que se enfrente al mercado laboral de hoy y del futuro.

La tecnología tiene un fuerte impacto en la sociedad y enseñando programación permitimos a los estudiantes comprender mejor la tecnología, ser usuarios responsables, más informados y conscientes.

Sin embargo, enseñar y aprender a programar es una tarea compleja. Las dificultades en la enseñanza y el aprendizaje de la programación han sido un tema recurrente a lo largo de los tiempos. Muchos estudiantes se enfrentan a dificultades en las fases iniciales del aprendizaje de la programación. Enseñar y aprender programación requiere mucho trabajo, dedicación y compromiso por parte de todos los implicados en este proceso.

Esta tesis doctoral aborda el reto de mejorar el proceso de enseñanza y el aprendizaje de la iniciación de la programación, en un contexto muy concreto, en la unidad curricular de Introducción a la Programación de la carrera de Ingeniería Informática del Instituto Politécnico de Guarda.

Nuestro grupo de estudiantes tiene características muy particulares. La mayoría del estudiantado de nuestro curso tienen bajos niveles de motivación y escasos conocimientos generales. Un gran porcentaje de estudiantes no adquiere conocimientos básicos de programación. Este problema es extremadamente importante, ya que unos conocimientos de programación deficientes siempre serán un obstáculo importante para cursar estudios de ingeniería informática.

Con el objetivo principal de paliar este problema, investigamos, experimentamos e implementamos un conjunto de actividades y metodologías didácticas que nos permitieron mejorar las competencias básicas de programación. Para conseguir este objetivo, y dadas las características de nuestro grupo de estudio, utilizamos metodología de investigación científica, con el desarrollo de un artefacto, denominado *Design Science Research Methodology*.

Esta investigación ha dado lugar a una estrategia para la enseñanza y el aprendizaje de la iniciación de la programación que ha mejorado notablemente las competencias de nuestros alumnos.

Las actividades de pensamiento computacional utilizadas como introducción y un conjunto de actividades de elección múltiple y *problemas de parson*, con realimentación inmediato, puestas en práctica

para los conceptos iniciales del aprendizaje de la programación, contribuyeron claramente a esta mejora. Pero también, el gran número de ejercicios de codificación, también con realimentación inmediato, puestos a disposición del alumnado para practicar y consolidar los diferentes conceptos de aprendizaje de la programación inicial.

En esta estrategia, hemos incluido un modelo predictivo del fracaso escolar, basado en una red neuronal, con el objetivo de detectar precozmente las dificultades del alumnado y actuar con rápidamente para ayudarles.

Toda la estrategia implicada en el proceso de enseñanza y aprendizaje de la programación para principiantes no sería posible sin la ayuda de una herramienta tecnológica de gestión. En este trabajo presentamos el desarrollo de una herramienta tecnológica de ayuda al proceso de enseñanza y aprendizaje.

Los resultados obtenidos en la evaluación del curso de Introducción a la Programación, en función de la estrategia y la herramienta adoptadas, muestran que hemos mejorado significativamente nuestro proceso de enseñanza y aprendizaje. Esto indica que vamos por el buen camino, y que los esfuerzos realizados para que un mayor número de alumnos adquieran las competencias básicas de programación, se han hecho realidad.

Nuestro trabajo tiene un origen y unas características muy particulares, pero estamos convencidos de que las medidas sugeridas supondrán mejoras significativamente en cualquier sistema de la enseñanza y el aprendizaje de la iniciación a la programación.

Concluimos este trabajo con una breve aproximación a lo que podría ser el uso e impacto de sistemas inteligentes, como *ChatGPT*, en la enseñanza y el aprendizaje de la iniciación a la programación.

Palabras Clave

Programación; Codificación; Enseñanza inicial de la programación; Aprendizaje inicial de la programación;

Índice general

1. Introducción	1
1.1. Problema	3
1.1.1. Caracterización del grupo de estudio	5
1.1.2. Causas del problema	5
1.1.3. Preguntas de investigación	7
1.2. Objetivos	8
1.3. Metodología	10
1.4. Contexto	11
2. Estado de la cuestión	13
3. Aplicación de <i>Design Science Research Methodology</i> en Introducción a la Programación	17
3.1. Identificación de problemas y motivación	19
3.2. Definición de objetivos para una solución	20
3.3. Diseño y desarrollo	21
3.4. Demostración	23
3.5. Evaluación	24
3.6. Comunicación	25
4. Discusión	27
5. Conclusión	31
5.1. Principales Contribuciones	33
5.2. Trabajo Futuro	36
Referencias	39

Acrónimos

CT	Computational Thinking
DSRM	Design Science Research Methodology
HTProgramming	Help To Programming
IDE	Integrated Development Environment
IPG	Instituto Politécnico da Guarda
MCQ	Multiple Choice Question
PALOP	Países Africanos de Língua Oficial Portuguesa
PP	Parson Problems

1

Introducción

Índice

1.1. Problema	3
1.2. Objetivos	8
1.3. Metodología	10
1.4. Contexto	11

Los conocimientos de programación son esenciales para todos los estudiantes. Los conocimientos informáticos básicos no son suficientes para cualquier estudiante que quiera ser competitivo en el mercado laboral actual [1]. Se ha vuelto esencial que los estudiantes desarrollen habilidades informáticas más profundas. El desarrollo de habilidades como la creatividad, la resolución de problemas, la persistencia, la colaboración, la comunicación y el pensamiento crítico son cruciales para el éxito de un estudiante frente a una tecnología en constante evolución [2,3]. En general, se considera que los cursos de programación son eficaces para fomentar y mejorar estas competencias. Por ello, muchos cursos de enseñanza superior incluyen una introducción a la programación en sus planes de estudio. Gran cantidad de estudiantes de nivel universitario aprenden a programar, pero desgraciadamente tienen dificultades. Estas dificultades y problemas tienen un impacto más significativo cuando la programación es una de las áreas centrales del curso. Como resultado, hay altos índices de fracaso, desmotivación y abandono de los cursos por parte de los estudiantes.

La enseñanza y el aprendizaje de la iniciación a la programación es una tarea difícil que requiere mucho trabajo, dedicación y práctica. Con el objetivo principal de enseñar y aprender los principios básicos de la programación, en los últimos años se ha investigado intensamente sobre las dificultades en la enseñanza y los problemas que tienen los alumnos para adquirir estas habilidades. Se han esgrimido diversas razones y causas del problema, y se han sugerido diversas estrategias, metodologías y técnicas para ayudar a minimizarlo. Sin embargo, en la mayoría de los casos, los problemas persisten y la falta de compromiso, motivación y trabajo conducen al abandono y al fracaso en la adquisición de las competencias iniciales de programación.

Bajo el tema general de las dificultades en la enseñanza y el aprendizaje inicial de la codificación, en este capítulo presentamos el problema, identificamos algunas de las razones que lo explican, caracterizamos a nuestro grupo de estudio y elaboramos nuestras preguntas de investigación. Definimos el objetivo a alcanzar y la metodología a aplicar en este trabajo. Por último, presentamos la estructura de la tesis.

1.1. Problema

A lo largo de los años dedicados a la enseñanza de la iniciación a la programación en la enseñanza superior, hemos observado y experimentado con diversos enfoques y métodos. Y casi invariablemente el resultado es idéntico: altas tasas de fracaso, abandono y desmotivación. Reducir significativamente estos resultados es nuestra principal motivación y la razón de llevar a cabo esta investigación. La enseñanza inicial de la programación es el objetivo del curso de introducción a la programación en el primer año del programa de ingeniería informática de la *Escola Superior de Tecnologia e Gestão*, Instituto Politécnico da Guarda (IPG). Con este curso, esperamos que los estudiantes adquieran las

habilidades fundamentales para construir programas informáticos capaces de resolver problemas. El fracaso en la consecución de este objetivo se refleja en las altas tasas de fracaso y abandono. Hemos identificado claramente nuestro problema: ***el estudiantado no adquiere los conocimientos básicos de programación para construir programas informáticos.***

Este problema es consecuencia de otra serie de factores, como son: la escasa motivación del alumnado por el curso; la deficiente formación general, especialmente en matemáticas; la no asistencia a las clases; los problemas en la construcción y aplicación de las diferentes estructuras del lenguaje de programación; los problemas en la capacidad de resolución de problemas; y los problemas en la transformación de la estrategia de resolución en un programa informático [4–7].

La programación tiene una connotación negativa. Enseñar y aprender a programar es difícil. La enseñanza y el aprendizaje de la iniciación a la programación se considera uno de los grandes retos en el campo de la enseñanza de la informática. Existen muchos trabajos que describen y argumentan estas dificultades, como los trabajos desarrollados en [8–10]. En estos trabajos se identifica claramente y es conocido por toda la comunidad científica, especialmente en el campo de la informática, que el estudiantado tiene grandes dificultades para programar.

La programación es el proceso de transformar un plan de términos ordinarios en términos compatibles con un ordenador [11]. En otras palabras, programar es el proceso de crear un conjunto organizado de instrucciones que permitan a un ordenador realizar una tarea. Pero para crear este proceso, el primer paso es que la persona construya un plan, una estrategia para resolver una tarea [4]. En el segundo paso, el o la estudiante debe conocer y ser capaz de aplicar la sintaxis y la semántica de un lenguaje de programación. A continuación, esta estrategia de resolución debe estar en consonancia con los recursos disponibles en el lenguaje de programación. Por todo ello, el objetivo principal de la enseñanza de la iniciación a la programación debe ser que el estudiantado adquiera las habilidades necesarias para crear estrategias de resolución de problemas, así como el conocimiento de la sintaxis y la semántica de un lenguaje de programación, todo ello en un breve espacio de tiempo [9, 12, 13]. Ante estos hechos, es difícil encontrar un método de enseñanza que sea eficaz y adecuado para todos los alumnos. Es importante que el profesor conozca un conjunto de estrategias, con ejemplos de buenos resultados, para que profesorado y alumnado puedan alcanzar sus objetivos.

Desde la aparición de los primeros lenguajes de programación, este problema ha sido objeto de diversos estudios y reflexiones. Desde entonces se han creado varios grupos de trabajo para analizar aspectos generales de la enseñanza y aprendizaje de la programación [14, 15]. Otros se han centrado en aspectos más concretos de las dificultades de enseñar y aprender programación [3, 9, 16–18]. Sin embargo, en todos ellos hay evidencias de dificultades en la enseñanza y el aprendizaje de la codificación, independientemente de la ubicación geográfica o el curso.

1.1.1. Caracterización del grupo de estudio

En nuestro grupo de estudio participan estudiantes con características muy particulares. Quienes asisten al curso de Introducción a la Programación en el primer año del programa de ingeniería informática en IPG. Los objetivos del curso son adquirir y aplicar conceptos básicos de programación. Se utiliza el lenguaje de programación C. La decisión de adoptar un lenguaje de programación es una decisión de la unidad técnica y científica de TI, y no de una persona en particular. Sin embargo, pensamos que para el estudiantado que cursa una carrera de ingeniería informática, es un lenguaje que cumple con los requisitos necesarios para el desarrollo de la carrera.

El programa de ingeniería informática de la IPG no suele ser la primera opción a la hora de solicitar estudios superiores. Esto puede indicar una falta de motivación por el curso. La media de notas de solicitudes se sitúa entre 11.45 y 13.48, mientras que la media de matemáticas se sitúa entre 10.62 y 11.96. Estas cifras revelan algunos problemas en la formación general de nuestro grupo de estudio, especialmente en matemáticas.

En los últimos años, hemos recibido estudiantes de Países Africanos de Língua Oficial Portuguesa (PALOP) que acceden a la enseñanza superior en virtud de acuerdos de cooperación con graves problemas en su formación general.

El número de personas que acceden a través del concurso nacional de acceso a la enseñanza superior es inferior al 30 %, con la excepción del último curso académico en el que fue del 45 %. También se observa que la mayoría del alumnado que asiste al curso de Introducción a la Programación accedió al curso a través de las oposiciones extraordinarias y repetidores. El número de personas matriculadas en el curso de introducción a la programación es de 96 de media. Sin embargo, solo entre el 50 % y el 60 % asiste a clase.

1.1.2. Causas del problema

¿Cuáles son las causas de todos los estudios y preocupaciones sobre la enseñanza y el aprendizaje de la programación? Para responder a esta pregunta es importante analizar qué es la programación. Como ya se ha mencionado, la programación es el proceso de creación de un conjunto organizado de instrucciones que permite resolver problemas mediante su ejecución por un ordenador. Al centrarse principalmente en la enseñanza y el aprendizaje de la iniciación a la programación, el profesorado debe ser capaz de impartir los conocimientos necesarios para que el alumnado adquiera las habilidades esenciales para llevar a cabo este proceso. En primer lugar, el alumnado debe conocer y aplicar un conjunto de estrategias para resolver un problema [4]. En segundo lugar, los y las estudiantes debe conocer y ser capaces de aplicar la sintaxis y la semántica de un lenguaje de programación. Después, tienen que coordinar la estrategia de solución con los recursos disponibles en el lenguaje de programación.

Todo ello en un breve espacio de tiempo. Con este complejo conjunto de tareas, es difícil encontrar un método de enseñanza eficaz y adecuado para todas las personas [19].

En el proceso de creación de un programa informático es necesario aprender un nuevo lenguaje. Este lenguaje es diferente de los lenguajes escritos y hablados ordinarios. Es un lenguaje de programación, caracterizado por un conjunto de conceptos abstractos y reglas muy específicas. El lenguaje de programación suele utilizarse con un entorno de desarrollo integrado, *Integrated Development Environment (IDE)*. La mayoría de las veces, este entorno de desarrollo es una herramienta útil para el programador. Esta aplicación incluye en su entorno recursos sobre el lenguaje de programación y el compilador y/o intérprete. También proporciona información adicional sobre errores de código y su localización. Sin embargo, a veces este entorno actúa más como un obstáculo para el aprendizaje inicial de la programación, que como una herramienta útil.

Una de las mejores habilidades para todos hoy y en el futuro es la resolución de problemas y el pensamiento computacional (*Computational Thinking (CT)*) [20, 21]. El pensamiento computacional, o la falta de capacidad para aplicar y utilizar el pensamiento computacional, es uno de los temas más debatidos y se identifica como una de las dificultades en el aprendizaje de la programación. El Computational Thinking (CT) es una de las formas de desarrollar la capacidad de resolución de problemas. Podemos definir el CT como un conjunto de habilidades para la resolución de problemas basados en [22] técnicas informáticas. Para muchos problemas, en las áreas más diversas, es una buena estrategia hacer un plan para resolverlos basado en algunas técnicas informáticas, como: descomponer un problema complejo en partes más simples; buscar similitudes en otras soluciones y experiencias; centrarse en lo realmente importante y eliminar partes específicas para obtener una solución más completa; y desarrollar una solución paso a paso. En otras palabras, los cuatro factores principales del pensamiento computacional: descomposición, reconocimiento de patrones, abstracción y construcción de algoritmos. En muchos de los trabajos que hemos realizado, hemos descubierto que los y las estudiantes nunca han tenido la oportunidad de practicar o utilizar estrategias y conocimientos de resolución de problemas, es decir, nunca han tenido la oportunidad de desarrollar sus habilidades de pensamiento computacional.

El aprendizaje inicial de la programación suele ser una unidad curricular del primer año de un curso de enseñanza superior. Para la mayoría del alumnado, el primer año es una época difícil. Se trata de adaptarse a un nuevo sistema educativo, vivir lejos de la familia, hacer nuevos amigos y amigas, un entorno totalmente nuevo al que hay que adaptarse y manejarse. La programación ha adquirido fama de difícil [12]. Esta opinión se transmite de los estudiantes mayores a los más jóvenes. Por otra parte, existe una imagen social de quienes desarrollan profesiones relacionadas con la programación como individuos socialmente alineados, que viven detrás de un ordenador y se alimentan de pizza y cafeína. No es una imagen muy agradable para utilizarla como referencia o motivación.

Aprender a programar requiere una cantidad considerable de esfuerzo y trabajo duro. Muchas per-

sonas no están dispuestas o motivadas para hacer ese esfuerzo. Como resultado, todo el proceso de enseñanza y aprendizaje se ve comprometido.

1.1.3. Preguntas de investigación

Tras identificar el problema, caracterizar a nuestro grupo de estudio y señalar algunas de las posibles causas del mismo, debemos elaborar nuestras preguntas de investigación, que nos permitirán definir nuestros objetivos.

Como se ha descrito anteriormente, la programación es el proceso de creación de un conjunto organizado de instrucciones que permiten a un ordenador llevar a cabo una tarea. Pero para que este proceso se lleve a cabo, se necesita una estrategia de resolución y conocer y aplicar la sintaxis y la semántica de un lenguaje de programación. Por último, la estrategia de resolución tiene que estar en consonancia con los recursos disponibles en el lenguaje. Como ya se ha dicho, este proceso es difícil de realizar para un gran número de estudiantes. Con este problema en mente, podemos formular nuestra primera pregunta de investigación.

PI1 - ¿Qué conjunto de actividades se deben utilizar para mejorar las competencias en el aprendizaje de la iniciación a la programación?

Con esta pregunta, pretendemos investigar qué conjunto de actividades debemos utilizar en nuestro proceso inicial de enseñanza y aprendizaje de la programación, con el fin de minimizar nuestros problemas y mejorar significativamente las competencias básicas del estudiantado en la resolución de problemas y la construcción de programas informáticos.

La siguiente pregunta de investigación está directamente relacionada con la identificación de las dificultades y los patrones de comportamiento que llevan a los alumnos a obtener peores resultados.

PI2 - ¿Cómo podemos predecir el fracaso de los y las estudiantes?

Con esta pregunta queríamos investigar qué características contribuyen más al fracaso en el alumnado. Estas características se identifican a partir del conjunto de acciones realizadas por los y las estudiantes en el proceso de enseñanza y aprendizaje de la programación. Con esta información queremos detectar precozmente los problemas y actuar con rapidez para diagnosticarlos y minimizarlos.

La enseñanza y el aprendizaje de la programación implican una gran cantidad de información en un corto espacio de tiempo. Por otra parte, las personas tienen diferentes ritmos de aprendizaje, conocimientos y motivación. Para gestionar el proceso se necesita una herramienta tecnológica que responda a las necesidades de estudiantado y profesorado. La siguiente pregunta de investigación nos permitirá evaluar la necesidad e importancia de una herramienta tecnológica en el proceso inicial de enseñanza y aprendizaje de la programación.

PI3 - ¿Qué puede aportar el uso de una herramienta tecnológica al proceso de enseñanza y el aprendizaje de la iniciación a la programación?

Una vez formuladas las preguntas de investigación, en la siguiente sección definimos nuestro objetivo general y los objetivos específicos que nos permitirán llevar a cabo esta investigación en busca de los conocimientos necesarios para realizarla.

1.2. Objetivos

El objetivo general de cualquier proyecto es la definición de lo que se quiere conseguir. El objetivo general debe demostrar el cambio en la realidad que se pretende conseguir, es decir, pretende dar respuesta al problema principal identificado. De acuerdo con el marco descrito en este trabajo, y su especificidad, el objetivo principal de nuestro trabajo es **que el estudiantado adquiera conocimientos básicos de programación para construir programas informáticos.**

Una vez identificado el problema principal, identificadas las causas del problema y definidas las preguntas de investigación, debemos explicar los detalles o etapas del trabajo para alcanzar el objetivo general. Estas etapas de trabajo u objetivos específicos deben corresponder a resultados intermedios que debemos conseguir para alcanzar el objetivo general.

Una vez formulada la pregunta de investigación *PI1 - ¿Qué conjunto de actividades se deben utilizar para mejorar las competencias en el aprendizaje de la iniciación a la programación?*, pretendemos investigar, con un conjunto de objetivos específicos, qué actividades utilizar en nuestro proceso inicial de enseñanza y aprendizaje de la programación, para minimizar nuestros problemas y mejorar significativamente las competencias básicas de los alumnos en la construcción y resolución de programas informáticos.

Para llevar a cabo esta investigación, es importante que de alguna manera consigamos dar respuesta o compensar la baja motivación quienes estudian programación, su escasa formación general, sobre todo en matemáticas, su absentismo en clase, sus dificultades para resolver problemas y, en consecuencia, transformarlos en un programa informático. Teniendo esto en cuenta, nos planteamos el objetivo de utilizar actividades que nos permitieran practicar conceptos informáticos más serios de forma lúdica, lo que aumentaría sus capacidades cognitivas de resolución de problemas.

A lo largo de los años dedicados a la enseñanza de la programación se ha infravalorado los conceptos iniciales del lenguaje de programación, como identificadores y variables, tipos de datos, formateo de datos de entrada y salida, instrucciones y expresiones aritméticas sencillas, expresiones booleanas, bloques de instrucciones, entre otros. Por otro lado, se valora saber hacer, resolver y codificar pequeños problemas. Sin embargo, somos conscientes de que muchos de los problemas en la codificación provienen de pequeños errores y dificultades en la aplicación de estos conceptos iniciales, propios de un

lenguaje de programación. Teniendo esto en cuenta, nuestro objetivo es implementar un conjunto de actividades que permitan practicar y consolidar sus conocimientos en esta fase inicial del aprendizaje de un lenguaje de programación. Para que alguien se sienta motivado y desarrolle las actividades a su ritmo, es importante que las actividades tengan realimentación inmediata. De esta forma, la persona puede evaluar el progreso de su aprendizaje y ver que sus acciones tienen una respuesta, manteniendo su confianza y motivación para resolver problemas. Por ello, nos hemos marcado el objetivo de implantar un sistema de feedback inmediato en todas las actividades a desarrollar.

La enseñanza y el aprendizaje de la iniciación a la programación requieren mucho trabajo y dedicación por parte de todos los implicados en el proceso. Para el profesor, requiere una búsqueda constante de los mejores métodos y los mejores ejemplos de actividades para presentar al estudiantado. Sin embargo, cada año nos damos cuenta de que el grupo de estudiantes es muy heterogéneo. Con ritmos de aprendizaje, niveles de conocimiento, comportamiento y motivación muy diferentes. Por ello, esto se convierte en un reto constante y extremadamente exigente para el profesorado. Igualmente para el estudiantado es un trabajo exigente que requiere mucha dedicación, compromiso y trabajo duro. Por ello, es importante contar con una amplia variedad de actividades guiadas, con realimentación, para que el estudiantado pueda trabajar y practicar el uso de los diferentes conceptos y técnicas de un lenguaje de programación. Teniendo en cuenta lo anterior, es importante fijarse el objetivo de realizar un conjunto organizado de actividades de codificación para que los alumnos puedan practicar y consolidar sus conocimientos.

Al formular la pregunta de investigación *PI2 - ¿Cómo podemos predecir el fracaso de los y las estudiantes?*, nuestro objetivo es investigar si existe un patrón o comportamiento en el alumnado que nos permita predecir su bajo rendimiento. La idea surge del volumen de información que genera el estudiantado. Predecir el fracaso de un o una estudiante sería una ayuda relevante, ya que nos permitiría actuar y trabajar sobre sus dificultades para contrarrestar esta predicción. Por ello, nos planteamos el objetivo de investigar qué variables debemos analizar e incluir, y si es factible un modelo de predicción del fracaso estudiantil.

Por último, de acuerdo con la pregunta de investigación *PI3 - ¿Qué puede aportar el uso de una herramienta tecnológica al proceso de enseñanza y el aprendizaje de la iniciación a la programación?*, parece inevitable que se utilice una herramienta tecnológica para gestionar todo el proceso. Proponer actividades al alumno con feedback no será posible sin una herramienta automática que gestione este proceso. Por otro lado, el modelo predictivo no será ejecutable sin esta herramienta automática de tratamiento y análisis de datos. Además, de lo contrario, el seguimiento por parte del profesorado de la actividad y el progreso de cada persona sería poco práctico o casi imposible, incluso para nuestro reducido número de estudiantes. En consecuencia, el objetivo de construir una aplicación que ayude al estudiantado y profesorado en el proceso de enseñanza y aprendizaje es demasiado obvio e impres-

cindible.

Con la definición del objetivo general y de los objetivos específicos, hemos establecido las etapas en el desarrollo de este trabajo. Con esta planificación esperamos poder sugerir una estrategia de enseñanza y aprendizaje, basada en un conjunto de actividades y con ayuda tecnológica, que ayude a todas las personas involucradas en este difícil y complejo proceso de enseñanza y el aprendizaje de la iniciación a la programación, y elimine o reduzca el problema. También esperamos que este trabajo sea un buen aporte para todos quienes se identifican con el área y enfrentan dificultades similares.

1.3. Metodología

Los trabajos realizados en esta área de conocimiento suelen seguir un patrón de investigación consistente en definir el problema, revisar la literatura, desarrollar hipótesis de investigación, recoger y analizar datos y discutir los resultados. En este trabajo, optamos por aplicar *Design Science Research Methodology* Design Science Research Methodology (DSRM), metodología de investigación científica con desarrollo de artefactos [23–25]. Dadas las características de nuestro trabajo y grupo de estudio, consideramos que esta es la mejor opción para su desarrollo y la consecución de nuestros objetivos.

En el trabajo desarrollado en [23], los autores proponen que DSRM incluya seis etapas: identificación y motivación del problema, definición de los objetivos de una solución, diseño y desarrollo, demostración, evaluación y comunicación.

En la primera etapa, identificación del problema y motivación, hay que definir el problema y justificar la importancia de una solución. Enmarcar la importancia del problema e identificar los puntos débiles de las soluciones actuales. La etapa siguiente, o etapa 2, consiste en definir los objetivos de una solución. Deben definirse objetivos generales como la viabilidad y el rendimiento, así como objetivos específicos que debe incluir la solución del problema. También deben considerarse los métodos, teorías y tecnologías que pueden ayudar a definir los objetivos. La etapa 3 consiste en crear un artefacto. Es decir, construir un modelo, aplicar una teoría, método o tecnología que resuelva el problema. La etapa 4, la demostración, consiste en probar que el artefacto funciona resolviendo uno o más casos del problema. La etapa 5 consiste en la evaluación. Observar y medir los resultados de la aplicación del artefacto para resolver el problema. Debemos evaluar los resultados y, si es necesario, repetir el paso 2 y/o el paso 3. El paso 6, y último, consiste en la comunicación. Comunicar y presentar el problema, su solución y su utilidad. Dar a conocer los resultados y la eficacia de la solución a la comunidad. En este último paso también debemos evaluar la solución y su utilidad y, si es necesario, repetir el paso 2 y/o 3.

Esta metodología DSRM es una buena opción para adoptar en el desarrollo de nuestro trabajo. En nuestro estudio, suele haber dos pequeños grupos de alumnos que comienzan a aprender a programar. No nos parece adecuado utilizar un grupo de prueba y un grupo de control. No queremos adoptar una

estrategia en la que sólo el grupo experimental utilice conceptos, métodos y medios tecnológicos para ayudar al alumno a motivarse y adquirir conocimientos.

1.4. Contexto

Esta investigación se ha realizado dentro del programa de doctorado en Ingeniería Informática de la Universidad de Salamanca. Formalmente, con el plan de estudios: D015 - Ingeniería Informática (R.D. 99/2011), en la Escuela de Doctorado "Studii Salamantini", de la Universidad de Salamanca, a partir del curso académico 2016 - 2017. El trabajo realizado contó con la participación y colaboración del grupo de investigación GRupo de Investigación en InterAcción y eLearning (GRIAL) [26]. El estudio y aplicación práctica se realizó en la Escola Superior de Tecnologia e Gestão del Instituto Politécnico de Guarda, dentro de la asignatura Introducción a la Programación de 1º curso de Ingeniería Informática.

La experiencia profesional de más de 32 años de docencia, siempre como profesor de la unidad curricular de Introducción a la Programación, hace que el interés por esta materia sea relevante, y a la vez de relevancia científica, en el desarrollo de estudios para mejorar el proceso de enseñanza y el aprendizaje de la iniciación a la programación.

2

Estado de la cuestión

El objetivo principal de la enseñanza inicial de programación es proporcionar a los y las estudiantes los conocimientos necesarios para construir programas informáticos capaces de resolver problemas. Este proceso requiere la adquisición de un gran conjunto de conocimientos en un corto espacio de tiempo. Este capítulo presenta el estado del arte en el área de enseñanza y el aprendizaje de la iniciación a la programación. A través de una revisión de la literatura, se buscó identificar los temas más discutidos, qué temas se identificaron como problemas, qué soluciones se propusieron, las metodologías de enseñanza y aprendizaje utilizadas, y también, la identificación de trabajos caracterizados como buenas prácticas en el área.

La acción de transmitir conocimientos y habilidades en el área de la programación es un área relativamente reciente, si se compara con otras áreas del conocimiento, como las matemáticas o la física. La enseñanza de la programación requiere diferentes metodologías.

Ningún método o herramienta tendrá efecto sin la motivación y el trabajo duro de todas las personas involucradas en el proceso de enseñanza y aprendizaje de la programación. Sin embargo, la falta de compromiso, dedicación, perseverancia, esfuerzo y trabajo, mucho trabajo, por parte del estudiantado puede comprometer el éxito en el curso. Los métodos y herramientas son solo medios que pueden ayudar y fomentar el proceso. A todos los grandes músicos o deportistas el talento ayuda mucho, pero hay que trabajar duro. Es necesario practicar y trabajar duro todos los días. Se requiere analizar y evaluar la realimentación sobre nuestro trabajo para poder mejorar.

En este capítulo abordamos el estado de la cuestión en el área de la enseñanza y aprendizaje de la programación, analizamos algunos de los temas más relevantes y que consideramos más pertinentes para una posible aplicación en nuestro trabajo. Abarcamos diez temas divididos en igual número de apartados. Comenzamos con la presentación del pensamiento computacional y su contribución al desarrollo de técnicas de resolución de problemas. En la sección dos, desarrollamos el tema de los conceptos erróneos y las dificultades preconcebidas del estudiantado al aprender inicialmente a programar, seguido de la necesidad de identificar a los y las estudiantes con aptitudes para la programación. En el cuarto apartado analizamos el trabajo en el área del aprendizaje autorregulado. En el apartado cinco se analizan los principales factores que favorecen el éxito del alumnado en el estudio de la programación. En el sexto apartado describimos el trabajo realizado con el objetivo de ayudar a los y las estudiantes a adquirir habilidades de resolución de problemas. La codificación en vivo en la enseñanza de la programación introductoria se explora en la sección siete. La sección ocho describe los errores más comunes cometidos por principiantes en programación. En la sección nueve, describimos el uso y los resultados de la aplicación de las actividades de los *Parson Problems* en la enseñanza inicial de programación. Finalmente, en el último apartado abordamos la gamificación y su importante papel en la enseñanza y aprendizaje de la programación.

3

Aplicación de *Design Science* *Research Methodology* en Introducción a la Programación

Índice

3.1. Identificación de problemas y motivación	19
3.2. Definición de objetivos para una solución	20
3.3. Diseño y desarrollo	21
3.4. Demostración	23
3.5. Evaluación	24
3.6. Comunicación	25

En este capítulo presentamos el conjunto de operaciones sistemáticas y racionales a realizar para lograr nuestro objetivo, trazar el camino necesario, detectar y corregir errores en el proceso de enseñanza y el aprendizaje de la iniciación a la programación. Describimos la aplicación de nuestra metodología de investigación, *Design Science Research Methodology*, en el desarrollo de nuestra propuesta inicial de enseñanza y aprendizaje de programación, teniendo en cuenta los objetivos definidos, reforzados por las características muy particulares de nuestro grupo de estudiantes, los problemas y dificultades identificadas, los diferentes aspectos de la literatura analizada y la experiencia vivida a lo largo de los años de enseñanza de la introducción a la programación.

Como se mencionó anteriormente, elegimos DSRM como nuestra metodología de investigación. Esta metodología consta de una secuencia de seis actividades: identificar y motivar el problema; en la definición de los objetivos para una solución; en la creación del artefacto; en manifestación; en evaluación; en la comunicación y presentación de resultados. Por las razones y características de nuestro grupo de estudio, antes enumeradas, consideramos que es la mejor opción para desarrollar y alcanzar nuestros objetivos.

Identificada la metodología a adoptar, describimos los seis pasos que la caracterizan, en cada uno de los apartados que componen este capítulo. Describimos detalladamente su aplicación con el fin de producir información confiable y verificable para lograr nuestros objetivos y en consecuencia minimizar nuestro problema.

3.1. Identificación de problemas y motivación

En esta primera etapa se debe definir el problema y justificar la importancia de una solución. Enmarque la importancia del problema e identifique las debilidades en las soluciones actuales.

En el capítulo 1 - Introducción, identificamos el problema principal es que los estudiantes no adquieren habilidades básicas para introducir la programación. Y, señalamos como principales causas del problema la débil motivación del estudiantado por el curso, la débil formación general, especialmente en matemáticas, la débil formación general y motivación de los estudiantes de PALOP, las ausencias a clases, la dificultad para construir y aplicar las diferentes estructuras del lenguaje de programación, la dificultad para construir una estrategia de resolución de problemas y transformar esta estrategia en programas informáticos.

También señalamos algunas de las posibles razones del problema. En este apartado complementamos la información con la presentación de datos que corroboraron nuestras afirmaciones.

Identificamos como una de las causas del problema la escasa motivación y formación general de los estudiantes, particularmente en matemáticas. Los resultados de las postulaciones a educación superior entre los años 2018 y 2022 revelan que la carrera de ingeniería informática del IPG no es la

primera opción de elección entre los estudiantes, en promedio solo el 37 % de los estudiantes eligieron la carrera como primera opción. Esto puede indicar una falta de motivación para el curso. Por otro lado, en la tabla mencionada también podemos ver que el promedio de la solicitud está entre 11.45 y 13.48 valores, y el promedio de la prueba de ingreso requerida, Matemáticas, está en el orden de 10.62 y 11.96 valores. Estos valores revelan una pobre formación general de los estudiantes. Además de estos indicadores, cabe destacar la existencia de un elevado número de estudiantes asistentes al curso, provenientes de concursos especiales, del orden del 31 %, destacando los estudiantes de PALOP, con graves deficiencias en su formación general y dificultades en su adaptación cultural.

En este proceso de identificación del problema y motivación, el porcentaje de estudiantes aprobados sobre el total de la matrícula es del orden del 30 %, en la unidad curricular de introducción a la programación, de la carrera de ingeniería informática, de la IPG, en los últimos 6 años. Otro dato preocupante es el número de estudiantes evaluados. Este valor ronda el 60 % de quienes están matriculados, valor muy similar al número de personas que asisten a clases.

En esta etapa, identificamos el problema y lo justificamos presentando los resultados de los últimos años de la unidad curricular de introducción a la programación. Estos resultados son de enorme importancia en el desarrollo general del estudiante de la carrera de ingeniería informática. No solo por la importancia que tienen las habilidades de programación, sino también por la fuerte desmotivación que puede provocar a los y las estudiantes en su formación como profesionales de la ingeniería informática. Todo esto combinado con el conjunto de factores que constituyen la nueva vida de un estudiante en la educación superior.

Identificamos como debilidades en nuestra programación del proceso de enseñanza y aprendizaje, el bajo nivel de conocimiento de nuestros estudiantes, la dificultad del docente para seguir todo el proceso de enseñanza aprendizaje y la evolución de cada estudiante, poca interacción entre estudiantes y docentes, los y las estudiantes no verbalizan sus dificultades y un gran porcentaje de estudiantes no asiste a clases. Todo ello dando como resultado altos índices de fracaso en la evaluación de habilidades de programación.

La principal motivación para desarrollar esta investigación es minimizar todos estos problemas. Hacer cada vez mejor para el aprendizaje de los estudiantes. Permitir que un mayor número de estudiantes alcancen sus objetivos cada año.

3.2. Definición de objetivos para una solución

La segunda etapa de nuestra metodología de investigación consiste en definir los objetivos que conducen a una solución. En esta etapa se deben definir objetivos generales como viabilidad y desempeño, y objetivos más específicos que debe incluir la solución al problema. También deben considerarse los

métodos, teorías y tecnologías que pueden ayudar a definir los objetivos.

Comenzamos con el objetivo de crear un prólogo de programación con el finalidad de desarrollar algunos conceptos del pensamiento computacional, y de una manera más informal, despertar el gusto y el interés por la programación. A continuación, se trazan objetivos que permitieron el desarrollo de habilidades en la aplicación de las diferentes estructuras que constituyen un lenguaje de programación y, en consecuencia, la construcción de programas informáticos para la resolución de problemas. En esta etapa, identificamos el uso de un gran conjunto de *Multiple Choice Question* (MCQ), *Parson Problems* (PP) y actividades de codificación como de gran importancia para que los y las estudiantes practiquen y consoliden sus conocimientos. Con el gran volumen de información que generan los estudiantes al resolver las diferentes actividades, definimos el objetivo de construir un medio tecnológico capaz de ayudar a estudiantes y docentes en la gestión de todas las actividades del proceso de enseñanza y aprendizaje. Por otro lado, consideramos que los datos producidos podrían utilizarse para predecir el fracaso del estudiantado y, para ello, nos planteamos un nuevo objetivo. En la siguiente etapa describimos el desarrollo de los objetivos definidos.

3.3. Diseño y desarrollo

En esta etapa describimos el desarrollo y consecución de los objetivos definidos, en tres áreas esenciales. Proporcionamos una descripción detallada de la implementación de los objetivos propuestos para la construcción de nuestra metodología de enseñanza y aprendizaje inicial de la programación.

El prólogo a la enseñanza y el aprendizaje de la programación, la primera de las áreas esenciales, debe incluir un conjunto diverso de actividades con el propósito de practicar y mejorar las habilidades cognitivas de cada estudiante, para el aprendizaje de la iniciación a la programación. La enseñanza de cómo pensar y la aplicación de técnicas de pensamiento computacional como proceso estratégico para la resolución de problemas, a través de un conjunto de actividades prácticas de manipulación y preferencias de los estudiantes, debe considerarse fundamental para alcanzar los objetivos de este prólogo a la enseñanza de la programación.

En el segundo área, describimos los pasos para lograr nuestros objetivos, en cuanto a los problemas en la adquisición de habilidades en el lenguaje de programación C, y su aplicación en la construcción de programas informáticos para la resolución de problemas. Al identificar el problema encontramos que existen dificultades para identificar y aplicar los conceptos básicos del lenguaje de programación. Descubrimos que los y las estudiantes cometen errores en conceptos iniciales del lenguaje de programación, como al definir tipos de datos, definir identificadores o formatear instrucciones de entrada y salida. Para afrontar este problema, nos planteamos como objetivo implementar un gran conjunto de actividades, con *feedback* inmediata, del tipo *Multiple Choice Question* (MCQ).

En este segundo ámbito también destacamos el uso de actividades tipo Parson Problems (PP), para que los estudiantes practiquen y consoliden sus conocimientos. Las actividades *Parson Problems* son ejercicios en los que se debe ordenar las instrucciones de un programa. Según [27–30], aprender y practicar usando actividades de *Parson Problems* se considera un buen ejercicio para cualquiera que comience a aprender programación. Este tipo de actividades despierta un gran interés y curiosidad en los estudiantes. Los estudiantes se sienten motivados al practicar este tipo de actividades, ya que no cometen errores de sintaxis y no requiere un gran esfuerzo cognitivo para definir la estrategia de resolución del problema. Desde un punto de vista estratégico para el proceso de enseñanza y aprendizaje, aprovechamos el entusiasmo en la resolución de estas actividades, e incluimos un conjunto de operaciones y técnicas de programación clásicas, para que el estudiantado se familiarice con estas técnicas y prácticas habituales.

Para lograr el objetivo relativo a los problemas en la adquisición de habilidades de programación y su aplicación en la resolución de problemas, finalizamos nuestra propuesta con un conjunto de actividades que denominamos actividades de codificación. En estas actividades de codificación, utilizamos un conjunto de casos de prueba para evaluar la solución propuesta por el estudiante, con retroalimentación inmediata.

El tercer y último ámbito imprescindible para la consecución de nuestros objetivos consiste en describir la construcción de la herramienta tecnológica, denominada **HTProgramming** – *Help To Programming*, que permitirá gestionar todo el proceso de enseñanza y aprendizaje. La aplicación desarrollada consta de dos módulos, el módulo de administración y el módulo de estudiantes.

El módulo de administración, lo utiliza el profesorado para gestionar todo el proceso de enseñanza y aprendizaje. Es en este módulo donde el personal docente crea y configura las actividades a presentar al estudiantado, pudiendo consultar los resultados de sus actividades, así como su resolución e intentos de resolución. Los resultados obtenidos por los y las estudiantes en diferentes actividades se utilizan en el modelo de red neuronal que predice el fracaso estudiantil, con el objetivo de indicar qué estudiantes son los que más necesitan atención y apoyo.

El módulo de estudiante es utilizado por todos matriculado en el curso de introducción a la programación de la carrera de ingeniería informática, de IPG. Cada estudiante debe pasar por un proceso de identificación antes de la aplicación *Help To Programming* (HTProgramming). Una vez validada la entrada, ya puede comenzar a utilizarla. El perfil de estudiante tiene a su disposición un conjunto organizado de actividades para complementar su proceso de aprendizaje. De esta forma, un o una estudiante podrá practicar, probar y evaluar sus conocimientos en las diferentes áreas de conocimiento que componen el aprendizaje inicial de programación. Los y las estudiantes podrán realizar las diferentes actividades a su propio ritmo y sin miedo a cometer errores. Por cada actividad realizada, se recibe inmediatamente una evaluación. Dependiendo del resultado obtenido, se podrán recibir sugerencias de

lectura o revisión, o sugerencias de nuevas actividades.

Finalmente, describimos el uso del modelo predictivo de fracaso del estudiantado. Definimos el conjunto de características que mejor caracterizan al modelo y reportamos su funcionamiento y objetivo, que es detectar precozmente las dificultades y problemas del alumnado. Una vez que alguien ha sido señalado, es posible intervenir rápidamente para que reciba ayuda, minimizando la posibilidad de fallar en el aprendizaje.

3.4. Demostración

El objetivo de esta sección es demostrar que el uso de las estrategias y la implementación de los objetivos propuestos permiten resolver una o más instancias del problema. En otras palabras, el conjunto de medidas adoptadas para mejorar las habilidades básicas de los y las estudiantes para la construcción de programas informáticos produce resultados significativos.

Para consolidar los conceptos iniciales del aprendizaje del lenguaje de programación C, implementamos la resolución de actividades tipo MCQ, con el objetivo de consolidar sus habilidades, pero con el objetivo de que cada persona adquiera el gusto y el interés por la programación.

Definimos como nuestro objetivo la implementación de actividades tipo PP. Con este tipo de actividades pretendemos que los y las estudiantes complementen sus conocimientos en la aplicación y práctica de conceptos de modelos para la resolución de problemas informáticos.

La práctica constante de actividades de codificación es de especial relevancia a la hora de aprender a programar. En particular, un conjunto de actividades orientadas de acuerdo con los objetivos definidos para la asignatura. Por otro lado, es importante tener realimentación, para que cada cual pueda evaluar sus conocimientos y mantener altos sus niveles de motivación. La retroalimentación es igualmente importante para el profesorado, para monitorear y evaluar el desarrollo del proceso de enseñanza y aprendizaje de la programación. La existencia de un conjunto variado y organizado de actividades de codificación con realimentación inmediata, e integradas en la aplicación Help To Programming (HTProgramming), es una de las estrategias utilizadas para lograr nuestro objetivo de adquirir habilidades básicas para la resolución de problemas informáticos.

En esta etapa de demostrar que el uso de estrategias y la implementación de los objetivos propuestos permiten resolver una o más instancias del problema, destacamos la asociación que se realiza entre las actividades PP. Como se mencionó, las actividades PP deben servir como introducción y presentación de técnicas de resolución de problemas y luego resolución de actividades de codificación para poder aplicar y consolidar estas habilidades.

3.5. Evaluación

En esta etapa de aplicación de DSRM en el aprendizaje de la iniciación a la programación, debemos medir y evaluar la estrategia adoptada, así como en qué medida el artefacto tecnológico apoya una solución al problema.

Abordamos este proceso de evaluación en tres áreas: la evaluación por parte del estudiantado en el uso de la aplicación HTProgramming, la evaluación del modelo predictivo de fracaso de los estudiantes y, finalmente, la evaluación basada en los resultados finales en la unidad curricular de introducción a la programación, del curso de ingeniería informática, de IPG.

Analizando los resultados arrojados por los estudiantes en la encuesta de opinión sobre el uso de la aplicación HTProgramming, encontramos que el 79,2 % del estudiantado calificó el uso de la aplicación como *Importante* o *Muy Importante* HTProgramming en actividades como *Conceptos básicos* y *Problemas de Parson*. Con respecto a las actividades de *Codificación*, el 67,9 % del estudiantado la calificó el uso de la aplicación en su proceso de aprendizaje de programación como *Importante* o *Muy Importante*. Atendiendo a la importancia de la aplicación, en general, y su contribución a su proceso inicial de enseñanza de la programación, el 67,9 % de los estudiantes la clasificaron como *Importante* o *Muy importante*, y el 32,1 % como *Muy poco importante* o *No muy importante*. Nadie respondió con el valor *Sin importancia* al uso de la aplicación HTProgramming.

Para evaluar el modelo predictivo, una de las medidas más utilizadas es el porcentaje de valores clasificados correctamente, es decir, la precisión de clasificación (en inglés *accuracy*) [31–33]. El cual representa la relación entre el número de valores clasificados correctamente por el modelo sobre el número total de valores clasificados. En nuestro caso solo hay dos resultados: fracaso y éxito. Por lo tanto, para calcular y evaluar el desempeño del modelo de clasificación podemos utilizar una matriz de confusión, o también llamada matriz de error.

De la aplicación del modelo predictivo de fracaso del estudiantado, resultaron 4 redes neuronales, una por cada fase, las cuales serán aplicadas a los conjuntos de pruebas del año académico 2022/2023. La precisión de la clasificación aumentó alrededor de un 23 %, es decir, pasó de un valor cercano al 75 % a un valor del 92 %. En los resultados de la fase 1 podemos ver que de las 23 personas que no tuvieron éxito, fueron identificados por el modelo 21 (VP). El modelo identificó 25 estudiantes con éxito (VN), de un total de 27 estudiantes. Solo 2 estudiantes se identificaron como que habían tenido éxito y realmente reprobados (FN). En la fase final, se identificaron los 23 estudiantes que no tuvieron éxito y solo 2 no fueron identificados como tales, de un total de 27 estudiantes. Los resultados obtenidos por las métricas utilizadas son muy buenos. Podemos destacar los valores obtenidos para la precisión de clasificación del modelo con un 96 %, la sensibilidad con un 100 % y, finalmente, el valor *F-score* con un 95,8 %.

De acuerdo con los resultados obtenidos, podemos afirmar que el modelo predictivo del estudiantado

do es una buena herramienta para ayudar al profesorado a identificar estudiantes con dificultades en el proceso de enseñanza y aprendizaje inicial de la programación.

Los resultados de la unidad curricular de introducción a la programación son un indicador muy relevante. Además del número de estudiantes matriculados, evaluados y aprobados, también podemos analizar el porcentaje de estudiantes aprobados sobre el total matriculados y el porcentaje de estudiantes aprobados sobre el total de evaluados. Los valores obtenidos en los cursos 2019 y 2020, años de pandemia, reflejan algo de lo que pasamos. En el primer año de la pandemia, todos se enfrentaron a la adaptación a un nuevo sistema de enseñanza y evaluación, con resultados muy bajos. En el segundo año de pandemia, valores muy altos, quizás por el hecho de que la mayor parte del proceso de enseñanza y aprendizaje se ha trasladado a la modalidad *online* y todas las consecuencias y facilidades que ello ha podido provocar. Encontramos que el porcentaje de estudiantes que aprobó sobre el total de matriculados con un valor del 25,0 %, en 2021, subió a un valor del 33,3 %, en 2022, es decir, un incremento del 32 %. El resultado del porcentaje de aprobados sobre el total de evaluados en 2021, del 53,3 %, ha pasado al 76,6 %, en 2022, lo que supone un aumento del 45 %. Destacamos los cursos 2021 y 2022, ya que consideramos que la herramienta de ayuda al aprendizaje inicial de programación desarrollada, la aplicación *HTProgramming*, se encuentra en un estado de desarrollo más avanzado.

Los resultados respecto al uso de la aplicación *HTProgramming*, en el contexto del proceso de enseñanza y aprendizaje, a pesar de que su uso fue bajo, los estudiantes que la utilizaron, entre el 68 % y el 70 % de los estudiantes, obtuvieron aprobación.

Si bien hay valores a mejorar, como el número de estudiantes que fue evaluado y el número de estudiantes que fue aprobado, los resultados de éxito de los y las estudiantes han ido mejorando. Por tanto, promover el uso de la herramienta por parte del estudiantado debe ser más claro y preciso. Comunicar estos resultados a los y las estudiantes puede actuar como un buen vehículo de difusión.

3.6. Comunicación

La última etapa de *DSRM* es la comunicación, difusión y presentación del problema, sus soluciones y la utilidad del trabajo desarrollado. Este trabajo es el resultado de varios años de trabajo y muchos más años de experiencia en la enseñanza de la iniciación a la programación.

En este apartado describimos los trabajos y publicaciones que hemos realizado a lo largo de esta trayectoria investigadora, en favor de la mejora del proceso de enseñanza y el aprendizaje de la iniciación a la programación. Con el propósito de mejorar significativamente las habilidades del estudiantado en la construcción de programas informáticos, capaces de resolver pequeños problemas.

4

Discusión

En este capítulo se presentarán y discutirán los resultados de este trabajo de investigación. Teniendo en cuenta la revisión bibliográfica y los resultados obtenidos, principalmente a través de las actividades realizadas por los estudiantes, las encuestas realizadas y a través de la observación, se buscó interpretar los resultados obtenidos y sus implicaciones en el proceso de la enseñanza y el aprendizaje de la iniciación a la programación.

Al analizar la implementación del prólogo de la programación de aprendizaje, consideramos que los resultados son muy positivos. Verificamos diferencias significativas en la motivación y el conocimiento de los estudiantes, antes y después de practicar las actividades implementadas. Sin embargo, como se mencionó, se aplica en un corto espacio de tiempo. Nuestra sugerencia es crear una actividad extracurricular con la temática de pensamiento computacional, a realizarse durante todo el semestre. Pero su implementación es difícil de lograr. La falta de tiempo mencionada por el estudiantado, la falta de disponibilidad de más profesorado, el hecho de que sea opcional y el estudiantado no participe, o incluso simples problemas administrativos, son algunas de las razones que justifican la dificultad de implementación.

También incluimos, en este apartado de discusión y análisis del perfil del estudiante, los resultados del ausentismo escolar. Los valores relacionados con el ausentismo en las evaluaciones presentadas confirman este problema. De media, el 52,7 % de las personas matriculadas en la unidad curricular de introducción a la programación participan en la evaluación. El número medio de estudiantes presentes en las clases es de 39, repartidos en dos clases, lo que corresponde a un porcentaje medio del 40,5 %. Estos valores expresan el abandono y la desmotivación del estudiantado ante las dificultades encontradas en el proceso inicial de enseñanza y aprendizaje de la programación. El número de asistencias a clases va disminuyendo paulatinamente a lo largo del semestre, con una reducción significativa en el último mes de clases. Señalamos como motivo de este fenómeno el aumento de las dificultades y exigencias de conceptos de programación más avanzados. La suma de dificultades y problemas se agrava y culmina con el abandono de la unidad curricular. Como se destaca, los contenidos de aprendizaje de programación están fuertemente interconectados, una deficiencia en uno de estos contenidos puede comprometer todo el proceso de aprendizaje. Respecto a la diferencia entre el porcentaje medio de estudiantes presentes en clases (40,5 %) y el porcentaje medio de alumnos evaluados (52,7 %), señalamos como justificación el número de repetidores que no asisten a clases, pero se presentan a evaluación. Con nuestro perfil estudiantil y el ausentismo escolar añadido, se cumplen las condiciones del fracaso académico.

Analizamos la adopción del lenguaje de programación C. Podemos preguntarnos si los problemas y dificultades en el aprendizaje inicial de programación se deben al uso del lenguaje C. En esta discusión sobre el lenguaje de programación y su aprendizaje inicial, encontramos que en todos los lenguajes hay problemas. El uso del lenguaje C no está fuera de sintonía con la realidad. Podemos considerar usar

otro lenguaje de programación, pero la mayoría de los problemas persistirán. Los problemas de falta de motivación del estudiantado por el curso, mala formación general, ausencias a clases, problemas en la capacidad de resolución de problemas y, también, problemas para transformar la estrategia de solución en un programa de computadora, no se pueden resolver sólo con el uso de otro. lenguaje de programación.

Las actividades implementadas presentan buenos resultados en el logro de nuestros objetivos. El conjunto de actividades tipo MCQ permite al estudiantado practicar un número variado de conceptos sencillos, de una forma muy activa y motivadora. Las actividades de tipo PP permiten al alumnado adquirir un conjunto de conceptos y prácticas comunes para el aprendizaje inicial de la programación, de una forma muy atractiva y de interés para los estudiantes. Finalmente, las actividades de codificación guiadas por el profesorado, divididas por niveles y temáticas, con retroalimentación inmediata, permiten al estudiante aplicar y consolidar sus conocimientos al aprender a programar.

Basado en el trabajo desarrollado y los resultados obtenidos en el modelo predictivo del fracaso del estudiantado, debemos valorar esta herramienta como una ayuda importante en el proceso de la enseñanza y el aprendizaje de la iniciación a la programación.

Para cumplir con el objetivo de realimentación de forma inmediata las actividades realizadas por los y las estudiantes, es imprescindible la construcción de una herramienta automática. Por otro lado, registrar las actividades realizadas a lo largo del proceso de enseñanza y aprendizaje resulta de especial importancia para el docente. Ya que permite evaluar el progreso de cada alumno, evaluar las dificultades y proponer actividades según las necesidades individuales y colectivas. En evaluación final, la aplicación se desarrolló de acuerdo a las necesidades e implementación de ideas, por lo que no cuenta con una estructura bien definida y organizada. También necesita un diseño y una usabilidad nuevos y más atractivos. Sin embargo, satisface nuestras necesidades de brindar un conjunto de actividades con realimentación inmediata, registrar las calificaciones obtenidas, calcular el perfil del estudiante y además permitir el uso del modelo predictivo de fracaso del estudiantado.

5

Conclusión

Índice

5.1. Principales Contribuciones	33
5.2. Trabajo Futuro	36

En este capítulo final presentamos nuestras conclusiones sobre el trabajo realizado. Destacamos las principales aportaciones de nuestro trabajo al responder a las preguntas de investigación. También abordamos las posibilidades de desarrollo y trabajos futuros, haciendo un énfasis en la implicación de ChatGPT en la enseñanza y el aprendizaje de la iniciación a la programación. Por último, presentamos las publicaciones realizadas con el objetivo de difundir y presentar los resultados y proporcionar una oportunidad para discutir el trabajo realizado con la comunidad académica.

5.1. Principales Contribuciones

El objetivo principal de este trabajo ha sido mejorar todo el proceso de enseñanza y aprendizaje de la programación, con el fin de que el estudiantado adquieran conocimientos básicos de programación para construir programas informáticos. Como resultado, un mayor número de estudiantes supera con éxito el curso de Introducción a la Programación. Destacar la importancia de una buena base de conocimientos en introducción a la programación para un buen rendimiento en el curso de ingeniería informática.

Objetivamente, dada la complejidad de la materia, nuestro trabajo consistió en investigar qué soluciones podemos incluir en nuestro proceso para minimizar los problemas en la enseñanza y aprendizaje inicial de la programación. Este trabajo tiene un origen y un destino muy particular asociado a nuestro perfil de estudiantes. Es el resultado de varios años de docencia e investigación. Esperamos que nuestro trabajo sea una buena aportación para todos aquellos que se enfrentan a problemas en la enseñanza y aprendizaje de la programación.

A continuación, presentamos las principales contribuciones de este trabajo, que nos permiten responder a las preguntas de investigación.

Respecto a la primera pregunta de investigación ***Q11 - ¿Qué conjunto de actividades se deben utilizar para mejorar las competencias en el aprendizaje de la iniciación a la programación?*** Nuestro objetivo era investigar qué conjunto de actividades deberían utilizarse en nuestro proceso de enseñanza y el aprendizaje de la iniciación a la programación para minimizar nuestros problemas y mejorar significativamente las competencias básicas de los estudiantes en la resolución de problemas y la construcción de programas informáticos.

En respuesta a la primera pregunta de investigación planteada y principal aportación de nuestro trabajo, identificamos un incremento de más del 100 %, concretamente del 107,2 %, en el éxito de los y las estudiantes. Esta cifra se obtiene calculando el porcentaje medio de estudiantes con éxito por número total de evaluados para el curso 2017-2019 (34,5 %) y el curso 2020-2022 (71,5 %). Esta es la principal evidencia de nuestro trabajo. Se ha demostrado que a través del conjunto de medidas adoptadas, las competencias iniciales de aprendizaje en programación han mejorado significativamente. Y, en consecuencia, el éxito de los alumnos en el curso de introducción a la programación.

El conjunto de actividades adoptadas en el proceso de enseñanza contribuyó significativamente a este resultado. Las propuestas de actividades de respuesta múltiple, en inglés *Multiple Choice Question*, el conjunto de actividades del tipo *Parson Problems* y las actividades de codificación, todas ellas con realimentación inmediata, resultaron especialmente importantes para los resultados obtenidos. En respuesta a la pregunta de investigación planteada, destacamos claramente estas actividades como las que más contribuyeron a mejorar las habilidades de programación.

Las actividades de respuesta múltiple permiten a los alumnos practicar y consolidar su aprendizaje en un conjunto de conceptos iniciales del lenguaje de programación, que, como hemos visto, tienen una fuerte influencia en la evolución de este aprendizaje.

Las actividades tipo *Parson Problems* juegan un papel importante en este proceso de enseñanza y aprendizaje. No solo el estudiantado muestra un especial interés en la resolución de estas actividades, sino que además permiten transmitir un conjunto de conceptos modelo de resolución de problemas y construcción de programas, que facilitan la adquisición y consolidación de las destrezas del estudiantado en la aplicación de las diferentes estructuras de programación a la construcción de programas informáticos.

Las actividades de codificación desempeñan un papel importante en todo el proceso de enseñanza y aprendizaje de la programación. Creemos que la práctica constante y el trabajo duro en la creación de programas es la mejor manera de aprender. La resolución de una amplia variedad de ejercicios de codificación, con realimentación inmediata, permite al el estudiantado adquirir experiencia en el uso de un conjunto de técnicas y patrones de resolución. Este conjunto de técnicas y patrones de resolución ayuda a combatir sus dificultades para aplicar las distintas estructuras del lenguaje de programación, las dificultades para resolver problemas y las dificultades para transformar la estrategia de resolución en un programa informático.

Hay que destacar lo prólogo de la enseñanza y el aprendizaje de la programación. Como ya se ha mencionado, poco podemos hacer, o controlar, cuando se trata de los conocimientos previos de nuestro estudiantado, o a la hora de motivar al estudiantado para que elijan nuestro curso e institución educativa. Sin embargo, sí podemos proporcionar un buen ambiente de aprendizaje, fortalecer y fomentar el compañerismo, la comunicación y la amistad entre estudiantado y profesorado a través del conjunto de actividades y juegos propuestos en este prólogo. Por otro lado, se proponen actividades que permiten al estudiantado conocer, practicar y desarrollar habilidades de pensamiento computacional. Y, quizás lo más importante, estimular el gusto por la programación.

En cuanto a la segunda pregunta de investigación: **Q12 - ¿Cómo podemos predecir el fracaso de los y las estudiantes?** Nuestro objetivo era identificar algún tipo de comportamiento, conjunto de características o perfil del estudiantado que pudiera indicar de antemano su fracaso. Y con esta información, poder actuar rápidamente para ayudar a las personas en riesgo de fracaso.

Para responder a esta pregunta, aplicamos un modelo de predicción del fracaso del estudiantado basado en una red neuronal. Se evaluaron varios conjuntos de datos de estudiantes, con el objetivo de construir el mejor conjunto de características para alcanzar nuestro objetivo. A la vista de los resultados obtenidos en la evaluación del modelo de predicción del fracaso estudiantil, con el conjunto de características seleccionadas, podemos concluir que es un excelente aliado para nuestro proceso de enseñanza y aprendizaje. Con este modelo será posible predecir qué alumnos tienen más probabilidades de fracasar, e intervenir rápidamente para ayudarles, minimizando sus problemas y dificultades, con el fin de combatir la predicción.

Por último, analizamos la tercera pregunta de investigación: ***Q13 - ¿Qué puede aportar el uso de una herramienta tecnológica al proceso de enseñanza y el aprendizaje de la iniciación a la programación?***

A lo largo de este trabajo, hemos demostrado que el uso de una herramienta tecnológica que centre en sí misma un gran conjunto de actividades, que permita evaluar y retroalimentar de forma inmediata al alumno y al profesor, es de gran importancia y esencial para la realización de nuestro objetivo. En nuestro trabajo, hemos construido y descrito una plataforma, HTProgramming, con el fin de satisfacer nuestras necesidades y los problemas que hemos identificado. HTProgramming integra diferentes tipos de actividades, desarrolladas específicamente por el profesor, con fines muy concretos y de acuerdo con las características muy particulares de nuestro grupo de trabajo. HTProgramming es una herramienta muy importante para la recogida de datos que se utilizarán en el sistema de predicción del fracaso escolar.

Con esta herramienta, HTProgramming, hemos ayudado a resolver algunos de los problemas que se han puesto de manifiesto. Como la dificultad en el seguimiento de las actividades realizadas por los alumnos y la respectiva evolución de su proceso de aprendizaje. En particular, aquellas personas que no dan a conocer sus acciones ni verbalizan sus dificultades. HTProgramming también permite superar la falta de realimentación inmediata que sienten muchas personas, así como motivar e incentivar el trabajo autónomo. Con esta herramienta, los y las estudiantes pueden comprobar y evaluar sus conocimientos y también practicar actividades a su propio ritmo y repetirlas tantas veces como necesiten.

Desde el punto de vista del profesorado, esta herramienta ayuda a supervisar todo el proceso de enseñanza y aprendizaje. El profesorado tiene una visión general de los resultados y las actividades realizadas por el alumnado. Con estos indicadores, el profesorado puede intervenir para ayudar mejor a las personas, ya sea mediante el envío de mensajes o a través de un contacto más individualizado.

En síntesis, la realización y utilización de la herramienta tecnológica es preponderante en todo el proceso de enseñanza y aprendizaje propuesto en este trabajo.

5.2. Trabajo Futuro

La enseñanza y aprendizaje de la programación es un área de investigación muy amplia y compleja. Se pueden seguir y probar muchos enfoques y líneas de investigación, y dependiendo de las características de los alumnos y del sistema de enseñanza, algunos enfoques funcionan mejor que otros. Sin embargo, es importante que el profesorado sea consciente de que existen problemas y estén abiertos a nuevas ideas y cambios en el sistema de programación de la enseñanza ante las dificultades y problemas del estudiantado.

Se debería mantener un debate fructífero entre todas las personas involucradas en el sistema de enseñanza y aprendizaje inicial de la programación, no solo por lo que representa la asignatura y su importancia dentro de un curso de informática, sino también por la poca importancia que en la gran mayoría de los casos se le da a la asignatura a la hora de asignar profesorado. Las asignaturas de iniciación a la programación deberían asignarse a profesorado con más experiencia y no al profesorado más joven y con menos experiencia en el área, ya que está en el inicio de su carrera profesional.

En lo que respecta a nuestra propuesta, es necesario introducir mejoras significativas en la aplicación desarrollada para ayudar a todo el proceso de enseñanza y aprendizaje de la programación, HTProgramming. En particular, en cuanto a la usabilidad y diseño de la aplicación. Por otro lado, su transformación en una aplicación web mejoraría notablemente la velocidad de acceso y respuesta de la aplicación.

Los resultados obtenidos con el sistema de predicción de fracaso de los estudiantes son buenos. Sin embargo, con un mayor volumen de datos, nuestro modelo *machine-learning*, basado en una red neuronal, debería producir mejoras significativas. También hay que prestar especial atención al conjunto de características. Por un lado, deberíamos seleccionar un conjunto de variables que puedan caracterizar mejor al estudiantado, pero que también sean fáciles de obtener mediante un proceso automático.

Tras la aparición de ChatGPT, no podemos evitar reflexionar y analizar su posible uso en el proceso de la enseñanza y el aprendizaje de la iniciación a la programación. A finales de 2022, empezamos a oír hablar mucho de un programa considerado como un gran paso evolutivo en la inteligencia artificial, ChatGPT.

Vivimos un momento único, con reacciones de gran entusiasmo y curiosidad conviviendo con muchos temores y dudas [34, 35]. Especialmente en el ámbito de la enseñanza, donde hay quien ve oportunidades, como la creación de contenidos para facilitar el aprendizaje, un acceso fácil a la información que desean, un asistente virtual de aprendizaje disponible 24 horas al día, 7 días a la semana. Sin embargo, también hay quien encuentra problemas, como el uso deshonesto de la herramienta, el aprendizaje rápido y superficial, las respuestas incorrectas, la posibilidad de sustituir al profesorado y la dependencia tecnológica [36, 37]. En estos momentos, es importante que todos los implicados en la enseñanza y el aprendizaje conozcan y utilicen la tecnología existente y la aprovechen en sus activida-

des.

Ante la reciente aparición de ChatGPT, nos vemos obligados a utilizarlo y probarlo para conocer sus potencialidades o debilidades en la resolución de las actividades propuestas. Si no lo hacemos nosotros, seguro que lo hará el alumnado. Lo importante es saber utilizar la tecnología y sacarle el máximo partido. En este contexto, es importante saber hacer preguntas. Para saber hacer preguntas, hay que tener un conocimiento básico de la materia y pensar de forma crítica. Por lo tanto, en el contexto de la enseñanza y el aprendizaje iniciales de la programación, debe utilizarse con cuidado. Dado que los conocimientos y la madurez del estudiantado en la materia son aún muy frágiles, pueden producirse errores, falta de rigor y dependencia de la tecnología, que adquiridos en esta fase inicial del aprendizaje pueden convertirse en graves problemas en el futuro.

En conclusión, podemos decir que ChatGPT es una buena ayuda para la enseñanza y el aprendizaje de la iniciación a la programación. Debería incluirse en el proceso de enseñanza lo antes posible porque los y las estudiantes ya lo utilizan sin juzgar, solo para facilitar su trabajo. ChatGPT puede ayudar en una amplia gama de ejercicios que suelen utilizarse en la enseñanza inicial de la programación. Sin embargo, el pensamiento crítico y el análisis cuidadoso de las respuestas deben estar siempre presentes, ya que hemos visto que hay algunas alucinaciones por parte de ChatGPT. En las actividades de tipo *Parson Problems*, las respuestas producidas contienen varias imprecisiones, lo que pone en peligro su utilización. ChatGPT produce excelentes resultados cuando se trata de solicitar actividades y ejemplos de aplicación de conocimientos sobre una amplia variedad de temas para el aprendizaje inicial de la programación.

Referencias

- [1] S. Dekhane, X. Xu, and M. Tsoi, "Mobile app development to increase student engagement and problem solving skills," *Journal of Information Systems Education*, vol. 24, pp. 299–308, 01 2013.
- [2] M. McCracken, V. Almstrum, D. Diaz, M. Guzdial, D. Hagan, Y. B.-D. Kolikant, C. Laxer, L. Thomas, I. Utting, and T. Wilusz, "A multi-national, multi-institutional study of assessment of programming skills of first-year cs students," *SIGCSE Bull.*, vol. 33, no. 4, p. 125–180, dec 2001. [Online]. Available: <https://doi.org/10.1145/572139.572181>
- [3] E. Lahtinen, K. Ala-Mutka, and H.-M. Järvinen, "A study of the difficulties of novice programmers," *SIGCSE Bull.*, vol. 37, no. 3, p. 14–18, jun 2005. [Online]. Available: <https://doi.org/10.1145/1151954.1067453>
- [4] D. Sleeman, "The challenges of teaching computer programming," *Commun. ACM*, vol. 29, no. 9, p. 840–841, sep 1986. [Online]. Available: <https://doi.org/10.1145/6592.214913>
- [5] M. McCracken, V. Almstrum, D. Diaz, M. Guzdial, D. Hagan, Y. B.-D. Kolikant, C. Laxer, L. Thomas, I. Utting, and T. Wilusz, "A multi-national, multi-institutional study of assessment of programming skills of first-year cs students," *SIGCSE Bull.*, vol. 33, no. 4, p. 125–180, dec 2001. [Online]. Available: <https://doi.org/10.1145/572139.572181>
- [6] A. Robins, J. Rountree, and N. Rountree, "Learning and teaching programming: A review and discussion," *Computer Science Education*, vol. 13, no. 2, pp. 137–172, jun 2003. [Online]. Available: <https://doi.org/10.1076/csed.13.2.137.14200>
- [7] U. Omer, S. Farooq, and A. Abid, "Introductory programming course: review and future implications," *PeerJ. Computer science*, vol. 7, p. e647, 07 2021.
- [8] J. Bennedsen, "Teaching and learning introductory programming : a model-based approach," phdthesis, University of Oslo, 2008. [Online]. Available: <http://hdl.handle.net/10852/9962>

- [9] A. Robins, J. Rountree, and N. Rountree, "Learning and teaching programming: A review and discussion," *Computer Science Education*, vol. 13, no. 2, pp. 137–172, jun 2003. [Online]. Available: <https://doi.org/10.1076/csed.13.2.137.14200>
- [10] A. d. J. Gomes, "Dificuldades de aprendizagem de programação de computadores : contributos para a sua compreensão e resolução," phdthesis, Faculdade de Ciências e Tecnologia da Universidade de Coimbra, 2010. [Online]. Available: <https://estudogeral.sib.uc.pt/jspui/handle/10316/14586>
- [11] J.-M. Hoc and A. Nguyen-Xuan, "Chapter 2.3 - language semantics, mental models and analogy," in *Psychology of Programming*, J.-M. Hoc, T. Green, R. Samurçay, and D. Gilmore, Eds. London: Academic Press, 1990, pp. 139–156. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/B9780123507723500148>
- [12] T. Jenkins, "On the difficulty of learning to program," in *Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences*, vol. 4, no. 2002. Citeseer, 2002, pp. 53–58.
- [13] A. Gomes and A. Mendes, "A teacher's view about introductory programming teaching and learning: Difficulties, strategies and motivations," in *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*, 2014, pp. 1–8.
- [14] C. Watson and F. W. Li, "Failure rates in introductory programming revisited," in *Proceedings of the 2014 conference on Innovation & technology in computer science education*, 2014, pp. 39–44.
- [15] A. Luxton-Reilly, Simon, I. Albluwi, B. A. Becker, M. Giannakos, A. N. Kumar, L. Ott, J. Paterson, M. J. Scott, J. Sheard, and C. Szabo, "Introductory programming: A systematic literature review," in *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*, ser. ITiCSE 2018 Companion. New York, NY, USA: Association for Computing Machinery, jul 2018, p. 55–106. [Online]. Available: <https://doi.org/10.1145/3293881.3295779>
- [16] S. Fincher, B. Baker, I. Box, Q. Cutts, M. de Raadt, P. Haden, J. Hamer, M. Hamilton, R. Lister, and M. Petre, "Programmed to succeed?: A multi-national, multi-institutional study of introductory programming courses," University of Kent, Canterbury, United Kingdom, Technical report, April 2005. [Online]. Available: <https://kar.kent.ac.uk/14335/>
- [17] S. Shuhidan, M. Hamilton, and D. D'Souza, "A taxonomic study of novice programming summative assessment," *Conferences in Research and Practice in Information Technology Series*, vol. 95, pp. 147–156, 2009.

- [18] J. Bennedsen and M. E. Caspersen, "Failure rates in introductory programming: 12 years later," *ACM Inroads*, vol. 10, no. 2, p. 30–36, apr 2019. [Online]. Available: <https://doi.org/10.1145/3324888>
- [19] M. Ahmadzadeh, D. Elliman, and C. Higgins, "An analysis of patterns of debugging among novice computer science students," *SIGCSE Bull.*, vol. 37, no. 3, p. 84–88, jun 2005. [Online]. Available: <https://doi.org/10.1145/1151954.1067472>
- [20] J. Wing, "Computational thinking," *Communications of the ACM*, vol. 49, pp. 33–35, 03 2006.
- [21] —, "Computational thinking and thinking about computing," *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, vol. 366, no. 1881, pp. 3717–25, 11 2008.
- [22] F. J. García-Peñalvo and J. Cruz-Benito, "Computational thinking in pre-university education," in *Proceedings of the Fourth International Conference on Technological Ecosystems for Enhancing Multiculturality*, ser. TEEM '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 13–17. [Online]. Available: <https://doi.org/10.1145/3012430.3012490>
- [23] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A design science research methodology for information systems research," *Journal of Management Information Systems*, vol. 24, no. 3, pp. 45–77, 2007. [Online]. Available: <https://doi.org/10.2753/MIS0742-1222240302>
- [24] J. Choi, D. L. Nazareth, and H. K. Jain, "Implementing service-oriented architecture in organizations," *Journal of Management Information Systems*, vol. 26, no. 4, pp. 253–286, apr 2010. [Online]. Available: <https://doi.org/10.2753/MIS0742-1222260409>
- [25] G. L. Geerts, "A design science research methodology and its application to accounting information systems research," *International Journal of Accounting Information Systems*, vol. 12, no. 2, pp. 142–151, jun 2011, special Issue on Methodologies in AIS Research. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1467089511000200>
- [26] F. García-Peñalvo, M. J. Conde, R. Therón, A. García-Holgado, F. Martínez Abad, and A. Benito Santos, "le comunicaciones grupo grial," pp. 33–48, 12 2019.
- [27] P. Denny, A. Luxton-Reilly, and B. Simon, "Evaluating a new exam question: Parsons problems," in *Proceedings of the Fourth International Workshop on Computing Education Research*, ser. ICER '08. New York, NY, USA: Association for Computing Machinery, 2008, p. 113–124. [Online]. Available: <https://doi.org/10.1145/1404520.1404532>
- [28] B. J. Ericson, "Adaptive parsons problems with discourse rules," in *Proceedings of the Tenth Annual Conference on International Computing Education Research*, ser. ICER '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 145–146. [Online]. Available: <https://doi.org/10.1145/2632320.2632324>

- [29] B. B. Morrison, L. E. Margulieux, B. Ericson, and M. Guzdial, "Subgoals help students solve parsons problems," in *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, ser. SIGCSE '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 42–47. [Online]. Available: <https://doi.org/10.1145/2839509.2844617>
- [30] Y. Du, A. Luxton-Reilly, and P. Denny, "A review of research on parsons problems," in *Proceedings of the Twenty-Second Australasian Computing Education Conference*, ser. ACE'20. New York, NY, USA: Association for Computing Machinery, 2020, p. 195–202. [Online]. Available: <https://doi.org/10.1145/3373165.3373187>
- [31] "Encyclopedia of machine learning," 2010.
- [32] D. Chicco and G. Jurman, "The advantages of the matthews correlation coefficient (MCC) over f1 score and accuracy in binary classification evaluation," *BMC Genomics*, vol. 21, no. 1, jan 2020.
- [33] A. Tharwat, "Classification assessment methods," *Applied Computing and Informatics*, vol. 17, no. 1, pp. 168–192, jul 2020.
- [34] B. Meyer, "What do chatgpt and ai-based automatic program generation mean for the future of software." [Online]. Available: <https://bit.ly/3LyAJLj>
- [35] F. J. García-Peñalvo, F. Llorens-Largo, and J. Vidal, "La nueva realidad de la educación ante los avances de la inteligencia artificial generativa," *RIED-Revista Iberoamericana de Educación a Distancia*, vol. 27, no. 1, pp. 9–39, 2024.
- [36] F. J. García-Peñalvo, "La integración de la inteligencia artificial generativa en la práctica docente," Apr. 2023. [Online]. Available: <https://doi.org/10.5281/zenodo.7853091>
- [37] A. Bozkurt, J. Xiao, S. Lambert, A. Pazurek, H. Crompton, S. Koseoglu, R. Farrow, M. Bond, C. Nerrantzi, S. Honeychurch, M. Bali, J. Dron, K. Mir, B. Stewart, E. Costello, J. Mason, C. M. Stracke, E. Romero-Hall, A. Koutropoulos, C. M. Toquero, L. Singh, A. Tlili, K. Lee, M. Nichols, E. Ossianilsson, M. Brown, V. Irvine, J. E. Raffaghelli, G. Santos-Hermosa, O. Farrell, T. Adam, Y. L. Thong, S. Sani-Bozkurt, R. C. Sharma, S. Hrastinski, and P. Jandrić, "Speculative futures on chatgpt and generative artificial intelligence (ai): A collective reflection from the educational landscape," 2023.

