Universidade Estadual de Feira de Santana

Programa de Pós-Graduação em Computação Aplicada

Uma Abordagem de Ensino-Aprendizagem de Programação na Educação Superior

Bianca Leite Santana

Feira de Santana

2018

Universidade Estadual de Feira de Santana

Programa de Pós-Graduação em Computação Aplicada

Bianca Leite Santana

Uma Abordagem de Ensino-Aprendizagem de Programação na Educação Superior

Dissertação apresentada à Universidade Estadual de Feira de Santana como parte dos requisitos para a obtenção do título de Mestre em Computação Aplicada.

Orientador: Roberto Almeida Bittencourt

Feira de Santana

2018

Ficha Catalográfica - Biblioteca Central Julieta Carteado - UEFS

S223 Santana, Bianca Leite
Uma abordagem de ensino-aprendizagem de programação na educação superior / Bianca Leite Santana. – 2018.
163 f.: il.

Orientador: Roberto Almeida Bittencourt.
Dissertação (Mestrado) – Universidade Estadual de Feira de Santana, Programa de Pós-Graduação em Computação Aplicada, 2018.

1. Programação – ensino-aprendizagem. 2. Educação superior. 3. Modelo ARCS. 4. *Non-majors*. 5. Motivação. I. Bittencourt, Roberto Almeida, orient. II. Universidade Estadual de Feira de Santana. III. Título.

CDU: 004.42:37

Elaboração: Luis Ricardo Andrade da Silva - Bibliotecário - CRB-5/1790

Bianca Leite Santana

Uma Abordagem de Ensino-Aprendizagem de Programação na Educação Superior

Dissertação apresentada à Universidade Estadual de Feira de Santana como parte dos requisitos para a obtenção do título de Mestre em Computação Aplicada.

Feira de Santana, 06 de abril de 2018

BANCA EXAMINADORA

Dr. Roberto Almeida Bittencourt (Orientador)
Universidade Estadual de Feira de Santana

Dr. Dalton Dario Serey Guerrero
Universidade Federal de Campina Grande

Dr. José Amâncio Macedo Santos
Universidade Estadual de Feira de Santana

Abstract

CS non-major students usually lack interest and have greater difficulties in learning programming when comparaed to CS major students. This work describes the design, application and evaluation of a teaching and learning approach for non-major students, whose purpose is to increase their motivation and decrease their difficulties in learning programming. The proposed approach combines the use of the Scratch environment in a context of game creation, the Python programming language associated with the turtle graphics library, and image manipulation with the Jython Environment for Students (JES). We conducted two exploratory case studies with Civil Engineering students attending a CS1 course at our institution to analyze the impact of this approach on student motivation and learning. Our results describe the motivation present during the course in terms of the Attention, Relevance, Confidence and Satisfaction (ARCS) model, and identify the practical factors that may contribute to increase or decrease student motivation. We also present a framework that shows the positive and negative impacts of the elements of our approach on each of the categories of the ARCS model. Various such elements are common in several teaching-learning situations. From a learning point of view, our findings also show that contextualized and spiral learning has enhanced the learning of concepts such as loops and functions. Scratch facilitates the learning of programming logic, select and repeat structures. Python with Turtle enhances learning of these same concepts with the addition of functions. Finally, the media computation approach has shown potential for learning the concepts of functions and arrays. We believe that the practical factors presented in this work can support the design of CS1 courses for non-majors.

Keywords: Programming learning, Computational Thinking, CS Non-Majors, Motivation, ARCS Model.

Resumo

Estudantes que não são da área de TI, chamados *non-majors*, usualmente apresentam falta de interesse e maiores dificuldades na aprendizagem de programação em relação a estudantes de cursos como Ciência da Computação. Neste trabalho descrevemos a concepção, aplicação e avaliação de uma abordagem de ensino-aprendizagem de Programação, destinada a estudantes *non-majors*, cujo intuito é aumentar a sua motivação e amenizar as suas dificuldades em aprender programação. A abordagem proposta combina o uso do ambiente lúdico *Scratch* em um contexto de criação de jogos, a linguagem de Programação Python associada à biblioteca *Turtle Graphics* e à manipulação de imagens por meio do ambiente de desenvolvimento *Jython Environment for Students (JES)*. Realizamos dois estudos de caso exploratórios com estudantes de Engenharia Civil cursando uma disciplina introdutória de programação em nossa instituição para analisar o impacto desta abordagem sobre a motivação e aprendizagem dos estudantes. Nossos resultados descrevem a motivação presente durante o curso em termos do modelo Atenção, Relevância, Confiança e Satisfação (ARCS), e identificam os fatores práticos que podem contribuir para aumentar ou diminuir a motivação dos estudantes. Geramos um quadro que evidencia os impactos positivos e negativos dos elementos de nossa abordagem sobre cada uma das categorias do modelo ARCS, sendo que muitos destes elementos são comuns em diversas situações de ensino-aprendizagem. Do ponto de vista da aprendizagem, nossos achados demonstram que o ensino contextualizado e em espiral potencializou a aprendizagem de conceitos como *loops* e funções. *Scratch* potencializa a aprendizagem de lógica de programação, *loops* e estruturas de seleção. Já Python com *Turtle* potencializa a aprendizagem destes mesmos conceitos com a adição de funções. Finalmente, a abordagem com mídias demonstrou potencial para a aprendizagem dos conceitos de funções, vetores e matrizes. Acreditamos que os fatores práticos apresentados nesse trabalho podem apoiar o design de disciplinas introdutórias de programação para *non-majors*.

Palavras-chave: Ensino-aprendizagem de programação, pensamento computacional, non-majors, motivação, modelo ARCS.

Prefácio

Esta dissertação de mestrado foi submetida à Universidade Estadual de Feira de Santana (UEFS) como requisito parcial para obtenção do grau de Mestre em Computação Aplicada.

A dissertação foi desenvolvida dentro do Programa de Pós-Graduação em Computação Aplicada (PGCA), tendo como orientador o Dr. **Roberto Almeida Bittencourt**.

Esta pesquisa foi financiada pela Fundação de Amparo à Pesquisa do Estado da Bahia – FAPESB.

Agradecimentos

Em primeiro lugar, agradeço a Deus pois ele tem me abençoado continuamente e me ensinado que é preciso pôr o amor incondicional a Ele e a nossos irmãos como medida para nossas ações.

Agradeço a minha mãe/pai Sônia Maria, por todo o esforço que tem feito por mim e por sempre ter me mostrado o poder transformador da Educação. Aos meus avós Darcy e Benjamim, este último *in memorian*, por serem as pessoas mais incríveis deste mundo e me ensinarem todos os dias o que é o amor. Agradeço a Felipe por todo seu amor, companheirismo, e por sempre estar ao meu lado.

Agradeço ao Professor Roberto Bittencourt por ter feito um excelente trabalho como orientador ao longo desses dois anos de pesquisa.

Agradeço também aos colegas do programa PGCA e do laboratório LESS, por todo apoio e troca de experiências.

Sumário

Lista de Publicações

Santana, B. L., Figuerêdo, J. S. L., and Bittencourt, R. A. "Motivação de Estudantes Non-Majors em uma Disciplina de Programação." In *WEI 2017 — XXV Workshop sobre Educação em Computação*. 2017.

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

Em 1961 Alan Perlis, o primeiro Cientista da Computação a receber o prêmio Alan Turing, ao palestrar no Massachusetts Institute of Technology (MIT), defendeu que a Ciência da Computação deveria fazer parte de uma educação liberal geral. Perlis justifica seu ponto de vista fazendo uma analogia com o ensino de cálculo, pois apesar de nem todos utilizarem cálculo na vida prática, aprender cálculo faz parte do que se considera boa educação (Guzdial and Ericson, 2009). Ao contrário desta época, atualmente a computação influencia a sociedade em muitos aspectos, uma vez que os computadores estão cada vez mais presentes na vida das pessoas.

Há um interesse crescente em ofertar educação em computação para todos. Nesse sentido, Wing (2006) argumenta que pensar como um cientista da Computação é mais do que saber como implementar programas, pois requer um pensamento em múltiplos níveis de abstração. O pensamento computacional envolve, entre outras coisas, a resolução de problemas, *design* e compreensão do comportamento humano, que são habilidades transferíveis para contextos fora da programação (Resnick et al., 2009). Outro argumento sobre a importância do ensino de computação, nesse caso essencialmente programação, dado por Resnick et al. (2009), é que fluência digital requer não só a habilidade de usar o computador para conversar e navegar na internet, mas a habilidade de projetar, criar e inventar novas mídias. E isso só é possível de se fazer aprendendo a programar.

Cursos introdutórios de programação, principalmente aqueles voltados para o nível superior, são muitas vezes conhecidos como cursos introdutórios de Ciência da Computação (Computer Science 1 - CS1). A Sociedade de Computação do Instituto de Engenheiros Eletricistas e Eletrônicos (IEEE-CS) e a Association for Computing Machinery (ACM) estabelecem diretrizes curriculares, constantemente revisadas, para cursos de graduação em computação incluindo diretrizes para cursos introdutórios (Curricula, 2001; Shackelford et al., 2006; Joint Task Force on Computing Curricula and Society, 2013). Em linhas gerais, essas diretrizes descrevem as categorias de abordagens mais comuns nos cursos do tipo CS1, e estabelecem as características principais que devem ser atendidas ao formatar estes cursos. De

maneira geral, os cursos de CS1 são caracterizados pelo uso do paradigma imperativo (*Imperative-first*), pelo uso do paradigma Orientado a Objetos (*Objects-first*), pelo uso do paradigma funcional (*Functional-first*), por introduzir os conceitos da computação sem foco na programação (*Breadth-first*), por utilizar pseudo-código (*Algorithms-first*) ou por introduzir os conceitos da programação a partir do nível da máquina (*Hardware-first*) (Joint Task Force on Computing Curricula and Society, 2013). A grande maioria dos cursos CS1 é focada na programação, na qual os alunos aprendem conceitos da ciência da computação através das tarefas explícitas de aprender uma determinada linguagem de programação e construir artefatos de *software*.

Dada a importância da computação na vida moderna e a relevância que há em saber programar, disciplinas do tipo CS1 têm sido adicionadas às matrizes curriculares de vários cursos de graduação fora da área de TI, principalmente nas áreas de ciências exatas e engenharias, que demandam força de trabalho conhecedora de tecnologia, a exemplo do Georgia Institute of Technology (Georgia Tech), onde CS1 é uma exigência para todos os estudantes (Forte and Guzdial, 2005). Na Universidade Estadual de Feira de Santana (UEFS), quatro componentes curriculares são ofertados nos moldes de CS1:

- EXA 801 – Algoritmos e Proramação I: ministrada para o curso de Engenharia de Computação;
- EXA 630 – Informática Aplicada: ministradas para os cursos de Agronomia e Química;
- EXA 170 – Introdução à Ciência da Computação (ICC): ministrada para os cursos de Engenharia de Alimentos e Engenharia Civil;
- EXA 196 – Introdução à Informática F: ministrada para os cursos de Física e Matemática.

Neste trabalho, vamos chamar de ICC as variantes EXA 630, EXA170 e EXA 196, oferecidas na UEFS. E vamos denominar de *non-majors* os estudantes que são de cursos que não são da área de Tecnologia da Informação (TI).

Existe todo um corpo do conhecimento desenvolvido ao longo de 30 a 40 anos sobre como ensinar programação efetivamente no nível superior (Dijkstra et al., 1989; Jenkins, 2002; Robins et al., 2003). Várias abordagens são propostas, mas apesar de todos os esforços, cursos introdutórios de programação muitas vezes têm altas taxas de evasão (Bennedsen and Caspersen, 2007; Watson and Li, 2014). Embora programar seja considerada uma habilidade importante, é usualmente aceito que aprender a programar é difícil, pois exige a aquisição de habilidades múltiplas, como a capacidade para resolução de problemas, lógica matemática, entender a sintaxe de uma nova linguagem, dentre outros aspectos (Jenkins, 2002; Azzam and Career, 2006).

Devido a essas dificuldades, ao longo dos anos, inúmeras iniciativas foram propostas na tentativa de melhorar as taxas de aprovação. As soluções propostas, normal-

mente, alteram características do cursos de CS1 como currículo, pedagogia, escolha da linguagem ou ferramentas para apoiar a prática e o aprendizado (Pears et al., 2007). Assim, as abordagens propostas podem: reformular o currículo, identificando o que deve ser ensinado nos cursos de CS1 através das tendências nacionais e internacionais nos currículos das universidades; explorar a pedagogia do ensino, ou seja, a maneira com que o ensino e a aprendizagem são geridos; e redefinir a linguagem de programação a ser ensinada, ou promover a adoção de ferramentas para apoiar a aprendizagem. As ferramentas utilizadas para apoiar a aprendizagem de programação pode ser: ferramentas de visualização, de avaliação automatizada; ambientes de desenvolvimento, e ferramentas de ajuda ao programador de maneira geral (Pears et al., 2007). As abordagens propostas costumam interferir em um ou mais desses fatores, promovendo a colaboração e apoio entre pares, melhoria e contextualização do conteúdo, reconfiguração de recursos e da avaliação do curso, dentre outras (Vihavainen et al., 2014). Por outro lado, Watson and Li (2014) demonstram que as taxas de reprovação em cursos de CS1 não têm melhorado significativamente ao longo do tempo, apesar dos avanços na pedagogia, e que estas taxas de insucesso são influenciadas por diferentes aspectos do contexto educacional, como linguagem utilizada, país, nível escolar e tamanho das turmas.

Um fator apontado como crucial neste contexto é a motivação. Jenkins (2001a) sugere que entender a motivação pré-existente que leva um estudante a entrar em um curso de programação e a permanecer nele é um passo vital para promover uma aprendizagem melhor e mais eficiente. Estudantes de cursos da área de TI têm uma motivação implícita, uma vez que é objetivo da graduação saber programar, enquanto que estudantes de outros cursos normalmente não têm essa motivação e costumam se deparar com uma disciplina de CS1 fora de contexto de realidade, pois usualmente os departamentos aproveitam as disciplinas formatadas para uma estrutura curricular de cursos da área de TI (Forte and Guzdial, 2005). Cursos introdutórios tradicionais não conseguem motivar muitos estudantes e podem até mesmo desencorajá-los de prosseguir a aprendizagem futura em computação. Em contrapartida, cursos alternativos, projetados para acomodar uma variedade de interesses e *background*, oferecem um contexto mais motivador e atraente para a aprendizagem de conceitos de programação e computação (Forte and Guzdial, 2005).

Diversas abordagens já foram propostas para melhorar a motivação e o engajamento de *non-majors* nos cursos de CS1. Muitos pesquisadores defendem que os cursos de CS1 devem ser reformulados para atender as necessidades dos *non-majors* e oferecer um contexto motivador. Joyce (1998) apresenta um curso para *non-majors*, focado em desenvolver a habilidade de resolução de problemas. Comer and Roggio (2002) discutem alguns dos problemas associados ao ensinar um curso de CS1 baseado em Java, para um grupo cuja maioria são *non-majors*. Kaplan (2004) propõe um curso de CS1 para cientistas naturais, físicos e cientistas sociais. Forte and Guzdial (2005) propõem a contextualização dos cursos de CS1, de acordo com a audiência. Chilana et al. (2015) apresentam os resultados de um estudos de caso com estudantes calouros de engenharia de gestão matriculados em um curso de CS1.

Hromkovič et al. (2016) descrevem uma abordagem de ensino introdutório de programação utilizando Logo e Python. Dziallas et al. (2017) avaliam o programa *Year in Computing*, oferecido na Escola de Computação da Universidade de Kent, no Reino Unido, onde diversas disciplinas são ofertadas para os estudantes *non-majors*, como a compreensão de sistemas operacionais e redes de computadores, aprendizagem de programação, produção de páginas web usando HTML, CSS e JavaScript. Estas e outras inúmeras iniciativas vem sendo propostas para melhorar o engajamento dos *non-majors*, porém, a maioria dos trabalhos encontrados, principalmente os que propõem abordagens hibridas, utilizando diversas linguagens e ferramentas, não apresentam investigações bem formuladas sobre a efetividade do ensino dessas abordagens ou as melhorias na motivação e em demais fatores. Na maioria dos casos as abordagens são simplesmente apresentadas, sem um avaliação detalhada de sua eficácia. Quando as avaliações são propostas, muitas vezes se limitam a análises quantitativas, não abordando o problema de maneira aprofundada. Aspectos mais subjetivos, como a motivação, dificilmente são avaliados. Além disso, este é um problema que envolve inúmeros fatores, incluindo o contexto geográfico e espacial. Quase não existem trabalhos propondo avaliações com foco puramente nos estudantes *non-majors* e em sua motivação dentro do contexto brasileiro.

Este trabalho supõe que a solução para um problema com múltiplos fatores, envolve uma solução mista, com aprendizagem gradual e espiral de fundamentos básicos da programação, e que merece uma avaliação aprofundada, numa perspectiva de métodos quali-quantitativos e dentro do contexto da universidade brasileira.

Dada a relevância do ensino de programação para o público em geral, a crescente inclusão de disciplinas de CS1 em cursos de graduação fora da área de TI e os problemas relacionados ao ensino da programação brevemente listados, o principal objetivo deste estudo foi *conceber, aplicar e avaliar uma abordagem de ensino-aprendizagem de Programação, destinada a estudantes que não são de cursos de TI*. A abordagem proposta fez uso do ambiente lúdico Scratch, da linguagem de programação Python, e do ambiente de desenvolvimento Jython Environment for Students (JES). Aplicamos esta abordagem em turmas de ICC do curso de Engenharia Civil na Universidade Estadual de Feira de Santana ao longo de dois semestres consecutivos, seguindo uma metodologia pré-estabelecida (ver Capítulo 3).

São questões de pesquisa que nortearam este estudo:

(I) **Como a abordagem proposta influencia na motivação dos estudantes?** Tendo em vista que a motivação é um fator importante no processo de aprendizagem, que pode influenciar fortemente as taxas de reprovação e evasão. Pretendeu-se avaliar se a abordagem proposta influencia positivamente na motivação dos estudantes. Através da triangulação dos dados quantitativos e qualitativos foram contrastados os níveis de motivação apresentados pelos estudantes, nos diversos instrumentos de medição utilizados, como observações, entrevistas e questionários.

(II) **Quais competências tem o aprendizado melhor potencializado pela**

abordagem? Pretendeu-se identificar quais conceitos e habilidades de programação são melhor potencializados pela abordagem proposta e quais conceitos e habilidades têm sua aquisição prejudicada devido à abordagem.

Este trabalho justifica-se pela carência de análises mais aprofundadas e metodologicamente consistentes que observem as nuances do problema da motivação e da aprendizagem, ponderando elementos locais, como o perfil do estudante brasileiro. A relevância social reside na disponibilização gratuita à comunidade acadêmica, de uma abordagem de ensino-aprendizagem de programação projetada para motivar os estudantes e para aumentar os índices de retenção e aprovação. Vale ressaltar que a palavra retenção pode significar algo negativo, como retenção de alunos em uma disciplina, mas também pode ser positiva, como retenção de alunos que não abandonam um curso. Nesse estudo, esta palavra será empregada em seu sentido positivo.

O restante deste documento está dividido do seguinte modo: o Capítulo 2 traz a revisão bibliográfica pertinente ao tema; o Capítulo 3 descreve a metodologia aplicada; o Capítulo 4 apresenta a disciplina e os materiais e ferramentas utilizados; o Capítulo 5 apresenta um estudo sobre a aprendizagem dos estudantes; o Capítulo 6 trata da motivação dos estudantes ao utilizarem a abordagem proposta; no Capítulo 7, é feita uma discussão sobre as mudanças trazidas pela abordagem e descrevemos o cenário tradicional de ICC em nossa instituição; por fim, o Capítulo 8 traz as considerações finais para este trabalho, evidenciando os resultados alcançados, tratando da validade e confiabilidade da pesquisa, além de trabalhos futuros.

Capítulo 2

Revisão Bibliográfica

Neste capítulo, serão apresentados conceitos básicos necessários à fundamentação teórica deste trabalho. Primeiramente, é feita uma discussão sobre pensamento computacional e sua importância para embasar discussões no campo da educação em computação (Seção 2.1). Em seguida, apresentamos as taxas de aprovação, reprovação e evasão em cursos de programação a fim de entender as possíveis consequências que as dificuldades enfrentadas por novatos podem ocasionar (Seção 2.2). Busca-se entender quais as dificuldades encontradas pelos novatos em cursos de programação (Seção 2.3). São identificadas as soluções comumente exploradas para amenizar as dificuldades dos novatos (Seção 2.4). Em seguida, a partir da identificação das principais dificuldades enfrentadas pelos novatos procura-se entender a importância da motivação no papel da aprendizagem de programação e como o professor pode fomentar a motivação em seus alunos (Seção 2.5). A questão dos *non-majors* também é abordada de maneira específica, através de um levantamento de trabalhos relacionados (Seção 2.6). Por fim, cada uma das ferramentas, linguagens e materiais escolhidos para compor a abordagem propostas são descritas (Seção 2.7). Cada um desses temas são tratados nas seções a seguir:

2.1 Pensamento Computacional

Pensamento computacional é o termo popularizado por Jeannette Wing através de seu artigo "Computational Thinking" (Wing, 2006). Segundo a autora, o pensamento computacional envolve a resolução de problemas, a concepção de sistemas e a compreensão do comportamento humano, baseando-se nos conceitos fundamentais da computação. Apesar de fomentar uma ampla discussão no campo da Educação em Computação, Wing não define exatamente o que o termo Pensamento Computacional significa. Selby and Woollard (2013) trazem uma definição mais precisa através de uma revisão bibliográfica que evidencia os termos que aparecem consistentemente ao longo da literatura de educação em computação entre 2006 e 2013.

Na definição proposta, o pensamento computacional é um processo cognitivo ou de pensamento focado na resolução de problemas e que reflete a capacidade de pensar: em abstrações, em termos de decomposição, algoritmicamente, em termos de avaliações e em generalizações.

O pensamento computacional é tido como uma habilidade fundamental para todos em um mundo onde há forte presença das soluções proporcionadas pela Computação em outros campos do conhecimento, tais como estatística, biologia, economia, ciências e engenharias de maneira geral (Wing, 2008). Além disso, associando o ato de "pensar como um cientista da computação" à capacidade de computar, alguns estudos sugerem a existência de uma correlação entre capacidade mental de computar e desempenho do estudante em cursos universitários diferentes, como ciência da computação, direito, farmácia ou educação (Oliveira, 2012). A importância atribuída ao pensamento computacional contribui para fomentar uma discussão sobre a popularização do ensino da computação, não só no nível superior, mas também para estudantes de ensino médio e fundamental. Para Wing (2008), deve-se assegurar a todos uma base comum e sólida de compreensão e aplicação de pensamento computacional.

Admitindo a necessidade do ensino do pensamento computacional a todos, o foco das principais discussões é sobre como o pensamento computacional deve ser ensinado e como sua aquisição pode ser observada. Especificamente, para o caso dos *non-majors* em CS, o modo como as disciplinas introdutórias de Ciência da Computação são formatadas pode contribuir para potencializar o pensamento computacional. Um ponto pertinente de discussão é o papel da programação nestas disciplinas (mais adiante retomaremos esta discussão no tópico específico para *non-majors*). Uma das edições da revista ACM Inroads, em março de 2015, traz uma seção sobre *"O papel da programação em cursos de Ciência da Computação para non-majors"*, e as opiniões são divergentes (Imp, 2015). Alguns artigos são enfáticos em afirmar que é possível construir uma introdução rigorosa e significativa à Ciência da Computação sem programação. Um dos principais argumentos é que embora saber programar seja uma habilidade interessante no campo da Ciência da Computação, aprender a programar demanda tempo e existem outros tópicos da Ciência da Computação mais importantes para *non-majors* que podem ser abordados sem uso da programação e que são numerosos o suficiente para ocupar uma disciplina inteira (Goldweber, 2015; Walker, 2015a). Em contrapartida, argumentos favoráveis à utilização tem como base a aquisição do pensamento computacional em um nível adequado para estudantes de graduação. Para Cooper and Dann (2015), o pensamento computacional é uma habilidade que envolve necessariamente resolução de problemas, e a programação é um componente fundamental para potencializar isto. Mesmo um dos que argumentam em contrário, (Walker, 2015b) admite que a aquisição da habilidade de resolução de problemas em termos de pensamento computacional requer precisão de pensamento, exposição e consideração de detalhes, e só as linguagens de programação incentivam o nível de precisão necessário para desenvolver soluções propostas, executá-las, analisá-las e compará-las com outras soluções.

Para ensinar o pensamento computacional para todos, são necessárias abordagens diferentes das que são utilizadas para estudantes que querem se tornar profissionais de computação, pois estudantes novatos, principalmente *non-majors*, apresentam dificuldades em entender programação e não compreendem de modo natural as soluções que os cientistas de computação adotam. Pesquisas sugerem, por exemplo, que o pensamento "orientado a objetos" não é "natural", no sentido de ser característico das descrições de tarefas dos novatos, e isso encoraja a utilização de linguagens que se baseiam nas formas naturais de pensar sobre a computação, como o uso de programação baseada em eventos no Scratch (Guzdial, 2008). As linguagens de programação visuais geralmente são menos poderosas do que as linguagens tradicionais, mas usam representação mais próxima da linguagem humana, tornam as práticas computacionais, como testes e depuração cognitivamente menos exigentes e proveem visualização rápida dos resultados. Isto permite que os alunos adquiram práticas computacionais de resolução de problemas mais facilmente (Lye and Koh, 2014).

Ensinar programação por si só em uma disciplina não implica necessariamente que o pensamento computacional seja potencializado. Selby (2015) utiliza uma abordagem de teoria fundamentada em dados para desenvolver um modelo que representa o relacionamento entre o domínio cognitivo da taxonomia de Bloom, habilidades de pensamento computacional e o ensino de programação. O modelo criado sugere uma sequência comum ao ensino de programação, que independe de contexto, como linguagem de programação ou ambiente. Esta sequência é composta por: 1) construções, fatos, tipos; 2) como funcionam os construtos individuais; 3) usar construções de programação em contextos artificiais; 4) discriminar, decompor, abstrair; 5) criar programas, design de algoritmos; 6) testar, avaliar. O modelo também sugere os níveis de ensino de programação de acordo com a Taxonomia de Bloom, onde avaliação equivale a testar e avaliar; síntese equivale a criar programas e design de algoritmos; análise e aplicação equivale a resumir, decompor e discriminar; compreensão e conhecimento equivale a abstrair, decompor e discriminar. O modelo também ordena a dificuldade de assimilar as habilidades de pensamento computacional do mais fácil para o mais difícil: 1) avaliação; 2) projeto de algoritmos; 3) generalização, abstração de funcionalidade, abstração de dados; 4) decomposição. Este modelo sugere que a ordem em que as habilidades de programação são ensinadas reflete diretamente a ordem dos níveis no domínio cognitivo da taxonomia de Bloom e isto é um resultado norteador para a formatação da ordem em que os conteúdos são abordados.

A partir do que foi exposto, percebe-se que o pensamento computacional é uma habilidade relativamente complexa. Considerando o forte caráter de resolução de problemas, muitas vezes espera-se que as competências adquiridas sejam empregadas em contextos fora da programação. Consequentemente, as maneiras de avaliar sua aquisição acabam não sendo triviais e dependem muito do modo como o pensamento computacional foi mapeado para a abordagem. Basawapatna et al. (2011) investigam se estudantes são capazes de reconhecer padrões de pensamento computacional (CTP, da sigla em inglês Computational Thinking Partterns) a partir

de sua experiência de programação de jogos. Os autores definem padrões de pensamento computacional como as unidades específicas de transferência entre jogos e simulações científicas. Para avaliar estes modelos, os autores desenvolveram um questionário de Teste de Pensamento Computacional, com questões que incluem vídeos de fenômenos da vida real e pediram aos estudantes que relacionassem estes fenômenos com os padrões dos jogos programados anteriormente, além de questões para listar e descrever todos os padrões de pensamento computacional necessários para a implementação de determinado modelo. Os resultados apresentados pelos autores mostram que os participantes foram capazes de compreender e reconhecer padrões de pensamento computacional em um contexto diferente.

As linguagens visuais como o Scratch, que proveem uma comunidade onde usuários podem disponibilizar seus projetos online, podem ter a análise da aquisição do pensamento computacional automatizada. Um exemplo é a ferramenta Hairball que faz análise automática de projetos no Scratch. Esta ferramenta é disponibilizada para a comunidade como um conjunto de plugins para a avaliação de projetos em quatro dimensões: inicialização de estado, sincronização de blocos de fala e arquivos de som, transmissão e recebimento de mensagens para desencadear execução de blocos, e animação (Boe et al., 2013). Dr Scratch é uma aplicação web gratuita e de código aberto, baseada nos plug-ins Hairball, e que permite analisar projetos Scratch e obter feedback que pode ser usado para melhorar habilidades de programação e de pensamento computacional [1]. Dr. Scratch detecta certos maus hábitos de programação ou erros potenciais, como nomes de sprites não significativos, repetição de código, código que nunca é executado e inicialização incorreta de atributos de objeto. Nos aspectos em que há margem para melhorias, a ferramenta ainda fornece ao usuário links para informações que podem ser usadas para melhorar o código. Posteriormente os autores comparam as avaliações fornecidas pelo Dr. Scratch com avaliações fornecidas por especialistas em educação em computação. Os resultados mostram fortes correlações entre avaliações automáticas e avaliações manuais. Nos casos que apresentaram diferença notória entre as avaliações, o principal motivo para a discrepância foi devido à funcionalidade dos projetos. Enquanto os especialistas levam em consideração se os projetos atingiram seus objetivos, conforme indicado nas instruções ou a usabilidade do projeto, Dr. Scratch é incapaz de avaliar tais problemas (Moreno-León et al., 2017).

Resultados de estudos realizados com ferramentas de avaliação automatizada levam a crer que a análise automática pura não é suficiente, embora seja bastante eficaz para identificar a existência de comandos e padrões de construção de código. A avaliação da aquisição do pensamento computacional pode ser mais eficaz quando combinada diferentes instrumentos de avaliação. Brennan and Resnick (2012) descrevem um framework para avaliação do pensamento computacional a partir do Scratch que se baseia em conceitos, práticas e perspectivas computacionais observadas pelos autores em suas experiências com estudantes. De acordo com os autores, os conceitos de pensamento computacional mais úteis são: sequências, loops, paralelismos, eventos,

[1] http://www.drscratch.org

condicionais, operadores e dados. As principais práticas computacionais são: ser incremental e iterativo, testar e depurar, reutilizar e remixar, e abstrair e modularizar. E as mudanças de perspectivas observadas em jovens trabalhando com Scratch são: expressão (autoexpressão), conexão (interação com outras pessoas) e questionamento. Para avaliar o pensamento computacional, o framework provê a análise de portfólio de projetos, entrevistas baseadas em artefatos e o desenvolvimento de cenários de design. A análise de portfólio se dá através da avaliação automática, utilizando a ferramenta Scrape, para analisar os projetos de um membro da comunidade e gerar uma representação visual dos blocos utilizados em cada projeto. As entrevistas devem ser realizadas a partir de um guia, que investiga o background do estudante em relação ao Scratch, busca identificar conceitos e práticas de pensamento computacional a partir de dois projetos implementados pelo usuário e as impressões gerais do estudante em relação ao Scratch. A terceira abordagem de avaliação é o desenvolvimento de cenários de design, onde os autores desenvolveram três conjuntos de projetos Scratch com complexidade crescente. Em cada conjunto, há dois projetos que envolvem os mesmos conceitos e práticas, mas possuem uma estética diferente. Em uma série de três entrevistas, os alunos são apresentados aos cenários de design, selecionam um dos projetos de cada conjunto e explicam o que o projeto selecionado faz, descrevem como ele poderia ser estendido, corrigem um erro e adicionam um recurso. Esta última abordagem oferece uma oportunidade para explorar sistematicamente diferentes formas de conhecimento e a fluência do estudante com diferentes conceitos e práticas de pensamento computacional.

Seiter and Foreman (2013) apresentam um framework denominado Progression of Early Computational Thinking (PECT), destinado à compreensão e avaliação do pensamento computacional nas séries iniciais do ensino fundamental por meio de programas escritos em Scratch. Para os autores, uma estrutura conceitual de avaliação precisa envolver um conjunto de categorias que integram os principais conceitos envolvidos com o pensamento computacional, um conjunto de padrões de design, que servem como modelos primários para completar tarefas ou objetivos de cada programa e um conjunto de construções de programação explícitas sobre as quais os padrões de design dependem. Seguindo esta ideia, o modelo PECT é composto por três componentes fundamentais da abstração decrescente: 1) conceitos de pensamento computacional, 2) variáveis de padrões de projeto e 3) variáveis de evidência. As variáveis de evidência são um conjunto de características concretas e classificadas do código escrito em Scratch: Looks, Som, Movimento, Variáveis, e demais categorias de comandos do Scratch. As variáveis de padrão de design são um conjunto de habilidades contextuais baseadas em padrões de codificação comuns em Scratch: animação de aparência, animação de movimento, conversação, colisão, score, interação com usuário. Os conceitos de pensamento computacional são divididos em: Procedimentos e Algoritmos, Decomposição de Problemas, Paralelização e Sincronização, Abstração, e Representação de Dados. Estas variáveis são mais qualitativas em estrutura e servem como um meio para entender o nível geral de pensamento computacional de um estudante. Os testes realizados com o modelo PECT demonstram

a sua eficácia na detecção das diferenças no pensamento computacional entre estudantes de várias idades, bem como quaisquer progressões gerais claras no aumento da sofisticação computacional. Os autores verificaram que, ao delinear o trabalho do aluno através do uso de variáveis de padrão de design, o Modelo PECT é potencialmente um modelo efetivo para usar no estudo e compreensão do desenvolvimento do pensamento computacional.

Os frameworks de avaliação compostos por mais de um instrumento de medição potencializam a criação e análise crítica de projetos, especialmente porque se baseiam em aspectos dos artefatos desenvolvidos pelos estudantes. Essas abordagens se esforçam para identificar a evolução dos estudantes ao longo do período de instrução.

2.2 Taxas de Aprovação e Reprovação

Programação usualmente é tida como algo difícil de se aprender. Esta é uma percepção que professores de programação apresentam de modo geral e uma das justificativas que sustentam esta ideia são os altos índices de desistência e reprovação registrados em cursos de CS1.

Bennedsen and Caspersen (2007) fizem um estudo sobre as taxas de reprovação em cursos introdutórios de programação a nível universitário com 63 instituições de diversos países e identificam que o nível médio de aprovação é de 67%. A princípio pode parecer que este valor não é alarmante, mas, ao traduzí-lo para valores numéricos, o estudo estima que cerca de 650.000 estudantes em média sejam reprovados em CS1. Neste mesmo trabalho, ao comparar os dados obtidos com os dados da UNESCO sobre estudantes matriculados em cursos de educação superior em computação no ano de 1999 e estudantes graduados em 2004, observa-se que apenas 26,8% dos estudantes se graduaram. Watson and Li (2014), baseando-se no trabalho anterior, fazem uma análise estatística sobre os dados de taxa de aprovação extraídos de 161 cursos de CS1 de 51 instituições em 15 países e encontraram uma taxa de aprovação média de 67,7%. Eles concluem que não houve uma melhora significativa na taxa de aprovação ao longo dos anos.

No Brasil, Bosse and Gerosa (2015) fizeram um levantamento das taxas de reprovação e trancamento nas disciplinas de CS1 da Universidade de São Paulo (USP) no período de 2010.1 a 2014.2. Os resultados mostram que a taxa de reprovação e trancamento média é de 30% e que a incidência da reprovação é maior para estudantes que não são de cursos da área de computação.

2.3 As Dificuldades em Aprender Programação

Quando se deseja ensinar programação efetivamente, é preciso entender o que torna a aprendizagem de programação difícil para muitos estudantes (Jenkins, 2002).

Dijkstra et al. (1989) afirma que programação é uma "novidade radical", por ser um assunto novo para muitos estudantes, e o sistema educacional ao qual estamos habituados não funciona mais para este tipo de novidade. Ele considera que o processo de aprendizagem é algo lento e gradual que visa transformar o estranho em familiar. Os alunos que iniciam um curso de programação no ensino superior vieram de um ambiente acadêmico familiar e desenvolveram um conjunto de habilidades de aprendizado que podem não ser muito eficazes com essa "novidade radical".

Jenkins (2002) assume que a programação é uma habilidade complicada de se dominar, e portanto, aprender a programar é, correspondentemente, complexo. Ele descarta a aptidão como um fator mensurável e afirma que o foco para entender a dificuldade de aprender a programar deve se voltar para uma visão mais cognitiva da processo de aprendizagem.

Jenkins (2002) faz uma descrição sobre vários fatores que podem ajudar a entender as dificuldades enfrentadas por estudantes no processo de aquisição dessa habilidade. São eles: estilo de aprendizagem, motivação, múltiplas habilidades, múltiplos processos, linguagem de programação, novidade educacional, interesse, reputação e imagem, e ritmo. O estudo faz uma análise sobre como cada estudante apresenta diferentes preferências pessoais e estilos de aprendizagem. Além disso, a motivação, seja ela intrínseca, extrínseca ou social, é um fator determinante para que um estudante se mantenha num curso de programação. Outra dificuldade apontada é o fato de que programadores experientes devem possuir múltiplas habilidades, como capacidade de resolução de problemas, facilidade com matemática e facilidade com manuseio do computador. O entendimento sobre os múltiplos processos envolvidos na criação de um programa são difíceis para um estudante novato. A linguagem de programação escolhida também é apontada como dificuldade pois, usualmente, é escolhida uma linguagem que está em alta na indústria, projetada para oferecer o máximo de possibilidades possíveis e não para ensinar. Outra dificuldade é a chamada "novidade educacional", um termo usado para descrever o cenário com o qual os estudantes que chegam a um curso de CS1 no ensino superior são confrontados, já que a programação é um tópico totalmente novo que não responde às suas abordagens de estudo habituais. O interesse dos estudantes também é visto como um problema, e as atividades passadas podem tornar a tarefa de aprender a programar maçante, minando o interesse dos estudantes. Além disso, cursos de programação têm imagem de que são difíceis e esta reputação é passada pelos estudantes mais velhos para os novatos, e isso se soma à imagem comumente tida de que programadores são nerds que passam seu tempo produzindo código, o que dificilmente é aspirado pela maioria das pessoas. Outro problema apontado é que o ritmo das aulas, que normalmente é ditado pelo tempo e necessidade de avaliação, não esteja sujeito ao ritmo de aprendizagem dos estudantes.

Robins et al. (2003) fazem uma revisão da literatura acadêmica sobre ensino e aprendizagem de programação para novatos e identificam uma série de tendências que podem ajudar a entender as dificuldades encontradas pelos novatos. Uma delas é a necessidade de distinção entre novatos e especialistas em programação. Especialis-

tas usualmente apresentam um nível de entendimento, capacidade de resolução de problemas e práticas de design que estudantes novatos não apresentam. Segundo a pesquisa, as potencias dificuldades enfrentadas pelos novatos são: a orientação geral, como o conceito de programas e suas potencialidades; noções sobre modelo de computador e sua relação com a execução de programas; a sintaxe e semântica de uma linguagem particular; e construção de habilidades de planejamento, desenvolvimento, testes e depuração. O artigo realça a importância de distinguir estudantes novatos de acordo com suas habilidades, níveis de motivação, personalidade, estilo cognitivo e de programação. Nesse contexto, o estudo identifica três tipos de comportamento característicos de novatos denominando-os *stoppers*, *movers* e *tinkerers*. Quando confrontados com um problema ou a falta de uma direção clara para prosseguir, *stoppers* simplesmente param. Já os *movers* são estudantes que continuam tentando experimentando, alterando seu código, e utilizando o *feedback* dos erros de maneira efetiva. Os *tinkerers* são aqueles estudantes capazes de rastrear seus programas, e podem até fazer alterações mais ou menos ao acaso, mas como *stoppers*, têm pouca chance efetiva de progredir.

Robins et al. (2003) também identificam características dos novatos que podem justificar insucesso no processo de aprendizagem, como a falta de modelos mentais detalhados, estratégias específicas para resolução de problemas, e compreensão sobre a sequência natural da execução de um programa. Além disso, novatos gastam muito pouco tempo planejando e testando o código e seus conhecimentos tendem a ser sobre contextos específicos ao invés de gerais.

Eckerdal et al. (2005), ao entrevistar estudantes *non-majors* que fizeram CS1, sobre as suas experiências de aprendizagem de programação e sobre suas percepções a respeito da programação identificaram o que pode ser chamado de "pensamento de programação". Muitos dos entrevistados expressaram que a programação inclui uma maneira diferente de pensar e que tiveram dificuldades para descrever e obter um controle sobre como aplicar esse tipo de pensamento. Poucos estudantes chegam ao final do curso com a compreensão de que programação tem a ver com a resolução de problemas e com uma forma sistemática de raciocínio. A maioria dos estudantes normalmente buscam soluções prontas ou roteiros a seguir, o que é usual em disciplinas como matemática, mas não são necessariamente quando se aprende programação.

Uma suposição alternativa é dada por Robins (2010), que analisa as causas das reprovações e desistências descritas na literatura acadêmica para CS1. Sua análise considera fatores de desenvolvimento cognitivo, estilo de aprendizagem, atitude, motivação, dentre outros. Ele afirma que há poucas evidências que fundamentem justificativas como "os alunos naturalmente se dividem em populações de programadores e não programadores" e propõe uma explicação chamada efeito Learning Edge Momentum (LEM). Esse efeito, fundamentado na literatura psicológica educacional, opera de tal forma que o sucesso na aquisição de um conceito facilita a aprendizagem de outros conceitos intimamente ligados. Em contrapartida, a aprendizagem malsucedida de um conceito torna um pouco mais difícil a aquisição de

conceitos adicionais relacionados. O efeito LEM varia em intensidade, dependendo das propriedades do domínio alvo, e, dependendo da força desse efeito, o sucesso ou o fracasso inicial em adquirir conceitos pode se tornar auto-reforçado, ocasionando resultados extremos. Robins argumenta que os conceitos envolvidos em uma linguagem de programação são inusitadamente bem integrados, resultando em um efeito LEM mais forte do que a média. A interação entre a forma como as pessoas aprendem e a natureza do material do assunto CS1 cria um viés estrutural que age para levar os estudantes a resultados extremos. Robins acredita que ao concentrar os recursos nas fases iniciais do CS1, identificando e explorando os mecanismos de aprendizagem bem-sucedida, ou adaptando a forma como o currículo é oferecido, pode ser possível aumentar significativamente a taxa de resultados de aprendizagem bem-sucedidos.

Muitas das investigações a cerca das dificuldades em aprender programação buscam identificar especificamente os conceitos que os estudantes apresentam maiores dificuldades. Diversos artigos evidenciam as dificuldades com os conceitos de *loops* e estruturas condicionais. Kaczmarczyk et al. (2010) investigam os principais equívocos conceituais dos estudantes de CS1 através de entrevistas formais com estudantes. Dentre os achados, os alunos não entendem o processo de operação dos *loops*, principalmente *while*, e entendem mal a relação entre os elementos da linguagem e o uso da memória subjacente. Cherenkova et al. (2014) investigam quais os conceitos desafiadores em CS1 através da análise de 266.852 projetos de código em linguagem *Python*, provenientes de duas turmas de CS1 ministradas em uma universidade note-americana ao longo de 2011. Os pesquisadores identificaram que uma parcela significativa dos estudantes apresentam dificuldades com uso de estruturas condicionais e *loops*. Embora existam evidências de melhoras com a prática, esses conceitos ainda são desafiadores para os estudantes, principalmente *loops*. Neste trabalho também são evidenciadas as dificuldades com problemas estruturais básicos de um programa, como sintaxe e recuo, mas esse tipo de dificuldade é amenizada com a prática.

2.4 Soluções Comumente Exploradas

Diversas soluções são propostas para ajudar a melhorar a eficiência do ensino da programação. Vihavainen et al. (2014) fornecem uma análise do efeito que várias intervenções podem ter sobre as taxas de aprovação de cursos de CS1 e comparam quantitativamente o impacto que as diferentes abordagens tiveram sobre as taxas de aprovação de cursos de programação. O estudo analisa abordagens que promovem a colaboração e apoio entre pares, melhoria e contextualização do conteúdo, reconfiguração de recursos e metodologia de avaliação, e abordagens híbridas, dentre outras. As abordagens encontradas melhoram, em média, a taxa de aprovação de CS1 em um terço. Uma conclusão importante desde estudo é que mesmo as abordagens menos bem sucedidas ainda melhoram os índices de aprovação quando

comparados com a abordagem tradicional que substituíram. Ou seja, os educadores e pesquisadores estão fazendo a diferença quando experimentam novas intervenções de ensino e abordagens pedagógicas.

As soluções propostas também podem ser separadas em quatro categorias distintas: Currículo, Pedagogia, Escolha da Linguagem e Ferramentas para ensinar (Pears et al., 2007). As subseções a seguir utilizam esta divisão para facilitar a exposição das tendências que vêm sendo abordadas, mas deve-se admitir que uma dada abordagem deve olhar para todos estes fatores ao mesmo tempo.

2.4.1 Currículo

As abordagens que focam no currículo a ser adotado normalmente tentam entender como CS1 se encaixa em uma variedade tão grande de currículos, como fazer a distinção entre programação e codificação, e quais as tendências nacionais e internacionais nos currículos das universidades (Pears et al., 2007). As principais sociedades de computação, ACM e IEEE, têm trabalhado conjuntamente por muitos anos em recomendações para os currículos de computação, que incluem disciplinas introdutórias de programação.

Apesar de existirem diretrizes para um currículo de CS1, é possível fazer uma escolha explícita para enfatizar um tema particular que será recorrente ao longo do curso (Pears et al., 2007). Para Brilliant and Wiseman (1996) alterar um paradigma ou linguagem requer reestruturação de todo o currículo para sustentar esta mudança. Long et al. (1998) acreditam que as disciplinas de CS1 não devem abordar apenas tópicos de programação e define objetivos para CS1 a fim de que haja um tema intelectual fundamental que seja claramente visível para os estudantes e que transcenda as tendências de ensino, como paradigmas e linguagens.

O currículo de um curso de CS1 também deve levar em consideração o público-alvo para o qual será ministrado. Forte and Guzdial (2005) discorre sobre a necessidade de atender as diferentes audiências e apresenta a experiência, vivenciada no Instituto de Tecnologia da Geórgia (Georgia Tech), onde foram ofertados três cursos, com currículos e ferramentas diferentes para estudantes dos cursos de TI, estudantes de engenharia e estudantes de outras áreas.

2.4.2 Pedagogia

Enquanto o currículo define o que deve ser ensinado, as soluções que focam na pedagogia lidam com a maneira com que o ensino e a aprendizagem são geridos a fim de facilitar os resultados desejados para a aprendizagem (Pears et al., 2007).

Abordagens pedagógicas podem propor novas maneiras de ensinar programação. Palumbo (1990) revisa e analisa as evidências e modelos associados ao uso de linguagens de programação como um veículo para a aprendizagem de habilidades de

resolução de problemas, incluindo questões relacionadas ao modo como isso é ensinado. Linn and Clancy (1992) defendem o uso de estudos de caso para o ensino de programação em uma abordagem que, para cada atividade, inclui a descrição do problema, a descrição do processo utilizado por um especialista para resolver o problema, o código do especialista, questões de estudo e perguntas de teste. Kuittinen and Sajaniemi (2004) apresentam os "papéis de variáveis" como um conceito que pode ser usado para auxiliar os novatos no processo de escrita de código. Eles descobriram que 10 papéis podem descrever os usos de 99% de todas as variáveis nos programas introdutórios dos alunos e fornecem instruções detalhadas sobre técnicas para apresentar estes papéis aos novatos e como utilizar esse conhecimento na concepção do programa.

Em termos pedagógicos, as melhorias no ensino da programação podem fazer uso de diferentes teorias educacionais. A exemplo de Ben-Ari (2001), que discute as diferentes dimensões do construtivismo e sua utilização na educação científica e nos contextos de educação em ciência da computação. Para ele, construtivismo pode ajudar a sanar certas dificuldades que os estudantes têm, especialmente nos níveis mais básicos do ensino da computação, se sua aplicação for conciliada com a natureza dos tópicos abordados em ciência da computação.

Abordagens pedagógicas podem defender que atividades de aprendizagem interativa são mais poderosas, a exemplo de Zingaro (2014), que defende o uso da instrução entre pares como uma pedagogia interativa valiosa para o ensino de programação. Instrução entre Pares é uma abordagem que aplica testes de conceitos que são respondidos individualmente, e em seguida, discutido em pares e em grupo. Para ele, esta abordagem reduz taxas de falhas, aumenta a retenção e é desfrutada tanto pelos alunos como pelos instrutores.

Abordagens pedagógicas também podem considerar o ponto de vista do estudante para entender e aprimorar o processo de ensino. Eckerdal et al. (2005) tratam o que é preciso para aprender a programar. Kolikant (2005) tratam as correções dos programas através da visão pragmática identificada através da opinião dos estudantes.

2.4.3 Linguagens

A primeira linguagem de programação serve como uma referência para o aprendizado de linguagens de programação adicionais (Böszörményi, 1998). A escolha da linguagem a ser ensinada pode ser feita com base na preferência da faculdade, relevância para a indústria, aspectos técnicos da linguagem e na disponibilidade de ferramentas e materiais úteis.

Diversos trabalhos tentam entender o problema da escolha do paradigma e linguagem de programação ideal para cursos introdutórios de programação. Brilliant and Wiseman (1996) discutem o uso dos paradigmas procedural, funcional e orientado a objetos (POO), bem como seus prós e contras. O paradigma procedural é considerando de mais fácil entendimento pelos professores, mas estes desconsideram que

muitas linguagens que utilizam este paradigma, como a linguagem C, dificultam a aprendizagem de conceitos sobre tipos abstratos de dados. O paradigma funcional é aparentemente mais simples, mas, em contrapartida, há o fato de não ser amplamente utilizada no meio comercial e a dificuldade em promover a transição para outras linguagens. O uso do paradigma orientado a objetos (POO) em CS1 evita a mudança de paradigma, pois alguns professores acreditam que é mais difícil aprender o paradigma procedural e depois migrar para o POO do que a transição no sentido oposto. Para Böszörményi (1998) começar com POO, principalmente se utilizando a linguagem Java, coloca uma sobrecarga desnecessária para os estudantes, que podem não conseguir entender completamente conceitos básicos por um longo tempo. Ele exemplifica seu ponto de vista com os *arrays* e matrizes, que são uma abstração fundamental em programação e nada têm a ver com a orientação a objetos, mas em Java são tratados como objetos, confundindo sua compreensão.

As linguagens C, C++ e Java estão no topo das mais escolhidas, embora haja um debate sobre a eficácia do uso dessas linguagens para o ensino de programação, uma vez que não foram criadas para este propósito (Pears et al., 2007). Linguagens como Python, Logo, Eiffel e Pascal, dentre outras, podem ser mais adequadas por atenderem as demandas educacionais dos novatos (Pears et al., 2007).

Alguns pesquisadores defendem o uso da programação visual para simplificar o ensino de conceitos complexos de programação. O objetivo da programação visual é tornar a programação mais fácil para seres humanos através de expressões visuais, onde os usuários visualmente esboçam seu fluxo de programa, ao invés de usar comandos, ponteiros e símbolos abstratos (Burnett, 1999). As expressões visuais incluem diagramas, esboços a mão livre, ícones ou demonstrações de ações realizadas por objetos gráficos.

2.4.4 Ferramentas

Ambientes de desenvolvimento normalmente são desenvolvidos para o trabalho com programação e contém muitas ferramentas e funcionalidades difíceis e até mesmo desnecessárias para os novatos. As ferramentas utilizadas para apoiar a aprendizagem de programação podem ser: ferramentas de visualização, de avaliação automatizada; ambientes de desenvolvimento, e ferramentas de ajuda ao programador de maneira geral (Pears et al., 2007).

Kelleher and Pausch (2005) apresentam uma taxonomia de linguagens e ambientes de programação voltados para novatos. A análise feita organiza os sistemas em duas categorias principais: sistemas que tentam ensinar programação e sistemas que apoiam o uso de programação em busca de outro objetivo. Segundo os autores, os sistemas que tentam ensinar programação são projetados em torno da hipótese de que a barreira primária na aprendizagem encontra-se na mecânica de escrita do programa ou na dificuldade dos estudantes em entender o que acontece durante a execução de um programa. Esses sistemas podem oferecer linguagens mais fáceis de

aprender através da simplificação da entrada de código ou evitando erros de sintaxe (eg. Basic, Turing, Blue, Java Junior e Gnome). Outros sistemas buscam facilitar o processo de digitação de programas permitindo a construção de programas através de objetos gráficos e ações de interface (eg. Logoblocks, Alice, LegoSheets e Leogo). Sistemas dessa categoria podem explorar a estruturação de programas através de novos paradigmas para organização de código (eg. Pascal, Smalltalk, PlayGround e BlueJ). Os sistemas dessa categoria também podem ajudar os estudantes a entenderem a execução de um programa (eg. Turingal, TonTalk e Prototype 2). Sistemas como Robocode e AlgoBlock apoiam a aprendizagem através de contexto motivacional ou aprendizagem social.

Segundo Kelleher and Pausch (2005), outra importante categoria são os sistemas construídos com a crença de que o aspecto importante da programação é que ela permite que as pessoas construam coisas adaptadas às suas próprias necessidades. Sistemas nesta categoria tentam eliminar dificuldades mecânicas para criar programas, demonstrando ações na interface, demonstrando condições e especificando ações para o estudantes (eg. Pygamlion e AgentSheets). O mecanismo para criar programas também pode ser facilitado através de sistemas que melhoram linguagens de programação, tornando-as mais fáceis de entender, melhorando a interação com a linguagem ou com o ambiente (eg. Cobol, Logo, Alice, FLogo, Jive, Visual Agent Talk). Outros sistemas dessa categoria foca em atividades que são enriquecidas pela programação, oferecendo linguagens de domínios específicos como entretenimento e educação (eg. PinballConstructionSet, Mindrover, e StarLogo).

Guzdial (2004b) identifica três famílias de ambientes para novatos que têm sido influentes nas pesquisas em educação em ciência da computação: a família Logo, a família baseada em regras, e a família das linguagens de programação tradicionais. A família Logo, começou com a linguagem Logo, desenvolvida em meados de 1960 como uma ramificação da linguagem Lisp, e cujo objetivo é promover a alfabetização computacional através de uma linguagem simples e uma abordagem motivacional. A família Logo gerou uma rica variedade de ambientes de programação para novatos, como LogoWriter, StarLogo, e Smalltalk. A família baseada em regras possui ambientes de programação descrevem estados do mundo ao invés de dizer ao computador como operar sobre o mundo, em vez de programação imperativa ou processual tradicional. O autor cita Prolog, que resume a programação à atividade básica de estabelecer relações lógicas, como um exemplo de linguagem dessa família. Alguns ambientes dessa família estendem a programação baseada em regras para domínios gráficos, eg. TonTalk, um ambiente onde o programa de um estudante manipula explicitamente personagens que, por sua vez, manipulam dados e estruturas de dados que aparecem como peças de Lego. A família de linguagens de programação tradicionais é composta por ambientes que não alteram a linguagem mas proveem suporte centrado no estudante às linguagens de programação já existentes. Esse tipo de ambiente enfatiza o chamado *scaffolding*, que pode ser descrito como o ato de fornecer apoio adicional que os novatos precisam, mas os especialistas não, ao utilizar linguagens tradicionais. Em particular, a mais sintaxe mais complexa das lingua-

gens de programação tradicionais é vista como um obstáculo para programadores iniciantes, muito do *scaffolding* foi destinado a aliviar a complexidade da sintaxe. Guzdial aponta editores de Pascal, Genie e GPCeditor, como bons exemplos dessa família.

Dada a diversidade de tipos de ferramentas, percebe-se que muitos estudiosos que tentaram novas abordagens, focadas no lado pedagógico do ensino da programação, acabaram criando ferramentas próprias. Assim, muitas ferramentas com propósito educacional vêm sendo desenvolvidas. Porém nem todas são boas opções para serem importadas por outras instituições pois podem conter muito da natureza da instituição de origem (Pears et al., 2007). Mesmo assim, existem ferramentas disponíveis para uso por parte de qualquer instituição ou até mesmo por iniciativa própria do estudante.

O Scratch, Greenfoot , Alice, e o AppInventor são iniciativas, que utilizam conceitos de programação visual e são bem sucedidas na introdução à programação (Resnick et al., 2009; Kölling, 2010; Kelleher et al., 2007). Fincher et al. (2010) descreve e compara os ambientes de aprendizagem Scratch, Greenfoot e Alice. O Scratch é um ambiente de rede rico em mídia, onde os estudantes criam histórias animadas, jogos e apresentações interativas. Greenfoot é um ambiente de desenvolvimento educacional onde os novatos desenvolvem aplicações gráficas interativas, como jogos e simulações. Greenfoot baseia-se na linguagem Java, o que torna possíveis aplicações muito sofisticadas, e prepara os estudantes para ambientes de programação mais gerais. Alice é um ambiente de visualização de programas de animação interativos tridimensionais, onde os novatos criam filmes 3D animados e jogos à medida que aprendem conceitos introdutórios de POO. Para Fincher et al. (2010) os três ambientes visam promover o engajamento imediato em uma atividade atraente e introduzir os estudantes pré-universitários à programação, fornecendo um ambiente que tenta evitar que os estudantes cometam erros de sintaxe. Esses ambientes são usados atualmente em centenas de faculdades e escolas secundárias, em cursos pré-CS1, CS1 e cursos para *non-majors*.

2.5 Motivação

Além da linguagem, paradigma, ambientes e adequação pedagógica mais apropriados para ensinar programação, entender a motivação (ou a falta de) dos estudantes é uma questão chave ao propor experiências de aprendizagem. Disciplinas de programação usualmente tem um caráter prático, mesmo que não estejam previstas aulas práticas na carga horária. Para aprender a programar, o estudante deve gastar tempo praticando. Assim, não é exagero dizer que o estudante deve se sentir motivado a gastar este tempo mesmo que não haja recompensas explícitas em termos de notas em avaliações.

A motivação há muito tempo vem sendo apontada como um fator importante na

aprendizagem. A exemplo disso, Wlodkowski (1978) discute e analisa técnicas e estratégias motivacionais que podem ser usadas pelos professores para fortalecer o desempenho dos estudantes e reforçar atitudes positivas em relação à aprendizagem. Segundo ele:

[...] "A motivação lida com o comportamento humano. A maioria dos psicólogos e educadores, usa a motivação como uma palavra para descrever aqueles processos que podem (a) despertar e instigar o comportamento; (b) dar direção e propósito ao comportamento; (c) continuar a permitir que o comportamento persista; e (d) levar a escolher ou preferir um determinado comportamento."

A partir da definição dada, percebe-se que a motivação é um conceito abstrato e portanto, difícil de medir. Além disso, é natural que pessoas diferentes sintam-se motivados de maneiras diferentes. Considerando uma turma de programação, por exemplo, um aluno pode se sentir motivado pela vontade de aprender a programar, pelo simples medo do fracasso ou simplesmente porque enxerga a disciplina de programação como uma etapa a ser vencida no curso de graduação. Estudantes motivados por razões diferentes podem apresentar abordagens de estudos diferentes.

Jenkins (2001a) afirma que um professor não pode se dar ao luxo de supor que os alunos em uma classe de programação são motivados a aprender a programar, pois eles têm um conjunto de objetivos diferentes e nem sempre estão interessados em adquirir esta habilidade. Jenkins define cinco tipos de motivação: extrínseca, intrínseca, social, de realização e a motivação nula. Na motivação extrínseca, o principal motivador é a carreira e as recompensas associadas que seguirão a partir da conclusão bem-sucedida do curso. Um estudante com motivação extrínseca provavelmente se esforça muito pouco para atividades qual não contabilizam nota. Na motivação intrínseca, o principal motivador é um profundo interesse em computação, ou especificamente programação. Um estudante com motivação intrínseca tende agir por sua própria iniciativa, e formar suas próprias opiniões sobre os materiais que foram ensinados, e se dispor a finalizar as tarefas mesmo que não haja recompensa em termos de nota. Na motivação social o principal motivador é o desejo de agradar a algum terceiro cuja opinião é valorizada. Estudantes motivados socialmente não querem falhar e ser uma decepção para algum grupo cujas opiniões eles valorizam. Na motivação de realização, o principal motivador é ser bom para a satisfação pessoal, ou competir com seus colegas mesmo que não seja algo explícito. O estudante com vontade de realização adotará todas as estratégias que ele acredita que trarão os melhores resultados na forma das notas mais elevadas. A motivação nula é quando não há um fator maior motivador e o estudante "só quer passar".

Bergin and Reilly (2005) apresentam um estudo sobre o papel da motivação num módulo introdutório de programação orientada a objetos. Os resultados encontrados apontam que os alunos que são mais intrinsecamente motivados do que motivados extrinsecamente têm um desempenho melhor. O estudo sugere que quanto maior o nível de motivação intrínseca mais elevado o nível de programação. Já o papel da motivação extrínseca, quando considerado por si só, parece ter pouco efeito sobre o

desempenho em programação.

Dada a importância da motivação no processo de aprendizagem, o modo como uma disciplina de programação é ensinada pode contribuir para aumentar ou minar completamente a motivação dos estudantes. Nikula et al. (2011) relatam, através de estudo de caso longitudinal com duração de cinco anos, como um curso de programação, com altos índices de desistência e reprovação, teve suas taxas melhoradas em mais de 50%, promovendo alterações focadas no aumento da motivação dos estudantes. Foram identificados três problemas principais no curso: dificuldade na disciplina de programação, complexidade do arranjo do curso e motivação limitada dos estudantes. Através do estudo da motivação dos estudantes, foram propostas alterações que visam eliminar a desmotivação e fatores de insatisfação gerais, aumentar os motivadores intrínsecos no curso, tornando-o mais interessante e útil, e introduzir motivadores extrínsecos para aumentar a previsibilidade do comportamento do aluno.

Existe uma necessidade de formatar cursos com materiais e métodos que aumentem o apelo motivacional da instrução. Além disso, é preciso saber como avaliar o fator motivacional em uma turma. Keller (1987) apresenta o modelo Attention, Relevance, Confidence and Satisfaction (ARCS) que inclui conjuntos de estratégias utilizadas para aumentar o apelo motivacional da instrução. O ARCS é fundamentado na teoria valor-expectativa que pressupõe que as pessoas são motivadas a participar de uma atividade se perceberam que ela está ligada à satisfação das necessidades pessoais e se existe uma expectativa positiva de sucesso. O ARCS contém quatro categorias conceituais que suprem muitos dos conceitos e variáveis específicas que caracterizam a motivação humana. A primeira categoria é a atenção, que traz diretrizes para obter, manter, e dirigir a atenção dos estudantes para os estímulos apropriados. A segunda categoria é a relevância, que oferece diretrizes para que a instrução pareça relevante para as oportunidades de carreira presentes e futuras, além de aumentar a sensação de relevância do curso através do modo como ele é ensinado. A terceira categoria é a confiança, que traz diretrizes para fomentar o desenvolvimento da confiança do estudante ajudando a formar a impressão de que algum nível de sucesso é possível se o esforço é exercido. A quarta categoria é a satisfação, que incorpora pesquisas e práticas que ajudam a fazer as pessoas se sentirem bem sobre suas realizações.

O ARCS incorpora um processo sistemático de design, chamado design motivacional, que pode ser usado na criação de material e planejamento de aulas (Keller, 1987). São quatro etapas preestabelecidas; definir, projetar, desenvolver e avaliar. A etapa de definição prevê a classificação do problema motivacional a ser resolvido e a análise da audiência para identificar lacunas motivacionais, e preparar objetivos motivacionais. A etapa de design cria uma lista de potenciais estratégias para os objetivos motivacionais definidos. Na etapa de desenvolvimento são criados os materiais necessários. Na etapa de avaliação, são julgadas as consequências motivacionais do que foi proposto em termos de medidas diretas de persistência, intensidade de esforço, emoção e atitude. A maioria dos estudos sobre motivação em aprender centra-se

na descrição das estratégias motivacionais aplicadas sem o apoio de dados empíricos (Huang et al., 2006). Para diagnosticar problemas motivacionais dentro de materiais instrucionais, Keller (1993) desenvolveu um instrumento de medição chamado Instructional Material Motivation Survey (IMMS). Há 36 declarações em escala de *Likert* na pesquisa e todas elas foram desenvolvidas de acordo com componentes individuais do ARCS.

O modelo ARCS e o IMMS têm sido amplamente aplicados em estudos que visam promover e avaliar estratégias de motivação (Huang et al., 2006). Muitos estudos adaptam o instrumento original para seu próprio contexto. Hamada (2008) introduz um ambiente integrado *web* para ensino de tópicos e teoria da computação, cuja concepção e avaliação levam em consideração as diretrizes do modelo ARCS, e consideram as preferências de aprendizagem ativa e colaborativa de alunos de engenharia de computação. Cho (2014) apresenta uma ferramenta *web* para o ensino e aprendizagem de programação com linguagem C ou C ++ utilizando os critérios e estratégias do modelo ARCS. Savi et al. (2011) propõem um modelo para avaliar a qualidade de jogos educacionais, para ensino de engenharia de software, através das percepções dos estudantes sobre motivação, experiência do usuário e aprendizagem. O modelo proposto se baseia no modelo ARCS para mensurar se o jogo capta a atenção dos estudantes, demonstra relevância e motiva e utilização do conhecimento aprendido. von Wangenheim et al. (2012) apresentam um jogo de tabuleiro educacional para reforçar e ensinar a aplicação de conceitos de Gerenciamento de Valor Agregado no contexto de cursos de graduação em computação. Dentre outros fatores, a motivação é avaliada de acordo com as categorias do modelo ARCS. Esses e outros trabalhos demonstram a flexibilidade do uso do modelo ARCS e uma tendência em se preocupar com a motivação dos estudantes.

2.6 Programação para *Non-majors*

Em 1999 a Academia Americana de Ciências publicou o relatório "Ser Fluente com Tecnologia da Informação"que defende que, dado o papel cada vez mais importante que a tecnologia da informação desempenha no trabalho e na vida pessoal dos cidadãos, é indispensável que todos possam aprender tópicos essenciais desta área (on Information Technology Literacy, 1999). A fluência com a tecnologia da informação requer três tipos de conhecimento: habilidades contemporâneas, conceitos fundamentais e capacidades intelectuais (on Information Technology Literacy, 1999). Habilidades contemporâneas traduzem a capacidade de usar várias aplicações de computador. Conceitos fundamentais se referem aos princípios básicos e conceitos de computação que formam a base da ciência da computação. Já as capacidades intelectuais se referem à capacidade de aplicar a tecnologia da informação em situações específicas, incluindo a resolução de problemas. O relatório defende que a programação constitui os conceitos fundamentais mais importantes da área e

defende que os estudantes universitários devem aprender a programar, a fim de se tornarem fluentes em tecnologia da informação.

O papel cada vez mais marginal da programação no uso geral do computador, levou algumas instituições a eliminarem a programação totalmente em cursos de computação introdutório para estudantes *non-majors* (Forte and Guzdial, 2005). Algumas iniciativas incluem a solução de problemas por meio do uso de computadores mas sem ensinar efetivamente programação, a exemplo de Urban-Lurain and Weinshank (2000), que discordam da ideia de que para se tornar fluente em tecnologia da informação estudantes *non-majors* precisam aprender a programar. Em contra-partida, os autores oferecem uma abordagem construtivista e elaboram um curso de Conceitos e Competências Computacionais, aplicado na Michigan State University, onde conceitos de computação são apresentados aos estudantes fazendo com que eles resolvam uma série de problemas semelhantes aos que eles encontrarão no ambiente de trabalho. Marks et al. (2001) sugerem que a ênfase em programação dada aos cursos introdutórios de ciência da computação não contribui para a motivação e o engajamento dos estudantes. Eles oferecem uma abordagem baseada em estudo de casos para ideias chaves da computação aplicada, que pode ser utilizada em cursos de CS1 para *non-majors*. Esses trabalhos no entanto, não trazem nenhuma evidência científica empírica de que a sua abordagem é superior.

Soloway (1993) investiga a necessidade da universalização da programação através da opinião de vários pesquisadores em aplicações avançadas de tecnologia. Ele usa a metáfora da programação como o "novo latim". Aprender a programar normalmente é defendido como algo que traz múltiplos benefícios, assim como a aprendizagem do latim ajuda no desenvolvimento cognitivo geral. Ele conclui que embora não haja consenso sobre melhor linguagem e abordagem para o ensino da programação, saber programar facilita a aprendizagem de várias áreas temáticas e capacita os estudantes a desenvolver novas maneiras de olhar para o mundo. Outros pesquisadores defendem o ensino da programação para *non-majors* argumentando que, assim como a alfabetização tradicional envolve tanto a leitura e escrita, alfabetização computacional envolve necessariamente a capacidade de criar artefatos computacionais, e não simplesmente usá-los (Forte and Guzdial, 2005; Resnick et al., 2009). Esses autores também argumentam que ambientes e abordagens adaptadas para o contexto dos estudantes facilitam o engajamento.

Muitas instituições americanas têm uma grande quantidade de *non-majors* matriculados nos cursos de CS1. A maioria delas apresentam esta demanda porque tornaram essa disciplina um componente curricular obrigatório. Urban-Lurain and Weinshank (2000) exibem um levantamento das publicações focadas em cursos de CS para *non-majors*, entre os anos de 1979 e 1998, nas principais conferências de educação em computação da ACM e identificam que a maioria dos trabalhos encontrados descrevem cursos cuja finalidade é aprender a programar em uma variedade de linguagens diferentes, incluindo Basic, Scheme, Pascal e etc.

Comer and Roggio (2002) discutem alguns dos problemas associados a ensinar um

curso de CS1 baseado em Java, no Texas Christian University (TCU), para um grupo muito diversificado de estudantes, cuja maioria é de *non-majors*. O curso, oferecido nos moldes tradicionais, faz uso da linguagem Java para atender aos interesses dos estudantes dos cursos de TI e tem exercícios voltados para exploração da diversidade de aplicações possíveis com a linguagem. Os autores afirmam que exitem dificuldades iniciais por parte dos *non-majors* em compreender a orientação a objetos e os detalhes de Java. Não são descritas taxas de aprovação e abandono das turmas mas os autores afirmam, baseados no *feedback* dos estudantes e conselheiros de CS, que a popularidade de CS1 com Java é maior que do que as versões anteriores, com outras linguagens. Os autores atribuem o sucesso ao fato de Java permitir programas com interface gráfica.

Muitos pesquisadores defendem que os cursos de CS1 devem ser reformulados para *non-majors* em vez de ensinados com a mesma abordagem pensada para estudantes da área de TI.

Joyce (1998) apresenta um curso completamente formatado para *non-majors*, focado em desenvolver a habilidade de resolução de problemas. O curso é dividido em três unidades interligadas. A primeira unidade faz uma introdução geral aos computadores e seu uso como uma ferramenta de resolução de problemas. A segunda unidade estuda programação. E a terceira unidade é um estudo de tópicos mais avançados de algoritmos. Segundo o autor, o curso é um sucesso entre os *non-majors* mas não são apresentadas taxas de aprovação e abandono, nem evidências que justifiquem tal afirmação.

Kaplan (2004) discorre sobre como a computação deve ser incluída na educação de um aluno de graduação em ciências e propõe o design de um curso de CS1 para cientistas naturais, físicos e cientistas sociais. Para ele, ao desenvolver um curso de computação que possa ser adotado como um requisito para uma graduação em ciências é preciso um objetivo que seja relevante para os cientistas. Além disso, deve ser dado ao estudante um *background* adequado em computação para que ele possa trabalhar efetivamente com professores e pesquisadores em sua área de interesse e ser capaz de determinar quais áreas da matemática e ciência da computação irão contribuir de forma útil para o seu próprio desenvolvimento futuro. Kaplan elenca quais aspectos da programação devem ser ensinados, quais linguagens, e quais temas devem ser abordados nos exemplos. No curso proposto, é utilizado o Matlab para abordar conceitos básicos de programação, exemplos de aplicações extraídas de uma ampla gama de disciplinas científicas (eg., imagens, sons e relações matemáticas) para reforçar as habilidades de programação. Há ainda uma ênfase na produção de software, incluindo trabalhos em equipe e processo de documentação.

Forte and Guzdial (2005) acreditam que os cursos de CS1 para *non-majors* devem apresentar um contexto dentro da realidade do estudante. Para eles, cursos introdutórios tradicionais não conseguem motivar muitos alunos e podem até mesmo desencorajá-los de prosseguir a aprendizagem futura CS. Em contra partida, cursos alternativos, projetados para acomodar uma variedade de interesses e background,

oferecem um contexto mais motivador e atraente para a aprendizagem de conceitos de programação e CS.

Forte and Guzdial (2005) descrevem a experiência vivenciada no Instituto de Tecnologia da Geórgia (Georgia Tech), Atlanta, onde tradicionalmente, todos os estudantes tomaram o mesmo curso de CS1. No semestre da primavera 2003, foram oferecidos três cursos diferentes de CS1: CS1 tradicional, Introdução a computação para engenheiros e Introdução a computação de mídia. CS1 tradicional é ensinado com a linguaem Scheme tem foco em design e documentação. CS1 para engenheiros, é ensinado usando MATLAB, além de ser feita uma introdução à Java. CS1 com mídia é um curso adaptado para estudantes que não são nem de engenharia, nem ciência da computação, e é ensinado com a linguagem Python. Nesta última abordagem, estudantes aprendem como diferentes tipos de mídias são codificadas pelo computador e como escrever programas para manipulá-las. Os autores avaliam o impacto dos cursos adaptados na motivação dos estudantes. De maneira geral, percebeu-se que estudantes *non-majors* aproveitaram mais os cursos adaptados, e que dentre eles, estudantes de engenharia são os que mais vislumbraram utilidade da programação fora do curso de CS1. O autor identificou que os estudantes de CS1 com computação com mídia estavam aproveitando os aspectos criativos do curso usando suas novas habilidades fora da classe (eg., reproduzir músicas populares ao contrários, alterar fotografias pessoais).

Chilana et al. (2015) apresentam os resultados de um estudo de casos com estudantes calouros de engenharia de gestão matriculados em um curso de CS1 e fornece *insights* sobre motivações dos alunos, objetivos de carreira, percepções de programação e reações às linguagem Java e Processing. Os autores sugerem que, entre a classificação tradicional de não programadores versus programadores para usuários finais versus programadores profissionais, existe uma categoria de alunos que querem ser alfabetizados na programação para conversar na "linguagem do programador". Os autores denominam essa classificação intermediária de "programador conversacional". Esses "programadores conversacionais" percebem que as linguagens-padrão da indústria (por exemplo, Java) são mais benéficas do que linguagens específicas para ensino, que simplificam a sintaxe de programação. Os autores discorrem sobre os limites na ênfase em ambientes e linguagens com objetivos pedagógicos e linguagens e ambientes com objetivos comerciais, sugerindo que essa questão deve ser mais bem explorada.

2.7 Ferramentas, Linguagens e Materiais Escolhidos

A partir da revisão bibliográfica, que permitiu um estudo das soluções comumente exploradas e uma identificação das tendências mais bem sucedidas, foram definidas as ferramentas, linguagens e abordagens para compor a abordagem proposta. A

composição concebida, traz em sua fase inicial uma abordagem com a ferramenta Scratch (Subseção 2.5.1). Em um segundo momento, é feita a introdução à linguagem Python (Subseção 2.5.2), utilizando a ferramenta Turtle(Subitem 2.5.1) para reforçar os conhecimentos adquiridos e facilitar a transição de uma linguagem visual para uma linguagem textual. Ao final, a utilização da linguagem Python é mantida, mas em um contexto de mídias (Subitem 2.5.4).

2.7.1 Scratch

Scratch é uma ferramenta gratuita, criada pelo Media Lab do Massachusetts Institute of Technology (MIT), e que utiliza programação visual para facilitar a aprendizagem de programação através da construção de jogos, simulações, e animações, dentre outros contextos.

A gramática do Scratch baseia-se em uma coleção de "blocos gráficos de comandos de programação" que são unidos para criar programas. Os usuários arrastam blocos de uma paleta e montam-nos, como peças de quebra-cabeças, para criar pilhas de blocos que determinam o comportamento dos objetos (Fincher et al., 2010). Assim, o processo de programação dá uma sensação semelhante à de montar blocos de Lego. Os blocos são moldados para se encaixarem apenas em formas que fazem sentido sintático, e os conectores sugerem como eles devem ser unidos (Resnick et al., 2009). O Scratch dá *feedback* imediato para o estudantes, pois basta clicar em uma pilha de blocos que o seu código é executado imediatamente. Os principais critérios de design do Scratch são a diversidade e a personalização (Resnick et al., 2009). Ele apoia muitos tipos diferentes de projetos (eg., histórias, jogos, animações, simulações) e facilita a personalização dos projetos, importando mídias e criando gráficos. O Scratch está ligada à sua comunidade, organizada de maneira semelhante a uma rede social, onde é possível compartilhar e ver os projetos Scratch de outros usuários.

Muitas escolas de ensino médio e fundamental em todo o mundo, e até mesmo algumas universidades usam o Scratch como um primeiro passo para a programação (Resnick et al., 2009). Malan and Leitner (2007) defendem o uso do Scratch como primeira linguagem de programação em cursos introdutórios de programação, tanto para cursos de graduação na área de TI quanto para *non-majors*. O potencial do Scratch como ferramenta de introdução à programação no ensino superior é justificado pelo fato de ele permitir que os estudantes dominem as construções de programação e se concentrem em problemas de lógica antes da sintaxe. Os autores utilizaram o Scratch em uma versão de verão de um curso de CS1 na universidade de Harvard. Em parte do curso, conceitos básicos de programação foram ensinados através do Scratch e em seguida foi feita a transição para linguagem Java. Os autores constataram que, embora o Scratch não forneça todas as construções disponíveis em paradigmas e linguagens de programação comuns, ele motiva os estudantes e familiariza os inexperientes com fundamentos de programação sem a distração da sintaxe.

de Kereki (2008) apresenta uma abordagem que utiliza o Scratch nas primeiras semanas de num curso de CS1 para estudantes de engenharias e em um curso vocacional , ambos na universidade ORT, no Uruguai. Os autores constatam que o uso do Scratch no início do curso promoveu um alto nível de motivação, com uma percepção positiva dos estudantes em relação a programação, mas não foi encontrada melhoria mensurável nas taxas de aprovação e retenção das turmas que utilizaram o Scratch, em comparação com o curso normal. Mishra et al. (2014) descreve uma intervenção de duas semanas em um curso de CS1, ministrado para estudantes de engenharias, a maioria deles novatos. O objetivo de utilizar o Scratch é nivelar os estudantes novatos e manter o engajamento dos estudantes mais avançados através de desafios. Num segundo momento, o curso é focado em fazer a transição para a linguagem C++. Os resultados obtidos através de estudo de campo, demonstram que os estudantes novatos foram capazes de acompanhar os estudantes mais avançados e que a maioria deles reconhece que o Scratch ajudou-os a aprender conceitos de programação, facilitando a transição para C++.

Bittencourt et al. (2015) relatam o uso do Scratch em uma oficina de introdução à programação para calouros do curso de Engenharia da Computação da UEFS, que ocorreu antes do início das aulas de CS1, com linguagem de programação C. Na oficina, liderada por estudantes veteranos, os participantes desenvolveram animações e jogos no Scratch e recebiam desafios durante o processo de desenvolvimento, em vez de explicações detalhadas. O resultados da survey realizada com os estudantes e análise dos projetos demonstram que, embora a maioria dos estudantes não tenham utilizado o Scratch após a oficina, eles acreditam que o uso da ferramenta os ajudou com a disciplina de CS1, principalmente na aprendizagem de estruturas de controle e operações lógicas.

2.7.2 Python como Primeira Linguagem

Python é uma linguagem de programação de propósito geral, que combina os paradigmas procedural, funcional e orientado a objetos (Lutz, 2009). É comumente definida como uma linguagem de script orientada a objetos - uma definição que combina suporte a POO com uma orientação geral para funções de script. Códigos em Python têm tamanhos menores do que códigos equivalentes em C ++ ou Java. Além disso, Python se encarrega de manter registros de detalhes de memória de baixo nível. Objetos em Python são alocados automaticamente, tem seu espaço recuperado quando não são mais usados ea em sua maioria, podem crescer ou encolher sob demanda (Lutz, 2009).

Python é uma linguagem de propósito geral, muito utilizada por programadores profissionais. Seu uso como primeira linguagem em cursos introdutórios de programação não é tão popular como C, C++ e Java. No entanto, as iniciativas que empregam Python como primeira linguagem para novatos, principalmente em abordagens híbridas, apontam resultados satisfatórios.

Muitas das abordagens encontradas tratam o Python como uma linguagem mais enxuta sintaticamente e mais simples de entender do que outras linguagens de programação de propósito geral. Agarwal and Agarwal (2005) discutem Python e sua adequação a cursos de CS1, CS2 e outros cursos avançados de ciência da computação. São apresentados exemplos que evidenciam as características de Python como uma linguagem promissora para o ensino. Python possui uma sintaxe mais simples e oferece suporte para interfaces gráficas de usuário e POO. São vantagens apontadas: a ausência de símbolos para indicar um bloco de instruções, as variáveis não precisarem ser declaradas e poderem ser atribuídas a qualquer tipo de dado, os laços for podem variar ao longo de um conjunto de palavras, programas com interface GUI poderem ser criados e configurados de maneira rápida. Para os autores, Python é uma boa escolha para CS1 e CS2 e outros cursos de computação.

Jenkins (2004) apresenta características de Python que a torna interessante como primeira linguagem de programação para novatos e constata que existem menos obstáculos cognitivos para novatos superarem com Python do que com outras linguagens de programação. Jenkins compara Python e C++ em termos de detalhes da sintaxe e nos conceitos que um novatos deve entender. Um "Hello World!", por exemplo, em Python ocupa no máximo duas linhas enquanto que em C ++ ocupa aproximadamente sete. Para Jenkins, obstáculos práticos de criação e edição são mais difíceis para os novatos quando o programa é mais longo. Além disso, o número de conceitos que devem ser entendidos antes que um programa possa ser escrito em Python é menor do que em C++. Outras características interessantes de Python é que ela é interativa, permitindo uso do prompt para testar fragmentos de programa, o que torna a experiência de programação mais interativa e acessível. Como Python é interpretada, o programa executará até a parte que contém um erro, e isto facilita o entendimento dos estudantes mais do que um código que não compila. Outra vantagem do Python são as regras para indentação. Em Python, não é possível indentar incorretamente. Se a indentação não mostrar um fluxo de programa lógico é gerada uma mensagem de erro. Isso impõe aos estudantes a necessidade de recuarem corretamente e consistentemente seu código, garantindo melhor legibilidade.

Jenkins (2004) também relata, de maneira informal, a experiência de usar Python como primeira linguagem em um curso de CS1 para estudantes de cursos da área de TI, ensinado com C++. Nas sete primeiras semanas do curso foi feita a introdução com Python, em uma abordagem procedural, e no restante do tempo, 13 semanas, foi feita a transição para a linguagem C++. O plano de estudos começou com as noções básicas, variáveis, valores e instruções de controle e repetição. Jenkins afirma que ensinar C++ após a introdução com Python foi mais fácil, já que todos os alunos agora tinham algum *background* de programação. Para o autor, a transferência entre Python e C++ foi dificultada por conceitos em Python que não são diretamente mapeados para um equivalente em C++, como operadores de multiplicação operando como repetições de string. Jenkins também relata um tutorial extra de Python, ministrados para uma parte da turma, escolhida aleatoriamente. Com isso, ele conseguiu comparar as versões de Python do material de C++, e notou que elas

exigiam menos explicações por parte do professor e menos detalhes para o estudante dominar. A classe com Python foi muito à frente da classe principal em termos de material coberto.

Enbody et al. (2009) analisam a viabilidade do uso do Python em CS1 descrevendo a experiência de troca de linguagem do curso de CS1 de C++ para Python, na Michigan State University. A abordagem proposta apenas muda a linguagem de programação. O impacto dessa mudança foi analisado no segundo curso da sequência, CS2, ministrado com C++. Foram comparadas as notas dos estudantes das turmas de CS2 que haviam participado do CS1 com Python e os que haviam cursado CS1 com C++. O principal resultado foi que não houve diferenças significativas entre os grupos. Isso mostra que o curso CS1 baseado em Python preparou os estudantes para o curso de CS2 e não dificultou a transição entre linguagens. Para os autores, os estudantes de Python tinham a vantagem de conhecer outra linguagem de programação, e por não sentirem dificuldades na transição, Python se mostrou uma alternativa viável para o CS1, mesmo quando as disciplinas subsequentes se baseiem em linguagens diferentes.

Tendo em vista a simplicidade do Python, facilitando o contato inicial dos novatos, esta linguagem também aparece como mais tangível para turmas de CS1 com *non-majors*. Shannon (2003) descreve a alteração no currículo de um curso CS1 para estudantes da área de TI e *non-majors*, que tradicionalmente era ensinado com C++ ou Java. O objetivo da abordagem proposta é atender as necessidades do público diverso, principalmente os *non-majors*, mantendo a ênfase na programação e aumentando a gama de tópicos da ciência da computação. A abordagem proposta usa a linguagem Python, aborda temas relacionados à Internet, questões sociais da computação, e introduz os estudantes a robôs programáveis e a um banco de dados SQL. O Python é utilizada na primeira metade do curso, que tem duração total de 10 semanas. No início é feita uma introdução aos conceitos básicos da programação em Python, e na terceira semana é introduzido um módulo visual, contendo um pacote de gráficos 3D, VPython. Os resultados dessa abordagem mostram que, do ponto de vista do instrutor, a escolha do Python facilita o ensino da programação introdutória, pois a linguagem tem uma sintaxe muito simples. Os professores notaram que nas disciplinas subsequentes, muitos alunos optam por usar Python na escrita de seus programas se eles são autorizados a usar a linguagem de sua escolha.

Para Radenski (2006), a linguagem de programação ideal para cursos de CS1 deve ser simples o suficiente para que possa ser tratada por iniciantes, e ser comercial e orientada a objetos, o que implica complexidade significativa. Ele acredita que Python é a linguagem que consegue cumprir esses requisitos conflitantes. O autor descreve a experiência de reformular o curso de CS1 na Universidade Chapman, Califórnia, EUA, a fim de torná-lo mais atraente para um público diversificado, que inclui estudantes da área de computação e *non-majors* de ciências exatas. A abordagem proposta adota Python e coloca os recursos do curso disponíveis em um pacote de estudo on-line, com programas de amostra, questionários e slides. O foco do curso não é aprender a programar em Python, e sim aprender sobre

computação programando em Python. Uma pesquisa anônima sobre as percepções dos estudantes sobre o curso oferecido revelou que mais da metade dos alunos nunca tiveram contato com programação e que eles gostam e recomendariam Python para seus colegas. Além disso, o Fato do Python ser utilizada por organizações de alto perfil, como Yahoo, Google, e Disney, gera um fator motivacional entre os estudantes, trazendo uma sensação de utilidade.

2.7.3 Turtle

Turtle é a parte mais conhecida da linguagem Logo, uma linguagem derivada de Lisp concebida para tornar a programação acessível às crianças, permitindo que elas explorassem uma grande variedade de tópicos de matemática, ciências, linguagens e música (Guzdial, 2004b). Logo começou justamente como uma "tartaruga" robótica que podia desenhar no chão e foi substituída, mais tarde, por um "ator" simulado em um mundo gráfico bidimensional que pode se mover, girar, e deixar trilhas (Guzdial, 2004b). Com o Turtle, a linguagem Logo foi utilizada para gráficos, e em sentido inverso, os gráficos podem ser utilizados para compreender matemática. Por causa de sua ligação com os gráficos de Turtle, Logo foi bastante popular nas aulas de ciências e matemática (Guzdial, 2004b).

Papert (1980) descreve o Turtle como um estilo computacional da geometria, comparando-o com a geometria euclidiana, que possui um estilo lógico, e a geometria de Descartes, que possui um estilo algébrico. A essência dessa ferramenta está nos comandos e no modo como as construções exigem conceitos básicos de lógica de programação, a medida em que a complexidade da figura desenhada aumenta. Os comandos *forward* e *back* fazem com que a "tartaruga" se mova em linha reta, alterando sua posição, mas não sua direção. Os comandos *right* e *left* mudam a direção da "tartaruga" sem afetar a posição. Esses comandos recebem valores numéricos como parâmetros. Para desenhar uma figura regular de maneira otimizada é preciso repetir as mesmas instruções de posição e direção, levando ao conceito de laços. Para andar em espiral, é preciso dar um passo, em seguida, virar um pouco menos, dar um passo, e depois virar cada vez um pouco menos (ou mais). Para traduzir isso em instruções para a tartaruga, é necessário expressar uma quantidade variável. Assim, o Turtle se destaca com possibilidades que permitem a exploração de lógica e de conceitos chaves da programação (Papert, 1980).

O Turtle do Logo pode ser utilizado para promover uma introdução aos conceitos básicos de programação em cursos de CS1. Hromkovič et al. (2016) descrevem uma abordagem de ensino introdutório de programação utilizando Logo e Python, para ensino fundamental, médio e para *non-majors* no ensino superior. Cada nível de ensino teve seu próprio foco específico e currículo adaptado. No ensino superior, o objetivo é oferecer uma abordagem motivadora e eficiente. Nas primeiras unidades é feita uma introdução geral de como nos comunicamos com os computadores para que eles executem programas e algoritmos. Após esta introdução inicia-se à programação

com Logo. A parte final do curso é ministrada com Python, com exemplos que envolvem uma ampla gama de dados e problemas científicos. Segundo os autores, as primeiras lições aprendidas com o Logo foram valiosas para os estudantes, que foram capazes de aplicar muitos métodos aprendidos em algoritmos mais complexos.

Scratch e Python possuem implementações semelhantes ao Turtle. O Scratch possui comandos para um objeto caneta, semelhantes aos do Turtle. E Python possui a biblioteca Turtle, que possui comandos similares aos do Turtle de Logo. As abordagens para ensino da programação, podem se beneficiar da similaridade entre a caneta do Scratch, o Turtle do Logo e de Python. O ensino de conceitos básicos de programação através do Scratch, que é uma ferramenta visual, facilitam uma transição suave para Python, reforçando os conceitos aprendidos. Dorling and White (2015) abordam o processo de transição d Scratch para Python, aplicado em turmas de nível fundamental e médio. Na abordagem proposta os estudantes aprendiam os conceitos básicos de sequência, estruturas condicionais, de repetição e comandos Turtle, com ambiente visual e depois focavam em mapear esses conceitos para a linguagem baseada em texto.

Vidal Duarte (2016) descreve uma abordagem de ensino para CS1 que faz uso do Python, com sua biblioteca gráfica Turtle. A abordagem é toda voltada para construção de jogos, visando aumentar o interesse e a motivação dos alunos. O trabalho não faz análises muito rigorosas sobre o impacto dessa nova abordagem, mas identifica melhoria nas notas dos estudantes, em comparação com os semestres que não utilizaram essa abordagem.

Um traço comum na maioria dos trabalhos encontrados, principalmente os que propõem abordagens híbridas com Python, é que eles não apresentam investigações bem formuladas sobre a efetividade do ensino dessas abordagens ou as melhorias nos índices de motivação e demais fatores.

2.7.4 Computação com Mídias

Computação com mídia é o termo utilizado para descrever abordagens que contextualizam o ensino de programação através da criação e manipulação de mídias digitais. Além de oferecer contextualização para uma audiência que é normalmente mal servida por cursos CS1 tradicionais, a criação de mídias digitais pode ser numerada entre as novas habilidades de alfabetização para a era da computação (Forte and Guzdial, 2004). Esse tipo de curso cria um contexto criativo para a programação, onde os estudantes podem entender como as mídias são codificadas pelo computador e criar suas próprias mídias, enquanto aprendem conceitos de introdutórios de programação e ciência da computação. Normalmente, o objetivo dos cursos CS1 que utilizam computação com mídia não é tornar o aprendizado da programação mais fácil e, sim, motivar e aumentar o engajamento dos estudantes para, a partir daí, reduzir as taxas de reprovação e desistência.

Para Forte and Guzdial (2004) a disciplina de ciência da computação genérica não consegue satisfazer as necessidades de um corpo estudantil diversificado. As altas taxas de reprovação e desistências de *non-majors* em CS1, principalmente estudantes do sexo feminino, é sintomática de uma falha em comunicar o valor da computação para os estudantes. Se a ciência da computação não é percebida como interessante ou útil, os estudantes falham ou abandonam. Eles descrevem a experiência de utilizar computação com mídia em uma turma de CS1, para 120 estudantes fora da área de TI nem de engenharias, no Georgia Institute of Technology (Georgia Tech). No curso oferecido, os estudantes implementam filtros estilo *PhotoShop* em imagens, efeitos especiais de vídeo digital e sons, utilizando linguagem Python. Os estudantes são apoiados por um site colaborativo que permite publicação de perguntas, discussão de problemas. Foram coletadas quantitativas e qualitativas para mensurar os resultados. Ao final do curso, apenas tês estudantes desistiram e a taxa de reprovação e desistência foi de 11,5%, em comparação com os 42,9% registrados na versão tradicional. Além disso, 60% dos estudantes demonstraram interesse em fazer uma versão avançada do curso, e 94% não estariam interessados em um curso de ciência da computação em outros moldes. Os autores não investigam, no entanto, se os estudantes são capazes de transferir para outros contextos os conceitos computacionais adquiridos no curso.

Tew et al. (2005) investigaram se os resultados obtidos por Forte and Guzdial (2004) são replicáveis para outras instituições, utilizando a mesma abordagem de CS1 com computação com mídia em uma turma similar para *non-majors*, no Gainesville College, Gainesville. Vale ressaltar que o Georgia Tech é uma universidade voltada para pesquisa, enquanto Gainesville College é uma faculdade pública de dois anos. Os resultados finais, após a utilização de computação com mídia ao longo de quatro semestres, apontam uma melhoria significativa nos níveis de retenção, assim como no estudo anterior. Em comparação com os resultados do Georgia Tech, a parcela de estudantes interessados em fazer uma versão avançada do curso foi ainda maior, provando que a abordagem é bem sucedida mesmo em outra instituição. No entanto, não foram feitas quaisquer análises qualitativas, para obter mais informações sobre a adaptação desta inovação em outras instituições, o que pode enfraquecer os resultados obtidos.

A eficácia do uso da computação com mídia tem sido comprovada também para estudantes dos cursos da área de TI. Sloan and Troy (2008) descrevem o uso da computação com mídia em um curso introdutório de programação para calouros de computação, denominado CS 0.5, ministrado antes do curso de CS1 na University of Illinois, Chicago. O curso CS 0.5 era ensinado da maneira tradicional e seu objetivo é nivelar os estudantes recém chegados e aumentar as taxas de retenção, mas da maneira tradicional não vinha obtendo bons resultados. Os autores utilizam a mesma abordagem descrita por Forte and Guzdial (2004) para reformular o curso de CS 0.5. Eles consideram Python uma boa escolha de linguagem e acreditam que, como a maioria dos alunos desfruta de multimídia, provavelmente se sentiriam motivados ao manipular imagens, músicas e vídeos. Os autores investigam a efetividade do

ensino com a nova abordagem e o quão boa ela é para melhorar a retenção ao longo das disciplinas seguintes de programação. Os resultados obtidos ao longo de três anos aplicando a nova abordagem sugerem que a taxa de aprovação aumentou em média 10% em relação a abordagem antiga. Já o índice médio de retenção aumentou em mais de 20%. Este estudo se limita a uma análise puramente quantitativa. Por tanto, apesar dos bons resultados a efetividade do ensino não é comprovada, pois o design experimental não leva em conta fatores confundidores.

Simon et al. (2010b) descrevem o uso da computação com mídia no curso de CS1, para estudantes da área de TI, na University of California, San Diego. A abordagem proposta difere de Forte and Guzdial (2004), pois utiliza a linguagem Java, conforme abordagem descrita por Guzdial and Ericson (2009). Os autores escolheram a linguagem Java, pela proximidade dos instrutores com esta linguagem e para diminuir fatores confundidores na comparação com a versão antiga do curso, ensinada em Java. Os autores comparam quatro cursos de 10 semanas, ministrados pelo mesmo professor e na mesma instituição. Dois cursos foram ensinados com a abordagem tradicional, utilizando um livro texto popular em Java e os outros dois cursos foram ensinados usando a nova abordagm. As duas abordagens são comparadas em termos de competências desejadas. Os estudantes da nova abordagem tiveram melhor desempenho em construções do tipo *if*, *loops*, *loops* aninhados, vetores e matrizes. Os autores analisam também o uso dessas construções fora do contexto de mídias e a média de acerto em todas as construções foi maior que 65%, ultrapassando 90% nas construções *ifs* e *loops* aninhados. As taxas de aprovação também apresentam um aumento significativo em relação a abordagem tradicional.

Jython Enviroment for Students

O Jython Environment for Students (JES) é um ambiente de desenvolvimento, criado em 2002 para apoiar o curso de Computação com Mídia na Georgia Tech. Muitas instituições que adotam a abordagem com mídia também utilizam o JES, como Sloan and Troy (2008) e Tew et al. (2005).

O JES é um ambiente que utiliza Jython. O Jython é uma implementação da linguagem de script Python escrita em Java puro, que é executado em qualquer Máquina Virtual Java (JVM) compatível (Pedroni and Rappin, 2002). Usando o Jython, é possível escrever programas Python que interagem com qualquer código Java, permitindo o uso simplificado de bibliotecas e APIs do Java, como o *toolkit* de interface gráfica Swing (Pedroni and Rappin, 2002).

O JES foi projetado com foco pedagógico para facilitar a programação dos alunos em computação com mídia com um mínimo de sobrecarga cognitiva (Barr and Guzdial, 2016). Ao contrário das IDEs de uso geral, sua interface não possui muitas opções e só é possível editar um arquivo de cada vez. Os estudantes podem programar qualquer coisa em JES, pois é uma implementação completa em Python. Porém, o JES possui recursos adicionais para facilitar a manipulação de sons, imagens e vídeos,

como, por exemplo, acessar pixels individuais, alterar cores e acessar amostras de som, sem bibliotecas importadas e sem detalhes extras (Barr and Guzdial, 2016). Para ajudar os estudantes com a indentação do código, o JES desenha uma pequena caixa azul em torno das linhas no mesmo bloco contendo a linha atual (que pode ser determinada pelo cursor).

O livro "Introduction to Media Computation: A Multimedia Cookbook in Python", Guzdial (2004a), aborda o uso da computação com mídia, utilizando a linguagem Python, mas a partir do JES. Os nomes das funções e exemplos de códigos nesse livro consideram apenas o ambiente JES e não fazem parte de uma distribuição normal do Jython, embora a sintaxe da linguagem seja Python normal. Esse livro pode ser utilizado como livro-texto em disciplinas que ensinam programação no contexto de mídias.

Capítulo 3

Metodologia

Este trabalho foi conduzido utilizando métodos mistos (quantitativo e qualitativo) de pesquisa de maneira concomitante. Segundo Creswell (2010), a pesquisa de métodos mistos é uma abordagem da investigação que associa abordagens quantitativas e qualitativas a fim de realizar uma análise abrangente de um problema de pesquisa onde a força geral deste estudo possa ser maior que a da pesquisa quantitativa ou qualitativa isoladas. Ainda segundo este autor, os procedimentos de métodos mistos concomitantes são aqueles onde o pesquisador coleta as duas formas de dados ao mesmo tempo e integra as informações na interpretação dos resultados gerais. Com base nas definições formais para a metodologia, a razão de se combinar dados quantitativos e qualitativos é entender melhor as relações envolvidas no ambiente de ensino, sobretudo utilizando a abordagem proposta. Ao convergir as tendências numéricas da pesquisa quantitativa e os detalhes da pesquisa qualitativa, espera-se fazer uma análise exploratória do problema de pesquisa que resulte na melhoria da abordagem proposta e em *insights* significativos para a comunidade de pesquisa em educação em computação.

No contexto de métodos mistos, decidimos adotar uma metodologia de estudos de caso na condução desta pesquisa. Estudos de caso são uma estratégia de investigação em que o pesquisador explora profundamente um programa, um evento, uma atividade, um processo, ou um ou mais indivíduos Creswell (2010). Segundo este autor, num estudo de caso, o pesquisador explora um sistema limitado (um caso ou múltiplos casos) ao longo do tempo, através da coleta de dados envolvendo múltiplas fontes de informação, como observações, entrevistas, documentos, relatórios, dentre outros, e relata uma descrição do caso. Estudos de caso usualmente são vistos como uma metodologia de pesquisa qualitativa, mas Yin (2013) defende abordagens quantitativas e qualitativas para o desenvolvimento de estudos de caso. Foi escolhida a metodologia de estudos de caso devido à natureza do problema abordado, que exige uma análise exploratória do problema de pesquisa, onde não há controle sobre os eventos comportamentais envolvidos e o foco do estudo está em eventos contemporâneos.

Neste trabalho, a avaliação da abordagem proposta foi realizada através da condução de dois estudos de caso. Foi conduzido um Estudo de Caso Piloto com uma turma de ICC na UEFS ao longo do semestre letivo 2016.1. A partir dos resultados obtidos, foram promovidas melhorias na metodologia proposta e nos instrumentos de avaliação. O Estudo de Caso Final foi conduzido em uma turma de ICC na UEFS ao longo do semestre letivo 2017.1[1]. Para ambos os estudos de Caso, houve etapas de planejamento, realização e análise dos dados. Também investigamos o cenário tradicional de ICC em nossa instituição através de entrevistas com professores mais frequentemente alocados para ministrar esta disciplina. Os subtópicos a seguir explicam os principais aspectos metodológicos deste estudo.

3.1 Elaboração da Abordagem de Ensino-Aprendizagem

A abordagem de ensino-aprendizagem proposta se baseia em soluções comumente exploradas, identificadas através da revisão de literatura, e nos requisitos de ensino estabelecidos no currículo de ICC. A partir das definições de conteúdo, abordagem, linguagens e ferramentas preparou-se um plano de curso para a disciplina de ICC. Em seguida, foram elaboradas as aulas e instruções de conduta que foram disponibilizadas para os professores participantes dos estudos de caso e que, posteriormente, serão disponibilizados para a comunidade científica. Considerando o calendário acadêmico, as aulas foram agrupadas em 3 unidades.

A partir dos resultados alcançados através do Estudo de Caso Piloto, foram identificados os pontos fortes e fracos da abordagem proposta. Sendo assim, nesta etapa a abordagem foi reformulada a fim de encorajar as práticas que influenciaram positivamente a motivação e o aprendizado dos estudantes.

3.2 Estudo de Caso Piloto

Na Universidade Estadual de Feira de Santana (UEFS), as turmas de Engenharia Civil e Engenharia de Alimentos possuem em sua matriz curricular o componente obrigatório EXA 170 – Introdução à Ciência da Computação (ICC). A disciplina possui carga horária total de 60 horas, divididas meio a meio entre aulas teóricas e práticas. Para o curso de Engenharia Civil, esta disciplina faz parte da oferta do primeiro semestre, juntamente com as disciplinas de Cálculo I e Geometria Analítica, dentre outras.

O Estudo de Caso Piloto foi conduzido em uma turma de ICC do curso de Engenharia Civil, durante o semestre 2016.1 (entre os meses de agosto de 2016 e fevereiro de

[1] 2016.1 e 2017.1 são dois semestres subsequentes, de acordo com o calendário da instituição.

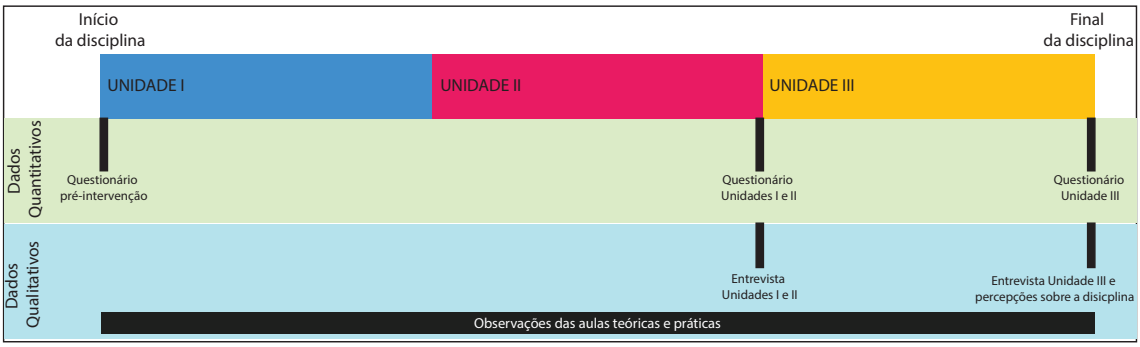

Figura 3.1: Coleta de dados durante o Estudo de Caso Piloto.

2017). A turma do semestre 2016.1 foi composta por 40 calouros de Engenharia Civil e 14 estudantes veteranos dos cursos de Engenharia Civil e Engenharia de Alimentos que já cursaram a disciplina em algum outro momento, totalizando 54 estudantes. As aulas teóricas foram ministradas em sala de aula para todos os estudantes. As aulas práticas foram ministradas em laboratório, em três horários diferentes. Os estudantes calouros tiveram aulas práticas em duas turmas, com 20 estudantes cada, enquanto os estudantes veteranos tiveram as aulas práticas em uma terceira turma. Por decisões do departamento a disciplina contou com quatro professores, três deles ministraram as aulas teóricas, cada um responsável por uma unidade diferente. Dado que o público alvo deste estudo foram os calouros do Curso de Engenharia Civil, acompanhamos a turma com aula teórica e apenas as turmas práticas que eles participaram.

Os professores da disciplina adotaram os materiais e procedimentos definidos na abordagem. Após cada aula, os slides e exemplos utilizados foram postados no site da disciplina[2].

3.2.1 Coleta de Dados

A coleta de dados foi feita através da aplicação de três questionários em momentos distintos do curso, da realização de entrevistas em dois momentos diferentes do curso e das observações feitas nas aulas teóricas e práticas. A Figura 3.1 exibe a organização da coleta de dados qualitativos e quantitativos durante este estudo de caso.

O primeiro questionário, foi concebido para ser aplicado no início do curso, juntamente com o Termos de Consentimento Livre e Esclarecido (TCLE). Ele tem como objetivo fazer o levantamento do perfil dos participantes da pesquisa e entender a percepção da importância da computação na vida acadêmica, profissional e pessoal dos estudantes, além de expectativas mantidas em relação a disciplina. O questionário foi elaborado a partir do questionário utilizado por Bittencourt et al. (2015), em

[2]https://sites.google.com/site/uefsicc20161/aulas

sua experiência com um mini-curso de Scratch para calouros do curso de Engenharia de Computação da UEFS. Neste Primeiro questionário, também foram utilizados alguns construtos, retirados do questionário apresentado por Liebenberg et al. (2015), que visam medir a familiaridade do estudante com a execução de tarefas triviais relacionadas a computadores, como enviar *e-mails* e realizar pesquisas.

O segundo questionário, foi concebido para ser aplicado ao final da segunda unidade. Seu objetivo é conhecer a opinião da turma a respeito do ambiente Scratch e da segunda unidade com Python. As questões referentes ao Scratch foram desenvolvidas a partir da adaptação do IMMS, sugerida por Huang et al. (2006) e tentam inferir o nível de motivação dos estudantes ao utilizarem o Scratch de acordo com as categorias Atenção, Relevância, Confiança e Satisfação do Modelo ARCS. As demais questões procuram levantar a opinião dos estudantes em relação à abordagem de ensino nas aulas teóricas e práticas, na transição de Strach para Python, e na dificuldade em entender conceitos de programação.

O terceiro questionário, tem como objetivo conhecer a opinião da turma a respeito da unidade com computação com mídia, do ambiente JES, colher a opinião dos estudantes sobre a abordagem empregada nas aulas teóricas e práticas e as dificuldades em entender conceitos de programação com a linguagem Python. Assim como no segundo questionário, os primeiros construtos buscam inferir o nível de motivação dos estudantes em relação aos materiais e ferramentas da terceira unidade a partir de uma adaptação do IMMS.

As entrevistas foram aplicadas no fim da segunda unidade e ao final do curso, com sete estudantes participantes escolhidos aleatoriamente. A primeira entrevista, tem o objetivo de entender as expectativas dos estudantes em relação à disciplina, levantar a opinião deles sobre o Scratch, a linguagem Python, as aulas teóricas e práticas, e as atividades realizadas. A segunda entrevista busca colher a opinião dos estudantes em relação à Unidade III, e avaliar o uso de mídias, além de colher a opinião dos estudantes em relação à disciplina de maneira geral.

3.2.2 Análise de Dados

Antes de fazer a triangulação dos dados qualitativos e quantitativos, cada conjunto de dados recebeu tratamento e análise individual. Para o conjunto de dados quantitativos, composto pelos três questionários desenvolvidos, os dados foram tabulados e foram geradas estatísticas descritivas, incluindo tabelas de frequência e gráficos de barra empilhadas. Para os construtos repetidos entre os questionários, como por exemplo os referentes a percepção dos estudantes sobre a utilidade da disciplina, foram submetidos ao teste não paramétrico de Wilcoxon, adequado para testar diferenças entre duas condições onde diferentes participantes foram selecionados em cada uma delas.

Os dados qualitativos somaram 37 arquivos de texto contabilizando entrevistas e observações das aulas. Para realizar a codificação dos dados, utilizamos técnicas de

análise de conteúdo. A primeira etapa analítica foi a codificação aberta dos dados, através da nomeação dos segmentos de dados mais significativos utilizando termos adequados para categorizar, resumir e representar cada trecho de texto. Nestas etapas iniciais não foram aplicados categorias ou códigos preconcebidos aos dados. Além disso, em alguns trechos foi empregada a codificação *invivo* com o intuito de preservar termos mais expressivos do discurso dos participantes. A codificação inicial gerou 127 códigos que, posteriormente, foram analisados e agrupados de acordo com o significado e contexto de cada um. Esta etapa de agrupamento seguiu procedimentos típicos de codificação axial. Além das categorias que emergiram nos dados, realizamos a codificação axial em torno dos eixos Atenção, Relevância, Confiança e Satisfação do Modelo ARCS. A partir das categorias mais altas foram elaborados memorandos com as definições das categorias, descrição das relações identificadas entre elas e suas possíveis causas e consequências. Os resultados obtidos serviram em sua maioria para promover mudanças na abordagem, nos materiais utilizados, e nos próprios questionários e entrevistas.

3.2.3 Participantes da Pesquisa

O projeto de pesquisa foi apresentando aos estudantes na primeira aula e foram distribuídos o TCLE (Apêndice A) e, após concordância na participação, o primeiro questionário. Para este estudo, foram considerados apenas os dados dos estudantes calouros, por considerar que estes possuem um perfil semelhante e estariam nivelados, em termos de conhecimentos gerais, pelo vestibular. Dentre os calouros, um total de 36 estudantes autorizou o uso dos dados levantados a partir da disciplina, sendo 27 do sexo masculino e 9 do feminino.

Através do primeiro questionário, fizemos um breve levantamento sobre o perfil dos participantes da pesquisa. Em relação ao uso de internet, 80,6% admitiram que acessam a internet todos os dias e 19,4% acessam quase todos os dias. 76,5% acessam a internet na maioria das vezes através de celular ou tablet, enquanto o restante o faz através de computador desktop ou notebook.

Investigamos a frequência com que os estudantes realizam atividades básicas com o computador ou relacionadas ao uso de tecnologia e os resultados podem ser vistos na Figura 3.2. Percebemos que, para a maioria dos estudantes, tarefas relacionadas ao uso de internet e diversão, buscar informações sobre algo na internet, baixar músicas e jogar games são realizadas com bastante frequência. Já atividades relacionadas a computação ou engenharia, como escrever um programa de computador, ler sobre computadores em livros e revistas, "construir" um dispositivo ou abrir um dispositivo para descobrir como funciona, para a maioria dos estudantes nunca, ou poucas vezes foi feita. Apenas dois estudantes admitiram já ter visto algo sobre programação: um deles teve aulas online sobre HTML e CSS e o outro já havia cursado uma disciplina de introdução a programação com Portugol e linguagem C em outra instituição de ensino superior.

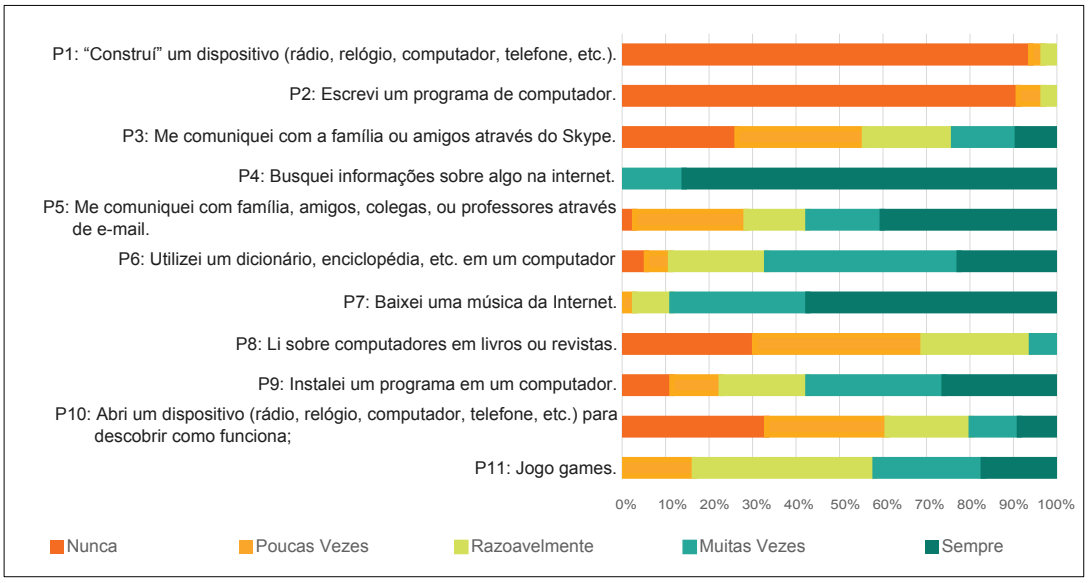

Figura 3.2: Perfil do uso do computador.

3.3 Estudo de Caso Final

O Estudo de Caso Final foi conduzido em uma turma de ICC do curso de Engenharia Civil, mesmo componente curricular em que foi conduzido o Estudo de Caso Piloto. O período de execução é ao longo do semestre 2017.1 (período entre março e agosto de 2017).

A turma do semestre 2017.1 foi composta por 40 calouros de Engenharia Civil. As aulas teóricas foram ministradas em sala de aula para todos os estudantes, enquanto as aulas práticas foram ministradas em laboratório, em dois horários diferentes. Ao contrário do semestre 2016.1 toda as aulas foram ministradas pelo mesmo professor, que é também orientador deste trabalho. Como toda a turma foi composta exclusivamente por calouros, todas as aulas foram acompanhadas para realização de observações. As aulas práticas contaram com monitoria de um estudante de Engenharia de Computação.

O professor da disciplina adotou os materiais e procedimentos definidos na abordagem. Após cada aula, os slides e exemplos utilizados são postados no site da disciplina[3].

[3]https://sites.google.com/site/uefsicc20171/aulas

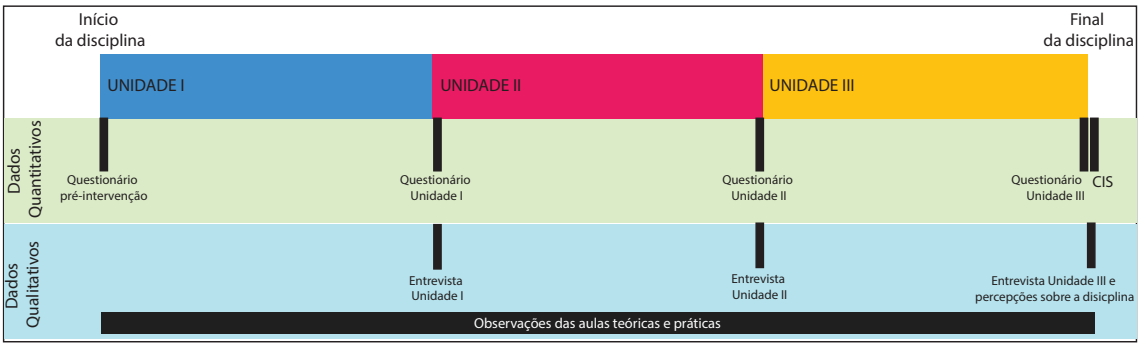

Figura 3.3: Coleta de dados durante o Estudo de Caso Final.

3.3.1 Coleta de Dados

A coleta de dados se deu através da aplicação de cinco questionários em momentos diferentes do curso, da realização de entrevistas em três momentos diferentes do curso e das observações feitas nas aulas teóricas e práticas. Os instrumentos de coleta de dados que já haviam sido desenvolvidos para o estudo de caso anterior foram adaptados para comportar as mudanças na abordagem. A Figura 3.3 exibe a organização da coleta de dados qualitativos e quantitativos durante este estudo de caso.

O primeiro questionário, que pode ser visto no Apêndice B foi concebido para ser aplicado no início do curso, juntamente com o TCLE (Apêndice A). Ele tem como objetivo fazer o levantamento do perfil dos participantes da pesquisa e entender a percepção da importância da computação na vida acadêmica, profissional e pessoal dos estudantes, além de expectativas mantidas em relação a disciplina. O questionário foi adaptado do questionário utilizado no estudo de Caso Piloto.

O segundo questionário, foi concebido para ser aplicado ao final da primeira unidade (Apêndice C). Este questionário foi elaborado a partir do questionário empregado na Unidade I do estudo de Caso Piloto. Seu objetivo é conhecer a opinião da turma a respeito do ambiente Scratch e da Unidade I. As questões referentes ao Scratch foram desenvolvidas a partir da adaptação do IMMS, sugerida por Huang et al. (2006) e tentam inferir o nível de motivação dos estudantes ao utilizarem o Scratch de acordo com as categorias Atenção, Relevância, Confiança e Satisfação do Modelo ARCS. As demais questões procuram levantar a opinião dos estudantes em relação à abordagem de ensino nas aulas teóricas e práticas.

O terceiro questionário, foi concebido para ser aplicado ao final da segunda unidade (Apêndice D). Seu objetivo é conhecer a opinião da turma a respeito da linguagem Python e da Unidade II. Os construtos do IMMS, sugerido por Huang et al. (2006), foram adaptados para corresponder à configuração desta unidade e tentam inferir o nível de motivação dos estudantes ao utilizarem a linguagem Python de acordo com as categorias do Modelo ARCS. As demais questões procuram levantar a opinião dos estudantes em relação à abordagem de ensino nas aulas teóricas e práticas, em

relação à transição de Strach para Python, e às dificuldades em entender conceitos de programação em uma linguagem textual.

O quarto questionário, tem como objetivo conhecer a opinião da turma a respeito da Unidade III, incluindo a abordagem no contexto de mídias, o ambiente JES e a abordagem empregada nas aulas teóricas e práticas (Apêndice E). Este questionário deve ser uma adaptação do questionário empregado na Unidade III do Estudo de Caso Piloto. Os primeiros construtos buscam inferir o nível de motivação dos estudantes em relação aos materiais e ferramentas da terceira unidade a partir de uma adaptação do IMMS.

O quinto questionário foi aplicado ao fim da disciplina, com o intuito de levantar a motivação dos estudantes em relação à abordagem de ensino-aprendizagem empregada durante toda a disciplina (Apêndice F). Este questionário é baseado no Course Interest Survey (CIS), proposto por Keller (2009), com o objetivo de medir a motivação dos estudantes em relação a um determinado curso. Assim como o IMMS, o CIS foi elaborado para estar em correspondência com a base teórica representada pelos conceitos motivacionais e teorias que compõem o modelo ARCS. Ao adicionar este questionário, pretendemos reforçar os estudos acerca da motivação dos estudantes ao participarem deste estudo de caso.

As entrevistas foram aplicadas ao final de cada unidade, com alguns dos estudantes participantes escolhidos aleatoriamente. A primeira entrevista, realizada com sete estudantes, tem o objetivo de entender as expectativas dos estudantes em relação à disciplina e levantar a opinião deles sobre o Scratch(Apêndice G). A segunda entrevista, realizada com quatro estudantes, tem como objetivo colher a opinião dos estudantes em relação à Unidade II, a utilização da linguagem Python, e a transição entre Scratch e Python (Apêndice H). A terceira entrevista, realizada com oito estudantes, busca colher a opinião dos estudantes em relação à Unidade III, e avaliar o uso de mídias, além de colher a opinião dos estudantes em relação à disciplina de maneira geral (Apêndice I).

3.3.2 Análise dos Dados

A análise dos dados ocorreu de maneira semelhante à empregada no Estudo de Caso Piloto. Cada conjunto de dados recebeu tratamento e análise individual antes da triangulação.

Para o conjunto de dados quantitativos, os dados foram tabulados gerando estatísticas descritivas e inferenciais. Com os dados provenientes de construtos baseados em IMMS e CIS foram realizados testes para normalidade do conjunto de dados e posteriormente testes para categorizar a significância das mudanças entre os resultados para as categorias do modelo ARCS ao longo das três unidades. Foram realizados testes do tipo *Kolmogorov-Smirnov* e Teste t. Vale ressaltar que, neste estudo de caso, pretende-se analisar a aprendizagem dos estudantes através da análise dos artefatos coletados em sala de aula, como os projetos Scratch e o questionário

de aprendizagem aplicado ao final da disciplina. Com isso, os dados quantitativos devem ser analisados em relação à dois aspectos diferentes.

Os dados qualitativos, compostos por 42 arquivos de texto, dentre entrevistas e observações das aulas, também foram tratados em um processo semelhante ao empregado no Estudo de Caso Piloto. A primeira etapa analítica é a codificação aberta dos dados, que foi realizada à medida que os artefatos foram coletados. Nesta etapa inicial, categorias ou códigos preconcebidos não foram aplicados aos dados. A partir dos códigos gerados pela codificação inicial, foi realizada a codificação axial a partir de categorias emergentes, descobertas em um trabalho de análise e agrupamento dos códigos de acordo com o significado e contexto de cada um. Além das categorias que emergiram nos dados, realizamos a codificação axial em torno dos eixos Atenção, Relevância, Confiança e Satisfação do Modelo ARCS. A partir das categorias descobertas, foram elaborados memorandos com as descrições das categorias, descrição das relações identificadas entre elas e possíveis relações de causa e consequência. Também foi feita a triangulação entre a codificação do Estudo de Caso Piloto e do Estudo de Caso Final, a fim de obter uma análise mais rica em torno dos dados qualitativos. A última etapa do processo de análise foi a triangulação entre os dados qualitativos e quantitativos a fim de fortalecer a interpretação dos resultados.

3.3.3 Participantes da Pesquisa

O projeto de pesquisa foi apresentando aos estudantes na primeira aula e foi distribuído o TCLE (Apêndice A). Após concordância com a participação, o primeiro questionário foi distribuído. Dentre os estudantes, um total de 37 estudantes autorizou o uso dos dados levantados a partir da disciplina, sendo 25 do sexo masculino e 12 do feminino.

Através do primeiro questionário, fizemos um breve levantamento sobre o perfil dos participantes da pesquisa. Em relação ao uso de internet, 91,9% admitiram que acessam a internet todos os dias e 5,4% acessam quase todos os dias. 80,6% acessam a internet na maioria das vezes através de celular ou tablet, enquanto o restante o faz através de computador desktop ou notebook.

Investigamos a frequência com que os estudantes realizam atividades básicas com o computador ou relacionadas ao uso de tecnologia e os resultados podem ser vistos na Figura 3.4. A maioria dos estudantes costuma realizar tarefas relacionadas ao uso de internet e diversão enquanto atividades relacionadas a computação ou engenharia, como escrever um programa de computador, “construir” um dispositivo ou abrir um dispositivo para descobrir como funciona, para a maioria dos estudantes, é realizada com pouca ou nenhuma frequência. Apenas três estudantes admitiram já ter estudado algo sobre programação.

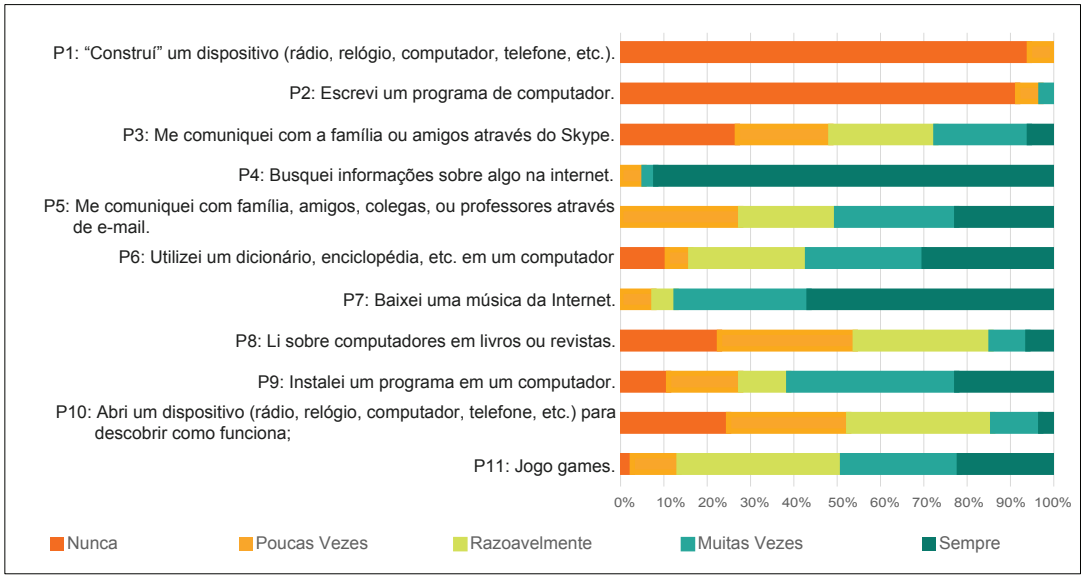

Figura 3.4: Perfil do uso do computador.

3.4 Estudo Sobre o Cenário Tradicional de ICC

Para promover uma análise mais aprofundada, buscamos entender o contexto tradicional de ICC. Entrevistamos os três professores de nossa instituição que são mais frequentemente alocados para ministrar disciplinas de ICC para non-majors. Nenhum dos professores entrevistados participaram dos estudos de caso fazendo uso de nossa abordagem. Também aplicamos um questionário CIS em uma turma de ICC para engenharia civil no semestre 2017.2. A turma foi composta apenas por calouros do curso de engenharia civil e ministrada por professores que não participaram no semestre 2016.1 ou 2017.1. Cerca de 39 participantes concordaram com o TCLE, e portanto, tiveram suas respostas contabilizadas.

Os dados provenientes do questionário CIS foram tabulados, passaram por teste de normalidade e, em seguida, foram submetidos a testes de comparação entre médias, com os dados provenientes do questionário CIS aplicado no Estudo de Caso Final. Os dados provenientes das entrevistas foram codificados nosoftware *NVivo*,num processo semelhante ao empregado para análise dos dados qualitativos dos estudos de caso realizado.

3.5 Padrão de Referência para os Extratos Qualitativos

Para garantir a fácil identificação da origem dos extratos qualitativos citados neste documento, organizamos a referências do seguinte modo:

- Trechos de entrevistas com estudantes: são referenciados pelo estudo de caso, EP se estudo de caso piloto e EF se estudo de caso Final, seguido pela letra S e o código atribuído ao estudante.
- Trechos de observações das aulas: são referenciados pelo estudo de caso, pela unidade da aula que foi observada, pelo código AP se for aula prática ou AT se for aula teórica, e pelo número da aula. Se houver comentário com impressões do observador, estará sinalizado com CO.
- Entrevistas com professores: são referenciadas pela letra T seguida do código atribuído ao professor.

Capítulo 4

Abordagem e Materiais

Para a abordagem de ensino-aprendizagem proposta foram identificados os objetivos educacionais, os assuntos a serem abordados, a forma como esses assuntos devem ser abordados em sala de aula e foram elaborados os materiais a serem utilizados em sala de aula pelos professores. Sob uma perspectiva de design, o modelo ARCS Keller (1987) foi consultado durante o processo de elaboração da abordagem, principalmente na formatação da apresentação do conteúdo e desenvolvimento dos materiais utilizados em aula.

Neste capítulo, apresentamos a abordagem que foi empregada durante a intervenção feita no Estudo de Caso Final. Também apresentamos os materiais utilizados para apoiar o processo de ensino-aprendizagem, incluindo as atividades avaliativas.

4.1 Organização da Disciplina

A disposição das aulas e das atividades considera uma disciplina com carga horária de 60 horas, das quais 30 horas correspondem à carga horária de aulas teóricas e 30 horas correspondem à carga horária de aulas práticas. A partir dos resultados obtidos com o Estudo de Caso Piloto, alguns aspectos da abordagem foram modificados, como por exemplo à adição ou remoção de algumas aulas e à reformulação de algumas fases do processo avaliativo, mas os objetivos educacionais, os assuntos abordados e as atividades realizadas em sala de aula permaneceram os mesmos.

Os temas foram divididos em três unidades, a Unidade I utiliza o ambiente lúdico Scratch num contexto de criação de Jogos, a Unidade II utiliza linguagem Python com a biblioteca Turtle num contexto de desenhos geométricos e a Unidade III utiliza a linguagem Python e o ambiente JES num contexto de edição de imagens. Uma visão geral pode ser obtida na Tabela 4.1.

A Unidade I, descrita detalhadamente na tabela do Anexo J, utiliza o ambiente lúdico Scratch para fazer a introdução à lógica e aos conceitos básicos de programa-

Tabela 4.1: Organização da Disciplina.

	Objetivos de Aprendizagem	**Conteúdos Abordados**	**Atividades Realizadas**
Uni I	-Ser capaz de se expressar através da construção de animações e jogos no Scratch. - Aplicar conceitos de algoritmos para projetar a solução de pequenos problemas.	- Ambiente Scratch: visão geral. - Comandos de aparência, movimento, caneta, sensores, controle, variáveis e expressões. - Entrada de dados através de mouse e teclado. - Saída de dados na tela. - Variáveis e expressões aritméticas. - Estruturas de seleção e expressões lógicas. - Estruturas de repetição. - Comunicação entre objetos.	- 3 aulas teóricas, com tutoriais de construção de jogos em Scratch. - 3 aulas práticas, com exercícios em Scratch; - Avaliação extraclasse através de dois desafios; - Avaliação prática, em laboratório.
Uni II	- Ser capaz de criar programas simples para desenhos de figuras através da biblioteca Turtle; - Ser capaz de implementar e utilizar funções de maneira consistente.	- Expressões lógicas e estruturas de seleção. - Estruturas de repetição. - Estruturas de dados básicas. - Funções em Python (declaração, utilização e parâmetros). Biblioteca Turtle. - Bibliotecas básicas de Python.	- 4 aulas teóricas com tutoriais de desenhos de figuras geométricas. - 3 aulas práticas com exercícios para desenho de figuras geométricas. - Avaliação extraclasse através de dois desafios. -Avaliação prática em laboratório.
Uni III	- Ser capaz de se expressar, em nível básico, utilizando a linguagem de programação Python, através de estruturas de controle, processamento e entrada e saída; - Compreender, de maneira introdutória, a estrutura de armazenamento de mídias digitais. - Ser capaz de implementar funções para manipulação de elementos de cor e estrutura de uma imagem; - Compreender estruturas de dados básicas em memória e usá-las para armazenar dados em memória. - Aplicar os conceitos de arquivos para armazenamento e recuperação de informação.	- Entrada e saída. - Expressões aritméticas. - Arquivos de mídia. - Expressões lógicas e estruturas de seleção. - Estruturas de repetição. - Estruturas de dados básicas. - Funções e módulos em Python. - Bibliotecas para manipulação de imagens. - Erros e exceções.	- 2 aulas teóricas com jogos textuais e programas com uso de entrada e saída no JES. - 1 aula prática com implementação de programas com entrada e saída. - 6 aulas teóricas com tutoriais de manipulação de imagens em Python. -Projeto Final Editor de Imagens Tabajara.

ção, como estruturas de controle e variáveis. Além destes conceitos, são introduzidos ainda o conceito de procedimentos, possibilitado pela versão do Scratch utilizada. Nas primeiras lições, estes temas são abordados utilizando a caneta do Scratch. Nas aulas subsequentes, esses temas são novamente abordados no contexto da construção de jogos clássicos, como *Pong, Space Invaders, Bow and Arrow,* e Interlagos. Além dos conceitos iniciais, são abordados conceitos comuns no *design* de jogos, como animação de aparência e movimentação dos objetos, colisão entre objetos e pontuação. Nas aulas teóricas, são utilizados poucos slides explicativos e o professor utiliza a projeção para mostrar como construir os exemplos. Em cada prática laboratorial, os estudantes são desafiados a construir um jogo diferente, com dificuldade de desenvolvimento maior que o desafio apresentado na semana anterior. Os estudantes foram avaliados através da proposição de dois desafios extra-classe e da avaliação final da unidade, realizada em laboratório.

A Unidade II, descrita detalhadamente no Apêndice K, utiliza a linguagem de programação Python e a biblioteca Turtle do Python. É utilizado o ambiente padrão de desenvolvimento IDLE. Na primeira aula, mostra-se como os desenhos feitos com a caneta do Scratch são implementados de maneira equivalente com o Turtle. As demais aulas se dedicam a explorar os mesmos conceitos abordados na unidade anterior e, adicionalmente, o conceito de função, prevendo a declaração, uso correto de funções e passagem de parâmetros. Nas aulas teóricas, o professor explica esses conceitos, através da construção de exemplos que fazem desenhos mais complexos, como caleidoscópios e tabuleiros de xadrez. Nas práticas laboratoriais os estudantes recebem um guia de desafios com atividades semelhantes aos exemplos explorados na aula teórica anterior. A avaliação do desempenho dos estudantes foi feita através da proposição de dois desafios extra-classe e da avaliação final da unidade, realizada em laboratório. Antes da avaliação, há uma aula teórica de revisão, cujo objetivo é abordar o conceito de variáveis e de funções de um ponto de vista mais formal, explicando sobre os diferentes tipos de variáveis e como cada tipo de variável é codificado pelo computador.

A Unidade III, descrita detalhadamente no Apêndice L, reforça os conceitos já aprendidos e, adicionalmente, conceitos de vetores, matrizes, entrada e saída de dados e construções mais complexas como laços aninhados. O ensino é contextualizado pela computação com mídia, e os estudantes continuam utilizando o Python, mas através do ambiente JES. As três primeiras aulas, duas teóricas e uma prática, visam promover a introdução ao ambiente JES e trabalham exemplos de propósito geral, como jogos textuais e cálculo de áreas de figuras geométricas. Estas aulas foram adicionadas devido aos resultados do Estudo de Caso Piloto, que levam a crer que uma transição de contexto mais leve é benéfica aos estudantes. Além disso, o uso de programas de propósito geral no início dessa unidade contribui para reforçar a utilidade do Python como linguagem mais profissional. As demais aulas teóricas abordam tópicos sobre a criação e manipulação de imagens. Os estudantes entendem como uma imagem é codificada pelo computador e como manipular as propriedades de uma imagem. Os exemplos feitos em sala de aula vão desde a implementação de

filtros de imagens (e.q., negativo, escala de cinza), até a implementação de efeitos como *Chroma Key*. A avaliação dos estudantes foi realizada por meio de um projeto de implementação de um editor de imagens, com alguns dos filtros implementados em sala, mas com exigências específicas para garantir que o estudante explore, por conta própria, os conceitos ensinados. As práticas laboratoriais foram utilizadas para acompanhar o desenvolvimento do projeto de cada estudante, nos mesmos moldes das unidades anteriores, com um guia descritivo das especificações do editor de imagens.

4.2 Materiais Utilizados

Não foram empregados livros-texto para a disciplina. Todos os materiais necessários para as aulas foram desenvolvidos e compartilhados previamente com os professores. À medida em que as aulas ocorrem, os slides e exemplos são postados, junto à programação da disciplina no site de apoio.

Para cada aula teórica planejada, os professores receberam uma pasta contendo os slides que poderiam ser utilizados e os exemplos a serem implementados durante a aula. Nas pastas da terceira unidade, foram adicionados exemplos de imagens para manipulação dos exemplos oferecidos. Para cada aula prática planejada, os professores receberam uma pasta contendo o jogo ou os exemplos a serem desenvolvidos pelos estudantes, um guia de desafio para ser distribuído para os estudantes com as atividades a serem desenvolvidas.

Durante a primeira unidade, todas as atividades, inclusive as avaliativas, foram apoiadas pelo ambiente Scratch versão 2.0. A Figura 4.1 exibe a tela do Scratch com o jogo *Space Invaders* implementado. A visualização instantânea do resultado do programa escrito é um dos pontos fortes dessa ferramenta e auxilia os estudantes novatos. Durante as aulas teóricas, foram implementados no Scratch exemplos diversos com o uso da caneta, o jogo *Space invaders* e *Bow Arrow*. Em cada uma das aulas práticas os estudantes implementaram um jogo: *Pong*, Arco e Flecha, e o jogo de corrida Interlagos. Estes jogos apresentam um nível de complexidade crescente. Todas as atividades avaliativas utilizaram o Scratch. Dois desafios foram propostos onde, a partir de projeto base no Scratch, os estudantes deveriam adicionar uma funcionalidade pedida.

A avaliação prática da Unidade I foi elaborada seguindo algumas diretrizes para avaliação do pensamento computacional descritas em Brennan and Resnick (2012) e em Seiter and Foreman (2013). A prova foi composta por 3 questões de dificuldade incremental. Cada questão, com dificuldade incremental dava ao estudante a possibilidade de escolher entre dois projetos distintos para implementar a atividade pedida. As questões solicitavam ações como conserto de *bugs* e implementação de novas funcionalidades.

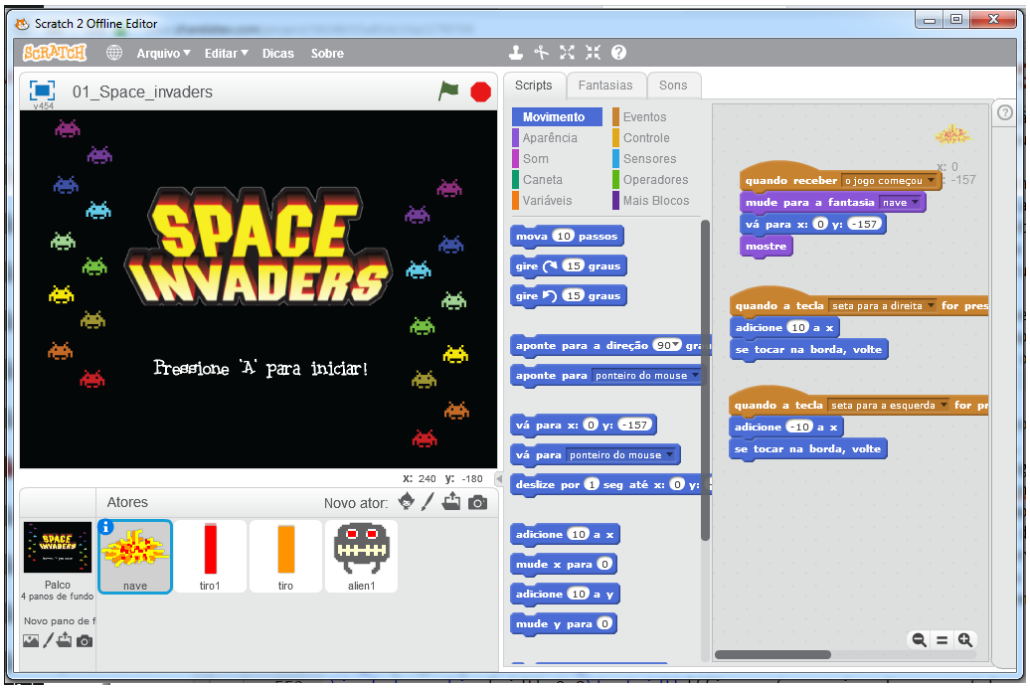

Figura 4.1: Tela do Scratch 2.0 com o jogo Space Invaders.

Durante a segunda unidade, todas as atividades utilizaram exemplos com o objetivo de desenhar figuras geométricas através da biblioteca *Turtle*. Os exemplos iniciais implementavam polígonos regulares, e foram utilizados para introduzir conceitos de estruturas de seleção e repetição e comandos da biblioteca *Turtle*. Os exemplos de nível intermediário consistiram de implementações de combinações de polígonos regulares, e foram utilizados para introduzir a necessidade de funções, aprimorar o uso de estruturas de seleção e de repetição, e introduzir o conceito de variável. Os exemplos finais, implementaram desenhos de estrelas e caleidoscópios.

A Figura 4.2 traz um dos desafios da aula prática, que solicitava a criação deu uma função céu, que recebesse como parâmetro a quantidade de estrelas, se estas estrelas terão a forma preenchida, e a cor das estrelas, como resultado a função desenha estrelas com estes parâmetros em posições aleatórias da tela. Ao longo da Unidade os estudantes implementaram um desafio para desenhar sus próprios nomes com um formato de letra especificado e um desafio para criar uma função que desenha um tabuleiro estilo xadrez mas com tamanho variável. A avaliação prática também foi completamente baseada em questões do Python com a biblioteca *Turtle* e conta com quatro questões de dificuldade crescente. A primeira questão pede que o estudante faça o desenho de uma figura geométrica e as demais solicitam a adição de funcionalidades a este programa inicial de maneira que o estudante precisa aplicar os conceitos de estruturas condicionais, estruturas de repetição, funções com parâmetros, variáveis e comandos da biblioteca *Turtle*.

Na Unidade III, a linguagem Python foi mantida. O ambiente de desenvolvimento

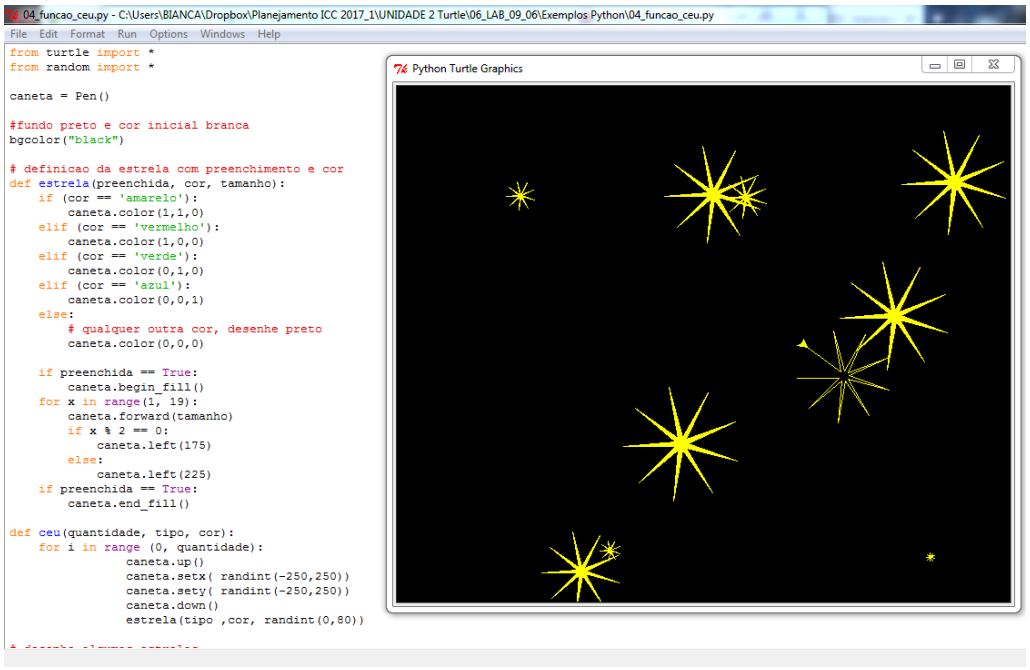

Figura 4.2: Desafio Céu, solicitado como atividade avaliativa da Unidade II.

JES foi utilizado por possuir diversas funções nativas, descritas em um guia detalhado, que facilitam a manipulação de imagens, e por sua interface simplificada, sem funcionalidades complexas pensadas para programadores profissionais.

A Figura 4.3 mostra a interface do JES, com a ajuda aberta para a função *addTextWithStyle* que adiciona textos à uma imagem passada como parâmetro. Outro ponto interessante é o console do JES que é incorporado a janela principal. Nas primeiras aulas, foram apresentados exemplos de propósito geral, como jogos textuais de adivinhação de números e programas para cálculo de notas. A partir da terceira semana, os exemplos utilizados estavam no contexto de manipulação de imagens e eram apresentados em um nível de complexidade crescente. Os primeiros exemplos tinham como objetivo carregar automaticamente diversas imagens de uma mesma pasta ou adicionar textos as imagens através de funções prontas do JES. Os exemplos intermediários exigiam a implementação dos efeitos através de varreduras nas imagens e utilizando funções pré-definidas do JES para obtenção de informações sobre os objetos de imagens e propriedades de cor. Os exemplos mais avançados modificavam a estrutura da imagem, através da manipulação de matrizes de maneiras mais complexas, como no exemplo de imagem espelhada da Figura 4.4. Exemplos mais complexos, como *Chromakey* na Figura 4.5, também foram utilizados durante as aulas teóricas. A atividade avaliativa desta unidade consistiu da implementação de um projeto chamado Editor de Imagens Tabajara. O projeto do editor de imagens exigiu a implementação de alguns dos filtros mostrados durante a aula teórica, mas com parâmetros de implementação diferentes. Para garantir a originalidade no trabalho dos alunos, foram elaboradas mais de uma descrição para o projeto, com

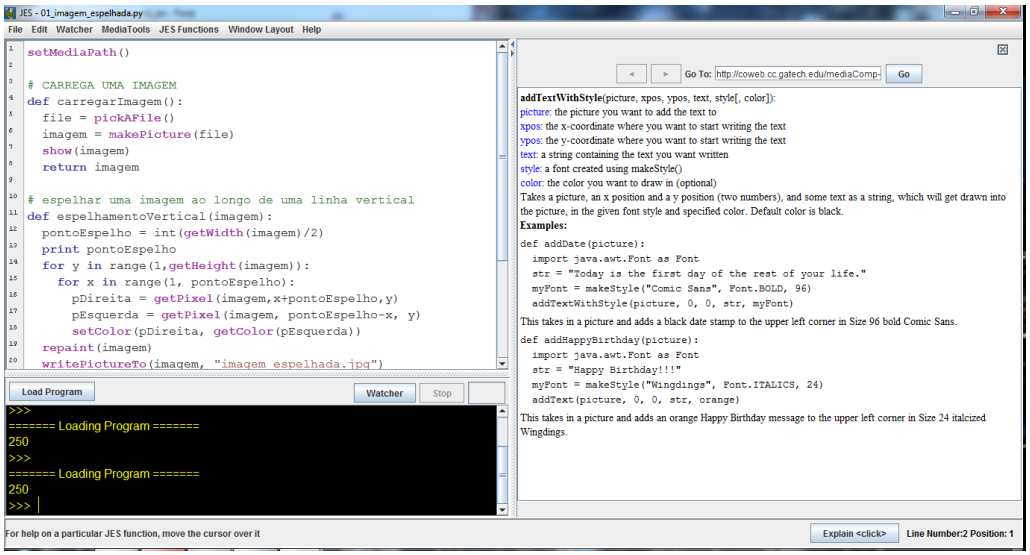

Figura 4.3: Interface do JES.

funcionalidades diferentes em cada uma.

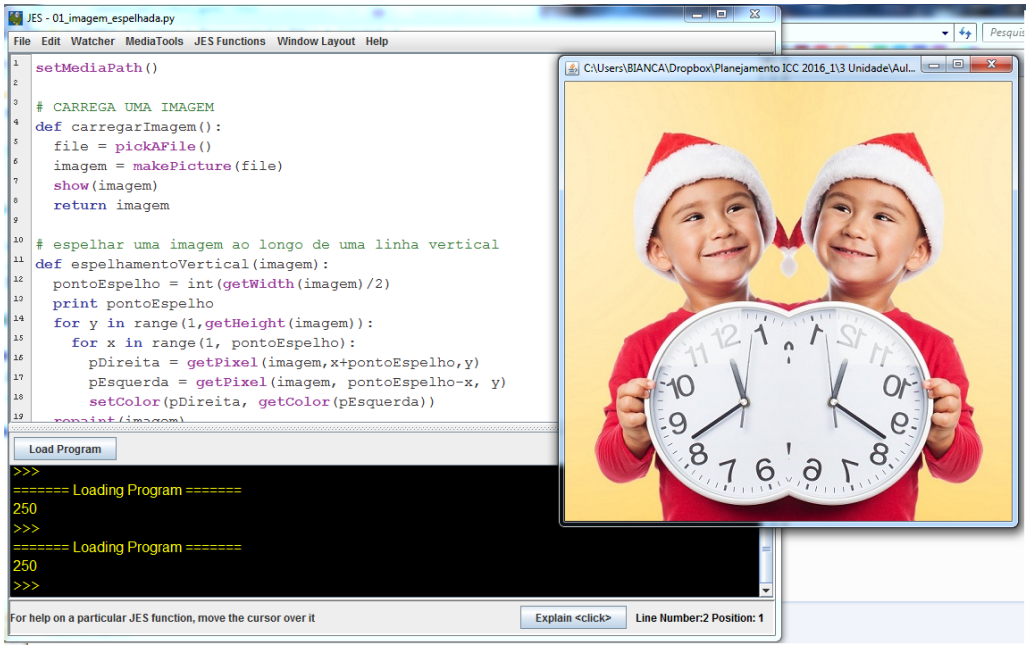

Figura 4.4: Exemplo imagem espelhada.

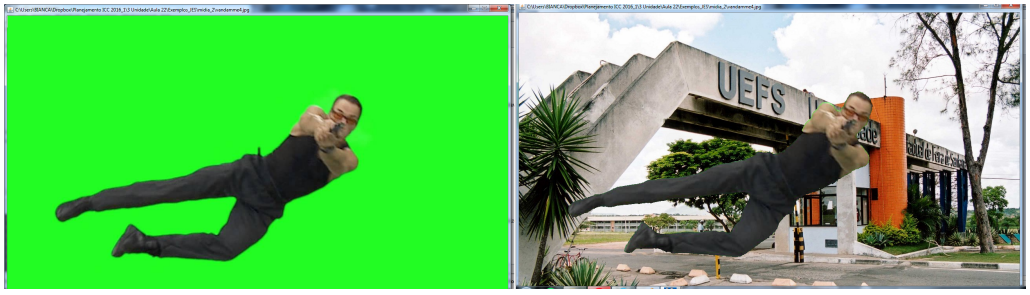

Figura 4.5: Resultado do efeito chromakey.

Capítulo 5

Aprendizagem

Avaliamos a aprendizagem dos estudantes sob diferentes pontos de vista. Através dos questionários, perguntamos sobre a facilidade que os estudantes tiveram em aprender os conceitos ensinados e, através das observações, procuramos identificar os temas que geraram mais dúvidas. Também analisamos os resultados das avaliações práticas realizadas nas Unidades I e II no Estudo de Caso Final. Buscamos identificar as dificuldades dos estudantes, bem como fazer uma avaliação da aprendizagem de conceitos considerando que, em cada unidade, o ensino de programação se deu através de contextos diferentes. Apresentamos os resultados e discussões para cada unidade a seguir.

5.1 Unidade I

Para avaliar a aprendizagem dos estudantes durante a Unidade I, consideramos os resultados das observações das aulas teóricas e práticas, os resultados da avaliação prática realizada ao final da unidade e a opinião dos estudantes através do questionário II. Os dados qualitativos são provenientes dos estudos de caso Piloto e Final, enquanto os dados quantitativos são provenientes apenas do Estudo de Caso Final. Para analisar os resultados da avaliação prática realizada ao final da unidade, agrupamos as questões de acordo com as categorias definidas no *framework* PECT, que categoriza o pensamento computacional através do nível de habilidade utilizado para a implementação de programas em Scratch (Brennan and Resnick, 2012). No *framework* PECT, existem três níveis de variáveis: Variáveis de evidência, que são um conjunto de construções de programação explícitas; Variáveis de padrão de projeto, que são modelos para completar determinadas tarefas; e conceitos de pensamento computacional. Este *framework* é descrito detalhadamente no Capítulo 2. A Tabela 5.1 descreve as variáveis identificadas em cada questão. Como foram elaborados quatro modelos de prova, os estudantes não responderam às mesmas questões e, consequentemente não foram testados para as mesmas variáveis de padrão de *design*, mas o nível de dificuldade foi respeitado de maneira que a primeira questão em

Tabela 5.1: Classificação das questões da Avaliação Prática da Unidade I.

Questão	**Variáveis de padrão de design / nível**	**Variáveis de padrão de design mapeadas para conceitos de Pensamento Computacional**	**Estudantes que responderam a questão**
Q1 - 1	Animar aparência / básico	Procedimentos e Algoritmos	44,44%
Q1 - 2	Manter score / básico	Procedimentos e Algoritmos	55,56%
Q2 - 1	Animar movimento / proficiente	Procedimentos e Algoritmos Representação de dados	44,44%
Q2 - 2	Animar aparência / proficiente	Procedimentos e Algoritmos Paralelização e sincronização Representação de dados	55,56%
Q3 - 1	Manutenção de score / proficiente	Procedimentos e Algoritmos Decomposição de problema Abstração	28,13%
Q3 - 2	Animar aparência / proficiente	Procedimentos e Algoritmos Representação de dados	43,75%
Q3 - 3	Animar aparência / proficiente + Manter score / básico	Procedimentos e Algoritmos Decomposição de problema Abstração Representação de dados	28,13%

todos os modelos exigiam nível similar de habilidades. Em cada modelo de prova, o estudante respondia a um tipo de Q1, um tipo de Q2 e um tipo de Q3.

O gráfico da Figura 5.1 mostra o desempenho dos estudantes em cada questão. Consideramos "não proficiente" o estudante que obteve abaixo de um terço da nota, "proficiente" o estudante que obteve nota acima de 2 terços e "parcialmente proficiente" estudantes que não se encaixaram em nenhum dos grupos anteriores. A questão Q1-2 obteve menor parcela de estudantes que puderam ser considerados proficientes (55%) enquanto todas as outras questões obtiveram nível de proficiência maior que 80%. Este resultado leva a crer que há uma dificuldade maior por parte dos estudantes em resolver questões que envolvam uso de variáveis, principalmente se considerarmos que as questões Q1-1 e Q1-2 envolvem as mesmas habilidades em termos de pensamento computacional. No entanto, outra questão que envolve exclusivamente manutenção de score, Q3-1, apresentou resultado satisfatório, mesmo envolvendo manutenção de escore em um nível proficiente. Porém, esta questão foi respondida apenas por 28,13% dos estudantes. Como no modelo PECT os níveis de proficiência nos padrões de design estão mapeados em conceitos de pensamento computacional, percebemos que, com o Scratch, os estudantes apresentaram poucas dificuldades com procedimentos e algoritmos, decomposição de problemas, paralelização, abstração e representação de dados.

Ao final da unidade, solicitamos aos estudantes que destacassem a facilidade com que tiveram para aprender determinados conceitos traduzidos como categorias de blocos de Scratch. Os resultados encontram-se na Figura 5.2. Utilizamos uma escala de

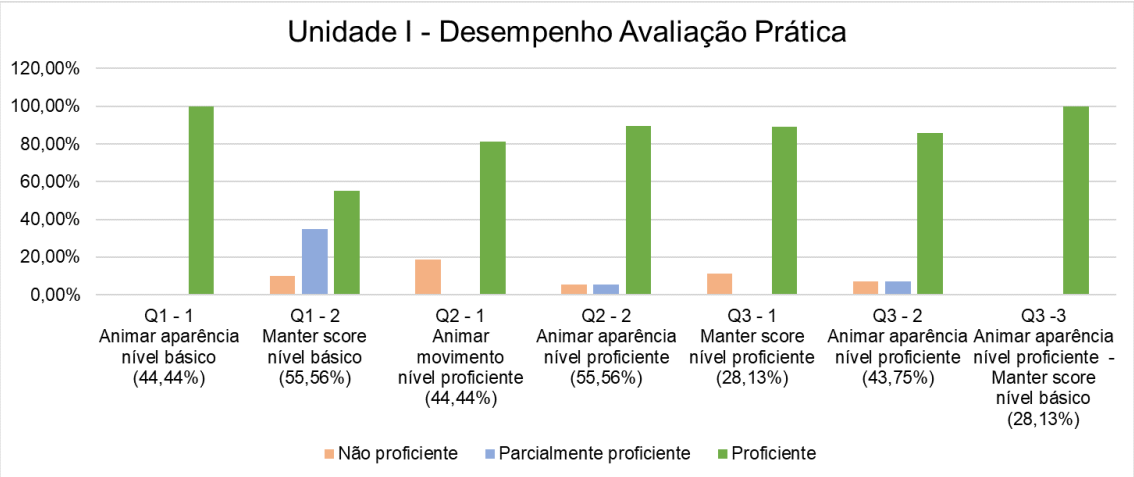

Figura 5.1: Desempenho dos estudantes na Avaliação Prática da Unidade I.

Likert com cinco pontos de muito difícil a muito fácil. Nenhum estudante escolheu as opções difícil ou muito difícil para as categorias de movimento e aparência. Para a categoria referente ao uso de variáveis, 41,17% dos estudantes consideraram difícil ou muito difícil e 26,7% consideraram regular a facilidade em seu uso, enquanto as demais categorias foram avaliadas como fácil ou muito fácil por mais de 50% dos estudantes.

A dificuldade em entender o conceito de variáveis a partir de Scratch também foi evidenciada através das observações das aulas teóricas e práticas, uma vez que os estudantes apresentaram um volume muito maior de dúvidas sobre este conceito em relação aos outros. As dificuldades com uso de variáveis foram predominantes na Unidade I em comparação com as unidades subsequentes. Durante as observações das aulas práticas na Unidade I são comuns trechos que evidenciam as dificuldades dos estudantes em utilizar variáveis, tanto no Estudo de Caso Piloto, quanto no Estudo Final: "Alguns não zeram a variável eles apresentam de modo geral dificuldade com o conceito de variáveis. (...) Peruca não sabe como contar as voltas. Eu falei que é com uma variável e que coisa parecida ele tinha feito com o arqueiro. (CO: Peruca tem dificuldade com variável, ele não entendeu o conceito)" (EP-UI-AP04); "Camisa de fogo está tirando dúvidas sobre como implementar o placar ele não tem noção de como usar variáveis, inteligentinho tbm não sabe acabou de me perguntar como zerar o placar, sendo que estava na aba de variáveis. Eles não estão entendendo o conceito." (EF-UI-AP03).

Durante a Unidade I, os estudantes também apresentaram dificuldades com o fluxo de execução de um programa, utilização de números aleatórios, sensores e dificuldades em trabalhar com o plano cartesiano e, em menor destaque, estruturas de repetição. O conceito de números aleatórios parece ter ficado mais bem entendido após utilizar funções de Python responsáveis por gerar números aleatórios. Já a dificuldade em trabalhar com o plano cartesiano pode se relacionar com falta de base

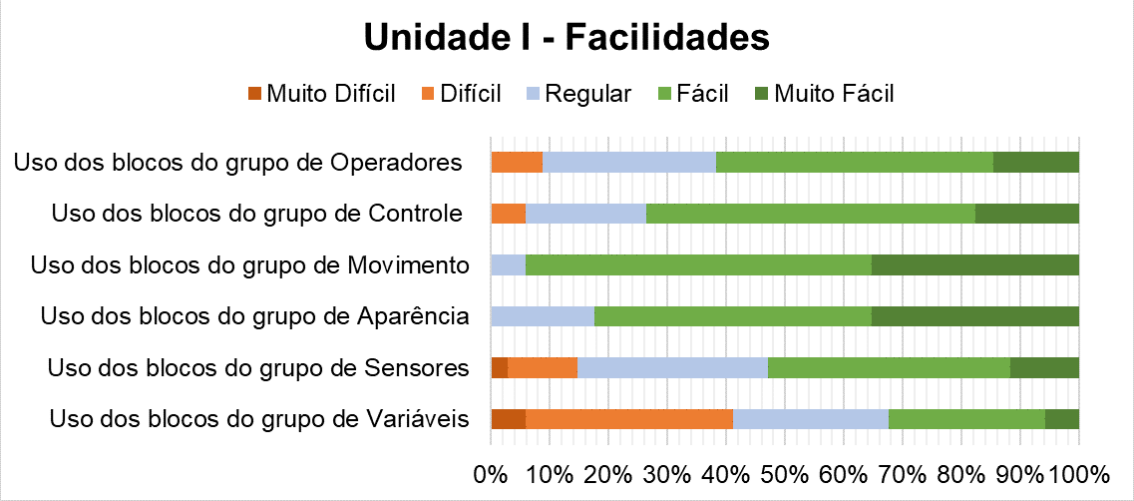

Figura 5.2: Opinião dos estudantes sobre a facilidade em aprender os conceitos de programação na Unidade I.

matemática, mas não temos nenhuma evidência comprobatória.

5.2 Unidade II

Para avaliar a aprendizagem dos estudantes durante a Unidade II, consideramos os resultados das observações das aulas teóricas e práticas, os resultados da avaliação prática realizada ao final da unidade e a opinião dos estudantes através do questionário III. Assim como na Unidade I, os dados qualitativos são provenientes dos estudos de caso piloto e final, enquanto os dados quantitativos são provenientes apenas do estudo de caso final.

Para a avaliação prática desta unidade, elaboramos quatro modelos similares de prova, cada um com 4 questões organizadas em nível de dificuldade crescente. Considerando que as questões foram similares, agrupamos os resultados para questões correspondentes nos diferentes modelos de prova. A tabela 5.2 abaixo lista os tipos de questões e as habilidades a serem avaliadas.

Classificamos as notas obtidas pelos estudantes nas categorias Não proficiente, Parcialmente proficiente e Proficiente. É considerado proficiente o estudante que obteve nota acima de 2 terços do valor da questão e é considerado não proficiente o estudante que obteve nota abaixo de um terço do valor da questão. Os resultados evidenciam que, que a medida em que o programa solicitado envolve combinações mais complexas de comandos e funções, o desempenho da turma cai consideravelmente. Os resultados podem ser vistos na Figura 5.3. Para a questão 1, que testa os conhecimentos a respeito dos comandos da biblioteca Turtle, todos os estudantes demonstraram proficiência. Para a questão 2, onde foi solicitado a implementação

Tabela 5.2: Descrição das questões da avaliação prática da Unidade II.

Questão	Atividade solicitada	Habilidades avaliadas
Q1	Desenho de um polígono regular utilizando comandos da biblioteca Turtle.	Uso dos comandos da biblioteca Turtle.
Q2	Implementação de um programa que tenha uma função responsável pelo desenho de um polígono regular e que receba como parâmetro o tamanh(o da figura a ser desenhada. A função implementada deve ser utilizada para formar uma figura que combine desenhos sucessivos deste polígono.	Criação e uso de funções simples.
Q3	Implementação de um programa que tenha uma função responsável pelo desenho de um polígono. Esta função recebe como parâmetros o tamanho do polígono a ser desenhado e se este polígono será preenchido ou terá uma cor diferente. A função deve ser utilizada para formar uma figura que combine desenhos sucessivos deste polígono.	Criação e uso de funções simples. Uso de estrutura condicional.
Q4	Implementação de um programa em Python que tenha uma função responsável pelo desenho de um polígono regular, semelhante à solicitada no programa da questão Q3, e outra função responsável pelo desenho de uma figura que combina o desenho de diversos polígonos. As funções devem ser utilizadas para desenhar a figura apresentada.	Criação e uso de funções.

de uma função com um parâmetro e sua respectiva utilização, 88,57% dos estudantes demonstraram proficiência. Para a questão 3, 60,0% dos estudantes demonstraram proficiência. Para a questão 4, que solicitava a implementação de funções chamando outras funções e construções lógicas um pouco mais complexas, apenas 41% dos estudantes apresentaram proficiência. Os estudantes não apresentam dificuldades com os comandos da biblioteca Turtle, nem com o uso de loops não aninhados e criação de funções com apenas um parâmetro.

Os estudantes não apresentam dificuldades em escrever uma função e utilizá-la, mas ao serem solicitados para escrever mais de uma função e utilizá-las de maneira combinada, eles aparentam não entender bem a modularização oferecida pelas funções. O uso de estrutura condicional também parece ser um ponto de entrave para parte dos estudantes, considerando que a maioria dos estudantes que não alcançaram proficiência na questão 3 apresentaram dificuldade com uso do *if-else*. As estruturas de seleção e expressões lógicas não apareceram como entraves durante a Unidade I, mas ao revermos estes conceitos na Linguagem Python os estudantes apresentaram certa confusão com a sintaxe, principalmente das expressões lógicas. Como as questões pouco exigiam a criação e uso de variáveis, este conceito não pôde ser bem avaliado.

No questionário III, aplicado ao final desta unidade, solicitamos aos estudantes que destacassem a facilidade que tiveram para aprender determinados conceitos e os resultados podem ser vistos no gráfico da Figura 5.2. Semelhante ao gráfico apresentado para a Unidade I, utilizamos uma escala de Likert com cinco pontos de

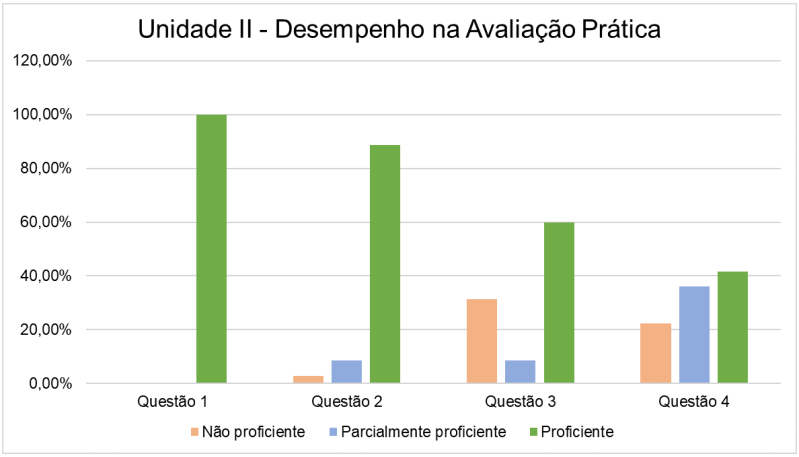

Figura 5.3: Desempenho dos estudantes na Avaliação Prática da Unidade II.

muito difícil a muito fácil. O uso dos comandos de movimento da biblioteca *Turtle*, uso de estrutura de repetição e uso de funções foram os quesitos mais bem avaliados pelos estudantes. O uso de estruturas condicionais, operadores lógicos e variáveis foram apontados como regular ou difícil de aprender pela maioria dos estudantes. As dificuldades em utilizar estrutura condicional e operadores lógicos ficaram explícitas no resultado da questão 3 da avaliação prática. Quando ao uso de variáveis, este conceito foi pouco explorado ao longo da Unidade II.

A partir das observações das aulas teóricas e práticas, identificamos os temas com dúvidas mais frequentes. Analisamos as dificuldades conceituais que os estudantes tiveram para aprender os conceitos teóricos na Unidade II, onde foram explorados os comandos do *Turtle*, noções de variáveis, estruturas condicionais, estruturas de repetição, funções e parâmetros. Os estudantes apresentaram, de maneira predominante, dificuldades com uso de funções, estruturas condicionais, variáveis e sintaxe da linguagem. O uso de funções gerou dúvidas durante as primeiras aulas em que este conceito foi introduzido, mas a medida em que os estudantes praticavam durante as aulas práticas, as dúvidas diminuíram significativamente. Apesar de haver evidências nas observações e no resultado da avaliação prática das dificuldades dos estudantes em utilizar funções, a grande maioria dos estudantes considerou este conceito fácil de aprender. Já as dúvidas relacionadas à estrutura condicional foram recorrentes ao longo de toda a Unidade, tanto no Estudo de Caso Piloto, quanto no Estudo de Caso Final. Durante as aulas práticas no Estudo de Caso Final, foram comuns momentos em que os estudantes travavam a resolução dos exercícios envolvendo *ifs*: "Amiguinho e ruiva tem dúvida com o *elif*. Cacheada tbm. Todos tem dúvida sobre isso." (EF-UII-AP07).

Em menor grau, os estudantes apresentaram dificuldades com estruturas de repetição e o fluxo de execução de um programa. Principalmente no Estudo de Caso Piloto, houve dúvidas sobre o uso do laço *for*, principalmente relacionadas a condição de parada: "Eles [os estudantes] não entendem o como funciona o incremento. Nota-

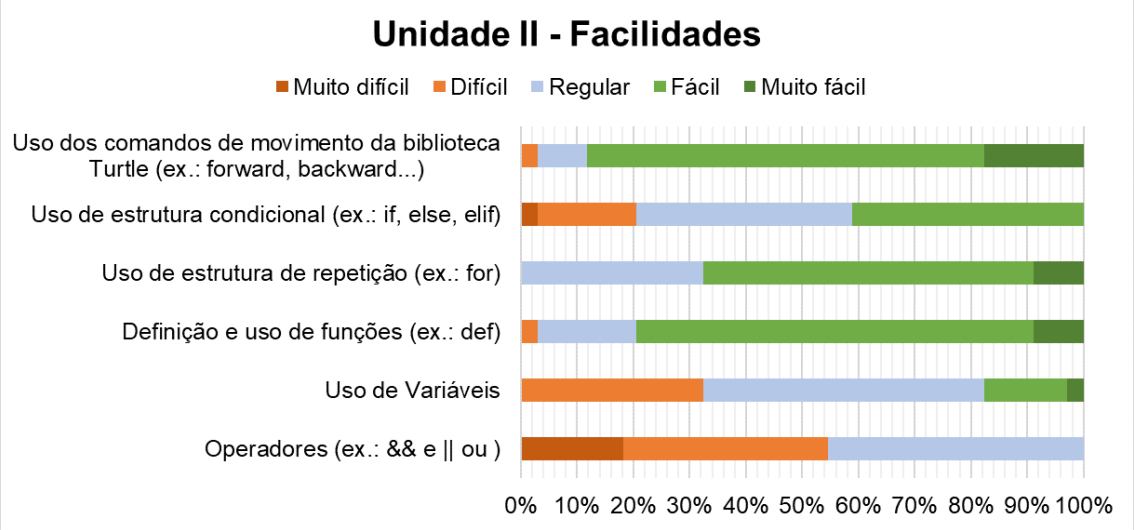

Figura 5.4: Opinião dos estudantes sobre a facilidade em aprender os conceitos de programação na Unidade II.

se isso pois o professor em alguns momentos fez algumas perguntas acerca disso, e eles não respondiam." (EP-UII-AT07). Percebemos que os estudantes apresentam alguma dificuldade em entender *loops* com condições assim como na Unidade I, onde a maioria das dúvidas sobre utilização de *loops* se relacionavam com o uso do comando repita até, que precisa de um critério de parada. Ao final da Unidade II nenhum estudante considerou estrutura de repetição como um conceito difícil de aprender.

Acreditamos que, assim como na Unidade I, os resultados da avaliação prática e das respostas ao questionário confirmam boa parte das dificuldades observadas durantes as aulas. Os estudantes apresentaram poucas dúvidas em relação aos comandos da biblioteca *Turtle* e, quando estas surgiam, estava mais relacionada à sintaxe da linguagem e à escrita das palavras em inglês do que com o entendimento das funções desta biblioteca.

5.3 Unidade III

Na unidade III procuramos reforçar os conceitos já aprendidos e, adicionalmente, conceitos de vetores, matrizes e construções mais complexas como laços aninhados. Durante as primeiras aulas buscamos introduzir o ambiente JES trabalhando exemplos de propósito geral, como jogos textuais e cálculo de áreas de figuras geométricas, e as demais aulas abordam tópicos sobre a criação e manipulação de imagens, com exemplos de implementação de filtros de imagem.

Como esta unidade está em um contexto mais complexo, a avaliação de aprendizagem se deu por meio de um projeto de implementação de um editor de imagens, dividido

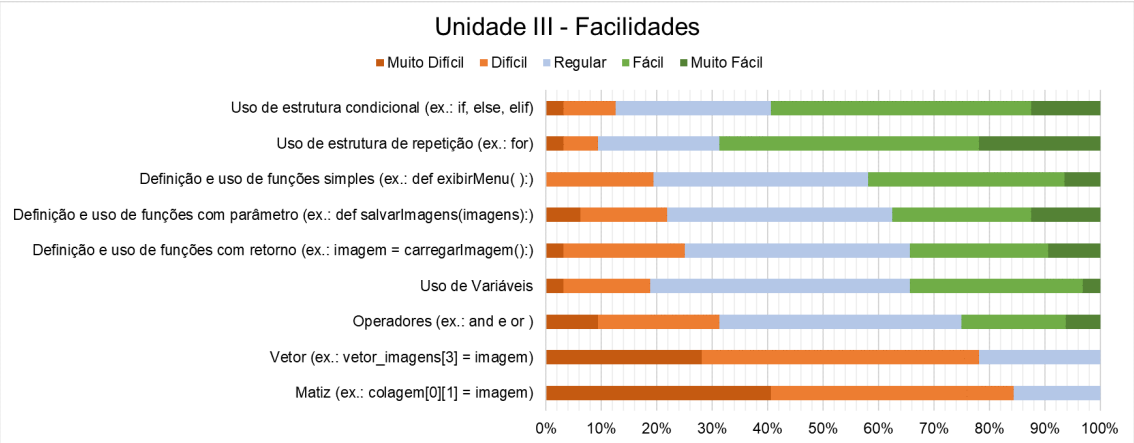

Figura 5.5: Opinião dos estudantes sobre a facilidade em aprender os conceitos de programação na Unidade III.

em três partes, com alguns dos filtros implementados em sala, mas com exigências específicas para garantir que o estudante explore, por conta própria, os conceitos ensinados. Como a atividade avaliativa também contou com desenvolvimento extra-classe, decidimos não incluir os resultados deste projeto em nossas análises devido à dificuldade em analisar as construções de código em cada projeto individualmente e garantir a autoria dos mesmos. Assim, para avaliar a aprendizagem dos estudantes durante a Unidade III, consideramos os resultados das observações das aulas teóricas e práticas, a opinião dos estudantes através do questionário IV e respostas da entrevista final da disciplina. Os dados qualitativos são provenientes dos dois estudos de caso realizados, enquanto os dados quantitativos são provenientes apenas do Estudo de Caso Final.

A Figura 5.5 ilustra a opinião dos estudantes quanto à facilidade para aprender os conceitos de programação durante a Unidade III. O uso de estruturas condicionais e de repetição são os únicos conceitos avaliados como fácil ou muito fácil por mais de 50% dos estudantes. Já os conceitos de Vetor e Matriz foram avaliados como difícil ou muito difícil por mais de 75% dos estudantes. Esses resultados, junto a opinião de alguns estudantes durante a última entrevista ao serem perguntados sobre a Unidade III, evidenciam que os estudantes tem a percepção de que a Unidade III foi difícil: “Essa terceira unidade mesmo foi, em relação a mim, a pior. Tanto em aprender, tipo, tipo, para estudar mais. E foi mais cansativa.”(EP-S20); “Acho que não teve facilidade não. Acho que o primeiro projeto foi bem complicadinho porque a gente teve que criar todas as definições. Eu acho que o segundo teve coisas mais legais de fazer. Mas dificuldade tudo pra mim foi dificuldade.” (EF-S32).

A partir das observações realizadas durante as aulas desta unidade, identificamos as dúvidas mais recorrentes apresentadas pelos estudantes. Os exemplos das três primeiras aulas utilizavam variáveis, estruturas de seleção e repetição além e funções. O único conceito novo apresentado nas aulas introdutórias foi de funções com retorno.

Nesse período, as participações observadas eram todas relacionadas a dúvidas e não sobre sugestões de implementação, como ocorria nas unidades anteriores. Os estudantes apresentaram muitas dificuldades com o uso de funções com retorno e com a utilização das funções pré-definidas do JES dedicadas à manipulação de imagens. No projeto editor de imagens, cada filtro a ser aplicado deveria estar implementado em uma função diferente, retornando um objeto de imagem com o filtro aplicado. Um dos requisitos do projeto, era a criação de uma função que combinasse três tipos e efeitos em uma mesma imagem. Apesar do suporte oferecido durante as aulas práticas, apenas 13 estudantes, conseguiram completar esta função corretamente.

Os estudantes também apresentaram muitas dúvidas relacionadas com a entrada e saída de dados. Vale ressaltar que foram utilizados tanto o console do JES quanto as funções de entrada e saída disponíveis no ambiente. Nas observações feitas durante o Estudo de Caso Final os estudantes se mostraram confusos: "Uma garota perguntou qual a diferença de *print* para *show information*? (...) Outra garota perguntou se o *return* já não exibia o resultado na tela..." (EF-UIII-AT02).

A partir das aulas que trataram de efeitos que manipulam a estrutura de uma imagem, como colagem de duas imagens, os estudantes aparentavam muitas dúvidas, relacionadas ao entendimento da manipulação de vetores e matrizes. Esses conceitos, além de ser apontados como os mais difíceis de aprender pelos estudantes, foram os que mais geraram dúvidas: "A garota do canto esquerdo não entende a diferença entre referenciar um vetor e referenciar um elemento do vetor (...) As pessoas tinham dúvidas sobre como fazer os efeitos, mas especificamente sobre como fazer a varredura da imagem." (EF-UIII-AP05); "O estudante vizinho confundiu um vetor com objetos de imagens com uma imagem. Os estudantes estão com dúvidas." (EP-UIII-AT03). As dúvidas quanto a manipulação foram recorrentes e não cessaram nas ultimas aulas da disciplina.

O conceito de variáveis foi mais explorado nas primeiras aulas da unidade. Não surgiram tantas dúvidas como em vetores, matrizes e funções com retorno. Apesar da grande maioria dos estudantes conseguiram finalizar os exercícios das aulas práticas que envolviam variáveis, parece haver equívocos quanto ao conceito. A resposta de um estudante sobre as dificuldades encontradas na Unidade III ilustra bem a falta de entendimento deste conceito: "Como acho que, como manipula as variáveis, como... é isso, como manipula. O jeito de você pensar para dar aquele resultado... como você vai fazer, tipo assim, a equação? não... para calcular, como você usava para dar o resultado. Achei difícil."(E-S07). As dúvidas relacionadas a variáveis demonstravam que a Unidade I, com o Scratch não amenizou as dificuldades com uso de variáveis.

5.4 Discussão

A partir dos resultados expressos nas seções anteriores, percebemos que, embora a abordagem utilizada não amenize todas as dificuldades dos estudantes, alguns conceitos tem a aprendizagem potencializada pela visualização instantânea do programa em execução, proporcionada pelo Scratch, pelo Tutle e pelo contexto de manipulação de imagens.

Constatamos que o Scratch potencializa o entendimento de lógica de programação e execução de programas, além de conceitos básicos como estruturas de seleção e repetição. Essa constatação se assemelha aos resultados de Bittencourt et al. (2015), que sugerem que os blocos Sensores, Variáveis e Operadores são menos praticados no Scratch, enquanto Controle, Aparência e Movimento são mais comumente utilizados. Embora Scratch não seja suficiente para ensino de uma linguagem de programação em nível de graduação, consideramos que a adaptação de cursos CS1 envolve uma reavaliação de objetivos de aprendizagem (Forte and Guzdial, 2005). Neste contexto, a escolha de Scratch para a primeira etapa da disciplina teve o intuito de promover uma introdução mais branda aos conceitos iniciais de programação. Conforme identificado nos dados coletados, a ferramenta exerceu de fato este papel.

O entendimento de conceitos básicos de programação, facilitado através de Scratch, foi potencializado quando retomados com a linguagem Python. Através das observações das primeiras aulas teóricas da Unidade II, percebemos que o Scratch cria uma espécie "noção visual"sobre a execução de comandos como for e *if*, e isso é facilmente transferido para o contexto de Python com *Turtle* empregado. Muitos estudantes entrevistados conseguiram associar conceitos que aprenderam em Scratch, com a linguagem Python e a biblioteca *Turtle*: "Foi bom por que ele deu uma noção ampla, deu uma noção do que é parâmetro, de variáveis, deu uma noção de. . . deu uma noção bem legal sobre o que a gente tá utilizando agora no Python." (EP-S02); "Eu acho que o Scratch ajudou mais porque você consegue entender a lógica que a programação é construída. Tipo, as definições *for, else...* todas essas definições, o que é uma função, uma variável. A gente conseguiu criar uma base muito boa no Scratch, que é mais didático. É mais interativo, pelo menos eu achei, aí isso foi uma base muito boa na hora de você ir executar no python.- (EF-S32).

O uso de *loops* e estrutura condicionais, deixa de ser um entrave para a maioria dos estudantes logo no início da Unidade II. Estes são conceitos apontados na literatura como difíceis de se aprender. É sabido que os estudantes possuem dificuldade em entender o processo de operação dos *loops* (Kaczmarczyk et al., 2010), e embora existam evidências de melhoras com a prática, esses conceitos ainda são desafiadores para os estudantes Cherenkova et al. (2014).

Como já era esperado, os estudantes consideram a linguagem Python mais difícil que o Scratch, principalmente na unidade final. Apesar da Unidade I trazer um aporte ao entendimento da linguagem Python, a medida em que os exercícios de programação se tornam mais complexos o desempenho dos estudantes cai. O uso

de funções não parece ser um entrava na Unidade II, mas na Unidade III onde este conceito é revisitado com uma abordagem mais complexa, os estudantes apresentam dúvidas. Apesar disso, podemos afirmar que em nossa abordagem, as pessoas adquirem alguma proficiência precoce em utilizar funções, e no modo tradicional esses conceitos acabam sendo apresentados no final e podem ficar desprezados. Durante as entrevistas, alguns estudantes foram específicos e consideram a Unidade III mais difícil devido ao uso do JES complicado. Essa percepção pode ser acentuada pelo fato do help do JES, que tem justamente a função de amenizar as dificuldades dos estudantes através de explicações detalhadas sobre manipulação de imagens estar em inglês: "especialmente por utilizar a língua dominante, o inglês. Isso foi um... pelo menos pra mim foi um pouco... acrescentou a dificuldade."(EP-S02). Além disso, durante a Unidade III vetores e matrizes, que não foram apresentadas nas unidades anteriores, foram explorados exaustivamente, uma vez que este é o principal conceito na hora de representar imagens.

Capítulo 6

Motivação

Avaliamos o nível de motivação dos estudantes através das quatro categorias do modelo ARCS. Consideramos as observações realizadas em sala de aula nos dois estudos de caso realizados e os questionários aplicados durante o Estudo de Caso Final. Em relação aos dados qualitativos, foi realizada a codificação conjunta dos arquivos dos estudos de caso Piloto e Final. Para os questionários IMMS, analisados apenas para o Estudo de Caso Final, onde as questões usam escala de Likert.Consideramos discordância as ocorrências das opções Discordo Parcialmente e Discordo Totalmente. De maneira análoga, consideramos concordância as ocorrências de Concordo Parcialmente e Concordo Totalmente. Para o questionário CIS, aplicado apenas com os participantes do Estudo de Caso Final, utilizamos uma escala de Likert diferente, com as opções *não é verdade*, *um pouco verdadeiro*, *moderadamente verdadeiro*, *principalmente verdadeiro* e *muito verdadeiro*. Para ambos os questionários foram atribuídos valores de 1 a 5 a estas opções e computadas as médias. Os aspectos metodológicos estão descritos detalhadamente no Capítulo 3.

A Figura 6.1 mostra o gráfico para média da opinião dos estudantes que participaram do Estudo de Caso Final sobre aspectos gerais de cada Unidade. Os estudantes atribuíram, para cada categoria, um escore de 1 a 5. Quanto mais próximo de 1, a opinião do estudante se aproxima mais da característica negativa e, quanto mais próximo de 5, a opinião do estudante se aproxima mais da característica positiva. As duas primeiras categorias apresentam avaliações mais negativas do que as demais categorias. A Unidade I apresenta, em todas as categorias, avaliações mais positivas, enquanto a Unidade III apresenta avaliações menos positivas que as unidades anteriores. A disciplina foi mais cansativa do que leve na Unidade III. Para as demais categorias, onde a disciplina obteve média de avaliação acima de 3,5 em todas as unidades, podemos afirmar que a disciplina teve boa didática, foi proveitosa, foi organizada e facilitou o aprendizado.

De maneira geral, podemos afirmar que, em ambos os estudos de caso, os maiores níveis de motivação mensurados foram na Unidade I e os menores, na Unidade III. Em ambos os estudos foi aplicada a mesma abordagem, com diferenças pontuais em

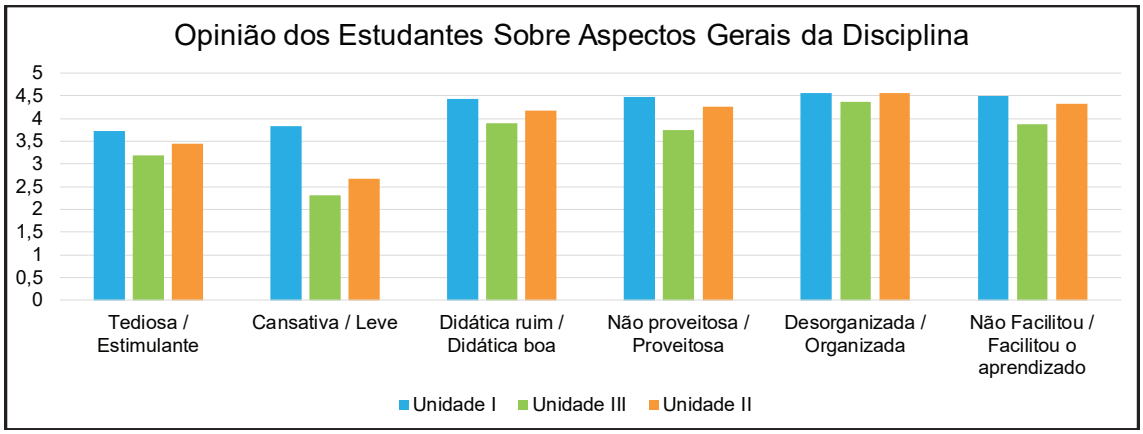

Figura 6.1: Opinião dos estudantes participantes sobre aspectos gerais da disciplina.

algumas aulas e no professor que ministrou a disciplina. Durante a Unidade I, os estudantes apresentaram-se bastante entusiasmados com a disciplina, principalmente nas aulas destinadas a construção de jogos. Na avaliação prática, os estudantes puderam escolher o projeto base, e receberam o resultado da avaliação assim que finalizaram a prova, junto com o *feedback* em relação aos erros cometidos.

Durante a Unidade II, os estudantes permaneceram entusiasmados com a disciplina. Na primeira aula foi feita a introdução a linguagem Python e comandos da biblioteca *Turtle*, reescrevendo os mesmos programas feitos com a caneta do Scratch. As aulas subsequentes trouxeram exemplos de desenhos cada vez mais complexos, onde, a cada exemplo eram introduzidos novos conceitos. A medida em que os exemplos ficavam mais complexos, a turma apresentava menos engajamento. Nas aulas práticas, observamos que os estudantes tinham mais demanda pela ajuda dos monitores do que na Unidade I e que nem todos conseguiam finalizar todas as atividades previstas no tempo da aula. A avaliação, assim como na Unidade I, teve correção e entrega dos resultados imediata.

Durante a Unidade III, foi apresentado o ambiente JES e foram revistos todos os conceitos de programação empregados nas unidades anteriores através da construção de exemplos de propósito geral. Nesse período, os estudantes apresentaram comportamento mais displicente em relação a disciplina, participando pouco das aulas. As participações observadas eram, em sua maioria, relacionadas a dúvidas e não sobre sugestões de implementação, como ocorria nas unidades anteriores. A participação dos estudantes aumentou quando a manipulação de imagens começou a ser tratada. As aulas práticas foram destinadas à elaboração de programas para a edição de imagens, inclusive o projeto final de um editor de imagens, que foi a atividade avaliativa desta unidade. Muitos estudantes demonstraram satisfação em relação a isto: "Que legal! Vamos virar o - *Zuckerberg*, fazer posts de *Orkut!*" (EF-UIII-AP03). De maneira geral, os estudantes apresentaram muitas dúvidas sobre a implementação dos filtros e dificuldades em gerenciar o tamanho dos programas, que tinham tamanho considerável comparado aos programas implementados na Unidade II. Apesar do

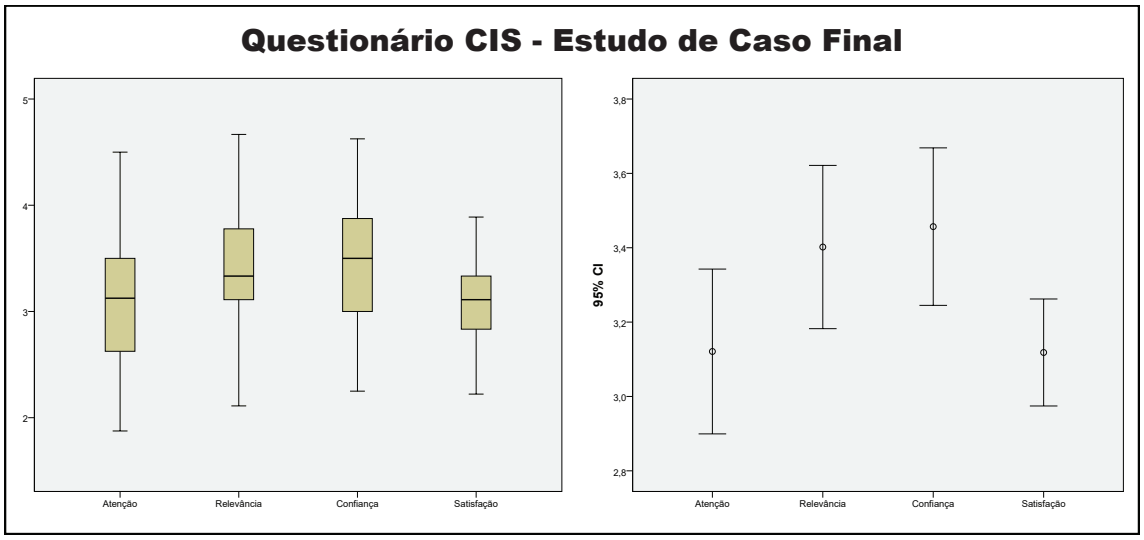

Figura 6.2: À esquerda: Box-plot para as médias do questionário CIS nas quatro categorias do ARCS. À direita: diagrama de barra de erros para as médias das quatro categorias.

suporte oferecido durante as aulas práticas, nem todos os estudantes conseguiram implementar todos os requisitos do editor de imagens. As dificuldades com o inglês ficaram evidentes, pois os estudantes apresentavam dificuldades em interpretar o *help* e a descrição dos erros dos programas no ambiente JES.

A Figura 6.2 mostra o box-plot e o diagrama de barra de erros das médias obtidas para as quatro categorias do ARCS a partir do questionário CIS. Para testar a significância entre a diferença nas médias aplicamos o teste-t pareado em cada um dos pares possíveis das categorias. A Tabela 6.1 traz o valor das médias e o respectivo desvio padrão para cada categoria e a Tabela 6.2 traz o resultado para o teste t, incluindo o cálculo do tamanho do efeito. A partir dos resultados, considerando apenas os resultados onde o nível de significância p foi inferior a 0,05, podemos afirmar que os níveis de relevância são maiores que os níveis de atenção e de satisfação. Com base nestes mesmos parâmetros também podemos afirmar que os níveis de confiança são maiores do que os de atenção e satisfação. No entanto, o tamanho do efeito calculado é pequeno para todos os casos testados. Fazendo uma correspondência com a escala de Likert utilizada, consideramos que as médias obtidas para as categorias de atenção e satisfação estão mais próximas da opção Moderadamente verdadeiro enquanto as médias para as categorias relevância e confiança estão entre as opções *moderadamente verdadeiro* e *principalmente verdadeiro*.

Os resultados do questionário CIS trazem um indicativo sobre a motivação dos estudantes ao longo de estudo de caso final, mas para fazer uma análise mais aprofundada dos níveis de motivação é preciso analisar cada uma das categorias em cada uma das unidades. Nas seções a seguir, apresentamos resultados específicos para cada categoria, seguidos das respectivas análises.

Tabela 6.1: Valores das médias para o Questionário CIS aplicado durante o Estudo de Caso Final.

Categoria	Média	Desvio Padrão
Atenção	3,1209	0,6042
Relevância	3,4018	0,5985
Confiança	3,4567	0,5774
Satisfação	3,1182	0,3922

Tabela 6.2: Resultado Teste t para médias do questionário CIS, no Estudo de Caso Final.

Par	t	df	Sig. (2 extremidades)	Tamanho do efeito r
Atenção - Relevância	-3,386	30	0,002	0,216
Atenção - Confiança	-2,188	30	0,037	0,072
Atenção - Satisfação	0,027	30	0,979	0,092
Relevância - Confiança	-,394	30	0,696	0,055
Relevância - Satisfação	2,787	30	0,009	0,216
Confiança - Satisfação	2,853	30	0,008	0,072

6.1 Atenção

A categoria atenção tem entre os objetivos principais capturar o interesse dos alunos e estimular a curiosidade para aprender. Nesse contexto, os construtos dos questionários CIS e IMMS dedicados a esta categoria buscam mensurar o quão estimulante e interessante foi a experiência de aprendizagem oferecida.

Considerando os dados quantitativos do IMMS para esta categoria, apesar de termos uma quantidade pequena de amostras, testamos os dados para a hipótese de distribuição normal no SPSS. As percentagens foram, para o teste *Shapiro-Wilk*, na Unidade I, $D(26) = 0{,}971$, $p > 0{,}05$, para os escores da Unidade II, $D(26) = 0{,}974$, $p > 0{,}05$, e para os escores da Unidade III, $D(26) = 0{,}930$, $p > 0{,}05$. Como a hipótese nula não é rejeitada aqui, consideramos as distribuições possivelmente normais.

A Figura 6.3 traz o box-plot da média das variáveis, à esquerda, e o gráfico de barras de erros para as médias de atenção, à direita. Os valores das médias e desvio padrão podem ser vistos na Tabela 6.3. Percebemos que a distribuição das avaliações nas unidades I e II é mais simétrica que na Unidade III, ou seja, as opiniões mais negativas estão mais balanceadas com as opiniões positivas. Na Unidade III, os escores superiores estão espalhados em um intervalo maior do que os escores inferiores, o que demonstra que há mais uniformidade entre as avaliações menos positivas. A média cai ao longo das unidades. Para testar se essa mudança é realmente significativa, realizamos o Teste t para amostras emparelhadas e os resultados podem ser vistos na Tabela 6.4. A único resultado com significância foi quando comparamos as Unidade I e III. Desta maneira, podemos afirmar que a média de atenção na Unidade I é ($M = 3{,}724359$, $DP = 0{,}5867847$) é maior que na Unidade III ($M = 3{,}456838$, $DP =$

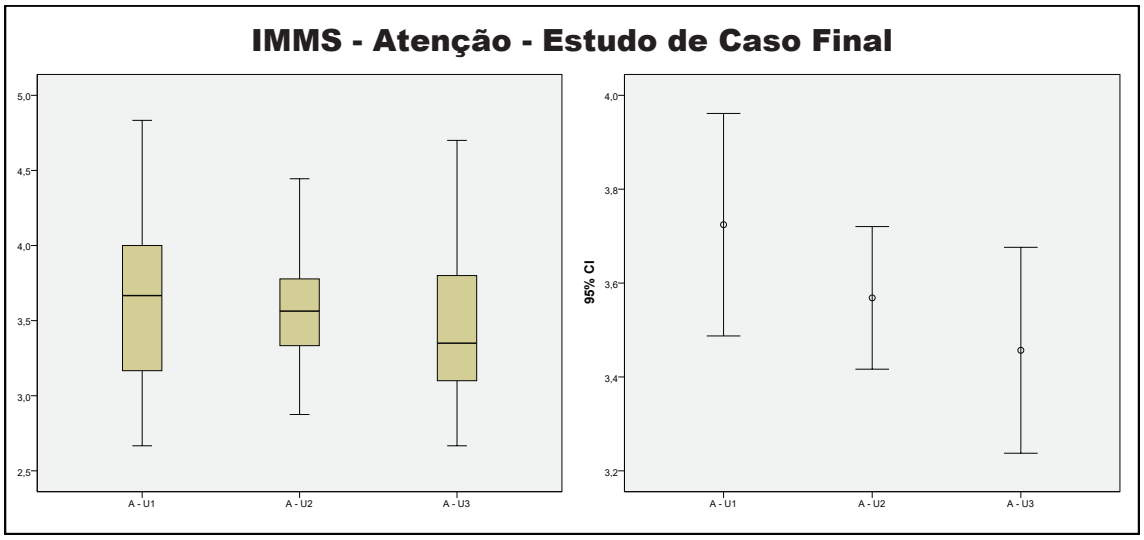

Figura 6.3: À esquerda: Box-plot para as médias de Atenção durante as três unidades. À direita: diagrama de barra de erros para as médias.

Tabela 6.3: Médias e desvio padrão para a categoria atenção.

Unidade	Média	Desvio Padrão
A - U1	3,7243	0,5867
A – U2	3,5684	0,3761
A3 – U3	3,4568	0,5430

0,5430352), onde t(28)= 2,182, p<0,05. Não podemos afirmar que houve mudança significativa entre as Unidades I e II ou Unidades II e III. Estabelecendo correspondência com a escala de Likert utilizada, podemos afirmar que as médias de atenção estão acima da neutralidade, mas apenas a Unidade I se aproxima do escore que caracteriza concordância parcial.

A Figura 6.4 traz, lado a lado, os gráficos de atenção para as três unidades evidenciando cada construto. Os construtos da unidade I obtiveram avaliação mais acentuada em termos de concordância e discordância, enquanto as demais unidades tiveram uma quantidade razoável de estudantes que se declararam neutros (razoável o suficiente para fazer com que não houvesse uma quantidade de aceitação ou rejeição tão expressivas). Os construtos A1, A2, A3, A4 e A5 buscam mensurar a

Tabela 6.4: Resultado Teste t para a categoria de Atenção durante as três Unidades do Estudo de Caso Final.

Par	t	df	Sig. (2 extremidades)	Tamanho do efeito r
AU1 - AU2	1,185	28	0,246	0,218
AU1 - AU3	2,182	28	0,038	0,381
AU2 - AU3	0,970	27	0,340	0,183

opinião dos estudantes sobre o quanto os aspectos/ferramentas/materiais/tema ajudam ou minam a atenção dos estudantes. Em todas as unidades, mais de 65% dos estudantes afirmaram que havia algo de diferente no início da unidade que chamou a atenção e mais de 55% afirmaram que o tema/ferramenta da unidade é atraente. Na unidade I, mais de 91,2% dos estudantes discordam que o Scratch seja abstrato a ponto de prejudicar a atenção, já nas unidades II e III o percentual de discordância do construto A3 cai para 61,8 e 42%, respectivamente. Em relação ao design das ferramentas/materiais serem atraentes (A4), nas unidades I e III, a concordância ultrapassa os 58% ao passo em que, na unidade II, o resultado é de apenas 35,3%, sendo inconclusivo. Em relação a maneira como a informação é organizada (A5), as unidades I e II obtiveram níveis de concordância favoráveis (67,6% e 52,9%), enquanto a unidade III obteve resultado inconclusivo.

No modelo ARCS, cada uma das quatro categorias também tem subcategorias baseadas nas principais variáveis motivacionais nelas incluídas. Estas subcategorias são úteis para identificar perfis motivacionais dos alunos e auxiliam na criação de táticas motivacionais adequadas aos problemas identificados. A categoria de atenção possui três subcategorias: Excitação perceptiva, *"Inquiry Arousal"* e Variabilidade. Os cinco primeiros construtos do IMMS estão relacionados a mais de uma dessas subcategorias, e principalmente com a excitação perceptiva.

Os construtos A6 e A8, estão relacionados com a subcategoria *"Inquiry Arousal"*, que busca estimular um comportamento de indagação por parte dos estudantes. Em todas as unidades, as avaliações para os construtos A6 e A8 foram bastante satisfatórias, de maneira que a maioria dos estudantes consideram que, em todas as unidades, aprenderam coisas surpreendentes ou inesperadas e que havia coisas que estimulavam a curiosidade. Acreditamos que, indiretamente, o uso do contexto de jogos e mídias ajudou a formulação de problemas que estimulam este comportamento "indagativo" e curioso dos estudantes como, por exemplo, na Unidade I, o contexto de jogos permitiu a apresentação de situações-problema comuns aos desenvolvedores de jogos.

Os construtos A7 e A9 estão ligados à subcategoria de Variabilidade, que busca manter a atenção dos estudantes através da variação de estímulos. Em todas as unidades, os estudantes consideraram que a variedade de exercícios, ilustrações, etc., ajudou a manter a atenção (61,8%, 66,7%, e 53,1%). Na Unidade I, 47,1% dos estudantes concordam que a repetição sobre os assuntos os deixa entediados, enquanto que, para as demais unidades, este índice é menor do que 30%.

A Figura 6.5 traz o gráfico para os construtos do questionário CIS referentes a categoria Atenção. As opções Principalmente verdadeiro e Muito verdadeiro não estão acentuadas. O construto CA2 é o onde mais de 50% dos estudantes marcaram Não é verdade ou Um pouco verdadeiro. Ao contrário do IMMS, os construtos do CIS levam em consideração a dinâmica de uma turma presencial, ponderando o papel do professor no processo de aprendizagem. O construto CA6 é onde mais de 50% dos estudantes marcaram a opção Principalmente verdadeiro ou Muito verdadeiro.

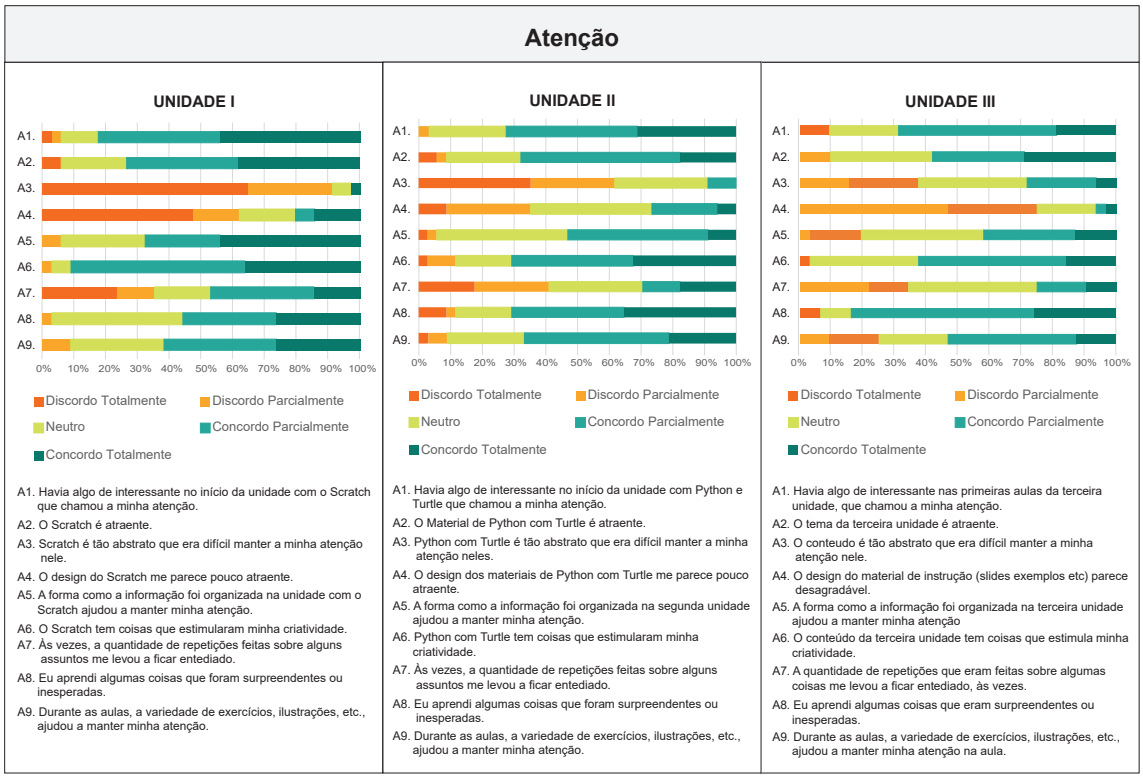

Figura 6.4: Gráfico com os construtos da categoria Atenção do IMMS.

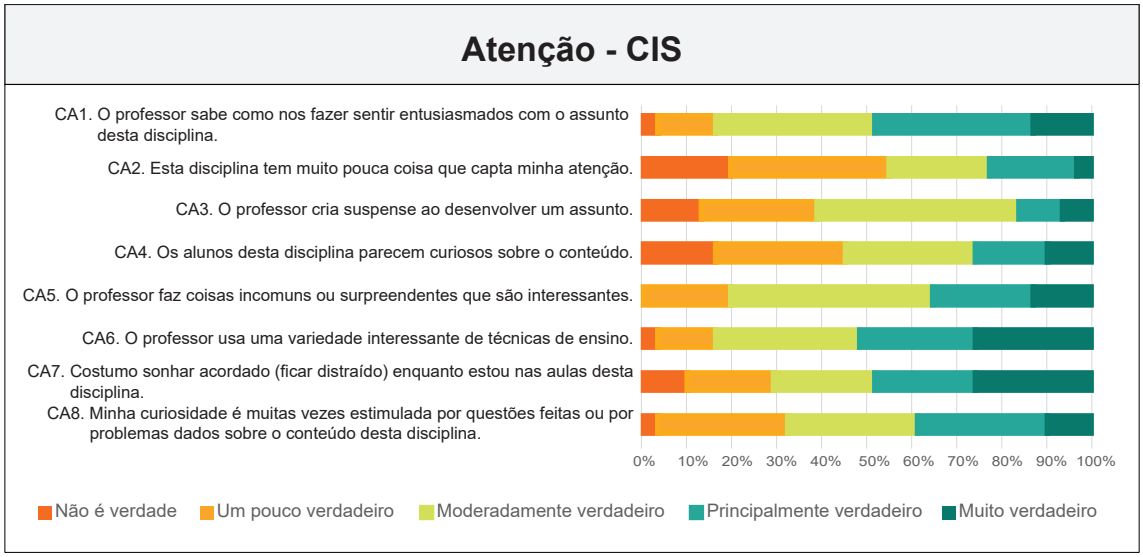

Figura 6.5: Gráfico com os construtos da categoria Atenção do CIS.

Podemos dizer que, para muitos estudantes, o professor fez uso de uma variedade de técnicas de ensino.

A grande maioria desses dados encontra embasamento quando confrontados com os dados qualitativos. A partir das observações, notamos que a organização da aula dá liberdade ao professor para alternar entre explicações utilizando quadro, projeção de slides explicativos ou códigos. Esse tipo de comportamento torna a aula mais dinâmica e, consequentemente, mais atrativa. Se o professor conseguir engajar os estudantes logo no início da aula, trazendo uma situação-problema ou exemplo de aplicação daquilo que será ensinado, é mais provável que o estudante preste atenção no restante da aula. Na última entrevista, perguntamos aos estudantes o que os levam a prestar atenção em uma aula. Parte das respostas atribuíram ao tema da aula ser atraente, mas a maioria das respostas apontaram situações que dependem do engajamento do professor: "Acho que com piadas para a gente distrair um pouco é... uma pausa durante a aula... essas coisas que levam a gente a descontrair."(EF-S05); "esse negócio de ficar fazendo a turma participar e ficar perguntando a turma, isso faz a turma ficar ligada."(EF-S04); "Quando fazer... tipo, mostrava coisa nova, tipo, que você nem imaginava o que era. Por exemplo, como o espelho da imagem, que era jogar os *pixels* pro outro lado. Isso chamou bastante atenção e a última aula do chroma key, eu achei... chamou bastante atenção, interessante. Coisas novas chamam atenção..." (EF-S26).

Em relação à Unidade I, a partir das observações das aulas notamos que os estudantes mantiveram um comportamento participativo durante toda a unidade. Constatamos que a utilização do Scratch num contexto de criação de jogos a partir de pequenos desafios estimula a criatividade, chama atenção dos estudantes e abre espaço para momentos de descontração, o que ajuda a captar a atenção do estudante. Grande parte dos entrevistados, nos dois estudos de caso conduzidos, avaliaram positivamente a utilização de jogos e consideram que o Scratch facilita o aprendizado e torna a aula mais divertida: "O Scratch foi interessante, [por]que desperta a criatividade, curiosidade também, porque a gente vai fazendo os desafios e vai incentivando a querer incrementar, colocar mais coisas." (EP-S35). "Eu achei bem interessante e interativo e se a aula fosse tipo algoritmo, essas coisas, ia ser bem cansativo pra gente. E como mexeu com jogos a gente conseguiu aprender e se divertir ao mesmo tempo. O que é um conceito meio contraditório na universidade né, que geralmente é bem maçante mas eu achei bem divertido e eu acho que eu aprendi. Acho que, agora, no final, o objetivo foi alcançado, que foi eu entender o que era aquilo ali." (EF-S32).

Na Unidade II, durante as primeiras aulas, que promoveram uma transição dos conceitos em Scratch para Python colocando lado a lado programas de mesmo objetivo implementados em ambas as ferramentas, observamos uma elevada participação dos estudantes. Mas a frequência das participações diminui à medida em que as aulas adquirem uma carga maior de conteúdo. Também notamos um maior número de estudantes que não conseguiram acompanhar o ritmo das aulas nesta unidade.

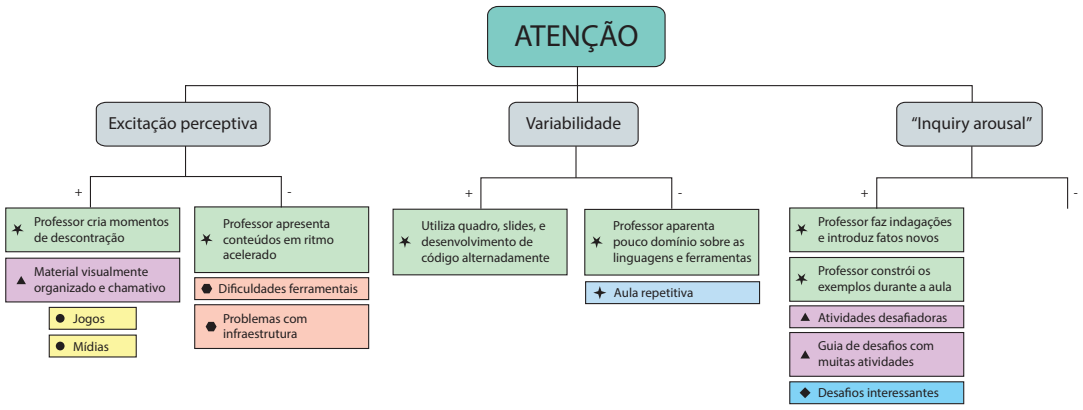

Figura 6.6: Fatores que influenciam na Atenção dos estudantes.

Na unidade III, as observações realizadas durante as aulas mostram que as aulas com curiosidades sobre codificação de imagens e exemplos mais significativos, como *chromakey*, conseguiram despertar a atenção dos estudantes, mesmo tratando de conceitos mais complexos. Quando as aulas teóricas não usaram esta abordagem, adquirindo um estilo mais expositivo e apresentando muito código, boa parte da turma encontrou-se frequentemente dispersa. Muitas vezes, aproveitaram o tempo da aula para resolver exercícios de outras disciplinas.

A Figura 6.6, traz um diagrama onde buscamos relacionar os aspectos da nossa abordagem com as subcategorias de Atenção. Por se tratar de um curso presencial, as atitudes do Professor acabam tendo um peso muito grande na hora de captar e manter a Atenção dos estudantes. Quando o professor apresenta o conteúdo em ritmo acelerado, ou quando surgem problemas de infraestrutura ou com ferramentas, o ambiente tende a ficar mais tedioso e a atenção dos estudantes é perdida. Quando o material da aula é visualmente chamativo, a atenção dos estudantes é retomada. Na Unidade III, muitas imagens utilizadas para aplicar os filtros continham frases engraçadas, e isso mexe com a percepção dos estudantes por algum tempo. Um ponto interessante que percebemos é que, para manter a atenção dos estudantes com melhor desempenho e estimular a sua capacidade de indagação, é preciso oferecer atividades desafiadoras. Nesse sentido, adicionar questões extras ao guia de desafios pode ser uma estratégia útil para acomodar os estudantes que possuem um ritmo rápido da hora de fazer as atividades.

6.2 Relevância

A categoria relevância tem entre os objetivos principais buscar atender às necessidades e objetivos pessoais do aluno para realizar uma atitude positiva. Nesse

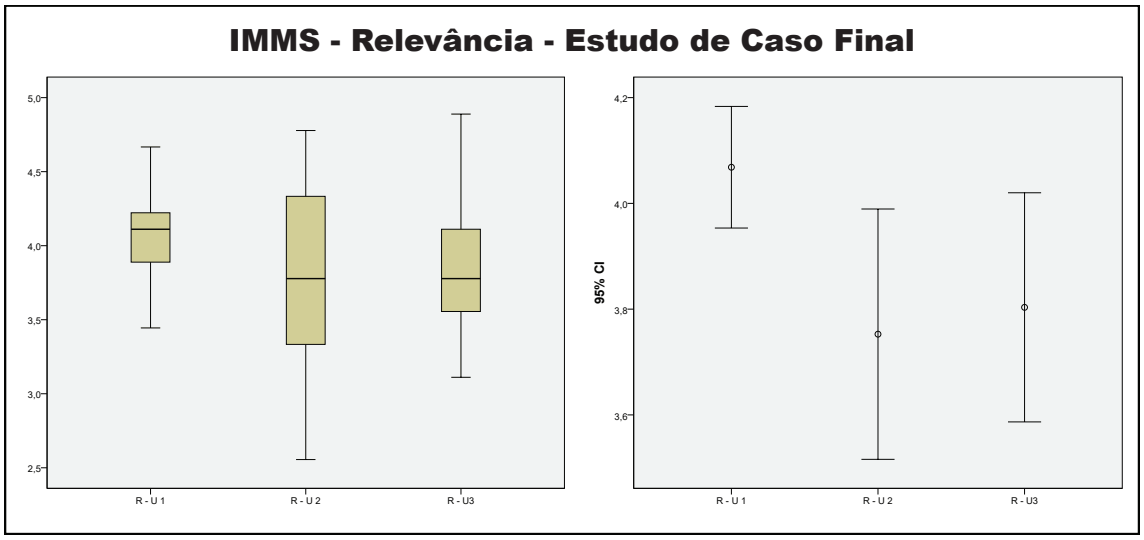

Figura 6.7: À esquerda: Box-plot para as médias de Relevância do questionário IMMS. À direita: diagrama de barra de erros.

contexto, os construtos dos questionários CIS e IMMS dedicados a esta categoria buscam mensurar o quão significativa foi a experiência de aprendizagem oferecida.

Considerando os dados quantitativos do IMMS para esta categoria, apesar de termos uma quantidade pequena de amostras, testamos os dados para a hipótese de distribuição normal no SPSS. As percentagens foram, para o teste Shapiro-Wilk, na Unidade I, D(25) = 0,972, p > 0,05, para os escores da Unidade II, D(25) =0,975, p>0,05, e para os escores da Unidade III, D(25)= 0,944, p > 0,05. Como a hipótese nula não é rejeitada aqui, consideramos as distribuições possivelmente normais.

A Figura 6.7 traz o box-plot das variáveis, à esquerda, e o gráfico de barras de erros para as médias de Relevância, à direita. Os valores das médias e desvio-padrão podem ser vistos na Tabela 6.5. Percebemos que a dispersão dos dados na Unidade II é maior do que nas demais unidades e, na Unidade I, os dados estão menos dispersos. Em relação ao valor da média, fica claro que ela cai entre as unidades I e II e tem um aumento na Unidade III em relação à Unidade II. Realizamos o Teste t para amostras emparelhadas a fim de testar se a mudança nas médias é de fato significativa, e os resultados podem ser vistos na Tabela 6.6. Desta maneira, podemos afirmar que a média de relevância na Unidade I (M = 4,075556, DP = 0,2881758) é maior que na Unidade II (M = 3,773889, DP = 0,1175658), onde t(27)= 3,157, p<0,05. Também podemos afirmar que a média de relevância na Unidade I (M = 4,075556, DP = 0,2881758) é maior que na Unidade III (M = 3,857778, DP = 0,4686905), onde t(25)= 2,526, p<0,05. Estabelecendo correspondência com a escala de Likert, podemos afirmar que a média da Unidade I aponta concordância parcial, e que as demais unidades também se aproximam deste grau de avaliação.

A categoria relevância também se subdivide em outras três categorias: Orientação do objetivo, Correspondência de motivos e Familiaridade. A orientação de objeti-

Tabela 6.5: Médias para a categoria Relevância.

Unidade	Média	Desvio Padrão
RU 1	4,0755	0,2881
RU 2	3,7738	0,1175
RU 3	3,8577	0,4686905

Tabela 6.6: Resultados do teste t para categoria Relevância.

Par	t	df	Sig. (2 extremidades)	Tamanho do efeito r
RU1 - RU2	3,157	27	0,004	0,5192
RU1 - RU3	2,526	25	0,018	0,4509
RU2 - RU3	-,357	28	0,724	0,0673

vos é o sentido mais comum de relevância por estar relacionado com a importância do conteúdo aprendido para atingir um objetivo no presente ou futuro como, por exemplo, emprego. Correspondência de motivos está relacionado à capacidade de oferecer ao estudante um ambiente de ensino que respeite e acomode seu comportamento e interesses e, para isso, busca entender as estruturas de motivação pessoal dos estudantes. A Subcategoria de Familiaridade busca amarrar as instruções às experiências dos estudantes.

A Figura 6.8 traz, lado a lado, os gráficos de relevância para as três unidades evidenciando cada construto. Os construtos da unidade I obtiveram avaliação mais acentuada em termos de concordância e discordância, e são visualmente diferentes das unidades II e III.É perceptível que houve uma diminuição na sensação de relevância após a Unidade I. Os construtos R2, R3, R4, R5 e R9 estão ligados à subcategoria de Orientação do Objetivo. E os construtos R1, R6, R7 e R8 estão ligados a subcategoria Familiaridade.

A Figura 6.9 traz a o gráfico obtido para a categoria Relevância através do questionário CIS. Assim como para os resultados obtidos na categoria Atenção, poucos construtos foram marcados como *principalmente verdadeiro* ou *muito verdadeiro* por mais de 50% dos estudantes. Como consequência, o valor da média foi menor do que o observado para os resultados obtidos com o IMMS em cada uma das unidades. Assim, parte dos estudantes considera verdadeiro que o professor faz com que o assunto da disciplina pareça importante (construto CR2). Os estudantes reconhecem que o conteúdo da disciplina está relacionado com coisas que eles já conheciam (construto CR3), e procuram estabelecer e alcançar altos padrões de excelência. Além disso, eles acreditam que serão beneficiados pela disciplina (construto CR8). Esses construtos também se relacionam diretamente com as subcategorias de relevância.

Buscamos embasamento nos dados qualitativos a fim de entender os resultados obtidos através dos questionários do ARCS. Em relação à orientação do objetivo, percebemos que, apesar de obter uma avaliação positiva, no IMSS, o Scratch não passa a sensação de utilidade prática na vida dos estudantes, o que diminui sua relevância

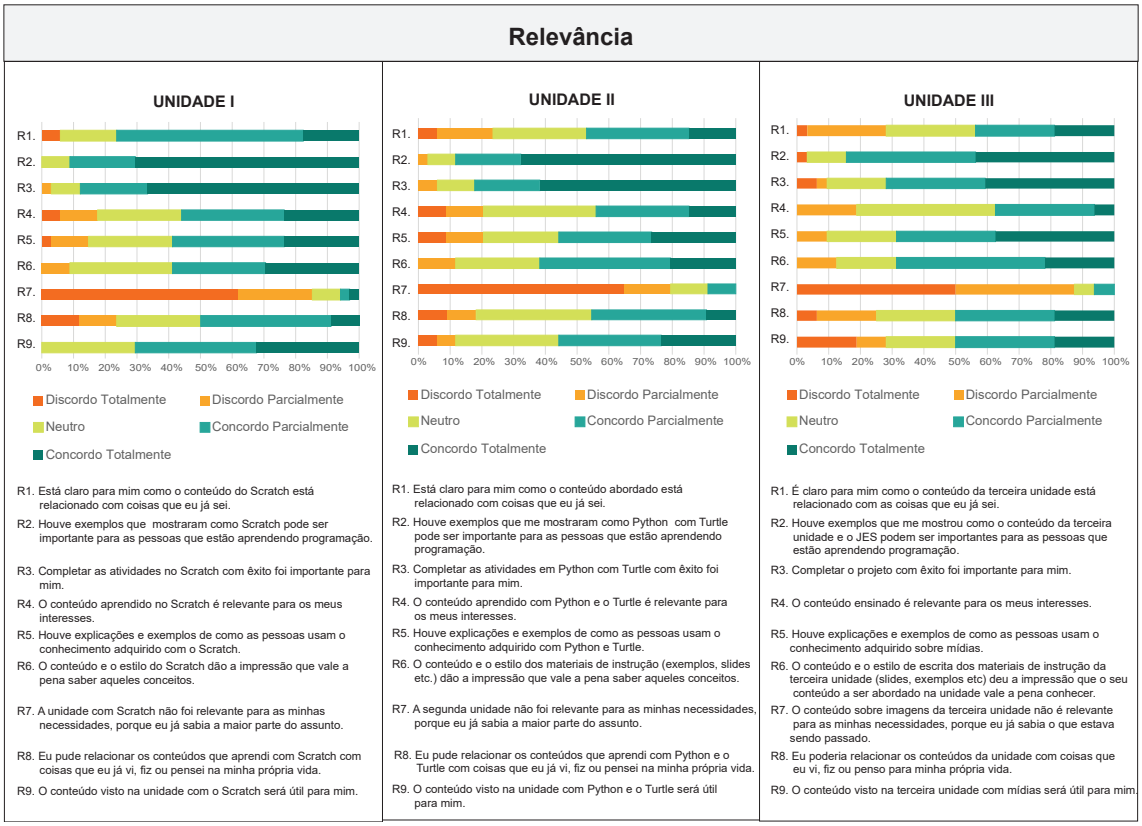

Figura 6.8: Resultados para a categoria Relevância ao longo das três unidades, considerando o questionário IMMS.

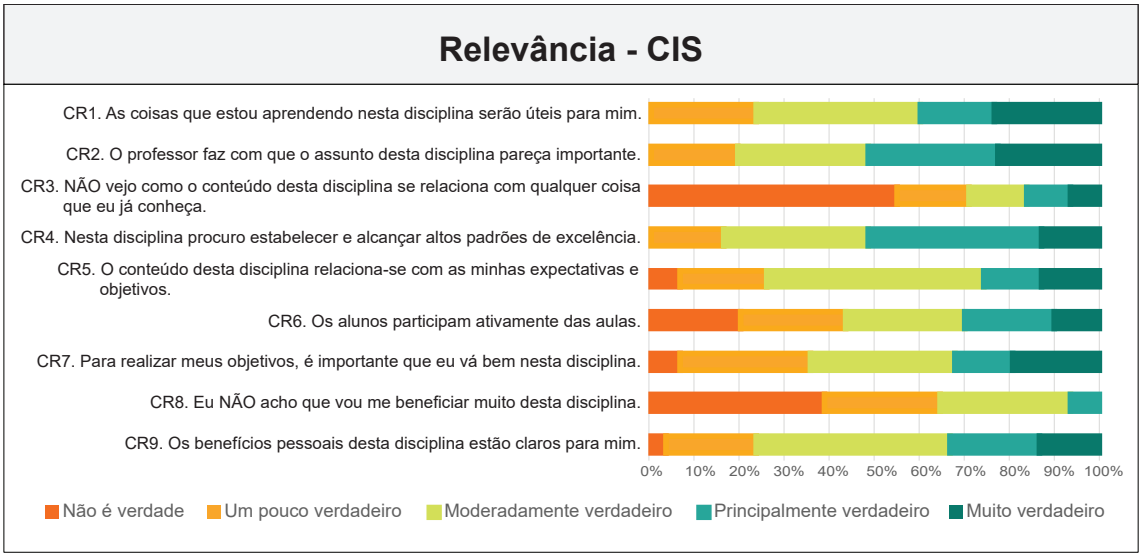

Figura 6.9: Resultados para a categoria Relevância no questionário CIS.

percebida em relação a futuras oportunidades de carreira. Durante as entrevistas, alguns estudantes, ao serem questionados sobre o que esperavam da disciplina, apontaram que esperavam aprender a utilizar softwares úteis para a engenharia civil: "Eu achei que seria coisas relacionadas a engenharia civil, por exemplo, como mexer em *softwares* de engenharia, como o *Autocad*, alguns desses." (EP-S18) ; "Sinceramente eu não acho que seja eficiente pro meu curso. Pode ser eficiente para um curso de computação por exemplo, mas pro meu curso não é algo que agregue muito." (EF-S15). Através das observações das aulas, percebemos que o professor tem um papel importante na hora de comunicar o valor daquilo que está sendo aprendido.

Tendo em vista que o procedimento para adaptar um curso CS1 para uma determinada audiência envolve o reconhecimento de interesses (Forte and Guzdial, 2005), sabíamos previamente que as ferramentas escolhidas para as três unidades não despertariam a sensação de relevância num sentido de utilidade prática, e por isso buscamos atender aos outros critérios de relevância. Segundo Keller (1987), a sensação de relevância pode surgir do jeito como algo é ensinado, satisfazendo certas necessidades pessoais do estudante como, por exemplo, dar a pessoas com "necessidade de realização"a oportunidade de estabelecer metas moderadamente desafiadoras e assumir a responsabilidade pessoal de alcançá-las. Neste sentido, o Scratch obteve avaliações positivas, tanto quantitativa quanto qualitativamente.

Anda tratando da orientação de objetivos, acreditamos que, embora os níveis de relevância para a Unidade I sejam maiores segundo os resultados do IMSS, a linguagem Python trabalhada em dois contextos diferentes, *Turtle* e Mídias, ajudou a aumentar a sensação de relevância da disciplina. Na Unidade II, para 64,71% dos estudantes Python parecia ser mais útil que o Scratch e na Unidade III esta razão sobe para 84,38%. Alguns estudantes entrevistados preferem Python pois esta linguagem aumenta a sensação de utilidade do que está sendo ensinado, uma vez que é uma linguagem usada no mundo profissional: "Quando a gente fala mais de mídia, aproximou mais, de certa forma, do lado profissional. Eu achei mais profissional a parte de mídia, a parte de Python do que o Scratch."(EF-S03). Os estudantes que preferem Python em relação a Scratch também apontam as possibilidades dessa linguagem como justificativa: "É. . . não. Tipo, a parte do Python mesmo, eu fiquei viajando porque sei fazer coisas incríveis. É a parte do Python, pra mim, foi a melhor. A segunda unidade, em relação à prática, foi a melhor." (EP-S20).

Em relação à correspondência de motivos, por se tratar de uma disciplina presencial e que não é autodirigida, esse tipo de variável não é mensurado diretamente no IMMS. No entanto, a partir das observações, percebemos alguns aspectos que contribuem para criar um ambiente de aprendizagem harmonioso para a maioria dos estudantes. Um desses fatores é o fato das atividades serem realizadas individualmente. A grande maioria dos entrevistados apontaram isto como um fator importante para o seu processo de aprendizagem: "Então, individual, eu acho que a gente aprende mais do que em grupo. Normalmente, em grupo muitos ficam isolados e deixam só pra uma pessoa. Só uma pessoa sabe e passa para as outras, então, ser individual é muito importante." (EF-S03). "Eu acho muito bom, por que cada pessoa tem o

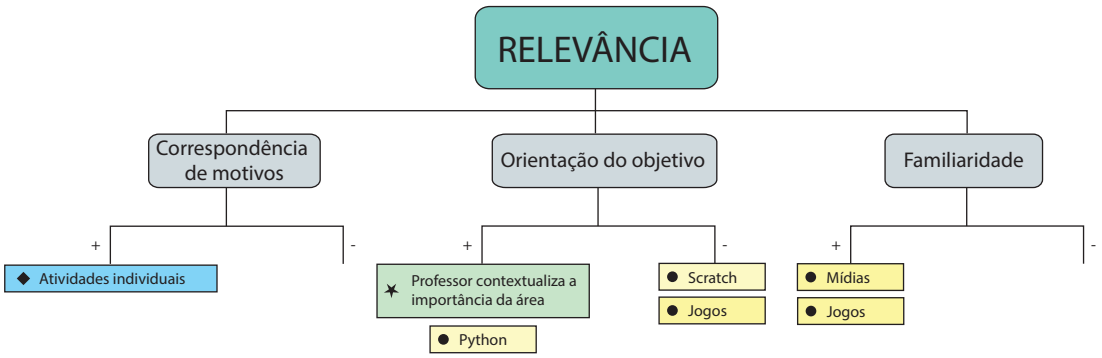

Figura 6.10: Fatores que influenciam a Relevância dos estudantes.

seu ritmo. Aí não tem como colocar, tipo, duas pessoas juntas, aí uma vai ser mais rápida que a outra, aí a outra pode ficar meio se batendo, tipo assim, ah! O que foi que você fez. Eu achei individual legal." (EP-S09).

A familiaridade foi explorada através dos contextos de jogos e mídias, uma vez que aproximam a programação de coisas que os estudantes lidam no dia a dia. Muitos estudantes entrevistados citaram a proximidade com o contexto como algo positivo: "Mas a primeira unidade, o que eu gostei mais foi a parte de fazer os joguinhos. Teve aquele da bolinha, eu achei bem interessante porque, tipo, eu não fazia a menor ideia, eu não tinha noção nenhuma de como é que fazia um jogo. E a partir do Scratch eu vi mais ou menos como é que eles faziam, tipo, eu tive uma noção bem básica de como é que acontece o joguinho que a gente joga." (EP-S14). O mesmo ocorre com a ideia de mídias.Neste contexto em específico, notamos que as mulheres tiveram a participação incrementada com uso de mídias. Além disso, explorar Python no contexto de mídias mostrou a versatilidade da linguagem e contribuiu também para o sentido de orientação do objetivo. Alguns estudantes viram utilidade explícita das funções pré-definidas do JES para manipular imagens: "eu gostei mais do Jython[Python] porque eu já estou até pensando em utilizar o Jython[Python] no caso para manipular imagens. Eu acho que, principalmente aquele *chromakey*. Oxe, aquilo ali tem muita utilidade. E eu já estou pensando em alguma forma de como utilizar." (EP-S01); "Achei bastante útil, porque a gente pode usar no dia a dia. É difícil mas, se a gente presta atenção e entende o básico, a gente vai dar conta e consegue entender." (EF-S12).

A Figura 6.10, aponta como algumas características de nossa abordagem influenciam na sensação percebida de relevância dos estudantes, considerando as três subcategorias anteriormente descritas. Diferente das demais categorias, não encontramos muitos fatores que influenciam na relevância percebida.

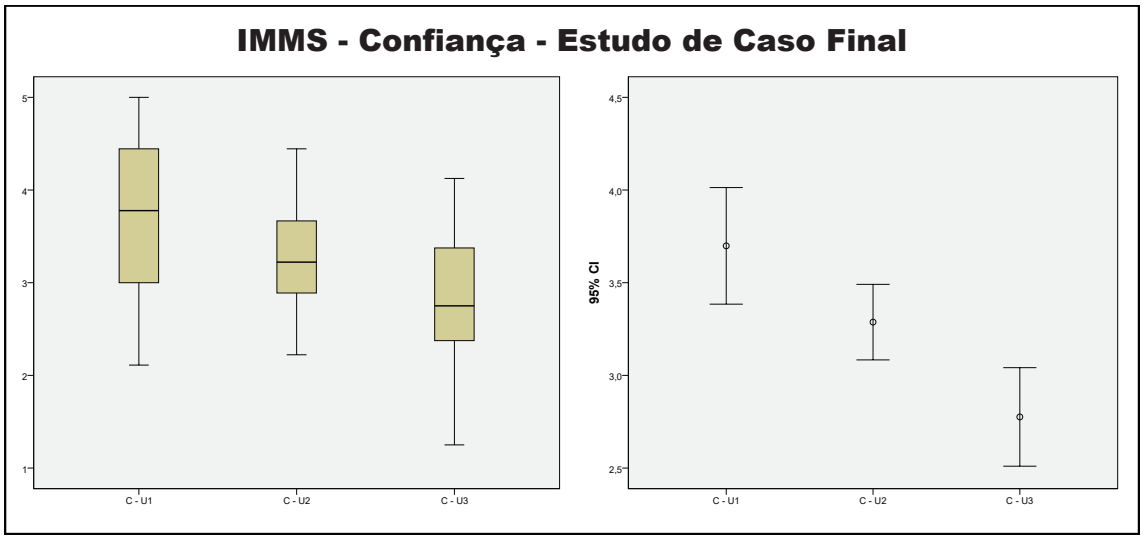

Figura 6.11: À esquerda: Box-plot para as médias de Confiança durante as três unidades. À direita: diagrama de barra de erros para as médias.

Tabela 6.7: Médias para a categoria Confiança no questionário IMMS.

Unidade	Média	Desvio Padrão
CU 1	3,6987	0,8267
CU 2	3,2873	0,5357
CU 3	2,7758	0,6986

6.3 Confiança

A categoria Confiança tem entre os objetivos principais ajudar os estudantes a acreditarem que que eles terão sucesso e que eles têm controle sobre seu sucesso. Neste contexto, os construtos dos questionários CIS e IMMS dedicados a esta categoria buscam mensurar o quanto a intervenção ajudou os estudantes a alcançarem o sucesso e o quanto permitiu-lhes controlar o seu sucesso.

Considerando os dados quantitativos do IMMS para a categoria confiança, testamos os dados para a hipótese de distribuição normal no SPSS. As percentagens foram, para o teste *Kolmogorov-Smirnov*, na Unidade I, $D(29) = 0{,}139$, $p > 0{,}05$, para os escores da Unidade II, $D(29) = 0{,}118$, $p > 0{,}05$, e para os escores da Unidade III, $D(29) = 0{,}101$, $p > 0{,}05$. Como a hipótese nula não é rejeitada aqui, consideramos as distribuições possivelmente normais.

A Figura 6.11 traz o box-plot das variáveis, à esquerda, e o gráfico de barras de erros para as médias de confiança nas três unidades, à direita. Os valores das médias e desvio-padrão podem ser vistos na Tabela 6.7. Percebemos que a dispersão nos dados da segunda unidade é menor do que nas demais. O valor da média cai ao longo das unidades. Para testar se esta mudança é realmente significativa, realizamos o

Tabela 6.8: Resultados para o Teste t para as médias da categoria Confiança no questionário IMMS.

Par	**t**	**df**	**Sig. (2 extremidades)**	**Tamanho do efeito r**
CU1 - CU2	2,918	30	0,007	0,4701
CU1 - CU3	5,087	29	0,000	0,6866
CU2 - CU3	4,577	30	0,000	0,641229

Teste t para amostras emparelhadas e os resultados podem ser vistos na Tabela 6.8. Para todos os pares, os resultados foram significativos. Dessa maneira, podemos afirmar que a média de confiança na Unidade I é (M = 3,698755, DP = 0,8267690) é maior que na Unidade II (M = 3,287356, DP = 0,5357726), onde t(30)= 2,918, p<0,05, e é maior que a média na Unidade III (M = 2,775862, DP = 0,6986749), onde t(29)= 5,087, p<0,05. Também podemos afirmar que a média para confiança na Unidade II é maior que na unidade III, uma vez que, t(30)= 4,577, p< 0,05. Fazendo correspondência com a escala de Likert utilizada, podemos afirmar que as Unidades I e II estão entre a neutralidade e concordância parcial, enquanto a Unidade III está entre a discordância parcial e a neutralidade.

A categoria Confiança possui três subcategorias: Requisitos de Aprendizagem, Oportunidade de sucesso e Controle pessoal. A subcategoria Requisitos de aprendizagem está relacionada com a necessidade de fazer com que o estudante construa uma expectativa positiva de sucesso a partir do momento em que ele sabe o que se espera dele, ou seja, quais os critérios de sucesso no curso ofertado. A subcategoria Oportunidades de sucesso busca oferecer tarefas desafiadoras e significativas para os estudantes, além de oferecer suporte para tornar o sucesso nestas tarefas possível. A subcategoria Controle Pessoal tem por objetivo deixar claro para o estudante que seu sucesso é baseado em seus esforços e habilidades, e não mera obra do acaso. Para tanto, é necessário oferecer ao estudante a sensação de controle sobre o próprio processo de aprendizagem.

A Figura 6.12 exibe, lado a lado, os gráficos de Confiança para as três unidades, evidenciando cada construto. Considerando a escala utilizada e o fato de que a média para esta categoria na Unidade III está abaixo de 3,0, que corresponde ao neutro da escala, é possível afirmar que nesta unidade a nossa abordagem não conseguiu potencializar a confiança dos estudantes como nas duas unidades anteriores.

No IMSS, os construtos C3 e C8 se relacionam com a subcategoria Requisitos de aprendizagem. O construto C3 obteve um nível de concordância de pouco maior que 30% na Unidade III, indicando que, nesta unidade, a medida em que as aulas avançavam, parte dos estudantes continuaram a não entender o que eles deveriam aprender, ao contrário das unidades anteriores. Os construtos C1, C2, C5, C6, C7 e C9 estão mais relacionados a subcategoria de Oportunidades de sucesso. Para estes construtos, a Unidade III apresenta uma média de respostas desfavorável, reforçando a ideia de que, na última unidade, o conteúdo foi difícil de aprender e a organização da disciplina e os materiais oferecidos não foram suficientes para poten-

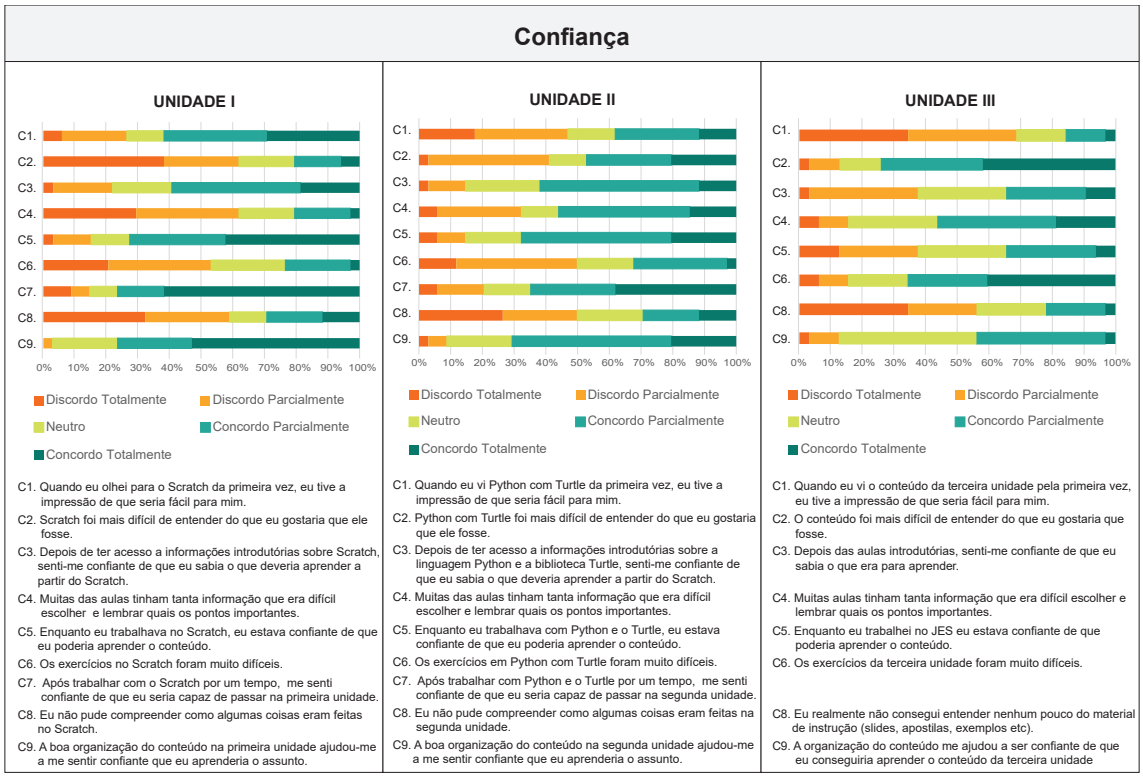

Figura 6.12: Resultados para a categoria Confiança ao longo das três unidades, considerando o questionário IMMS.

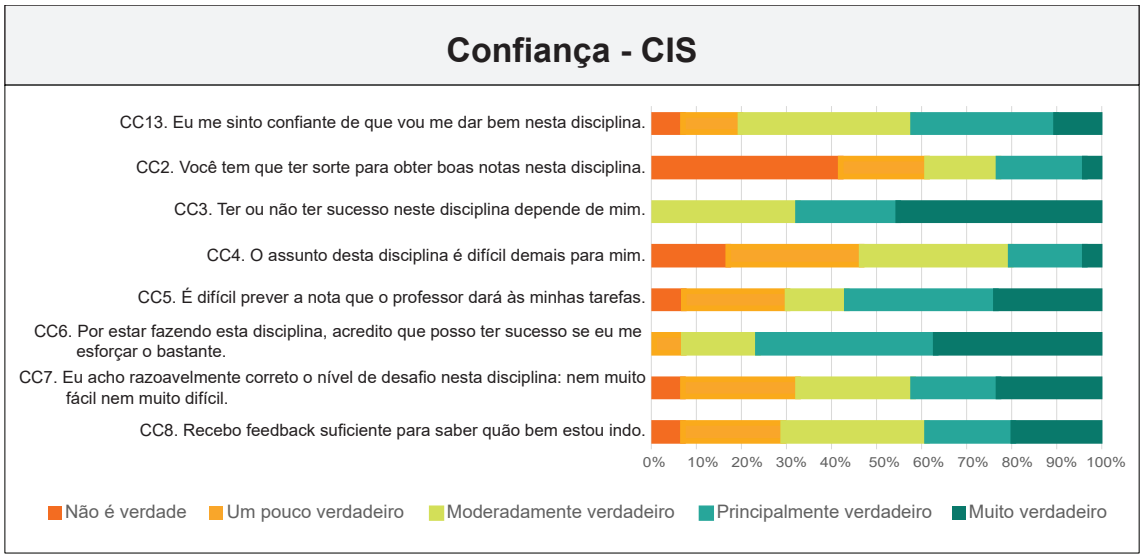

Figura 6.13: Resultados para a categoria Confiança no questionário CIS.

cializar a confiança. O construto C4 se relaciona diretamente com a subcategoria de Controle pessoal. Considerando este construto, para a maioria dos estudantes, as aulas das Unidades II e III tiveram uma quantidade de informações que atrapalhou o entendimento do conteúdo.

A Figura 6.13 apresenta os resultados para a categoria confiança obtidos partir do questionário CIS. O construto CC5 está relacionado à subcategoria de Requisitos de aprendizagem. O fato de a maioria dos estudantes marcarem como principalmente verdadeiro ou muito verdadeiro a afirmação de que é difícil prever a nota atribuída pelo professor indica que eles não têm total ciência dos requisitos de avaliação. Já os construtos CC2, CC3, e CC6 estão relacionados à subcategoria controle pessoal e refletem diretamente a teoria do valor-expectativa na qual se baseia o modelo ARCS. Os resultados positivos nestes construtos indicam que a abordagem empregada ajudou os estudantes a se perceberem como agentes ativos no processo de aprendizagem, responsáveis pelo próprio sucesso.

Em relação à subcategoria Requisitos de aprendizagem, percebemos durante as observações que o professor tem um papel fundamental para evidenciar o que é esperado dos estudantes. Quando o professor constrói os códigos exemplos durante a aula, os estudantes acompanham o processo de confecção do código, o que é útil para o entendimento do conteúdo mas também elucida como devem ser realizadas as atividades avaliativas. Em relação à categoria Oportunidades de sucesso, o fato de os estudantes não se sentirem sobrecarregados com o conteúdo e nem sentirem dificuldades para realizar as atividades durante a Unidade I demonstra que o Scratch torna o conteúdo inicial da programação mais acessível. Scratch traz confiança também para aprendizagem posterior, já que 82,35% dos estudantes concordam que, sem o Scratch, teriam demorado mais para entender a linguagem Python. Além disso, as aulas práticas foram projetadas para minimizar as dificuldades iniciais, utilizando um guia de desafios com dicas detalhadas de implementação. Essa percepção, reforçada pelas observações e entrevistas, indicam que a abordagem com Scratch proporcionou um scaffolding benéfico para a manutenção da confiança.

Durante as Unidades I e II, os estudantes tiveram como atividades a resolução de desafios que envolveram nível de dificuldade de resolução gradual. Ou seja, o primeiro desafio tinha um nível de dificuldade baixo e os desafios subsequentes tinham sempre um nível de dificuldade maior que o anterior. Em relação aos desafios no contexto de jogos, muitos estudantes elegeram como melhor jogo feito aquele que demandava maior desafio para sua implementação: “que teve mais desafios.”(EP-S02) ou “Pelo fato de ser desafios. . . desafiante e que você pode fazer o seu próprio jogo”(EP-S18).

Na Unidade III, muitos dos entrevistados consideraram a dificuldade para conseguir implementar o projeto: “Essa terceira unidade mesmo foi, em relação a mim, a pior. Tanto em aprender, tipo, tipo, para estudar mais. E foi mais cansativa.” (EP-S20). A abordagem de mídias não foi planejada para ser mais fácil que uma abordagem de ensino tradicional de programação, mas para ser mais motivadora

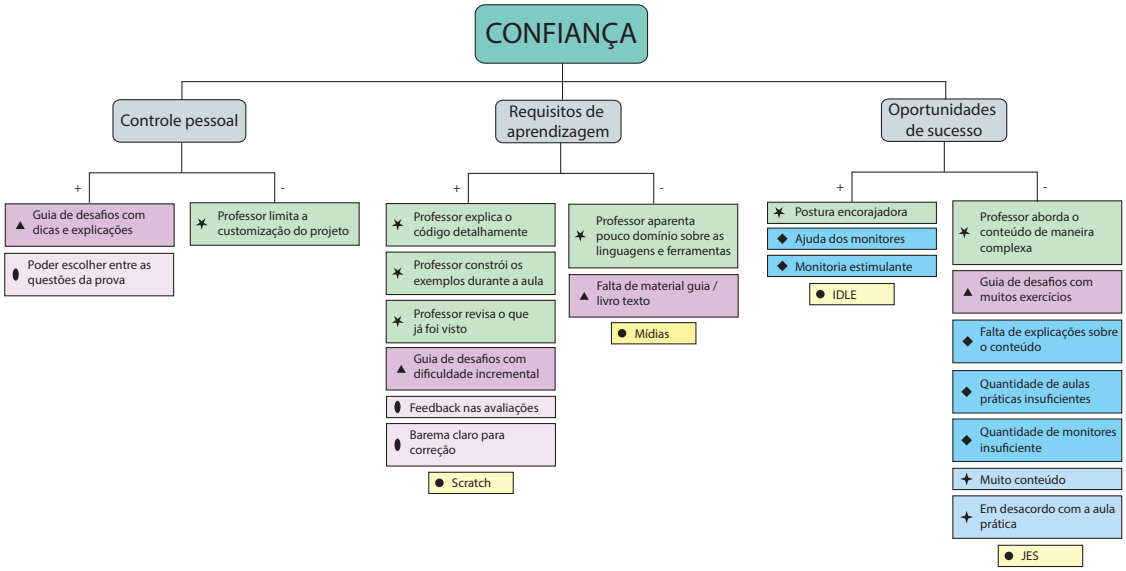

Figura 6.14: Fatores que influenciam a Confiança dos estudantes.

Forte and Guzdial (2004). Os estudantes que se mostraram bastante interessados em Python com mídias ressaltam seu interesse apesar das dificuldades "Eu achei interessante. Eu achei difícil, mas interessante. Quando você começa a entender, você acaba gostando (. . .) Essa terceira, eu estou achando boa, mas a melhor, eu achei com Scratch, por que foi mais fácil. Mas não quer dizer que a difícil seja ruim. Eu gostei da terceira e da primeira. (. . .) agora, eu comecei a entender o programa e tô muito interessado" (EP-S09); "Agora na terceira, com a manipulação de imagens,mesmo sendo difícil eu achei bastante legal." (EF-S07). Uma possível explicação para os baixos indicadores de atenção e confiança é o formato da aula teórica, que passou a contar com exemplos de código maiores, além de um estilo mais expositivo.

A dificuldade com a língua estrangeira foi um fator inesperado que demonstrou ter influência negativa na confiança do estudante. Na unidade II, o inglês foi considerado empecilho para 44,2% dos estudantes, enquanto que, na Unidade III, esta porcentagem foi de 37,50%. Embora esses índices não sejam maiores que 50%, simboliza uma parcela significativa de estudantes dentro do ambiente pesquisado. No estudo de caso piloto, esta dificuldade também foi evidenciada durante as entrevistas: "Tipo, o jeito que eu escrevo. Algumas funções mesmo. São funções gigantes. Alguma coisa lá, a maioria inglês. Essa parte pega muito. Inglês para quem não sabe já dificulta. E alguns detalhes, às vezes uma letra, aí você fica lá se matando duas horas com um erro que você... tipo... banal. (....) Mas, tipo, a parte do inglês mesmo é na hora do erro. Que o erro vem tudo em inglês. A parte da aba, não, por que antes acaba decorando alguma coisa. Eu não sabia inglês, mas sabia que aquele negócio fazia aquilo. Mesmo não sabendo seu significado. Agora quando dava o erro. Aí quem não sabe inglês, tipo, ele não vai consegui ver aquilo" (EP-S20).

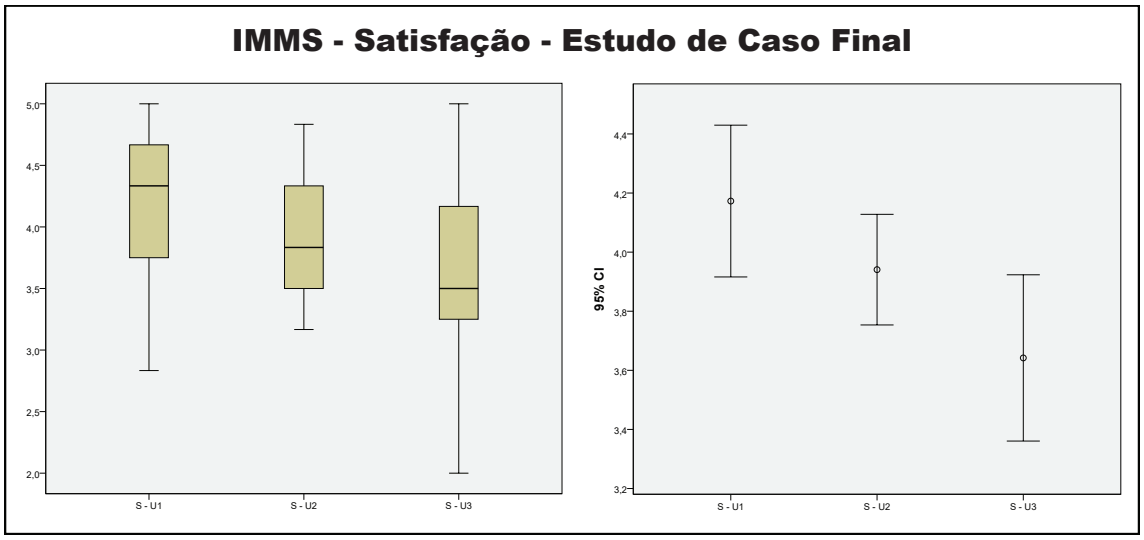

Figura 6.15: À esquerda: Box-plot para as médias de Satisfação do questionário IMMS. À direita: diagrama de barra de erros.

Em relação à subcategoria Controle pessoal, encontramos poucos fatores que influenciam explicitamente. De maneira geral, consideramos os fatores que dão a sensação de autonomia ao estudante.

A Figura 6.14, exibe a relação de alguns aspectos de nossa abordagem com as subcategorias da Confiança. A partir do que foi exposto, percebemos que minimizar as dificuldades dos estudantes é o principal fator que potencializa a confiança.

6.4 Satisfação

A categoria Satisfação tem entre os objetivos principais ajudar os estudantes a se sentirem bem com sua experiência e desejar continuar aprendendo. Nesse contexto, os construtos dos questionários CIS e IMMS dedicados a esta categoria buscam mensurar o quanto a intervenção ajudou os estudantes a se sentirem bem e realizados na experiência ofertada.

Considerando os dados quantitativos do IMMS para esta categoria, testamos os dados para a hipótese de distribuição normal no SPSS. As percentagens foram, para o teste Kolmogorov-Smirnov, na Unidade I, $D(27) = 0{,}153$, $p > 0{,}05$, para os escores da Unidade II, $D(27) = 0{,}167$, $p > 0{,}05$, e para os escores da Unidade III, $D(27) = 0{,}112$, $p > 0{,}05$. Como a hipótese nula não é rejeitada aqui, consideramos as distribuições possivelmente normais.

A Figura 6.15 traz o box-plot das variáveis, à esquerda, e o gráfico de barras de erros para as médias de confiança nas três unidades, à direita. Os valores das médias e desvio padrão podem ser vistos na Tabela 6.9. Percebemos que a dispersão nos dados

Tabela 6.9: Médias para a categoria Satisfação do questionário IMMS.

Unidade	Média	Desvio Padrão
SU1	4,1728	0,6491
SU2	3,9407	0,4731
SU3	3,6419	0,7111

Tabela 6.10: Resultado Teste t para as médias da categoria Satisfação do questionário IMMS.

Par	t	df	Sig. (2 extremidades)	Tamanho do efeito r
SU1 - SU2	2,328	29	0,027	0,3968
SU1 - SU3	3,845	28	0,001	0,5878
SU2 - SU3	2,178	28	0,038	0,380622

da Unidade III é maior do que nas demais, ou seja, na Unidade III, as opiniões não são uniformes como nas unidades anteriores. O valor da média cai ao longo das unidades. Para testar se esta mudança é realmente significativa, realizamos o Teste t para amostras emparelhadas e os resultados podem ser vistos na Tabela 6.10. Para todos os pares, os resultados foram significativos. Dessa maneira, podemos afirmar que a média de confiança na Unidade I (M = 4,172, DP = 0,6491) é maior que na Unidade II (M = 3,940, DP = 0,473), onde t(29)= 2,318, p<0,05, e é maior que a média na Unidade III (M = 3,641, DP = 0,711), onde t(28)=3,845, p<0,05. Também podemos afirmar que a média para confiança na Unidade II é maior que na unidade III, uma vez que, t(28)= 2,178, p< 0,05. Fazendo correspondência com a escala de Likert utilizada, a média da Unidade I está acima da concordância parcial e a média da Unidade II se aproxima da concordância parcial, enquanto a média da Unidade III está entre a neutralidade e a concordância parcial.

A categoria de Satisfação pode ser dividida entre três subcategorias: Consequências naturais, Consequências positivas e Equidade. Consequências Naturais é uma subcategoria relacionada à capacidade de oferecer condições aos estudantes de aplicar os conhecimentos recém-adquiridos. Consequências positivas envolve a utilização de recompensas extrínsecas, como maneira de demonstrar reconhecimento pelo avanço do estudante. Em nossa abordagem, não planejamos este tipo de incentivo. A Equidade diz respeito a dar ao estudante a sensação de que a sua nota é justa, principalmente quando comparada com a dos colegas.

A Figura 6.16 exibe, lado a lado, os resultados para cada construto de Satisfação nas 3 Unidades. É notória a diminuição nos níveis de concordância para os construtos quando comparamos as Unidades entre si. Os índices de concordância são mais acentuados na Unidade I do que nas demais. Os níveis de concordância para os construtos S1 e S5, acima dos 50% em todas as unidades, demonstram o quanto os estudantes sentem-se bem com suas realizações. O construto S4, que relaciona-se com a subcategoria equidade, obteve avaliação acima de 75% nas duas primeiras unidades, enquanto que, na unidade III o índice de concordância foi de 50%. O

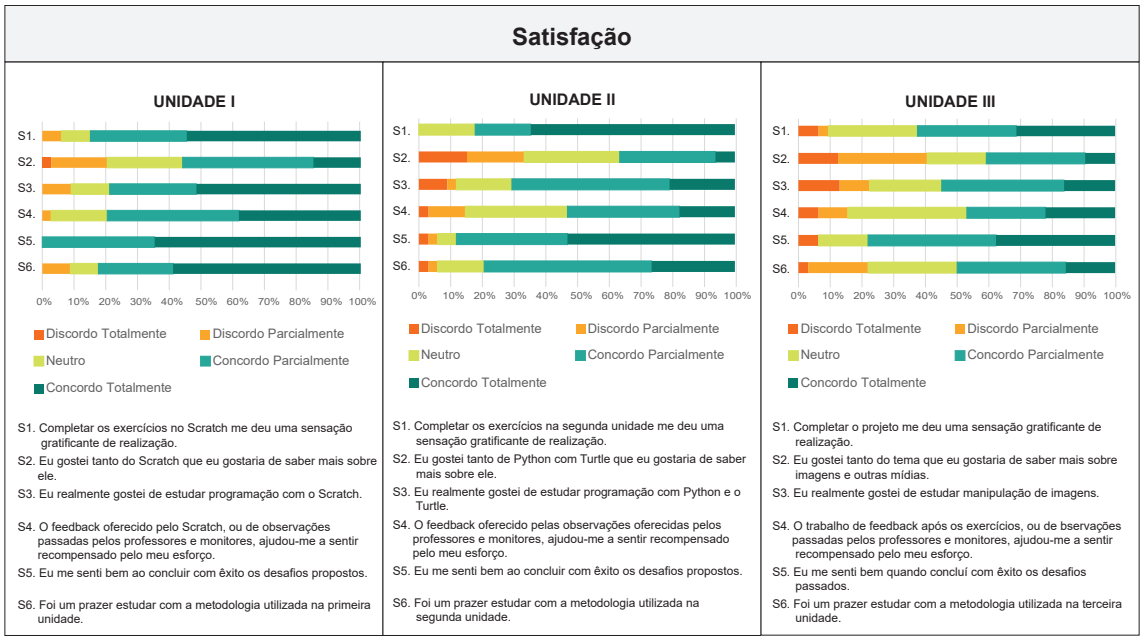

Figura 6.16: Resultados para a categoria Satisfação ao longo das três unidades, considerando o questionário IMMS.

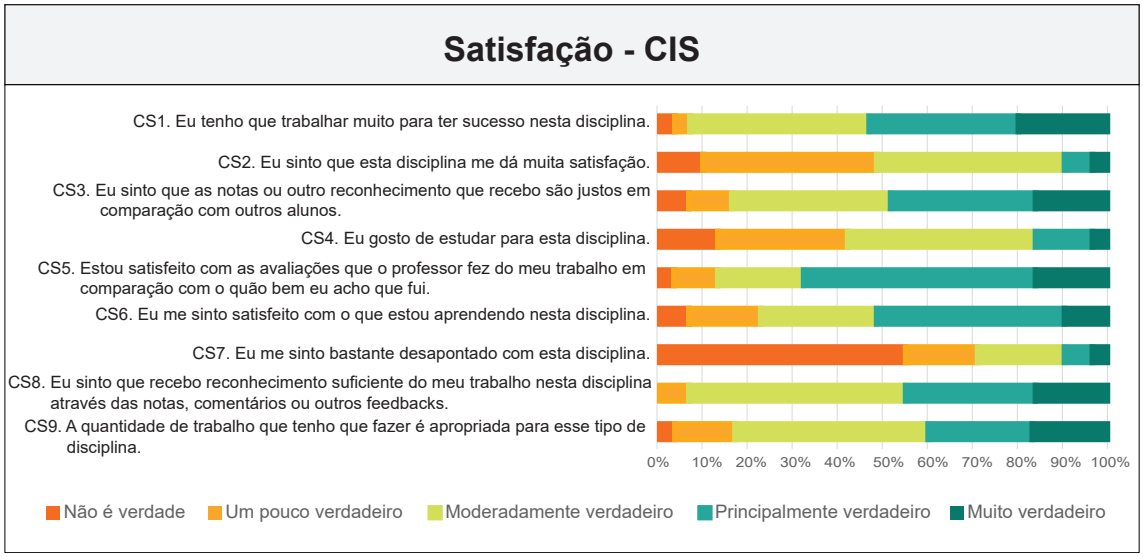

Figura 6.17: Resultados para a categoria Satisfação no questionário CIS.

construto S2 obteve índice de concordância de 55,89% para a Unidade I, e para as demais este incide foi inferior a 50%. Este resultado insinua que mesmo Scratch não apresentando uma sensação de relevância para aspectos profissionais, a experiência de aprendizagem apoiada nesta ferramenta foi satisfatória a ponto de querer saber mais.

A Figura 6.17 apresenta os resultados obtidos para a categoria Satisfação de acordo com o questionário CIS. Os construtos CS3 e CS5 relacionam-se diretamente com a subcategoria Equidade. Os resultados para estes construtos demonstram que, embora a maioria dos estudantes não considerem principalmente verdadeiro ou muito verdadeiro que o reconhecimento recebido é justo em comparação ao dos colegas, eles estão satisfeitos com o as avaliações que o professor faz do seu trabalho. Os demais construtos relacionam-se com mais de uma subcategoria. Apenas os construtos CS1, CS6 e CS7 possuem resultados acentuados. Estes resultados demonstram que os estudantes acreditam que devem trabalhar muito para obter sucesso nesta disciplina, que eles se sentem satisfeitos com o que aprenderam e não estão desapontados com a disciplina.

Buscamos embasamento nos dados qualitativos a fim de entender os resultados obtidos através dos questionários do ARCS. Em relação à subcategoria Consequências naturais, percebemos que ela depende da confiança do estudante para ser potencializada, uma vez que o estudante precisa sentir-se capaz de resolver os desafios relacionados ao conceito aprendido. A contextualização também tem um papel importante. Os contextos de jogos e mídias facilitaram a aplicação imediata do conteúdo ensinado e permitem a visualização instantânea do que está sendo feito. Em especial, as práticas de ensino utilizadas durante a Unidade I contribuíram para que os estudantes se sintam bem com suas realizações. Muitos estudantes entrevistados citaram que quando vencem as dificuldades e conseguem implementar os desafios solicitados, eles se sentem animados: “Eu gostei. Agora no final eu estou super empolgado. (...) Por que quando você começa entender alguma coisa que você vê que tá dando certo, você sente uma sensação prazerosa. Aí muito agradável com isso.” (EP-S09). “Quando eu conseguia completar um desafio e aí esse desafio... não tenho como explicar... era legal eu me sentia animado quando eu conseguia fazer as coisas. Quando eu não conseguia, eu ficava mal: não quero mais não, deixa lá....”(EF-S26).

A subcategoria Consequências positivas não foi explorada pela nossa abordagem. Durante o estudo de caso final, em algumas ocasiões os estudantes receberam recompensa em forma de pontos extra na na nota da unidade. Em nossas observações, este demonstrou ser o elemento de motivação extrínseca mais eficiente. Alguns estudantes citaram a recompensa por nota como algo que motiva, “Quando é, principalmente se valer alguma coisa, aí que a pessoa quer fazer mesmo. (...) O objetivo final vai ser sempre a nota. Se for alguma coisa: ah! não vale ponto, dificilmente eu irei fazer.” (EP-20).

Em relação a Equidade, acreditamos que o formato da avaliação prática realizada nas Unidades I e II ajudou a fortalecer a sensação de que as notas atribuídas são justas.

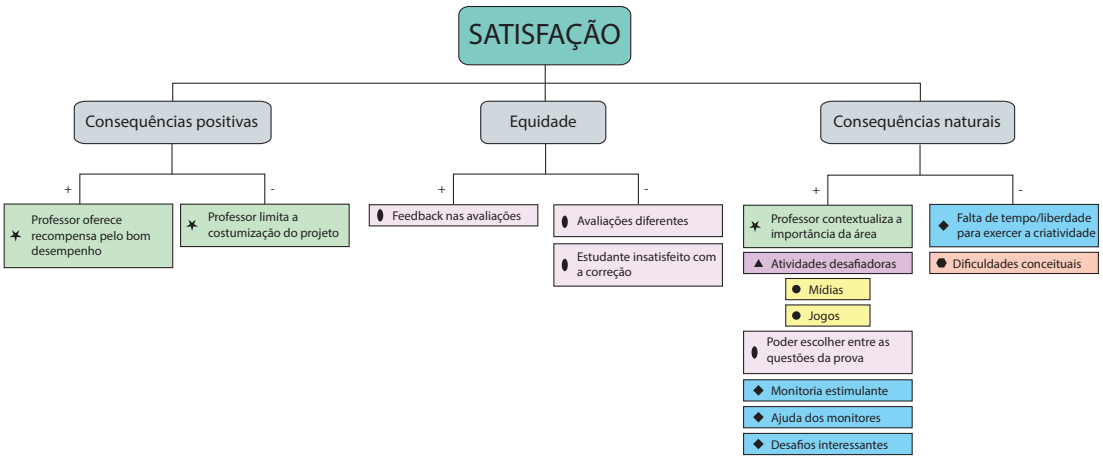

Figura 6.18: Fatores que influenciam na Satisfação dos estudantes.

O *feedback* oferecido no momento de entrega dos resultados foi elogiado por quase todos os entrevistados em ambos os estudos de caso: "Eu falava: ô professora, o por quê dessa nota? Aí ela mostrava na hora o que foi que eu tinha errado. Entendeu? E talvez se ela demorasse mais tempo eu não lembrasse mais nem o que foi que eu fiz ali." (EP-S14) "Ahh... eu acho bom porque aí não mostra só 'você errou', mas explica porque você errou e como poderia ser feito. Acho isso importante porque, como a avaliação é pra testar o nosso conhecimento, se tem alguma coisa que a gente não sabe ali, o *feedback* vai mostrar como fazer." (cod 03 EF).

Outro ponto de satisfação percebido foi que os estudantes comentaram fora da classe o que estavam aprendendo na disciplina. 94,12% comentaram com alguém, que não é da sua turma de ICC (família, amigos etc.), que estavam mexendo numa ferramenta que pode ser usada para fazer jogos e animações. E 78,57% comentaram com alguém, que não é da sua turma de ICC (família, amigos etc.), que estavam aprendendo como manipular imagens. Alguns estudantes compartilharam durante as entrevistas como isso acontecia: "Aprender novas coisas foi bom, mas o principal motivo foi que eu chegava em casa e falava "Mainha, eu fiz um joguinho" (...) era bastante divertido contar essas coisas!" (EF-S05)

A Figura 6.18 exibe alguns aspectos de nossa abordagem agrupados de acordo com sua influência nas subcategorias de Satisfação. Percebemos que a Satisfação tem como pré-requisito a confiança do estudante. Nesse sentido, percebemos que as aulas práticas, tiveram um papel muito importante para potencializar a confiança e satisfação dos estudantes. Notamos que nosso formato de aulas práticas potencializa a aprendizagem e respeita o ritmo de cada estudante sem subestimá-los. Por disseminar a prática dos conteúdos vistos na aula teórica prévia, os estudantes tiveram a possibilidade de treinar suas habilidades com o apoio do professor e dos monitores.

6.5 Aspectos de Nossa Abordagem e sua Influência na Motivação dos Estudantes

A partir dos resultados obtidos para cada categoria e suas respectivas subcategorias, já apresentados nas Figuras 6.6, 6.10, 6.14, e 6.18, montamos um quadro geral que evidencia o impacto, seja positivo ou negativo, dos elementos de nossa abordagem nas categorias do modelo ARCS. É importante salientar que muitos dos elementos são comuns em diversos ambientes de ensino-aprendizagem.

6.5.1 Linguagens e Ferramentas

Os resultados obtidos demonstram que o uso de Scratch potencializa a confiança dos estudantes, diminuindo o medo de errar, e isso influencia diretamente na persistência e realização do estudante. Eliminando a possibilidade de erros de sintaxe, todo o esforço em aprender a programar concentra-se em desenvolver as habilidades de lógica e no entendimento dos conceitos básicos.

Notamos também que a introdução com Scratch facilita a introdução à linguagem Python. Neste caso, por já conhecer alguns conceitos mais básicos, o(a) estudante concentra seus esforços iniciais em aprender detalhes da sintaxe da linguagem. Por outro lado, o aspecto infantil de Scratch não passa a sensação de que o conteúdo abordado é relevante para os objetivos de carreira e vida acadêmica. Os estudantes só enxergam a relevância de Scratch quando percebem que os conceitos vistos nessa ferramenta são transferíveis para linguagens de programação mais profissionais.

A linguagem Python permite trabalhar com um modo de programar mais próximo da prática profissional, e isso aumenta a sensação dos estudantes de que estão aprendendo algo relevante. No entanto, ao se depararem com os erros de sintaxe e demais dificuldades, a atenção e confiança dos estudantes diminuem e, consequentemente, a persistência. Durante a Unidade II, a combinação feita permitiu fazer aulas comparativas, pareando exemplos do Turtle com exemplos com a caneta de Scratch. Todos os estudantes entrevistados conseguiram relacionar Scratch e Python: “ajudou a entender parâmetros, variáveis, funções, um pouco do raciocínio lógico, do passo a passo, pra chegar na solução do problema. (. . .) É. . . por exemplo, quando é para desenhar alguma figura, é a mesma função, tipo assim, só muda o nome, por exemplo, mover em x e backward, forward, que mais. . . ” (S02EP). Com isso, percebemos que, ao trocar de linguagem ou ferramenta, deve-se gastar algum tempo relacionando a ferramenta atual com a anterior.

6.5.2 Contextualização

Ao formatarmos uma disciplina de CS1 no contexto de jogos e mídias, esperávamos potencializar a motivação dos estudantes em todos os aspectos. De maneira geral,

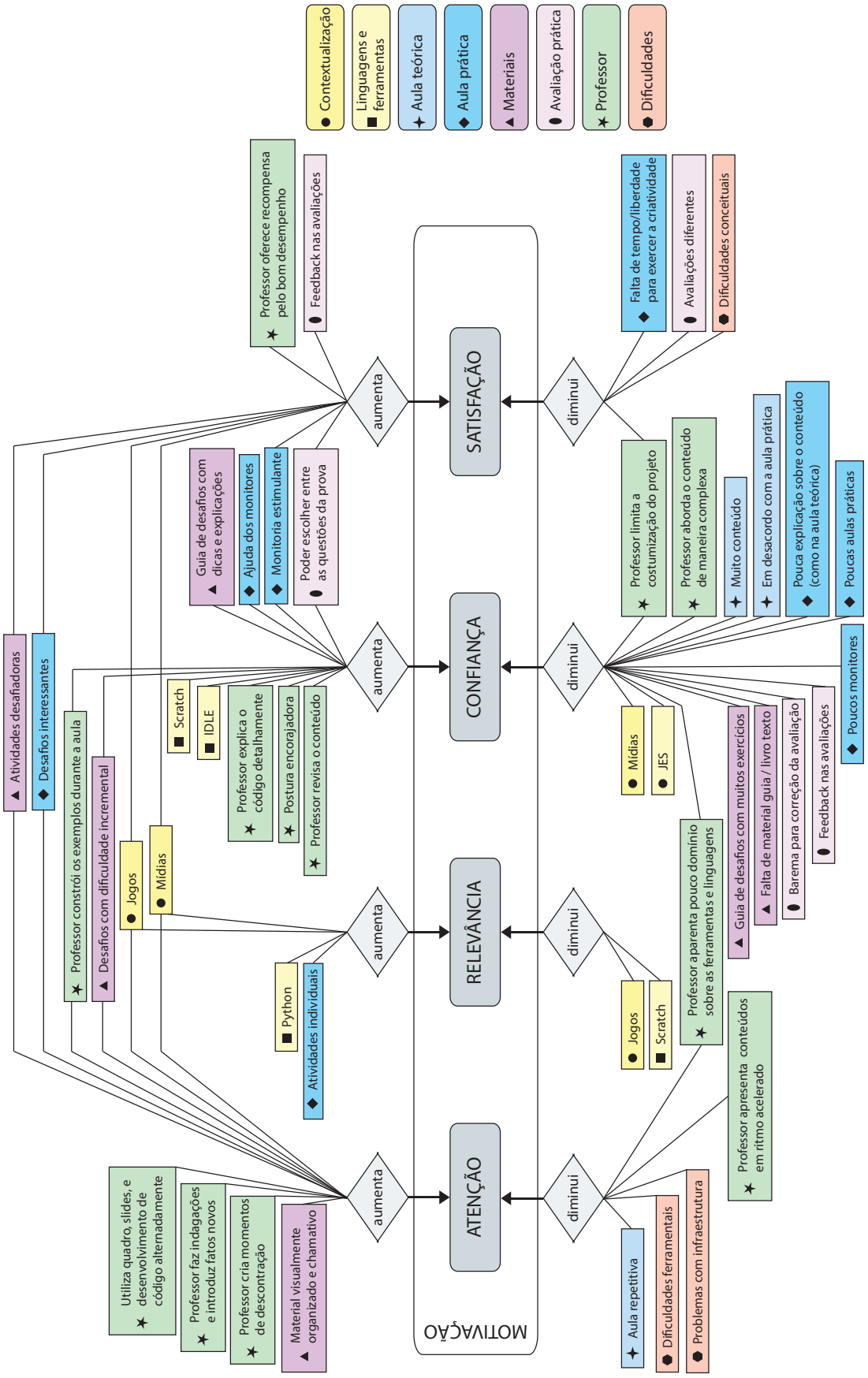

Figura 6.19: Fatores que influenciam na Motivação dos estudantes.

os resultados sugerem que os estudantes ficaram muito mais satisfeitos em aprender a criar jogos do que em manipular imagens. Parte desta impressão ocorreu por jogos terem sido criados numa ferramenta simples como o Scratch enquanto mídias foram exploradas usando uma linguagem de programação textual como Python. A utilização de jogos potencializou mais a confiança e a satisfação dos estudantes enquanto a utilização de mídias potencializou mais a relevância no sentido de orientação de objetivos, pois mostrou que a linguagem que estava sendo aprendida possui um contexto amplo de aplicação.

Mesmo havendo uma diferença no modo como as ferramentas influenciam a motivação, durante as entrevistas ficou claro que ambos os contextos são bons porque aproximam a programação de coisas que eles gostam de fazer. Alguns entrevistados afirmaram que fazer jogos é bom porque eles gostam de jogar e que foi bom aprender como jogos são feitos: "Eu não fazia a menor ideia, eu não tinha noção nenhuma de como é que fazia um jogo. E a partir do Scratch eu vi mais ou menos como é que eles faziam, tipo, eu tive uma noção bem básica de como é que acontece o joguinho que a gente joga." (EP-S14). A utilização de imagens também é apontada como interessante: "Foi bom porque foi, de certa forma, didático, assim como o Scratch, que o jogo fazia eu focar naquilo ali, o fato de eu utilizar imagem, eu puder utilizar isso na minha vida, no caso manipular imagens foi bastante interessante porque me ajudou a focar e conseguir a desenvolver os códigos." (EP-S01).

6.5.3 Aulas Teóricas

Nossa abordagem prevê, nas aulas teóricas, o uso combinado de explicações mais tradicionais utilizando o quadro branco, construção de exemplos passo a passo, e apresentação de slides. Estes são recursos comuns em qualquer ambiente de ensino-aprendizagem. Durante a Unidades I e II, conseguimos combinar estes elementos de maneira equilibrada. Durante a Unidade III, elaboramos muitos slides contendo curiosidades e prévias dos efeitos. Em compensação, devido à maior complexidade dos exemplos, não foi possível construí-los a partir de um projeto em branco em todas as aulas. Notamos que, quando isso ocorre, é mais difícil explicá-los. Notamos também que, se as aulas possuem um caráter mais formal, mostrando apenas conceitos teóricos e detalhes de implementação, os estudantes sentem-se facilmente entediados. Estes fatores contribuem para diminuição da atenção e confiança dos estudantes.

Em contrapartida, percebemos que, quando as aulas trazem alguma curiosidade sobre o tema abordado como, por exemplo, vídeos de *gameplay* de jogos clássicos ou aplicações dos filtros de imagem, mesmo que o conteúdo seja complexo, a atenção é mantida porque soubemos despertar a curiosidade. O verdadeiro desafio está em sustentar a atenção dos estudantes e, para fazer isso, é necessário responder às necessidades de busca de sensações dos estudantes e despertar sua curiosidade de busca de conhecimento sem superestimá-los Keller (1987). A sensação de relevância também

pode ser estimulada se, durante as aulas, as explicações forem contextualizadas com aplicações úteis para o dia-a-dia, a exemplo do uso de imagens em redes sociais.

6.5.4 Aulas Práticas

Notamos que nosso formato de aulas práticas potencializa a aprendizagem e respeita o ritmo de cada estudante sem subestimá-los. Por disseminar a prática dos conteúdos vistos na aula teórica prévia, os estudantes tiveram a possibilidade de treinar suas habilidades com o apoio do professor e dos monitores. Alguns estudantes se sentem confiantes durante as aulas práticas quando recebem instruções prévias.

No formato de aula baseada em desafios, os estudantes são estimulados, mas com a segurança de que, ao sentirem dificuldades, serão apoiados pelos professores e monitores. Este formato se mostrou interessante por apoiar tanto aqueles que são mais dependentes de instruções quanto os estudantes que gostam de desafios: "Assim. . . em todas as aulas práticas, eu me desafiava. Eu raramente pedia, perguntava aos monitores e à professora como fazer, porque eu me desafiava tentando fazer sozinho. Entendeu? E aí, como eu começava a fazer, que via os exemplos, e aí desenvolvia, ia desenvolvendo, aí eu me sentia confiante e conseguia terminar. Quando eu começo a fazer e tenho sucesso logo no início, aí eu me sinto confiante." (S06EP). O formato de monitoria, de não dar respostas prontas também foi avaliado positivamente. Além disso, o fato de todas as atividades serem individuais também obteve destaque. Todos os entrevistados afirmaram gostar das atividades realizadas individualmente pois isso respeita o ritmo de aprendizagem e contribui para reforçar os conceitos.

6.5.5 Materiais

Um aspecto emergente em nossas observações é que, durante as aulas que fizeram uso de slides visualmente organizados e chamativos, com diversas curiosidades sobre o tema, o nível de atenção da turma era maior. "As aulas teóricas eu achei legal, tipo os slides eram tipo assim faziam com que a gente prestasse atenção." (cod 07 EF).

Em relação aos materiais utilizados, destaca-se o guia de desafios entregue aos estudantes durante as aulas práticas. No guia, cada jogo ou exercício a ser feito era dividido em pequenos desafios organizados de acordo com a dificuldade de implementação. Para os estudantes, o guia deu direcionamento sobre a implementação dos exercícios: "Eu acho legal, por que o guia de exercício começa do mais básico até o avançado. E com isso, tipo, você vai evoluindo ao longo da aula." (S09EP). Um ponto negativo é que mesmo com todo o aporte oferecido, os estudantes ainda sentem falta de um livro texto, como ocorre nas disciplinas de cálculo e álgebra por exemplo.

6.5.6 Avaliações Práticas

As avaliações práticas foram a principal forma de avaliar o conhecimento adquirido durante as duas primeiras Unidades. A correção ocorria imediatamente após o término da prova e o estudante recebia o *feedback* sobre seu desempenho de imediato. Do ponto de vista operacional, esta forma de correção toma tempo e é inviabilizada quando não há existência de monitores além do professor. Mas a opinião dos estudantes entrevistados sobre os benefícios do *feedback* fornecido junto a entrega dos resultados foi bastante positiva. Além de melhorar o processo de aprendizagem, tirando as dúvidas dos estudantes, o *feedback* imediato também diminui a ansiedade do estudante em relação a seu desempenho.

6.5.7 Atitudes do Professor

Em nossa abordagem, as aulas possuem caráter menos formal e o professor deve estar mais próximo dos estudantes, demonstrando interesse no *feedback* por eles oferecido. Com este intuito, a partir de nossas observações e entrevistas, identificamos atitudes positivas e negativas dos professores que influenciam diretamente na boa aplicação de abordagem. Além disso, o professor tem um papel fundamental na hora de transmitir o valor daquilo que está sendo ensinado.

O ponto mais importante é que o professor teve ter domínio sobre as linguagens e ferramentas que estão sendo ensinadas. Quando isso não acontece, os estudantes ficam mais desatentos durante as aulas e levantam a ideia de que estão sendo cobrados por um conhecimento que não é comum ou relevante.

A partir das observações das aulas, notamos que a turma fica mais atenta quando o professor reforça periodicamente os objetivos de aprendizagem e a importância de determinados conceitos. Ao assumir uma postura encorajadora, incentivando os estudantes a se desafiarem, o professor estimula a confiança dos estudantes sem subestimá-los, fortalecendo o aspecto positivo de monitoria. Quando o professor explora as diferentes possibilidades de instrução, alternando entre explicações no quadro, a atenção da turma é captada. Já quando o professor apresenta um ritmo de instrução muito rápido, ou não constrói os exemplos durante a aula, é provável que os estudantes com maiores dificuldades deixem de prestar atenção na instrução e de se esforçar na implementação dos desafios.

6.5.8 Dificuldades

Ao analisar as observações, identificamos quatro tipos de dificuldades mais frequentes no nosso ambiente: conceituais, idioma, ferramentais e de infraestrutura.

As dificuldades conceituais se referem aos conceitos de programação. As dificuldades com o idioma inglês, apesar de não receberem concordância da maioria dos

estudantes nos questionários aplicados, foram bastante observadas durante as aulas. Esta foi a descoberta mais inesperada, pois imaginávamos que parte dos estudantes sem conhecimento de inglês saberiam contornar essas dificuldades. Ao esbarrar nesse tipo de dificuldade, a confiança do estudante é testada, e consequentemente a satisfação com a experiência de aprendizagem diminui. Pessoas confiantes tendem a atribuir as causas do sucesso a aspectos como capacidade e esforço em vez de sorte ou a dificuldade da tarefa. Já os inseguros, preocupam-se muito com as falhas. Quanto maiores as dificuldades, fica desacreditada a impressão de que algum nível de sucesso é possível se o esforço for exercido, e aumenta a impressão de que "isso não é pra mim, eu não nasci pra isso aqui." (S14 EP).

As dificuldades ferramentais são aquelas onde o estudante percebe algum *bug* ou limitação na ferramenta, enquanto as dificuldades de infraestrutura estão relacionadas com a falha de dispositivos ou falhas de organização que atrapalham o andamento da aula. Em nossas observações, notamos que sempre que uma dessas dificuldades surge, a atenção da turma é perdida, e dificilmente é retomada posteriormente.

Capítulo 7

Discussão

Neste capítulo, finalizamos a discussão acerca dos resultados obtidos, tomando como base uma comparação com o cenário tradicional da disciplina de ICC em nossa instituição e as questões de pesquisas que norteiam este trabalho. A abordagem de ensino-aprendizagem proposta visa aumentar o engajamento e melhorar a motivação dos estudantes, além de ser aplicável com diversos públicos de *non-majors*. Os estudos de caso foram conduzidos apenas com turmas de ICC no curso de Engenharia Civil porque essas turmas eram viáveis para a execução da pesquisa. Porém, em nossa instituição, estudantes de diversos cursos possuem em sua matriz curricular disciplinas do tipo ICC.

Procuramos entender o cenário tradicional da disciplina entrevistando três professores de nossa instituição que costumam ministrá-la com maior frequência. Também aplicamos o questionário CIS em uma turma de ICC para Engenharia Civil que não utilizou a nossa abordagem.

7.1 O Cenário Tradicional de ICC

A partir das entrevistas com professores, pudemos entender melhor sobre como esta disciplina vem sendo formatada para diferentes audiências em nossa instituição. Identificamos três grupos principais: estudantes de Engenharia Civil e de Engenharia de Alimentos; estudantes dos cursos de Licenciatura em Física, Matemática e Química, e Estudantes dos cursos noturnos de Administração, Ciências Econômicas e Ciências Contábeis. Este último teve a ementa da disciplina recentemente alterada, perdendo o foco em programação. O formato das disciplinas é sempre de 60 horas de carga horária total, normalmente divididas meio a meio entre aulas teóricas e práticas.

As ementas dos componentes curriculares variam mas, em geral, o foco está na aprendizagem de programação. No componente particular de EXA170 – Introdução

à Ciência da Computação, ofertado para Engenharia Civil e Engenharia de Alimentos, a linguagem a ser utilizada não é pré-definida, sendo de livre escolha do professor. Dentre os três professores entrevistados, dois consideram que conseguem cumprir a ementa da disciplina, mas com algumas ressalvas. Os tópicos gerais sobre ciência da computação usualmente são condensados em poucas aulas: "Consigo cumprir a ementa da disciplina. Agora, consigo cumprir porque, na ementa da disciplina fala em trabalhar com sistemas operacionais, por exemplo, e sistemas operacionais eu falo em metade de uma aula. (...) Eu poderia fazer um estudo sobre sistemas operacionais mas eu não faço isso. Falo sobre sistemas operacionais, caracterizo e cito alguns sistemas operacionais. (...)"(T02).

Também perguntamos aos professores quais conteúdos de programação costumam ser abordados. Usualmente, os conteúdos de programação cobrem tipos de dados, variáveis, expressões lógicas e aritméticas, seleção, repetição, até o tópico de matrizes e vetores. Em poucas turmas, o conteúdo de registros e funções consegue ser abordado: "Nunca, enquanto eu estive, nunca passou de vetores e matrizes. Teve uma turma que eu comecei a esboçar estrutura mas é bem complicado. Eles ficam travados principalmente nessa coisa de repetição."(T03). Segundo os professores, o planejamento inicial da disciplina é muitas vezes alterado conforme o ritmo apresentado pela turma e há uma limitação por parte dos estudantes: "Não, sempre tem uma mudancinha e nunca é exatamente [como planejado]. Uma coisa ou outra sempre muda. A turma também, tem dia que a turma tá mais né... menos atenta... No geral pode até se dizer que sempre o que você planeja, você consegue lecionar, mas sempre tem uma mudança ou outra." (T01).

Em relação à seção teórica da disciplina, todos os professores entrevistados descreveram basicamente as mesmas práticas. Na maioria das vezes, o assunto é explicado com o apoio do quadro e, em algumas aulas, o projetor é utilizado para exibição de slides com explicações mais teóricas. O formato de aulas práticas varia bastante entre os professores. Um ponto em comum com nossa abordagem é que as aulas normalmente seguem um roteiro com exercícios pré-definidos que demandam implementação de pequenos programas.

A avaliação de aprendizagem normalmente é realizada através de prova escrita. Todos os professores entrevistados citaram que as questões das avaliações são sempre voltadas para exercícios de programação e raramente para aspectos teóricos sobre ciência da computação ou até mesmo sobre as linguagens ensinadas. Não muito frequentemente, as notas são complementadas por outras atividades. Quando isto ocorre, pode ser através de listas de exercícios, avaliação oral ou seminários.

Todos os professores entrevistados citaram que já realizaram atividades lúdicas junto às turmas. Foram citadas a realização de atividades desplugadas: "Já fiz coisas como pegar um algoritmo e, tipo um quebra-cabeça. (...) É, fiz bastante isso no início, assim, quebrei, peguei papel cartão, peguei algoritmos grandes, tirei os algoritmos e fiz um quebra-cabeça pra testar justamente essa coisa da sequência lógica."(T03). Também foram citadas a realização de atividades desplugadas combinadas com exer-

cícios de programação: “Já tentei fazer programas com cubo mágico, levei o cubo mágico, dava sorte que havia algum ali que sabia resolver em menos de 2 minutos e empolgava eles. Já levei torre de Hanói, levei a torre de Hanói, brincamos e depois fizemos programas.”(T02). Os professores também apontaram experiências com ferramentas lúdicas como o Scratch: “(...) em algumas turmas, eu tentei trabalhar com o Scratch. Já tentei trabalhar, não é exatamente com uma linguagem, é mais um ambiente, o Alice. Eu já tentei fazer pelo menos para eles perceberem as estruturas de repetição e as estruturas de seleção. Mesmo que eu esteja trabalhando com C. Quando chega na estrutura, eu paro, primeiro faço uma atividade com o ambiente, Alice ou Scratch. Depois, como eles percebem muito rapidamente, aí, agora sim, vamos tentar escrever isso numa linguagem mais formal. E há muitos anos, tinha uma linguagem chamada Logo. E o Logo é uma linguagem muito forte, a parte de recursividade dela é muito forte. Mas só é muito divulgada a parte bem inicial. Aí eu tentava trabalhar com eles.”(T02). Ao perguntar quais turmas foram contempladas com estas experiências, os professores citaram com maior frequência as turmas dos cursos de licenciatura e uma experiência com Scratch em uma turma de ICC para Engenharia de Alimentos. Todos os professores admitiram que as experiências com atividades lúdicas pareceram bastante vantajosas e fomentaram a participação dos estudantes na disciplina. Porém, este tipo de atividade foi pouco explorada ou sequer foi continuada dentro da disciplina. A rotina corrida da vida docente foi citada como um empecilho para elaboração deste tipo de atividade.

Frequentemente, mais de um professor é alocado para ministrar a disciplina de ICC. O cenário mais comum é um professor ser alocado para disciplina teórica e um professor para as duas turmas práticas ou ainda dois professores, um em cada turma prática. Quando isto ocorre, normalmente eles se reúnem no início da disciplina e o professor da turma teórica é quem decide a linguagem a ser utilizada. Todos os professores entrevistados comentaram preferir ministrar a disciplina sozinhos. Segundo eles, há um esforço conjunto para conciliar o andamento em termos de conteúdo abordado nas turmas teórica e prática, no entanto, é difícil manter essa organização funcionando bem até o fim da disciplina: “A princípio, a teoria não consegue caminhar com a prática. A gente até tem conversado que seria melhor que sempre o professor da teoria fosse o da prática. Quando não acontece, as coisas não caminham no mesmo ritmo. As vezes acontece isso, o professor [da teórica] está um pouco mais atrasado e a gente [na turma prática] está além nos conteúdos programáticos.”(T03).

Para os professores entrevistados, é unanime a percepção de que há uma diferença no interesse apresentado pelas audiências. As turmas de licenciaturas, em especial o curso de Física, foram citadas como as que mais possuem casos de estudantes interessados em aprender programação, o que eles chamaram de “ponto fora da curva”. Enquanto as turmas de Engenharias, em especial as de Engenharia Civil, foram citadas como as que apresentam melhores resultados tanto em termos de avanço no cumprimento do conteúdo programático como nos resultados da disciplina.

A evasão é um problema mais comum entre os estudantes repetentes ou de semes-

tres mais avançados. Para os professores, parece haver uma associação clara entre a evasão em uma turma e existência de pouca motivação. Também percebemos, entre todos os professores entrevistados, uma associação entre motivação e atenção, esta última também como um indicador da existência de motivação. Ao perguntar se os professores costumam considerar a motivação como um fator importante no planejamento das aulas, os professores citaram uma preocupação em trabalhar os conteúdos de maneira a despertar e manter a atenção dos estudantes durante as aulas. Um dos entrevistados considera que uma das dificuldades em captar a atenção dos estudantes pode ser a existência de uma "incompatibilidade"entre o formato atual das disciplinas e a dinâmica atual da sociedade: "Hoje com, é, essa dinâmica que mudou na sociedade com relação a você ter aí na palma da sua mão tudo, né, então é tudo muito rápido. A questão da concentração para eles é terrível. Essa geração é terrível, porque é muita coisa ao mesmo tempo e aí para parar e prestar atenção numa aula é muito complicado. Então tem que ser uma aula mais dinâmica, agora, procurar coisas."(T01).

Os professores parecem considerar uma forte associação entre motivação como consequência da relevância percebida. Apesar de, em alguns momentos das entrevistas, os professores citarem a importância de aprender a programar como uma espécie de exercício para a mente, parece haver um consenso sobre a relevância estar associada à utilidade para a futura vida acadêmica ou profissional: "E principalmente essas disciplinas nas engenharias propiciam que eles trabalhem nas disciplinas do curso que exigem alguma coisa de programação (...) como também eu acho que nós termos um engenheiro civil, um engenheiro de alimentos com possibilidade maior de dialogar com o pessoal da área de TI que tenha uma empresa."(T02). Todos os professores relataram que é comum haver pouco engajamento nas turmas de ICC, onde poucos alunos apresentam curiosidade em aprender a programar. Eles também consideram a existência de uma dificuldade em comunicar o valor da programação para os estudantes. Esse problema parece ser maior para as turmas dos cursos noturnos e para as engenharias. Para as engenharias, apesar de apresentarem resultados melhores em relação ao entendimento do conteúdo: "Tinham algumas pessoas, até tem, até hoje eu imagino, que acham que aquilo não vai interessar para nada mais. Que é uma outra questão. Alimentos falava muito isso 'eu uso isso aqui pra que? Não vou usar mais pra nada'"(T03) ; "Civil, eles passam na disciplina, então eles estudam pra passar. É isso, eles não veêm a importância da disciplina."(T01). Ao perguntar sobre quais habilidades os professores acreditam que são importantes para os estudantes em ICC, além da possibilidade de profissionais de outras áreas que saibam dialogar com profissionais de TI, também são citadas uma possível melhoria na capacidade de raciocínio lógico dos estudantes.

Os professores acreditam que o momento em que as disciplinas de ICC são ofertadas no curso influenciam diretamente na sensação de relevância percebida pelos estudantes. Dentre os entrevistados, parece haver um consenso com o fato de que é mais fácil demonstrar a importância de aprender a programar aos estudantes calouros do que aos veteranos: "Eu acho que eles não percebem, mas o pessoal do primeiro

semestre é um pessoal que qualquer disciplina que for ofertada a eles, eles vão se envolver. Então cabe ao professor mostrar a eles a importância dessa disciplina para a vida acadêmica."(T02). Para os professores, os estudantes calouros, por ainda não terem uma vivência profissional de fato, são convencidos mais facilmente de que aquela disciplina será útil na vida acadêmica e profissional.

7.2 Aprendizagem

Uma das questões de pesquisa deste estudo busca identificar quais competências tem o aprendizado potencializado pela nossa abordagem. A abordagem utilizada não ameniza todas as dificuldades dos estudantes, mas encontramos diferenças significativas em relação ao nosso contexto tradicional. O Capítulo 5 apresenta os resultados de aprendizagem obtidos para cada unidade. Percebemos que Scratch potencializa a aprendizagem de lógica de programação, estruturas de repetição e, em menor nível, estruturas condicionais, ao mesmo passo em que conceitos como variáveis não são tão potencializados. O Python com a biblioteca Turtle facilita a aquisição dos conceitos de lógica de programação, estruturas de repetição, estruturas condicionais e funções. Já abordagem com mídias demonstrou potencial para os conceitos de funções, vetores e matrizes, embora não tenha trazido um aporte que diminuísse efetivamente as dificuldades dos estudantes em aprender estes conceitos.

Na abordagem tradicional, os professores relacionaram as dificuldades apresentadas pelos estudantes com o fato de que a programação envolve o pensamento computacional: "Porque uma linguagem de programação é uma formalização de uma forma de pensar que é muito diferente da nossa."(T02). Os professores também citaram a existência de uma deficiência prévia em relação à base matemática e a percepção de que isso vem aumentando ao longo dos anos, embora não haja nenhum estudo comprobatório. Em nossos estudos de caso, também percebemos que a falta de base matemática pode aumentar as dificuldades dos estudantes ao realizar alguns exercícios.

Todos os professores entrevistados foram unânimes em citar que os estudantes dos cursos de Engenharia de Alimentos e Engenharia Civil, em média, apresentam menos dificuldades na compreensão dos conteúdos que os demais cursos. Em relação a conteúdos, dois dos professores entrevistados afirmaram que, no cenário tradicional, há uma notória dificuldade em entender estruturas de repetição. Em nossa abordagem, o uso de estrutura de repetição é abordado em cada uma das unidades: na Unidade I, os estudantes aprendem o conceito no Scratch; na Unidade II, os estudantes aprendem a utilizar o laço for, e veêm uma analogia com tudo o que já foi aprendido no Scratch, de maneira que a novidade em aprender *loops* reside mais na sintaxe do comando do que em entender o conceito.

Nenhum dos professores citou a dificuldade em aprender variáveis ou estruturas como vetores e matrizes, que apareceram bastante em nossos estudos de caso. No caso

destes conceitos, é preciso ressaltar que a abordagem de mídias faz uso exaustivo de vetores e matrizes enquanto que, no cenário tradicional, o nível de dificuldade e exposição destes conteúdos podem ser mais facilmente definidos pelo professor. Além disso, estes conceitos foram apresentados apenas na Unidade III e não foram abordados de maneira espiral. Os professores também citaram "uma dificuldade associada à própria sintaxe. Às vezes, os estudantes sabem o que fazer mas não sabem como, ou sabem a sintaxe mas não sabem de que maneira usar aquilo."(T03). As dificuldades com sintaxe da linguagem não são destaques em nossa abordagem.

Em se tratando de aprendizagem, acreditamos que a abordagem em espiral dos conteúdos contribui significativamente para amenizar as dificuldades dos estudantes. Esse tipo de abordagem oferece uma revisão iterativa de tópicos e, em cada repetição, além da revisão há também o aprofundamento sobre o assunto em relação à etapa anterior Harden (1999). Em nossa abordagem, cada unidade repetia parte dos conceitos da unidade anterior antes de apresentar novos conceitos. Essas revisões demandam bastante tempo e, considerando o tempo limitado de uma disciplina, diminui a gama de conteúdos novos que podem ser abordados. No entanto, esta abordagem dá aos estudantes com mais dificuldades a oportunidade de rever explicações e se nivelar aos demais colegas. Ao fazer repetições de conteúdo, é preciso tomar cuidado para não entediar os estudantes que estão em um ritmo mais adiantado. Como nossa abordagem faz uso de contextos e ferramentas diferentes em cada unidade, este problema foi amenizado.

Percebemos que a transição entre contextos demanda tempo mas potencializa a abordagem em espiral. A escolha do Scratch para a primeira unidade objetivou promover uma introdução mais branda aos conceitos iniciais de programação. Por ser uma linguagem de blocos e visual, os estudantes não precisam se preocupar com detalhes de sintaxe. Por estar em português e seus comandos terem representação mais próxima da linguagem humana, as práticas computacionais se tornaram cognitivamente menos exigentes. Utilizar a caneta do Scratch nas primeiras lições da unidade I facilitou a transição para a linguagem Python com a biblioteca Turtle, uma vez que os códigos se tornaram altamente comparáveis. Tivemos um cuidado semelhante ao promover a transição de contexto entre a segunda e a terceira unidades. Antes de tratar da manipulação de mídias, reservamos algumas aulas para explorar exercícios com Python sem utilizar a biblioteca Turtle. Implementamos alguns jogos textuais e programas para cálculo de áreas de figuras geométricas, por exemplo. É inegável que, à medida em que os detalhes de sintaxe e a complexidade do código crescem, os estudantes apresentam mais dificuldades. No entanto, acreditamos que as dificuldades dos estudantes com o entendimento de Python foram amenizadas por nossa abordagem. Outro ponto é que a diversidade de exercícios utilizando mesmos conceitos de programação em diferentes contextos pode ter reforçado a utilidade da programação para resolução de problemas.

Considerando nossos resultados e a percepção apresentada pelos professores de nossa instituição, acreditamos que nossa abordagem se sai melhor em relação à aprendizagem de programação do que o cenário tradicional. No entanto, devido a limitações

de tempo e disponibilidade de recursos, não foi possível realizar um levantamento sobre as competências dos estudantes ao final do curso em comparação com uma disciplina de ICC tradicional, principalmente levando em consideração o contexto com mídias. Na literatura, não há um consenso se os estudantes que fazem cursos de CS1 com o contexto de mídias aprendem tanto quanto estudantes que aprendem em outros contextos. Algumas investigações demonstram que estudantes que aprendem programação em um contexto baseado em mídias também conseguem resolver problemas de contextos diferentes de mídia Maxwell and Taylor (2017); Simon et al. (2010a), enquanto outros estudos demonstram que estudantes que cursam CS1 num contexto de mídias apresentam desempenho inferior em relação a outros contextos Guzdial (2013).

7.3 Motivação

Neste trabalho, uma de nossas questões de pesquisa procura entender como a abordagem proposta influencia na motivação dos estudantes. No Capítulo 6, apresentamos os resultados obtidos a partir dos estudos de caso e promovemos uma discussão sobre os fatores da abordagem proposta e sua influência em diversas variáveis que afetam a motivação dos estudantes, de acordo com o modelo ARCS. Embora não seja intenção deste estudo promover uma comparação definitiva entre nossa abordagem e o cenário atual, é importante entender as mudanças promovidas.

Por isso, aplicamos o questionário CIS numa turma de ICC para Engenharia Civil composta apenas por calouros. A turma de ICC do segundo semestre de 2017 segue o mesmo padrão de abordagem didático-pedagógica descrito na Seção 7.1. A disciplina é ministrada por três professores diferentes, um na teórica e os demais, cada um em uma turma prática. Nas aulas teóricas, o professor aborda o conteúdo de programação através do Portugol, com explicações escritas no quadro ou projetadas em slides, mas não utiliza nenhum tipo de programa 'compilador' de Portugol, de modo que as atividades são realizadas em papel. Nas aulas práticas, os estudantes podem praticar exercícios de programação em linguagem C. Obtivemos 39 respostas para o questionário CIS, que utiliza uma escala de Likert com as opções "Não é verdade","Um pouco verdadeiro", "Moderadamente verdadeiro", "Principalmente verdadeiro"e "Muito verdadeiro", convertidas posteriomente numa escala numérica de 1 a 5. Consideramos a média como orientação para entender o resultado dentro da escala utilizada. Testamos os dados para a hipótese de distribuição normal no SPSS. As percentagens foram, para o teste Kolmogorov-Smirnov, na categoria Atenção, $D(39) = 0{,}099$, $p > 0{,}05$, para Relevância, $D(39) = 0{,}115$, $p > 0{,}05$, para Confiança, $D(39) = 0{,}101$, $p > 0{,}05$, e para Satisfação $D(39) = 0{,}135$, $p > 0{,}05$. Como a hipótese nula não foi rejeitada aqui, consideramos as distribuições possivelmente normais e promovemos comparações com os dados do Estudo de Caso Final.

A Figura 7.1 apresenta o gráfico de barra de erros onde pode ser feita a comparação entre as médias obtidas para o questionário CIS em nosso estudo de caso final e

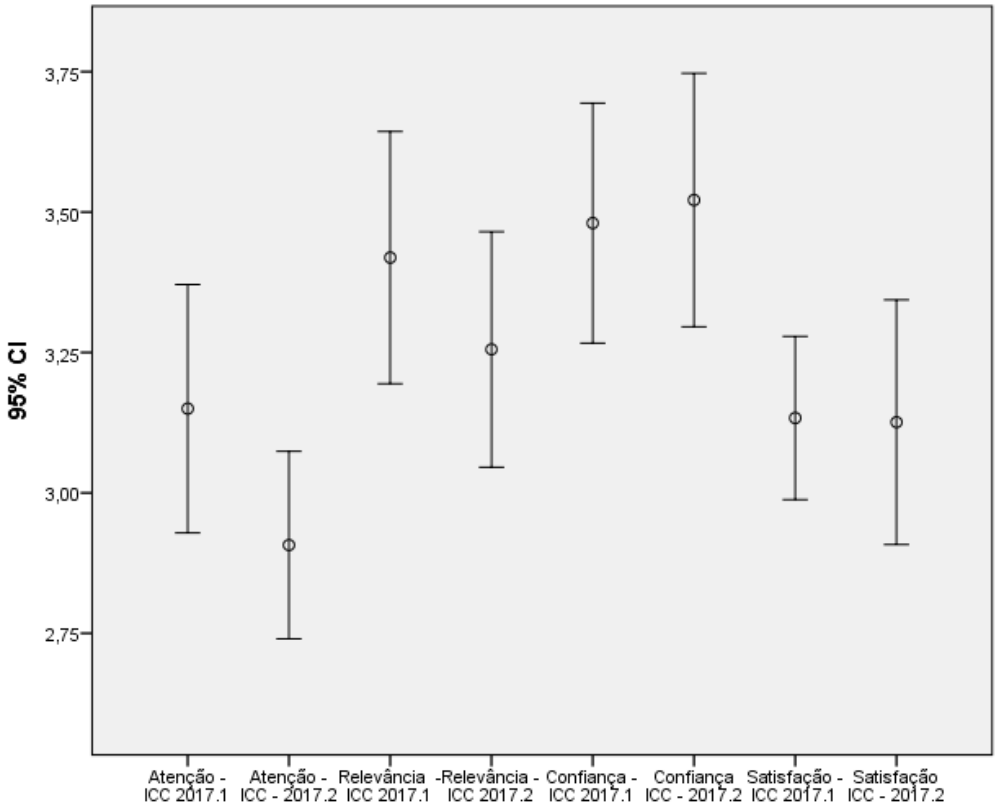

Figura 7.1: Gráfico de barra de erros com a comparação entre as médias das categorias ARCS nos semestres 2017.1 e 2017.2.

Tabela 7.1: Estatísticas para médias das categorias do modelo ARCS, mensuradas pela aplicação do questionário CIS.

Categoria	**Grupos**	**N**	**Média**	**Desvio Padrão**	**Erro padrão da Média**
Atenção	ICC 2017.1	31	3,1209	0,6042	0,1085
	ICC 2017.2	39	2,8933	0,4404	0,0705
Relevância	ICC 2017.1	31	3,4018	0,5985	0,1075
	ICC 2017.2	39	3,3198	0,5496	0,0880
Confiança	ICC 2017.1	31	3,4567	0,5774	0,1037
	ICC 2017.2	39	3,5682	0,6267	0,1003
Satisfação	ICC 2017.1	31	3,1182	0,3922	0,0704
	ICC 2017.2	39	3,1709	0,5823	0,0932

Tabela 7.2: Resultado para o teste t comparando médias do questionário CIS.

	Teste de Levene para igualdade de variâncias		**Teste t**			**Tamanho do**
Categoria	**Z**	**Sig.**	**t**	**df**	**Sig.**	**r**
Atenção	3,453	0,067	1,823	68	0,073	0,215
Relevância	0,068	0,795	0,597	68	0,555	0,072
Confiança	0,359	0,551	-0,765	68	0,447	0,092
Satisfação	8,992	0,004 (Variâncias iguais não assumidas)	-0,451	66,371	0,654	0,055

no semestre 2017.2, onde nossa abordagem não foi aplicada. A tabela 7.1 mostra os dados das médias e desvio-padrão. Também aplicamos o teste t para variáveis independentes a fim de verificar a diferença entre as médias das categorias do modelo ARCS nos dois semestres. A tabela 7.2 apresenta os valores obtidos.

Para a categoria Atenção, o semestre 2017.1 obteve maior média. Esta diferença foi significativa, se considerarmos a hipótese inicial de que as médias no semestre 2017.1 seriam maiores, $t(68) = 1{,}823$, $p < 0{,}05$; entretanto, ela representou um tamanho de efeito médio $r = 0{,}21$. Para a categoria Relevância, o semestre 2017.1 obteve maior média, mas esta diferença não foi significativa $t(68) = 0{,}597$, $p > 0{,}05$. Para as categorias Confiança e Satisfação, o semestre 2017.2 obteve as maiores médias, porém a diferença não foi significativa e o tamanho do efeito calculado foi pequeno com $r<0{,}1$ em ambas as categorias. Com base nas estatísticas apresentadas, podemos concluir que apenas a atenção foi maior no semestre em que a nossa abordagem foi empregada, mas nas demais categorias não houve diferença significativa. Uma possível explicação para esse resultado na categoria atenção pode estar relacionada aos fatores que influenciam nas variáveis e Excitação perceptiva e *Inquiry Aurousal*. Na Figura 6.6, apresentamos um diagrama com fatores que influenciam a categoria de atenção. A contextualização com jogos e Mídias é um dos principais diferenciais de nossa abordagem em relação ao cenário tradicional. Esses contextos influenciam diretamente na variável de excitação perceptiva, captando a atenção dos estudantes em um primeiro momento.

Os resultados para a categoria Relevância, de certa forma, já eram esperados uma vez que a Figura 6.10 sugere haver poucos fatores em nossa abordagem que aumentem a sensação de relevância percebida. É preciso ressaltar que nossa abordagem busca fomentar a relevância em aspectos diferentes do que a relevância para oportunidades de carreiras futuras, enquanto que, na abordagem tradicional, os professores demonstraram consenso sobre a relevância estar associada principalmente à utilidade para a vida acadêmica ou profissional. Estes resultados podem tornar questionáveis nossa escolha em trabalhar com jogos e mídias, em vez de tópicos de engenharia. Fundamentamos nossa escolha em três fatores: a própria literatura demonstra que, considerando a teoria do valor-expectativa, os estudantes são moti-

vados pelas próprias realizações e não somente na carreira futura Jenkins (2001b); os contextos escolhidos tornam nossa experiência replicável para outras audiências de *non-majors*; e o contexto de jogos, isoladamente, permite a adoção de práticas de ensino focadas em diminuir as dificuldades dos estudantes.

Conforme pode ser visto na Figura 6.14, identificamos praticamente a mesma quantidade de fatores que aumentam e diminuem a confiança dos estudantes. Boa parte destes fatores se relacionam a aspectos que acompanham a contextualização com mídias. Isso ocorre devido a este contexto não ser pensado para tornar a aprendizagem de programação mais fácil. Além disso, comparando o cenário tradicional e o cenário com nossa abordagem, percebemos que, em nossa abordagem conceitos como funções são trabalhados logo no início da disciplina e conceitos como matrizes e vetores são mais explorados, o que não ocorre no cenário tradicional. Mesmo havendo diversos recursos pensados para que os estudantes acreditem na possibilidade de sucesso e que eles tenham controle sobre seu sucesso, abordar conceitos mais complexos adiciona algum grau de dificuldade na aprendizagem, o que pode desencorajar parte dos estudantes. Além disso, embora nossa abordagem tente promover novas práticas, o controle que o professor exerce sobre o processo de ensino-aprendizagem permanece e sabemos que os estudantes gostam de sentir que estão no controle de seu próprio destino e de seu próprio sucesso ou falhas Jenkins (2001b); Keller (2009).

Em relação à categoria Satisfação, com nossa abordagem a avaliação cai à medida em que a disciplina caminha. Percebemos que quanto mais lúdico é o processo de ensino-aprendizagem, maior é a satisfação apresentada pelos estudantes. A satisfação diminui ao mesmo passo em que a forma com que os conteúdos são abordados se aproximam do ensino tradicional de programação e sua complexidade. Com base na literatura apresentada na Seção 2.3, podemos afirmar que a complexidade dos conteúdos de programação pode ser evitada em algum grau, mas não completamente. Ensinar vetores e matrizes apoiando-se no contexto de mídias aumenta o interesse dos estudantes em um primeiro momento, mas não diminui a dificuldade em explicar estes conceitos.

Devido às dificuldades e ao grau de esforço e recursos necessários para promover um estudo de caso aprofundado, não foi possível promover uma análise mais rica sobre o cenário tradicional de ICC. É preciso considerar que há uma limitação em promover algum grau de comparação baseando-se apenas nos relatos dos professores e resultados de um questionário.

Capítulo 8

Considerações Finais

Neste trabalho, descrevemos a concepção, aplicação e avaliação de uma abordagem de ensino-aprendizagem de programação destinada a estudantes que não são de cursos de TI (*non-majors*). A abordagem proposta foi formatada para uma disciplina com carga horária de 60 horas, divididas meio a meio entre aulas teóricas e práticas. A abordagem está organizada em três unidades. A Unidade I apresenta os conceitos básicos de programação com a ferramenta Scratch, exemplificados através da construção de jogos. Na Unidade II, é utilizada a linguagem de programação Python combinada com a biblioteca Turtle, e todos os exemplos e exercícios utilizados implementam rotinas para desenhos de figuras geométricas. A Unidade III utiliza a linguagem de programação Python e o ambiente de desenvolvimento JES e todos os exemplos e exercícios utilizam o contexto de manipulação de mídias.

Realizamos dois estudos de caso, denominados neste trabalho de Estudo de Caso Piloto e Estudo de Caso Final, ambos realizados com turmas de ICC para o curso de Engenharia Civil. Procuramos entender como a abordagem proposta influência na motivação dos estudantes e quais competências têm o aprendizado melhor potencializado. No estudo de Caso Piloto, 36 estudantes participaram da pesquisa e, no Estudo de Caso Final, contamos com 37 participantes. Aplicamos uma metodologia quali-quantitativa, com levantamento de dados através da aplicação de questionários, entrevistas individuais e observação das aulas teóricas e práticas. Elaboramos cinco questionários e três roteiros de entrevistas, com o intuito de levantar o perfil dos estudantes e suas expectativas em relação à disciplina, obter as percepções sobre as aulas, conteúdos abordados, materiais e, principalmente, sobre a motivação dos estudantes. Todo o estudo sobre motivação foi baseado no modelo ARCS.

Apresentamos os resultados referentes a aprendizagem e motivação em capítulos distintos. E buscamos aprofundar a discussão no Capítulo 7, onde fazemos também, uma descrição do cenário tradicional de ICC em nossa instituição. Do ponto de vista da aprendizagem, os pontos fortes deste trabalho residem em identificar, em nossos casos, o Scratch como um facilitador da aprendizagem de lógica de programação e estruturas de controle como loops e condicionais, enquanto Python com

Turtle melhora as habilidades com lógica de programação, estruturas de repetição, estruturas condicionais e funções. A abordagem com mídias, em nossos casos, traz muitas oportunidades em trabalhar os conceitos de funções, vetores e matrizes, embora não tenha trazido um aporte que diminuísse efetivamente as dificuldades dos estudantes em aprender estes conceitos. Acreditamos que os pontos fortes das linguagens e ferramentas foram evidenciados pela abordagem dos conteúdos de maneira espiral, que traz segurança para os estudantes, principalmente. Vale ressaltar que este estudo não tem como intenção comparar linguagens em bloco com linguagens baseadas em texto, mas explorar a evolução dos estudantes em uma abordagem de ensino-aprendizagem que combina estes elementos.

Do ponto de vista da motivação, identificamos como aspectos gerais da abordagem proposta influenciaram a motivação e o aprendizado dos estudantes, tanto positivamente quando negativamente. Na Figura 6.16, apresentamos estes fatores associados a cada uma das categorias do modelo ARCS. Descobrimos que, ao utilizar Scratch nas primeiras aulas da disciplina, a confiança dos estudantes foi potencializada e, ao eliminar a possibilidade de erros de sintaxe, o esforço inicial dos estudantes concentrou-se em praticar as habilidades de lógica e no entendimento dos conceitos básicos. Ao introduzir Python fazendo uso de exemplos que podem ser implementados tanto em Scratch como em Python, os estudantes conseguiram perceber que já têm algum conhecimento sobre conceitos básicos de programação e concentraram seus esforços em aprender detalhes da sintaxe de Python. Também descobrimos que nosso formato de práticas laboratoriais, atividades direcionadas por guias de desafios com dificuldade incremental e *feedback* imediato influenciaram positivamente na motivação dos estudantes, oferecendo confiança para que eles se sintam capazes de realizar as atividades, sem subestimá-los. Além disso, ao aplicar uma abordagem como a nossa, o professor também teve que adotar uma postura menos formal.

Os resultados obtidos sugerem a eficácia da abordagem proposta. No entanto, é preciso reconhecer que não temos uma bala de prata para o problema da aprendizagem e motivação dos estudantes *non-majors*. Ao desenvolver nossa abordagem e analisar os resultados obtidos, consideramos o que a literatura sobre o tema de ensino-aprendizagem de programação tem apresentado ao longo dos últimos anos. Para ensinar programação efetivamente, procuramos entender o que torna a aprendizagem de programação difícil para muitos estudantes e levamos em consideração que o processo de aprendizagem de programação é lento e gradual Jenkins (2002); Dijkstra et al. (1989). Levamos em conta diferentes estilos de estudantes, acreditando que nossa audiência se assemelha mais aos *stoppers* ou *tinkers*, que são estudantes que pouco tentam ou desistem de continuar as atividades quando confrontados com um problema ou a falta de uma direção clara para prosseguir Robins et al. (2003). Consideramos principalmente que cursos alternativos, projetados para acomodar uma variedade de interesses e *background*, oferecem um contexto mais motivador e atraente para a aprendizagem de conceitos de programação e computação Forte and Guzdial (2005). Mesmo assim, não resolvemos todos os problemas em relação à falta de engajamento e às dificuldades dos *non-majors* de nossa instituição. Um

contexto divertido incentiva a participação dos estudantes, mas eles ainda apresentam a percepção de que a programação é difícil de aprender. Isso ficou evidente na Unidade III onde, embora as aulas tenham se mostrado interessantes e mantido a atenção dos estudantes, alguns não conseguiam aplicar os conceitos corretamente e finalizar todas as funcionalidades do projeto de editor de imagens, evidenciando que os conceitos não foram de fácil compreensão. Ainda precisamos melhorar a experiência de aprendizagem oferecida, inclusive aspectos relacionados ao fator regional como o fato de que muitos estudantes demonstrarem dificuldades com linguagens e ambientes de desenvolvimento em inglês.

Finalizamos este trabalho apresentando à comunidade acadêmica dois tipos de resultados. Do ponto de vista pedagógico, apresentamos:

- Uma abordagem de ensino-aprendizagem de programação, adaptada à realidade de turmas de graduação que não são de cursos de TI;
- Sugestões de materiais para as aulas, com slides, projetos e códigos; Sugestões de artefatos para medição da aprendizagem: exercícios, projetos e provas.

Do ponto de vista científico, que é o mais relevante para este trabalho, apresentamos:

- Uma avaliação da metodologia proposta através de uma análise exploratória e aprofundada do problema de pesquisa definido;
- Um esquema de fatores associados as categorias do Modelo ARCS, que podem afetar positiva ou negativamente a motivação dos aprendizes de programação, conforme a Figura 6.16;
- Um conjunto de questionamentos para educadores sobre motivação em uma disciplina introdutória de programação;
- Instrumentos científicos que podem ser utilizados por pesquisadores para replicar ou adaptar a experiência relatada neste estudo.

8.1 Validade e Confiabilidade da Pesquisa

Independente do tipo de pesquisa, a validade e confiabilidade são preocupações que podem ser abordadas através de uma atenção cuidadosa à conceituação do estudo, ao modo como os dados são coletados, analisados e interpretados e ao modo como os achados são apresentados Merriam (2009). Assim, durante as diversas etapas da condução deste trabalho buscamos identificar ameaças potenciais à validade de nossas práticas e minimizá-las. Nesta seção, discutimos a validade e confiabilidade de nossa pesquisa, identificando possíveis ameaças, sob os pontos de vista de Creswell (2010) e Merriam (2009) a respeito deste tema em pesquisas quantitativas, qualitativas e de métodos mistos.

No que diz respeito aos dados qualitativos, procuramos manter procedimentos de confiabilidade, para garantir que abordagem utilizada seja consistente na obtenção

e tratamento desses dados. Para isso, verificamos as transcrições das entrevistas a fim de minimizar erros de entendimento e digitação. Também revisamos toda a codificação a fim de evitar desvios na definição de códigos ou erros no entendimento do significado das passagens.

As ameaças à validade internas são procedimentos, tratamentos ou experiências dos participantes da pesquisa que ameaçam a possibilidade de extrair inferências corretas dos dados Creswell (2010). Ao longo dos estudos de caso, principalmente do estudo de caso Piloto, acontecimentos externos podem ter influenciado indevidamente os resultados. Durante o Estudo de Caso Piloto, a ocorrência de greves e paralisações e o fato de três professores diferentes ministrarem a disciplina teórica podem ter influência sobre os resultados obtidos. Durante o Estudo de Caso Final, a disciplina foi ministrada por apenas um professor que não participou do Estudo de Caso Piloto e que, além disso, é orientador deste trabalho. Outro fator que dificultou a inferência sobre os dados quantitativos foi a alteração realizada nos questionários após o Estudo de Caso Piloto. Embora estas alterações tenham nos permitido uma análise mais aprofundada para o Estudo de Caso Final, retirou a possibilidade de unificação dos dados quantitativos. A fim de minimizar este tipo de ameaça, desconsideramos os dados quantitativos referentes ao Estudo de Caso Piloto.

As ameaças à validade externa dizem respeito à extração de inferências incorretas dos dados para outras pessoas, outros locais, ou outras situações passadas ou futuras Creswell (2010). A fim de minimizar este tipo de ameaça, em nossas análises, evitamos generalizar os resultados para indivíduos que não tenham as características dos participantes, que estejam em locais diferentes ou em situações passadas ou futuras. Uma observação a ser feita é que, em se tratando de estudos de caso com métodos mistos e da natureza exploratória, não é intenção deste trabalho generalizar os resultados para situações diferentes das que foram estudadas. Acreditamos que o valor da pesquisa apresentada reside justamente na descrição específica no contexto específico de uma disciplina de Introdução à Ciência da Computação para *non-majors* com foco no ensino de programação em uma instituição pública de ensino.

Uma linha de investigação, que não foi explorada em nosso estudo, é a motivação dos professores em ensinar ICC. Não fazemos uma investigação para identificar até que ponto a motivação dos professores afeta os resultados dos estudantes. Reconhecemos, no entanto, que isto pode ser uma ameaça à validade de nosso estudo. Outro fator externo que não foi investigado é como o cansaço dos estudantes do final do semestre pode ter influenciado, por exemplo, na queda nos indicadores de motivação da Unidade III.

Com relação a estratégias de validade qualitativas, promovemos a triangulação dos dados, procuramos fazer uma descrição rica e densa para comunicar os resultados, apresentando também as informações discrepantes que foram identificadas. Outra limitação referente à pesquisa está no viés dos pesquisadores envolvidos. A autora deste trabalho é graduada em Engenharia de Computação e não possui experiência em lecionar. Além do viés do pesquisador, ressaltamos também a possibilidade dos

participantes da pesquisa fornecerem respostas enviesadas, uma vez que os objetivos deste estudo foram explicitados no início de cada estudo de caso.

8.2 Trabalhos Futuros

Ao considerar o tema da motivação e aprendizagem dos estudantes *non-majors* em um curso de CS1 empregando nossa abordagem, surgiram algumas questões que podem servir de base para trabalhos futuros:

- Como é a motivação dos estudantes em outros formatos de CS1?
- O que os estudantes aprendem em nossa abordagem é transferível para outros contextos?
- Quão bem a nossa abordagem serve aos estudantes com diferentes *backgrounds*?

Investigamos o tema da motivação em uma abordagem que combina diferentes linguagens e ferramentas. Embora tenhamos oferecido uma descrição do cenário tradicional em nossa instituição, não promovemos um estudo mais aprofundado considerando cada uma das categorias do Modelo ARCS. Os artefatos de coleta elaborados e metodologia de análise deste trabalho, podem ser adaptados para investigar o tema da motivação em outros contextos de CS1. Este tipo de análise torna possível um estudo comparativo mais aprofundado entre os contextos no que diz respeito a motivação.

Analisamos a aprendizagem dos estudantes e identificamos pontos fortes das ferramentas e linguagens utilizadas, mas não analisamos o desempenho dos estudantes em exercícios fora do contexto de nossa abordagem. Outro ponto que não foi nosso objetivo consiste em descobrir se os estudantes que cursaram nossa disciplina aprendem tanto quanto os estudantes que aprenderam a programar com outros conceitos. Percebemos um problema no contexto de mídias com vetores e matrizes. Esses conceitos em específico podem ser alvo de uma investigação mais aprofundada, e os resultados podem ajudar a melhorar a abordagem proposta. Uma investigação mais aprofundada sobre aprendizagem de conceitos de programação também pode ser conduzida no intuito de identificar, mais especificamente, até que ponto as dificuldades apresentadas pelos estudantes são acidentais ou essenciais.

Em nossas investigações, identificamos uma série de fatores relacionados às atitudes dos professores que podem contribuir para aumentar ou diminuir a motivação dos estudantes, mas não investigamos qual a motivação dos professores envolvidos com as disciplinas de ICC. Uma vez que a motivação, ou a falta dela, influencia no comportamento das pessoas em relação a uma dada atividade e, considerando a natureza complexa do ambiente de ensino, convém promover investigações futuras sobre a motivação dos professores e como isso afeta os resultados dos estudantes.

Nos estudos de caso realizados, os participantes da pesquisa eram todos calouros do curso de Engenharia Civil. Esta audiência é restrita a apenas um curso, mas como qualquer outra turma, contém estudantes com uma diversidade de gêneros, etnias e *backgrounds* como, por exemplo, cotistas e não cotistas. Entender a motivação em cada um desses grupos contribui para oferecer uma análise mais aprofundada sobre os efeitos de nossa abordagem.

Diversos caminhos podem ser trilhados trazendo continuidade a esta pesquisa. Consideramos que o trabalho apresentado aqui é apenas um ponto de partida. Além das questões apresentadas, pretendemos aplicar a nossa abordagem a fim de obter amostras quantitativas maiores. Também pretendemos aprofundar os estudos sobre a abordagem em espiral dos conteúdos, que se mostrou um dos principais pontos fortes em ambos os estudos de caso realizados.

Referências Bibliográficas

ACM Inroads, 6(1), 2015. ISSN 2153-2184.

K. K. Agarwal and A. Agarwal. Python for cs1, cs2 and beyond. *Journal of Computing Sciences in Colleges*, 20(4):262–270, 2005.

A. Azzam and Y. Career. Why students drop out cs1 course? In *Proceedings of the Second International Workshop on Computing Education Research*, volume 64, pages 97–108, 2006.

V. Barr and M. Guzdial. Introducing cs to newcomers, and jes as a teaching tool. *Commun. ACM*, 59(11):10–11, Oct. 2016. ISSN 0001-0782. doi: 10.1145/2994590. URL `http://doi.acm.org/10.1145/2994590`.

A. Basawapatna, K. H. Koh, A. Repenning, D. C. Webb, and K. S. Marshall. Recognizing computational thinking patterns. In *Proceedings of the 42Nd ACM Technical Symposium on Computer Science Education*, SIGCSE '11, pages 245–250, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0500-6. doi: 10.1145/1953163.1953241. URL `http://doi.acm.org/10.1145/1953163.1953241`.

M. Ben-Ari. Constructivism in computer science education. *Journal of Computers in Mathematics and Science Teaching*, 20(1):45–74, 2001.

J. Bennedsen and M. E. Caspersen. Failure rates in introductory programming. *ACM SIGCSE Bulletin*, 39(2):32–36, 2007.

S. Bergin and R. Reilly. The influence of motivation and comfort-level on learning to program. 2005.

R. A. Bittencourt, D. M. B. dos Santos, C. A. Rodrigues, W. P. Batista, and H. S. Chalegre. Learning programming with peer support, games, challenges and scratch. In *Frontiers in Education Conference (FIE), 2015. 32614 2015. IEEE*, pages 1–9. IEEE, 2015.

B. Boe, C. Hill, M. Len, G. Dreschler, P. Conrad, and D. Franklin. Hairball: Lint-inspired static analysis of scratch projects. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, SIGCSE '13, pages 215–220, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1868-6. doi: 10.1145/2445196.2445265. URL `http://doi.acm.org/10.1145/2445196.2445265`.

Y. Bosse and M. A. Gerosa. As disciplinas de introdução à programação na usp: um estudo preliminar. In *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*, volume 4, page 1389, 2015.

L. Böszörményi. Why java is notmy favorite first-course language. *Software-Concepts & Tools*, 19(3):141–145, 1998.

K. Brennan and M. Resnick. New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada*, pages 1–25, 2012.

S. S. Brilliant and T. R. Wiseman. The first programming paradigm and language dilemma. *ACM SIGCSE Bulletin*, 28(1):338–342, 1996.

M. M. Burnett. Visual programming. *Wiley Encyclopedia of Electrical and Electronics Engineering*, 1999.

Y. Cherenkova, D. Zingaro, and A. Petersen. Identifying challenging cs1 concepts in a large problem dataset. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, SIGCSE '14, pages 695–700, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2605-6. doi: 10.1145/2538862.2538966. URL `http://doi.acm.org/10.1145/2538862.2538966`.

P. K. Chilana, C. Alcock, S. Dembla, A. Ho, A. Hurst, B. Armstrong, and P. J. Guo. Perceptions of non-cs majors in intro programming: The rise of the conversational programmer. In *Visual Languages and Human-Centric Computing (VL/HCC), 2015 IEEE Symposium on*, pages 251–259. IEEE, 2015.

S. Y. Cho. A web-based tool for teaching computer programming. , 17(4):58–61, 2014.

J. Comer and R. Roggio. Teaching a java-based cs1 course in an academically-diverse environment. In *ACM SIGCSE Bulletin*, volume 34, pages 142–146. ACM, 2002.

S. Cooper and W. Dann. Programming: A key component of computational thinking in cs courses for non-majors. *ACM Inroads*, 6(1):50–54, Feb. 2015. ISSN 2153-2184. doi: 10.1145/2723169. URL `http://doi.acm.org/10.1145/2723169`.

J. W. Creswell. Projeto de pesquisa métodos qualitativo, quantitativo e misto. In *Projeto de pesquisa métodos qualitativo, quantitativo e misto*. Artmed, 2010.

C. Curricula. Acm/ieee-cs joint task force, 2001.

I. F. de Kereki. Scratch: Applications in computer science 1. In *2008 38th Annual Frontiers in Education Conference*, pages T3B–7. IEEE, 2008.

E. W. Dijkstra et al. On the cruelty of really teaching computing science. *Communications of the ACM*, 32(12):1398–1404, 1989.

M. Dorling and D. White. Scratch: A way to logo and python. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, pages 191–196. ACM, 2015.

S. Dziallas, S. Fincher, C. G. Johnson, and I. Utting. A first look at the year in computing. In *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education*, ITiCSE '17, pages 275–280, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4704-4. doi: 10.1145/3059009.3059049. URL `http://doi.acm.org/10.1145/3059009.3059049`.

A. Eckerdal, M. Thuné, and A. Berglund. What does it take to learn'programming thinking'? In *Proceedings of the first international workshop on Computing education research*, pages 135–142. ACM, 2005.

R. J. Enbody, W. F. Punch, and M. McCullen. Python cs1 as preparation for c++ cs2. *ACM SIGCSE Bulletin*, 41(1):116–120, 2009.

S. Fincher, S. Cooper, M. Kölling, and J. Maloney. Comparing alice, greenfoot & scratch. In *Proceedings of the 41st ACM technical symposium on Computer science education*, pages 192–193. ACM, 2010.

A. Forte and M. Guzdial. Computers for communication, not calculation: Media as a motivation and context for learning. In *System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on*, pages 10–pp. IEEE, 2004.

A. Forte and M. Guzdial. Motivation and nonmajors in computer science: identifying discrete audiences for introductory courses. *IEEE Transactions on Education*, 48 (2):248–253, 2005.

M. Goldweber. Programming should not be part of a cs course for non-majors. *ACM Inroads*, 6(1):55–57, Feb. 2015. ISSN 2153-2184. doi: 10.1145/2727128. URL `http://doi.acm.org/10.1145/2727128`.

M. Guzdial. *Introduction to media computation: A multimedia cookbook in Python*. Citeseer, 2004a.

M. Guzdial. Programming environments for novices. *Computer science education research*, 2004:127–154, 2004b.

M. Guzdial. Education: Paving the way for computational thinking. *Commun. ACM*, 51(8):25–27, Aug. 2008. ISSN 0001-0782. doi: 10.1145/1378704.1378713. URL `http://doi.acm.org/10.1145/1378704.1378713`.

M. Guzdial. Exploring hypotheses about media computation. In *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research*, ICER '13, pages 19–26, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2243-0. doi: 10.1145/2493394.2493397. URL `http://doi.acm.org/10.1145/2493394.2493397`.

M. J. Guzdial and B. Ericson. *Introduction to computing and programming in Python, a multimedia approach*. Prentice Hall Press, 2009.

M. Hamada. An integrated virtual environment for active and collaborative e-learning in theory of computation. *IEEE Transactions on Learning Technologies*, 1(2):117–130, 2008.

R. M. Harden. What is a spiral curriculum? *Medical teacher*, 21(2):141–143, 1999.

J. Hromkovič, T. Kohn, D. Komm, and G. Serafini. Combining the power of python with the simplicity of logo for a sustainable computer science education. In *International Conference on Informatics in Schools: Situation, Evolution, and Perspectives*, pages 155–166. Springer, 2016.

W. Huang, W. Huang, H. Diefes-Dux, and P. K. Imbrie. A preliminary validation of attention, relevance, confidence and satisfaction model-based instructional material motivational survey in a computer-based tutorial setting. *British Journal of Educational Technology*, 37(2):243–259, 2006.

T. Jenkins. The motivation of students of programming. In *ACM SIGCSE Bulletin*, volume 33, pages 53–56. ACM, 2001a.

T. Jenkins. The motivation of students of programming. September 2001b. URL `http://kar.kent.ac.uk/13559/`.

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*, volume 4, pages 53–58. Citeseer, 2002.

T. Jenkins. The first language-a case for python? *Innovation in Teaching and Learning in Information and Computer Sciences*, 3(2):1–9, 2004.

A. f. C. M. A. Joint Task Force on Computing Curricula and I. C. Society. *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science*. ACM, New York, NY, USA, 2013. ISBN 978-1-4503-2309-3. 999133.

D. Joyce. The computer as a problem solving tool: a unifying view for a non-majors course. In *ACM SIGCSE Bulletin*, volume 30, pages 63–67. ACM, 1998.

L. C. Kaczmarczyk, E. R. Petrick, J. P. East, and G. L. Herman. Identifying student misconceptions of programming. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education*, SIGCSE '10, pages 107–111, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0006-3. doi: 10.1145/1734263.1734299. URL `http://doi.acm.org/10.1145/1734263.1734299`.

D. T. Kaplan. Teaching computation to undergraduate scientists. In *ACM SIGCSE Bulletin*, volume 36, pages 358–362. ACM, 2004.

C. Kelleher and R. Pausch. Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Computing Surveys (CSUR)*, 37(2):83–137, 2005.

C. Kelleher, R. Pausch, and S. Kiesler. Storytelling alice motivates middle school girls to learn computer programming. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1455–1464. ACM, 2007.

J. Keller. Motivation by design. *Unpublished manuscript, Florida State University, Florida*, 1993.

J. M. Keller. Development and use of the arcs model of instructional design. *Journal of instructional development*, 10(3):2–10, 1987.

J. M. Keller. *Motivational design for learning and performance: The ARCS model approach*. Springer Science & Business Media, 2009.

Y. B.-D. Kolikant. Students' alternative standards for correctness. In *Proceedings of the first international workshop on Computing education research*, pages 37–43. ACM, 2005.

M. Kölling. The greenfoot programming environment. *ACM Transactions on Computing Education (TOCE)*, 10(4):14, 2010.

M. Kuittinen and J. Sajaniemi. Teaching roles of variables in elementary programming courses. *ACM SIGCSE Bulletin*, 36(3):57–61, 2004.

J. Liebenberg, M. Huisman, and E. Mentz. The relevance of software development education for students. *IEEE Transactions on Education*, 58(4):242–248, Nov 2015. ISSN 0018-9359. doi: 10.1109/TE.2014.2381599.

M. C. Linn and M. J. Clancy. The case for case studies of programming problems. *Communications of the ACM*, 35(3):121–132, 1992.

T. J. Long, B. W. Weide, P. Bucci, D. S. Gibson, J. Hollingsworth, M. Sitaraman, and S. Edwards. Providing intellectual focus to cs1/cs2. In *ACM SIGCSE Bulletin*, volume 30, pages 252–256. ACM, 1998.

M. Lutz. Learning python-powerful object-oriented programming: Covers python 2.6 and 3. x ., 2009.

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*, 41:51–61, 2014. ISSN 0747-5632. doi: https://doi.org/10.1016/j.chb.2014.09.012. URL `http://www.sciencedirect.com/science/article/pii/S0747563214004634`.

D. J. Malan and H. H. Leitner. Scratch for budding computer scientists. In *ACM SIGCSE Bulletin*, volume 39, pages 223–227. ACM, 2007.

J. Marks, W. Freeman, and H. Leitner. Teaching applied computing without programming: a case-based introductory course for general education. *ACM SIGCSE Bulletin*, 33(1):80–84, 2001.

B. A. Maxwell and S. R. Taylor. Comparing outcomes across different contexts in cs1. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '17, pages 399–403, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4698-6. doi: 10.1145/3017680.3017757. URL `http://doi.acm.org/10.1145/3017680.3017757`.

S. Merriam. *Qualitative Research: A Guide to Design and Implementation*. Higher and adult education series. John Wiley & Sons, 2009. ISBN 9780470283547. URL `https://books.google.com.br/books?id=tvFICrgcuSIC`.

S. Mishra, S. Balan, S. Iyer, and S. Murthy. Effect of a 2-week scratch intervention in cs1 on learners with varying prior knowledge. In *Proceedings of the 2014 conference on Innovation & technology in computer science education*, pages 45–50. ACM, 2014.

J. Moreno-León, M. Román-González, C. Harteveld, and G. Robles. On the automatic assessment of computational thinking skills: A comparison with human experts. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '17, pages 2788–2795, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4656-6. doi: 10.1145/3027063.3053216. URL `http://doi.acm.org/10.1145/3027063.3053216`.

U. Nikula, O. Gotel, and J. Kasurinen. A motivation guided holistic rehabilitation of the first programming course. *ACM Transactions on Computing Education (TOCE)*, 11(4):24, 2011.

O. L. Oliveira. Statistical evidence of the correlation between mental ability to compute and student performance in undergraduate courses. In *Proceedings of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education*, ITiCSE '12, pages 111–115, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1246-2. doi: 10.1145/2325296.2325326. URL `http://doi.acm.org/10.1145/2325296.2325326`.

N. R. C. U. C. on Information Technology Literacy. *Being fluent with information technology*. National Academies Press, 1999.

D. B. Palumbo. Programming language/problem-solving research: A review of relevant issues. *Review of educational research*, 60(1):65–89, 1990.

S. Papert. *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc., 1980.

A. Pears, S. Seidman, L. Malmi, L. Mannila, E. Adams, J. Bennedsen, M. Devlin, and J. Paterson. A survey of literature on the teaching of introductory programming. *ACM SIGCSE Bulletin*, 39(4):204–223, 2007.

S. Pedroni and N. Rappin. *Jython essentials*. "O'Reilly Media, Inc.", 2002.

A. Radenski. Python first: A lab-based digital introduction to computer science. *ACM SIGCSE Bulletin*, 38(3):197–201, 2006.

M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, et al. Scratch: programming for all. *Communications of the ACM*, 52(11):60–67, 2009.

A. Robins. Learning edge momentum: A new account of outcomes in cs1. *Computer Science Education*, 20(1):37–71, 2010.

A. Robins, J. Rountree, and N. Rountree. Learning and teaching programming: A review and discussion. *Computer science education*, 13(2):137–172, 2003.

R. Savi, C. G. von Wangenheim, and A. F. Borgatto. A model for the evaluation of educational games for teaching software engineering. In *Software Engineering (SBES), 2011 25th Brazilian Symposium on*, pages 194–203. IEEE, 2011.

L. Seiter and B. Foreman. Modeling the learning progressions of computational thinking of primary grade students. In *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research*, ICER '13, pages 59–66, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2243-0. doi: 10.1145/2493394.2493403. URL `http://doi.acm.org/10.1145/2493394.2493403`.

C. Selby and J. Woollard. Computational thinking: the developing definition. 2013. URL `https://eprints.soton.ac.uk/356481/`.

C. C. Selby. Relationships: Computational thinking, pedagogy of programming, and bloom's taxonomy. In *Proceedings of the Workshop in Primary and Secondary Computing Education*, WiPSCE '15, pages 80–87, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3753-3. doi: 10.1145/2818314.2818315. URL `http://doi.acm.org/10.1145/2818314.2818315`.

R. Shackelford, A. McGettrick, R. Sloan, H. Topi, G. Davies, R. Kamali, J. Cross, J. Impagliazzo, R. LeBlanc, and B. Lunt. Computing curricula 2005: The overview report. In *ACM SIGCSE Bulletin*, volume 38, pages 456–457. ACM, 2006.

C. Shannon. Another breadth-first approach to cs i using python. In *ACM SIGCSE Bulletin*, volume 35, pages 248–251. ACM, 2003.

B. Simon, P. Kinnunen, L. Porter, and D. Zazkis. Experience report: Cs1 for majors with media computation. In *Proceedings of the Fifteenth Annual Conference on Innovation and Technology in Computer Science Education*, ITiCSE '10, pages

214–218, New York, NY, USA, 2010a. ACM. ISBN 978-1-60558-820-9. doi: 10.1145/1822090.1822151. URL http://doi.acm.org/10.1145/1822090.1822151.

B. Simon, P. Kinnunen, L. Porter, and D. Zazkis. Experience report: Cs1 for majors with media computation. In *Proceedings of the fifteenth annual conference on Innovation and technology in computer science education*, pages 214–218. ACM, 2010b.

R. H. Sloan and P. Troy. Cs 0.5: a better approach to introductory computer science for majors. In *ACM SIGCSE Bulletin*, volume 40, pages 271–275. ACM, 2008.

E. Soloway. Should we teach students to program? *Communications of the ACM*, 36(10):21–25, 1993.

A. E. Tew, C. Fowler, and M. Guzdial. Tracking an innovation in introductory cs education from a research university to a two-year college. In *ACM SIGCSE Bulletin*, volume 37, pages 416–420. ACM, 2005.

M. Urban-Lurain and D. J. Weinshank. Is there a role for programming in non-major computer science courses? In *Frontiers in Education Conference, 2000. FIE 2000. 30th Annual*, volume 1, pages T2B–7. IEEE, 2000.

E. Vidal Duarte. Teaching the first programming course with python's turtle graphic library. In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*, pages 244–245. ACM, 2016.

A. Vihavainen, J. Airaksinen, and C. Watson. A systematic review of approaches for teaching introductory programming and their influence on success. In *Proceedings of the tenth annual conference on International computing education research*, pages 19–26. ACM, 2014.

C. G. von Wangenheim, R. Savi, and A. F. Borgatto. Deliver!–an educational game for teaching earned value management in computing courses. *Information and software Technology*, 54(3):286–298, 2012.

H. M. Walker. Priorities for the non-majors, cs course: Programming may not make the cut. *ACM Inroads*, 6(1):46–49, Feb. 2015a. ISSN 2153-2184. doi: 10.1145/2727127. URL http://doi.acm.org/10.1145/2727127.

H. M. Walker. Computational thinking in a non-majors cs course requires a programming component. *ACM Inroads*, 6(1):58–61, Feb. 2015b. ISSN 2153-2184. doi: 10.1145/2727126. URL http://doi.acm.org/10.1145/2727126.

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*, pages 39–44. ACM, 2014.

J. Wing. Computational thinking and thinking about computing. In *2008 IEEE International Symposium on Parallel and Distributed Processing*, pages 1–1, April 2008. doi: 10.1109/IPDPS.2008.4536091.

J. M. Wing. Computational thinking. *Communications of the ACM*, 49(3):33–35, 2006.

R. J. Wlodkowski. Motivation and teaching: A practical guide. 1978.

R. K. Yin. *Case study research: Design and methods*. Sage publications, 2013.

D. Zingaro. Peer instruction contributes to self-efficacy in cs1. In *Proceedings of the 45th ACM technical symposium on Computer science education*, pages 373–378. ACM, 2014.

Apêndice A

Termo de Consentimento Livre e Esclarecido

TERMO DE CONSENTIMENTO LIVRE E ESCLARECIDO

Título do projeto: Aprendizagem de Programação de Computadores em Ambientes Lúdicos
Pesquisador responsável: Roberto Almeida Bittencourt
Pesquisadores colaboradores: David M. B. Santos, Carlos A. Rodrigues, Washington P. Batista, Suenny M. Souza, Mirela S. Rios, Lucas C. Morais, Clêison S. Santos, Henderson S. Chalegre, João P. Cardoso, João Paulo S. Sena, Allen H. M. dos Santos, Icaro, Luis, Bianca, Gustavo, Solenir e Ricardo.

Convidamos você para participar desta pesquisa cujo objetivo é aplicar e avaliar a disciplina de Introdução à Ciência da Computação para estudantes de graduação em ciências exatas, e cuja metodologia de ensino-aprendizagem está alicerçada em atividades lúdicas. Dada a importância da programação em cursos de Computação e Ciências Exatas, a avaliação de elementos de tal metodologia é de extrema relevância para mensurar os seus resultados. O levantamento de informações será através da observação das aulas além da aplicação de questionários e entrevistas individuais. Para analisar os dados coletados nestes questionários, usaremos técnicas estatísticas e análise comparativa com outros estudos semelhantes encontrados a partir de revisão bibliográfica, enquanto que para analisar as entrevistas e observações usaremos a análise de conteúdo.

Um potencial benefício de participar desta pesquisa é aprender conhecimentos básicos de programação de forma lúdica, os quais poderão ajudá-lo em sua formação, melhorando seu raciocínio lógico e capacidade de abstração. Um possível risco seria se, por algum motivo, você se sentir constrangido ao responder o questionário ou a entrevista ou ainda ao ser observado. Porém, poderá abandonar a pesquisa a qualquer momento que desejar. De todo modo, estaremos atentos para perceber possíveis desconfortos e fazer propostas para saná-los. Se os mesmos permanecerem, a pesquisa poderá ser interrompida imediatamente sem qualquer tipo de penalidade, inclusive dando continuidade à disciplina. Além disso, garantiremos que o seu anonimato será mantido, respeitando sua integridade intelectual, social e cultural. Neste sentido, o questionário não exige identificação do seu nome ou de qualquer outro documento de identificação. O anonimato também será preservado durante as observações e após as entrevistas.

Não haverá remuneração ou qualquer custo com a participação na pesquisa. A escolha em participar desta pesquisa é livre e, se permitida, pedimos autorização de divulgação dos dados analisados em eventos científicos, lembrando que será mantido sigilo absoluto a respeito de seus dados pessoais. As respostas dos questionários serão tabuladas e comporão um banco de dados para futuras análises histórico-comparativas, porém os questionários respondidos em papel serão mantidos sob responsabilidade do pesquisador responsável no LESS – Laboratório de Engenharia de Software e Sistemas da UEFS, por um período de 5 anos, sendo destruídos logo após. Tais resultados poderão ser acessados assim que disponíveis através deste link: https://sites.google.com/site/robertoabprof/publicacoes. Caso haja qualquer dúvida antes, durante ou depois da realização da pesquisa, você poderá saná-la através do contato do pesquisador responsável, indicado abaixo.

Caso aceite participar desta pesquisa, indique o seu nome completo e assine as duas vias deste termo. Uma cópia será sua e a outra, do pesquisador. Caso você seja menor de idade, é necessário que este termo seja assinado pelo seu responsável legal.

Feira de Santana, _____ de _____ de _____.

_____ _____

Assinatura do participante

Assinatura do pesquisador responsável
Roberto Almeida Bittencourt

_____

Assinatura do responsável legal (se for caso)

Contato com o pesquisador responsável: Departamento de Ciências Exatas. Universidade Estadual de Feira de Santana (UEFS), BR 116, Km 03, Feira de Santana, BA. CEP 44031-460. Telefone: (75) 3161-8086. e-mail: roberto@uefs.br.

Apêndice B

Questionário Pré-intervenção

Questionário 1 ICC (Pré-disciplina) Cód:_____

Instruções: Para cada questão assinale uma alternativa entre as respostas disponíveis

1. **Qual sua idade? _____**

2. **Sexo:** ( ) Masculino ( ) Feminino

3. **Qual sua frequência de uso da Internet?**
   ( ) Todos os dias
   ( ) Quase todos os dias
   ( ) Algumas vezes na semana
   ( ) Algumas vezes no mês
   ( ) Nenhuma vez no mês ou quase nenhuma

4. **Através de que você mais acessa a Internet?**
   ( ) Através de um computador desktop ou notebook
   ( ) Através de celular ou tablet

Quantas vezes você já fez isso?

	Nunca	Poucas vezes	Razoável-mente	Muitas vezes	Sempre
5. Joguei games					
6. Abri um dispositivo (rádio, relógio, computador, telefone, etc.) para descobrir como funciona					
7. Instalei um programa em um computador					
8. Li sobre computadores em livros ou revistas					
9. Baixei uma música da Internet					
10. Utilizei um dicionário, enciclopédia, etc. em um computador					
11. Me comuniquei com família, amigos, colegas, ou professores através de e-mail					
12. Me comuniquei com família, amigos, colegas, professores ou até mesmo desconhecidos através de redes sociais (Facebook, whatsapp etc.)					
13. Busquei informações sobre algo na internet					
14. Me comuniquei com a família ou amigos através do Skype					
15. Escrevi um programa de computador					
16. "Construí" um dispositivo (rádio, relógio, computador, telefone, etc.)					

17. Você já teve aula de programação de computadores? ( ) Sim ( ) Não

Se sim, onde e qual ferramenta ou linguagem de programação foi utilizada?

Para cada uma das afirmativas marque a opção que mais se adequa a seu pensamento.

	Discordo totalmente	Discordo parcialmente	Neutro	Concordo parcialmente	Concordo totalmente
14. Eu sei o que programadores fazem.					
15. Programação de computadores é fácil					
16. Eu acho a profissão de computação divertida					
17. Eu acho a profissão de computação fácil					
18. O que vou aprender nesta disciplina será útil para a vida acadêmica.					
19. O que vou aprender nesta disciplina será útil para a minha carreira.					
20. O que vou aprender nesta disciplina será útil para a minha vida.					

Apêndice C

Questionário Unidade I

QUESTIONÁRIO ICC II – Scratch

Cód._____

Sobre a primeira unidade com o Scratch. Marque a opção que mais se adequa ao seu pensamento.

	Discordo totalmente	Discordo parcialmente	Neutro	Concordo parcialmente	Concordo totalmente
1. Quando eu olhei para o Scratch da primeira vez, eu tive a impressão de que seria fácil para mim.					
2. Havia algo de interessante no início da unidade com o Scratch que chamou a minha atenção.					
3. Scratch foi mais difícil de entender do que eu gostaria que ele fosse.					
4. Depois de ter acesso a informações introdutórias sobre Scratch, senti-me confiante de que eu sabia o que deveria aprender a partir do Scratch.					
5. Completar os exercícios no Scratch me deu uma sensação gratificante de realização.					
6. Está claro para mim como o conteúdo do Scratch está relacionado com coisas que eu já sei.					
7. Muitas das aulas tinham tanta informação que era difícil escolher e lembrar quais os pontos importantes.					
8. O Scratch é atraente.					
9. Houve exemplos que me mostraram como Scratch pode ser importante para as pessoas que estão aprendendo programação.					
10. Completar as atividades no Scratch com êxito foi importante para mim.					
11. Scratch é tão abstrato que era difícil manter a minha atenção nele.					
12. Enquanto eu trabalhava no Scratch, eu estava confiante de que eu poderia aprender o conteúdo.					
13. Eu gostei tanto do Scratch que eu gostaria de saber mais sobre ele.					
14. O design do Scratch me parece pouco atraente.					
15. O conteúdo aprendido no Scratch é relevante para os meus interesses.					
16. A forma como a informação foi organizada na unidade com o Scratch ajudou a manter minha atenção.					
17. Houve explicações e exemplos de como as pessoas usam o conhecimento adquirido com o Scratch.					
18. Os exercícios no Scratch foram muito difíceis.					
19. O Scratch tem coisas que estimularam minha criatividade.					
20. Eu realmente gostei de estudar programação com o Scratch.					
21. Às vezes, a quantidade de repetições feitas sobre alguns assuntos me levou a ficar entediado.					
22. O conteúdo e o estilo do Scratch dão a impressão que vale a pena saber aqueles conceitos.					
23. Eu aprendi algumas coisas que foram surpreendentes ou inesperadas.					
24a. Após trabalhar com o Scratch por um tempo, me senti confiante de que eu seria capaz de passar na primeira unidade.					
24b. A unidade com Scratch não foi relevante para as minhas necessidades, porque eu já sabia a maior parte do assunto.					
25. O feedback oferecido pelo Scratch, ou de observações passadas pelos professores e monitores, ajudou-me a sentir recompensado pelo meu esforço.					
26. Durante as aulas, a variedade de exercícios, ilustrações, etc., ajudou a manter minha atenção.					

27. Eu pude relacionar os conteúdos que aprendi com Scratch com coisas que eu já vi, fiz ou pensei na minha própria vida.					
28. Eu me senti bem ao concluir com êxito os desafios propostos.					
29. O conteúdo visto na unidade com o Scratch será útil para mim.					
30. Eu não pude compreender como algumas coisas eram feitas no Scratch.					
31. A boa organização do conteúdo na primeira unidade ajudou-me a me sentir confiante que eu aprenderia o assunto.					
32. Foi um prazer estudar com a metodologia utilizada na primeira unidade.					

Sobre as aulas práticas da primeira unidade. Marque a opção que mais se adequa ao seu pensamento.

	Discordo totalmente	Discordo parcialmente	Neutro	Concordo parcialmente	Concordo totalmente
33. Acredito que o uso de jogos motivou mais o meu processo de aprendizado.					
34. As atividades desafiadoras me estimulam mais o aprendizado do que quando os instrutores mostram diretamente a solução.					
35. Eu não saberia como realizar as atividades se não houvesse um guia de desafios durante as aulas práticas.					
36. A quantidade de monitores durante as aulas práticas foi suficiente.					

37. Ainda sobre as aulas práticas da primeira unidade. Qual foi a atividade que lhe deixou mais motivado?

( ) Pong ( ) Jogo de Corrida Interlagos

38. Ainda sobre as aulas práticas da primeira unidade. Qual foi a atividade que mais difícil de implementar?

( ) Pong ( ) Jogo de Corrida Interlagos

39. Você comentou com alguém que não é da sua turma de ICC (família, amigos etc.) que estava mexendo numa ferramenta que pode ser usada para fazer jogos e animações?

( ) Sim ( ) Não

Caracterize a facilidade que você teve em aprender os conceitos/recursos a seguir.

	Muito Difícil	Difícil	Regular	Fácil	Muito Fácil
40. Uso dos blocos do grupo de Operadores.					
41. Uso dos blocos do grupo de Controle.					
42. Uso dos blocos do grupo de Movimento					
43. Uso dos blocos do grupo de Aparência					
44. Uso dos blocos do grupo de Sensores					
45. Uso dos blocos do grupo de Variáveis					

46. De um modo geral, como você classifica as aulas da primeira unidade com Scratch em relação aos seguintes aspectos:

Tediosa						**Estimulante**
Cansativa						**Leve**
Didática ruim						**Boa didática**
Não proveitosa						**Proveitosa**
Desorganizada						**Organizada**
Não facilitou o aprendizado						**Facilitou o aprendizado**

Apêndice D

Questionário Unidade II

QUESTIONÁRIO ICC III – Python + Turtle Cód._____

Sobre a segunda unidade com a linguagem Python e a biblioteca Turtle. Marque a opção que mais se adequa ao seu pensamento.

	Discordo totalmente	Discordo parcialmente	Neutro	Concordo parcialmente	Concordo totalmente
1. Quando eu vi Python com Turtle da primeira vez, eu tive a impressão de que seria fácil para mim.					
2. Havia algo de interessante no início da unidade com Python e Turtle que chamou a minha atenção.					
3. Python com Turtle foi mais difícil de entender do que eu gostaria que ele fosse.					
4. Depois de ter acesso a informações introdutórias sobre a linguagem Python e a biblioteca Turtle, senti-me confiante de que eu sabia o que deveria aprender a partir do Scratch.					
5. Completar os exercícios na segunda unidade me deu uma sensação gratificante de realização.					
6. Está claro para mim como o conteúdo abordado está relacionado com coisas que eu já sei.					
7. Muitas das aulas tinham tanta informação que era difícil escolher e lembrar quais os pontos importantes.					
8. O Material de Python com Turtle é atraente.					
9. Houve exemplos que me mostraram como Python com Turtle pode ser importante para as pessoas que estão aprendendo programação.					
10. Completar as atividades em Python com Turtle com êxito foi importante para mim.					
11. Python com Turtle é tão abstrato que era difícil manter a minha atenção neles.					
12. Enquanto eu trabalhava com Python e o Turtle, eu estava confiante de que eu poderia aprender o conteúdo.					
13. Eu gostei tanto de Python com Turtle que eu gostaria de saber mais sobre ele.					
14. O design dos materiais de Python com Turtle me parece pouco atraente.					
15. O conteúdo aprendido com Python e o Turtle é relevante para os meus interesses.					
16. A forma como a informação foi organizada na segunda unidade ajudou a manter minha atenção.					
17. Houve explicações e exemplos de como as pessoas usam o conhecimento adquirido com Python e Turtle.					
18. Os exercícios em Python com Turtle foram muito difíceis.					
19. Python com Turtle tem coisas que estimularam minha criatividade.					
20. Eu realmente gostei de estudar programação com Python e o Turtle.					
21. Às vezes, a quantidade de repetições feitas sobre alguns assuntos me levou a ficar entediado.					
22. O conteúdo e o estilo dos materiais de instrução (exemplos, slides etc.) dão a impressão que vale a pena saber aqueles conceitos.					
23. Eu aprendi algumas coisas que foram surpreendentes ou inesperadas.					
24a. Após trabalhar com Python e o Turtle por um tempo, me senti confiante de que eu seria capaz de passar na segunda unidade.					
24b. A segunda unidade não foi relevante para as minhas necessidades, porque eu já sabia a maior parte do assunto.					
25. O feedback oferecido pelas observações oferecidas pelos professores e monitores, ajudou-me a sentir recompensado pelo meu esforço.					
26. Durante as aulas, a variedade de exercícios, ilustrações, etc., ajudou a manter minha atenção.					
27. Eu pude relacionar os conteúdos que aprendi com Python e o Turtle com coisas que eu já vi, fiz ou pensei na minha própria vida.					
28. Eu me senti bem ao concluir com êxito os desafios propostos.					
29. O conteúdo visto na unidade com Python e o Turtle será útil para mim.					
30. Eu não pude compreender como algumas coisas eram feitas na segunda unidade.					

31. A boa organização do conteúdo na segunda unidade ajudou-me a me sentir confiante que eu aprenderia o assunto.					
32. Foi um prazer estudar com a metodologia utilizada na segunda unidade.					

Sobre a primeira e segunda unidades. Marque a opção que mais se adequa ao seu pensamento.

	Discordo totalmente	Discordo parcialmente	Neutro	Concordo parcialmente	Concordo totalmente
33. Ter a primeira unidade com o Scratch foi inútil, poderia ter ido logo para Python					
34. Sem o Scratch eu teria demorado mais para entender os conceitos de programação com Python.					
35. Python é mais desafiador que o Scratch.					
36. Python é mais útil que o Scratch.					
37. Manipular a caneta do Python é melhor do que manipular a caneta do Scratch.					
38. O fato da linguagem Python usar somente palavras em inglês é um empecilho para mim.					
39. Eu consigo acompanhar o ritmo das aulas teóricas na segunda unidade.					
40. Eu senti dificuldades para fazer os probleminhas extra classe na segunda unidade.					

Sobre as aulas práticas da Segunda unidade. Marque a opção que mais se adequa ao seu pensamento.

	Discordo totalmente	Discordo parcialmente	Neutro	Concordo parcialmente	Concordo totalmente
33. Acredito que o uso de exemplos com desenhos de figuras geométricas motivou mais o meu processo de aprendizado.					
34. As atividades desafiadoras me estimulam mais o aprendizado do que quando os instrutores mostram diretamente a solução.					
35. Eu não saberia como realizar as atividades se não houvesse um guia de desafios durante as aulas práticas.					
36. A quantidade de monitores durante as aulas práticas foi suficiente.					

Caracterize a facilidade que você teve em aprender os conceitos a seguir.	Muito Difícil	Difícil	Regular	Fácil	Muito Fácil
37. Uso dos comandos de movimento da biblioteca Turtle (ex.: forward, backward...)					
38. Uso de estrutura condicional (ex.: if, else, elif)					
39. Uso de estrutura de repetição (ex.: for)					
40. Definição e uso de funções (ex.: def)					
41. Uso de Variáveis					
42. Operadores (ex.: && e || ou)					

43. De um modo geral, como você classifica as aulas da segunda unidade com Python em relação aos seguintes aspectos:

Tediosa						Estimulante
Cansativa						Leve
Didática ruim						Boa didática
Não proveitosa						Proveitosa
Desorganizada						Organizada
Não facilitou o aprendizado						Facilitou o aprendizado

Apêndice E

Questionário Unidade III

QUESTIONÁRIO ICC IV – Python + Mídias Cód._____

Sobre a terceira unidade com a linguagem Python no contexto de manipulação de mídias. Marque a opção que mais se adequa ao seu pensamento.	Discordo totalmente	Discordo parcialmente	Neutro	Concordo parcialmente	Concordo totalmente
1. Quando eu vi o conteúdo da terceira unidade pela primeira vez, eu tive a impressão de que seria fácil para mim.					
2. Havia algo de interessante nas primeiras aulas da terceira unidade, que chamou a minha atenção.					
3. O conteúdo foi mais difícil de entender do que eu gostaria que fosse.					
4. Depois das aulas introdutórias, senti-me confiante de que eu sabia o que era para aprender.					
5. Completar o projeto me deu uma sensação gratificante de realização.					
6. É claro para mim como o conteúdo da terceira unidade está relacionado com as coisas que eu já sei.					
7. Muitas aulas tinham tanta informação que era difícil escolher e lembrar quais os pontos importantes.					
8. O tema da terceira unidade é atraente.					
9. Houve exemplos que me mostrou como o conteúdo da terceira unidade e o JES podem ser importantes para as pessoas que estão aprendendo programação.					
10. Completar o projeto com êxito foi importante para mim.					
11. A qualidade do material utilizado nas aulas ajudou a manter minha atenção.					
12. O conteudo é tão abstrato que era difícil manter a minha atenção nele.					
13. Enquanto eu trabalhei no JES eu estava confiante de que poderia aprender o conteúdo.					
14. Eu gostei tanto do tema que eu gostaria de saber mais sobre imagens e outras mídias.					
15. O design do material de instrução (slides exemplos etc) parece desagradável.					
16. O conteúdo ensinado é relevante para os meus interesses.					
17. A forma como a informação foi organizada na terceira unidade ajudou a manter minha atenção					
18. Houve explicações e exemplos de como as pessoas usam o conhecimento adquirido sobre mídias.					
19. Os exercícios da terceira unidade foram muito difíceis.					
20. O conteúdo da terceira unidade tem coisas que estimula minha criatividade.					
21. Eu realmente gostei de estudar manipulação de imagens.					
22. A quantidade de repetições que eram feitas sobre algumas coisas me levou a ficar entediado, às vezes.					
23. O conteúdo e o estilo de escrita dos materiais de instrução da terceira unidade (slides, apostilas, exemplos etc) deu a impressão que o seu conteúdo a ser abordado na unidade vale a pena conhecer.					
24. Eu aprendi algumas coisas que eram surpreendentes ou inesperadas.					
25. O conteúdo sobre imagens da terceira unidade não é relevante para as minhas necessidades, porque eu já sabia o que estava sendo passado.					
26. O trabalho de feedback após os exercícios, ou de bservações passadas pelos professores e monitores, ajudou-me a sentir recompensado pelo meu esforço.					
27. Durante as aulas, a variedade de exercícios, ilustrações, etc., ajudou a manter minha atenção na aula.					
28. Eu poderia relacionar os conteúdos da unidade com coisas que eu vi, fiz ou penso para minha própria vida.					
29. Eu me senti bem quando concluí com êxito os desafios passados.					
30. O conteúdo visto na terceira unidade com mídias será útil para mim.					
31. Eu realmente não consegui entender nenhum pouco do material de instrução (slides, apostilas, exemplos etc).					
32. A organização do conteúdo me ajudou a ser confiante de que eu conseguiria aprender o conteúdo da terceira unidade					
33. Foi um prazer estudar com a metodologia utilizada na terceira unidade.					

Sobre a terceira unidade Python + Mídias através do ambiente JES. Marque a opção que mais se adequa ao seu pensamento.	Discordo totalmente	Discordo parcialmente	Neutro	Concordo parcialmente	Concordo totalmente
34. O Python é mais útil que o Scratch.					
35. Trabalhar com Mídias foi mais interessante do que trabalhar com o Turtle					
36. Tive mais facilidade de usar o ambiente JES do que o IDLE					
37. O fato da linguagem Python e do ambiente JES usarem somente palavras em inglês foi um empecilho para mim.					
38. Eu consegui acompanhar o ritmo das aulas.					
39. Eu senti facilidade em fazer os projetos fora da aula.					

Sobre as aulas práticas da terceira unidade. Marque a opção que mais se adequa ao seu pensamento.	Discordo totalmente	Discordo parcialmente	Neutro	Concordo parcialmente	Concordo totalmente
40. Acredito que os projetos motivaram mais o meu processo de aprendizagem.					
41. As atividades desafiadoras me estimulam mais a aprendizagem do que quando os instrutores mostram diretamente a solução.					
42. As descrições contidas no roteiro dos projetos ajudaram a entender como o trabalho deveria ser feito.					
43. A quantidade de monitores durante as aulas práticas foi suficiente.					

44. Você comentou com alguém que não é da sua turma de ICC (família, amigos etc.) que estava aprendendo como manipular imagens?

( ) Sim ( ) Não

Caracterize a facilidade que você teve em aprender os conceitos a seguir.	Muito Difícil	Difícil	Regular	Fácil	Muito Fácil
45. Uso de estrutura condicional (ex.: if, else, elif)					
46. Uso de estrutura de repetição (ex.: for)					
47. Definição e uso de funções simples (ex.: def exibirMenu( ):)					
48. Definição e uso de funções com parâmetro (ex.: def salvarImagens(imagens):)					
49. Definição e uso de funções com retorno (ex.: imagem = carregarImagem():)					
50. Uso de Variáveis					
51. Operadores (ex.: and e or)					
52. Vetor (ex.: vetor_imagens[3] = imagem)					
53. Matiz (ex.: colagem[0][1] = imagem)					

54. De um modo geral, como você classifica as aulas da terceira unidade com Python e Mídias em relação aos seguintes aspectos:

Tediosa						Estimulante
Cansativa						Leve
Didática ruim						Boa didática
Não proveitosa						Proveitosa
Desorganizada						Organizada
Não facilitou o aprendizado						Facilitou o aprendizado

Para cada uma das afirmativas marque a opção que mais se adequa a seu pensamento.	Discordo totalmente	Discordo parcialme	Neutro	Concordo parcialme	Concordo totalmente
55. Eu sei o que programadores fazem.					
56. Programação de computadores é fácil					
57. Eu acho a profissão de computação divertida					
58. Eu acho a profissão de computação fácil					
59. O que vou aprender nesta disciplina será útil para a vida acadêmica.					
60. O que vou aprender nesta disciplina será útil para a minha carreira.					
61. O que vou aprender nesta disciplina será útil para a minha vida.					

Apêndice F

Course Interest Survey - CIS

Course Interest Survey

Cód._____

Existem 34 itens neste questionário. Por favor, pense em cada declaração em relação a disciplina de ICC e indique o quão verdadeira cada declaração é. Dê a resposta que realmente se aplica a você, e não o que você gostaria que fosse verdade, ou o que você acha que outros querem ouvir. Pense em cada afirmação por si só e indique quão verdade ela é. Não seja influenciado pelas respostas dadas para outras declarações.

	Não é verdade	Um pouco verdadeiro	Moderadamente verdadeiro	Principalmente verdadeiro	Muito verdadeiro
1. O professor sabe como nos fazer sentir entusiasmados com o assunto desta disciplina.					
2. As coisas que estou aprendendo nesta disciplina serão úteis para mim.					
3. Eu me sinto confiante de que vou me dar bem nesta disciplina.					
4. Esta disciplina tem muito pouca coisa que capta minha atenção.					
5. O professor faz com que o assunto desta disciplina pareça importante.					
6. Você tem que ter sorte para obter boas notas nesta disciplina.					
7. Eu tenho que trabalhar muito para ter sucesso nesta disciplina.					
8. NÃO vejo como o conteúdo desta disciplina se relaciona com qualquer coisa que eu já conheça.					
9. Ter ou não ter sucesso neste disciplina depende de mim.					
10. O professor cria suspense ao desenvolver um assunto.					
11. O assunto desta disciplina é difícil demais para mim.					
12. Eu sinto que esta disciplina me dá muita satisfação.					
13. Nesta disciplina procuro estabelecer e alcançar altos padrões de excelência.					
14. Eu sinto que as notas ou outro reconhecimento que recebo são justos em comparação com outros alunos.					
15. Os alunos desta disciplina parecem curiosos sobre o conteúdo.					
16. Eu gosto de estudar para esta disciplina.					
17. É difícil prever a nota que o professor dará às minhas tarefas.					
18. Estou satisfeito com as avaliações que o professor fez do meu trabalho em comparação com o quão bem eu acho que fui.					
19. Eu me sinto satisfeito com o que estou aprendendo nesta disciplina.					
20. O conteúdo desta disciplina relaciona-se com as minhas expectativas e objetivos.					
21. O professor faz coisas incomuns ou surpreendentes que são interessantes.					
22. Os alunos participam ativamente das aulas.					
23. Para realizar meus objetivos, é importante que eu vá bem nesta disciplina.					
24. O professor usa uma variedade interessante de técnicas de ensino.					
25. Eu NÃO acho que vou me beneficiar muito desta disciplina.					
26. Costumo sonhar acordado (ficar distraído) enquanto estou nas aulas desta disciplina.					
27. Por estar fazendo esta disciplina, acredito que posso ter sucesso se eu me esforçar o bastante.					
28. Os benefícios pessoais desta disciplina estão claros para mim.					
29. Minha curiosidade é muitas vezes estimulada por questões feitas ou por problemas dados sobre o conteúdo desta disciplina.					
30. Eu acho razoavelmente correto o nível de desafio nesta disciplina: nem muito fácil nem muito difícil.					
31. Eu me sinto bastante desapontado com esta disciplina.					
32. Eu sinto que recebo reconhecimento suficiente do meu trabalho nesta disciplina através das notas, comentários ou outros feedbacks.					
33. A quantidade de trabalho que tenho que fazer é apropriada para esse tipo de disciplina.					
34. Recebo feedback suficiente para saber quão bem estou indo.					

Apêndice G

ROTEIRO DE ENTREVISTA ICC – I

São questões e Hipóteses de Pesquisa:

1. Como a abordagem proposta influencia na motivação dos estudantes?
2. A abordagem proposta ajuda a reduzir os índices abandono e reprovação?
3. A motivação dos estudantes é mantida ao longo de toda a disciplina ou muda a partir do momento em que os exemplos param de utilizar games?
4. A análise dos resultados das observações reforça as respostas dadas pelos estudantes no questionário sobre motivação?
5. O nível de entendimento dos conceitos básicos (loops, vetores etc) apresentado pelos estudantes nos testes de medição de aprendizagem está de acordo com a porcentagem de aprovações nas disciplinas ofertadas?

Roteiro de entrevista:

1. Antes dessa disciplina, você já havia tido aula ou estudado programação? Conte um pouco da sua experiência.
2. Você faltou ou se atrasou em algum dia das aulas? Você sente que isso lhe prejudicou?
3. Essa disciplina tem sido o que você achou que seria? (Probe: se ele tinha uma opinião formada sobre os conteúdos que iria ver e teve uma surpresa)
4. O que você achou de usar Scratch? Por que?
5. Como você avalia a construção de jogos durante as aulas da primeira unidade? (Probe: você chegou a mostrar seus projetos no Scratch para algum conhecido?)
6. De qual projeto você mais gostou de fazer no Scratch? Por que?

7. Qual projeto você achou mais desafiador? Por que?
8. Você conseguiu terminar todos os projetos durante as aulas práticas? (Probe: Você conseguiu fazer os projetos do Scratch com facilidade? quais suas dificuldades...)
9. Teve alguma coisa que você não conseguiu fazer? Por que? (Probe: o que ele achou mais difícil, por ex: fazer a nave morrer; tentar identificar qual o comando)
10. O que você achou das aulas teóricas?
11. O que você achou das aulas práticas?
12. O que você achou da avaliação da primeira unidade?
13. Como você estudou para esta disciplina na primeira unidade?
14. Como você acha que a primeira unidade poderia ser melhorada?
15. Você gostaria de falar mais alguma coisa?

Apêndice H

ROTEIRO DE ENTREVISTA ICC – II

Questões e Hipóteses de Pesquisa:

1.
2. Como a abordagem proposta influencia na motivação dos estudantes?
3. A abordagem proposta ajuda a reduzir os índices abandono e reprovação?
4. A motivação dos estudantes é mantida ao longo de toda a disciplina ou muda a partir do momento em que os exemplos param de utilizar games?
5. A análise dos resultados das observações reforça as respostas dadas pelos estudantes no questionário sobre motivação?
6. O nível de entendimento dos conceitos básicos (loops, vetores etc) apresentado pelos estudantes nos testes de medição de aprendizagem está de acordo com a porcentagem de aprovações nas disciplinas ofertadas?

Roteiro de entrevista:

1. Você faltou ou se atrasou em algum dia das aulas? Você sente que isso lhe prejudicou? Se sim de que forma?
2. O que você achou da linguagem de programação Python?
3. Como você avalia a construção de programas para desenhos com a biblioteca Turtle durante as aulas da segunda unidade?
4. De que maneira o Scratch ajudou (ou atrapalhou) no entendimento de Python?
5. Quais conceitos você viu nas duas linguagens e percebeu que podem ser usados de maneira semelhante? (Probe: pode citar comandos ou funcionalidades específicas)

6. O que você achou das aulas teóricas?
7. O que você achou das aulas práticas?
8. O que você achou da avaliação da segunda unidade?
9. O que você achou dos Desafios extra classe?
10. Qual exercício das aulas práticas você achou mais desafiador?
11. Você conseguiu terminar todos os desafios durante as aulas práticas? (Probe: quais as facilidades e quais suas dificuldades...)
12. Teve alguma coisa que você não conseguiu fazer? Por que? (Probe: o que ele achou mais difícil)
13. Como você estudou para esta disciplina na segunda unidade?
14. Como você acha que a segunda unidade poderia ser melhorada?
15. Você gostaria de falar mais alguma coisa?

Apêndice I

ROTEIRO DE ENTREVISTA ICC – III

Sobre a Unidade III:

1. O que você achou da linguagem de programação Python?
2. O que você achou do ambiente JES?
3. Como você avalia a construção de programas para a edição de imagens durante as aulas da terceira unidade?
4. O que você achou das aulas teóricas da terceira unidade?
5. O que você achou das aulas práticas da terceira unidade?
6. O que você achou dos projetos da terceira unidade?
7. Qual dos projetos você achou mais desafiador?
8. Você conseguiu terminar todos os projetos no prazo? (Probe: quais as facilidades e quais suas dificuldades...)
9. Teve alguma coisa que você não conseguiu fazer? Por que? (Probe: o que ele achou mais difícil)
10. Como você estudou para esta disciplina na terceira unidade?
11. Como você acha que a terceira unidade poderia ser melhorada?

Sobre a disciplina:

1. O que você acha da utilização de jogos para aprender a programar? Você costuma fazer "uso" de jogos?
2. O que você acha da utilização de mídias para aprender a programar?

3. Na sua avaliação, qual foi a melhor unidade? Por quê? (Probe: verificar se a ferramenta preferida do estudante coincide com a ferramenta da unidade escolhida.)
4. Já houve casos em que você ficou entediado com a aula? Se sim, com que frequência isso aconteceu? Como você agiu?
5. O que você pensa sobre a construção de exemplos (passo a passo) durante a aula teórica?
6. O que te leva a prestar atenção em uma aula teórica?
7. No laboratório, em que situação você se sente confiante a fazer os exercícios que são solicitados?
8. O que você pensa sobre as atividades de laboratório serem realizadas individualmente?
9. O que você pensa sobre o guia de exercícios do laboratório? (Com questões, cujo objetivo e dicas são bem definidos)
10. Qual a sua opinião a acerca do feedback imediato após a correção da sua avaliação?
11. Em quais situações você se sentiu mais "animado" com a disciplina?
12. Em algum momento da disciplina você pensou em desistir da mesma? Se houve, o que você fez?
13. Na sua opinião, de que forma esta disciplina será útil na sua profissão (ou até mesmo durante a graduação)?
14. Na sua opinião, para que a disciplina pudesse ser aprimorada, o que você acrescentaria ou modificaria?
15. Você gostou de aprender a programar? Por quê?
16. Em uma outra oportunidade, caso fosse oferecida alguma disciplina avançada nos mesmos moldes de ICC, você cursaria? Por quê?
17. Você teria mais algum detalhe a acrescentar?

Apêndice J

Descrição Unidade I

UNIDADE I **Conceitos básicos de programação com a ferramenta Scratch**			
Aula	Objetivos de Aprendizagem	Conteúdos Abordados	Atividades realizadas
01 TEO	- Entender o que é programação e sua importância na Ciência da Computação; - Conhecer a interface do Scratch; - Conhecer os comandos básicos do Scratch.	- O que é Ciência da Computação, programação e linguagem de programação; - Comandos básicos do Scratch; -Fluxo de execução de um programa no Scratch.	Aula expositiva: Exemplos utilizando a caneta do Scratch.
02 TEO	- Identificar os elementos principais de um jogo; - Utilizar os comandos do Scratch para construção de jogos; - Ser capaz de implementar a animação de um personagem através da troca de *sprites*.	- Aparência; - Sensores; - Controle; - Estruturas de seleção; - Estruturas de repetição; - Variáveis.	Aula expositiva: Jogo Arco e Flecha.
03 LAB	- Ser capaz de construir um jogo a partir de especificações de suas funcionalidades e pequenas dicas de implementação; - Ser capaz de implementar a pontuação em um jogo através de uma variável contadora.	- Aparência; - Sensores; - Controle; - Entrada e saída; - Estruturas de seleção; - Estruturas de repetição; - Variáveis.	Prática Laboratorial: Jogo Pong.
04 TEO	- Identificar os elementos principais de um jogo; - Utilizar os comandos do Scratch para construção de jogos; - Aprender o conceito de estrutura condicional e o conceito de estrutura de repetição; - Entender como é feita a movimentação e animação de personagens em um jogo.	- Aparência; - Controle; - Entrada e saída; - Estruturas de seleção; - Estruturas de repetição.	Aula expositiva: Space Invaders (Parte I).
05 LAB	- Ser capaz de implementar a comunicação entre objetos; - Ser capaz de gerenciar a atualização de múltiplas variáveis; - Ser capaz de utilizar variáveis temporais.	- Comunicação entre objetos; - Variáveis; - Operadores lógicos e aritméticos; - Estrutura de repetição (for).	Prática Laboratorial: Jogo Interlagos.
06 TEO	- Aprender o conceito de variáveis e como utilizá-las no Scratch; - Aprender as diversas maneiras de utilizar os sensores do Scratch.	- Aparência; - Sensores; - Controle; - Entrada e saída; - Estruturas de repetição; - Variáveis; - Comunicação entre objetos.	Aula expositiva: Space Invaders (Parte II).
07 LAB			Avaliação Prática

Apêndice K

Descrição Unidade II

UNIDADE II Programação em Python com a Biblioteca Turtle			
Aula	Objetivos de Aprendizagem	Conteúdos Abordados	Atividades realizadas
01 TEO	- Conhecer a linguagem Python, a biblioteca Turtle e o ambiente de desenvolvimento integrado IDLE; - Ser capaz de implementar programas para desenho de polígonos regulares;	- Comandos Turtle; - Estruturas de repetição;	Aula expositiva: Comparação Python e Scratch.
02 TEO	- Ser capaz de utilizar estruturas de repetição; - Entender o conceito de função e sua aplicação; - Ser capaz de utilizar parâmetros em funções.	- Comandos Turtle; - Estrutura de repetição (for); - Estruturas condicionais (if); - Variáveis (variável lógica); - Funções com uso de parâmetros.	Aula expositiva: Figuras geométricas.
03 LAB	- Ser capaz de criar programas simples em Python para desenho de figuras geométricas através do uso de funções; - Ser capaz de criar programas em Python para desenho que combinam polígonos; - Ser capaz de implementar funções com passagem de parâmetro; - Ser capaz de utilizar estrutura condicional if com variáveis lógicas;	- Comandos Turtle; - Estrutura de repetição (for); - Funções simples - Estrutura de repetição (for); - Estruturas condicionais (if); - Variáveis (variável lógica); - Funções com uso de parâmetros.	Prática Laboratorial: Figuras geométricas e combinações de figuras geométricas.
04 TEO	- Entender as funções com passagem de parâmetro; - Ser capaz de utilizar estrutura condicional (if, else e elif....); - Ser capaz de utilizar variáveis booleanas, textuais e numéricas.	- Estruturas condicionais (if, else, elif...); - Variáveis lógicas, textuais e numéricas; - Números randômicos; - Funções com uso de parâmetros.	Aula expositiva: Figuras com efeitos visuais.
05 LAB	- Ser capaz de criar programas em Python para desenho de figuras com efeitos visuais; - Ser capaz de utilizar estruturas condicionais corretamente; - Ser capaz de utilizar variáveis booleanas, textuais e numéricas.	- Estruturas condicionais (if, else, elif...); - Variáveis lógicas, textuais e numéricas; - Números randômicos; - Funções com uso de parâmetros.	Prática laboratorial: Estrelas e caleidoscópios.
06 TEO	- Conhecer os diferentes tipos de variáveis e como elas são codificadas pelo computador; - Entender a diferença entre declaração e chamada de uma função em Python.	- Variáveis lógicas, textuais e numéricas; - Funções com uso de parâmetros	Aula expositiva: exemplos diversificados.
07 LAB			Avaliação Prática

Apêndice L

Descrição Unidade III

UNIDADE III Programação em Python com mídias através do ambiente de desenvolvimento JES			
Aula	Objetivos de Aprendizagem	Conteúdos Abordados	Atividades realizadas
01 TEO	- Conhecer o ambiente JES; - Aprender a criar e executar programas no JES; - Aprender a utilizar a entrada e saída do JES.	- Estrutura Condicional (if); - Estrutura de repetição (while); - Funções; - Variáveis; - Entrada e Saída do JES.	Aula expositiva: jogo adivinhação.
02 TEO	- Ser capaz de utilizar a entrada e saída do JES; - Ser capaz de implementar programas em linguagem Python que utilizam entrada e saída de dados;	- Estrutura Condicional (if); - Estrutura de repetição (while); - Funções; - Variáveis; - Entrada e Saída do JES.	Aula expositiva: jogo enquete.
03 LAB	- Ser capaz de implementar programas em linguagem Python que utilizam estruturas de repetição e comandos de entrada e saída.	- Estrutura Condicional (if); - Estrutura de repetição (while); - Funções; - Variáveis; - Entrada e Saída do JES.	Prática laboratorial: Programa para cálculo de áreas.
04 TEO	- Entender como arquivos estão armazenados no computador; - Entender o conceito de vetores e matrizes. - Carregar e abrir arquivos.	- Vetores; - Matrizes; - Funções com parâmetros e retorno.	Aula expositiva: Carregando imagens e sons no programa.
05 TEO	- Entender como o computador codifica uma imagem; - Ser capaz de relacionar a estrutura de uma imagem com a de uma matriz; - Entender a composição de cor das imagens; - Ser capaz de implementar uma função que faça varredura de uma imagem pixel a pixel (utilizando o conceito de matriz).	- Codificação de imagens; - Modelos de cor; - Funções de manipulação de imagens do JES; - Estrutura de repetição (loops aninhados); - Matriz.	Aula expositiva: Varrer cada pixel de uma imagem.
06 LAB	- Ser capaz de escrever e executar programas no JES; - Ser capaz de ler a ajuda do JES para identificar a descrição das funções e utilizá-las; - Ser capaz de escrever programas que carreguem arquivos de imagem e som a partir de qualquer pasta do computador.	- Funções para carregar e abrir arquivos; - Funções com parâmetros e retorno; - Manipulação de Matriz.	Prática laboratorial: Gerador de Postagens.
07 TEO	- Entender como filtros de cor de imagens são implementados; - Entender como elementos de texto e desenhos são adicionados a imagens; - Saber implementações com loops aninhados.	- Modelos de cor; - Matriz; - Funções para manipulação de imagens do JES; - Entrada e Saída do JES; - Estrutura de repetição (while).	Aula expositiva: preto e branco; Escala de cinza; negativo; remoção de R, G ou B; adição de texto e borda.
08 LAB	- Ser capaz de implementar um programa com múltiplas tarefas a partir de especificação de projeto; - Ser capaz de implementar programas que carreguem pixels de uma imagem e modifique a composição de cor; - Se capaz de implementar um programa com entrada e saída de dados e menu inicial.	- Codificação de imagens; - Modelos de cor; - Matriz; - Entrada e saída do JES; - Estrutura de repetição (while).	Prática laboratorial: apresentação do projeto.

09 TEO	- Entender como a estrutura de uma imagem pode ser alterada; - Ser capaz de acessar cada elemento de uma matriz através de loops aninhados.	- Matriz; - Entrada e saída do JES; - Estrutura de repetição - Estrutura de repetição (loops aninhados e while).	Aula expositiva: redimensionamento, espelhamento.
10 LAB	- Entender como a estrutura de uma imagem pode ser alterada; - Ser capaz de acessar cada elemento de uma matriz através de loops aninhados.	- Modelos de cor; - Funções de manipulação de imagens do JES; - Matriz; - Entrada e saída do JES; - Estrutura de repetição - Estrutura de repetição (loops aninhados e while).	Prática laboratorial: Implementação do projeto
11 TEO	- Ser capaz de acessar cada elemento de uma matriz através de loops aninhados;	- Matriz; - Entrada e saída do JES; - Estrutura de repetição.	Aula expositiva: Colagens.
12 LAB	- Ser capaz de implementar um programa com múltiplas tarefas a partir da especificação de um projeto; - Ser capaz de utilizar loops aninhados; - Ser capaz de implementar funções que modifiquem propriedades de cor e estrutura de uma imagem.	- Modelos de cor; - Funções de manipulação de imagens do JES; - Matriz; - Entrada e saída do JES; - Estrutura de repetição (loops aninhados e while).	Prática laboratorial: Implementação do Projeto (Final).
13 TEO	- Ser capaz de alterar a estrutura de uma imagem para criação de efeitos como chromakey e extração de background; - Aprimorar suas habilidades de manipulação de matriz.	- Modelo de cor RGB; - Funções de manipulação de imagens do JES; - Matriz; - Entrada e Saída do JES.	Aula expositiva: Extração de background e chromakey.