Universidade Estadual de Feira de Santana

Programa de Pós-Graduação em Computação Aplicada

Avaliação de Uma Abordagem de Aprendizagem Baseada em Problemas no Ensino de Programação Orientada a Objetos

Ayala Lemos Ribeiro

Feira de Santana

2019

Universidade Estadual de Feira de Santana

Programa de Pós-Graduação em Computação Aplicada

Ayala Lemos Ribeiro

Avaliação de Uma Abordagem de Aprendizagem Baseada em Problemas no Ensino de Programação Orientada a Objetos

Dissertação de Mestrado apresentada no Programa de Pós-Graduação em Computação Aplicada como parte dos requisitos para a obtenção do título de Mestre em Computação Aplicada.

Orientador: Roberto Almeida Bittencourt

Feira de Santana

2019

Ficha catalográfica - Biblioteca Central Julieta Carteado - UEFS

Ribeiro, Ayala Lemos
R367a Avaliação de uma abordagem de aprendizagem baseada em problemas no ensino de Programação Orientada a Objetos / Ayala Lemos Ribeiro. - 2019.
131f. : il.

Orientador: Roberto Almeida Bittencourt
Dissertação (mestrado) - Universidade Estadual de Feira de Santana. Programa de Pós-Graduação em Computação Aplicada, 2019.

1. Programação Orientada a Objetos (computação). 2. Estrutura de dados (computação). 3. Aprendizagem baseada em problemas (PBL). 4. Computação – Ensino – Motivação. 5. Modelo ARCS (motivação). I. Bittencourt, Roberto Almeida, orient. II. Universidade Estadual de Feira de Santana. III. Título.

CDU:.004..42

Rejane Maria Rosa Ribeiro – Bibliotecária CRB-5/695

Ayala Lemos Ribeiro

Avaliação de Uma Abordagem de Aprendizagem Baseada em Problemas no Ensino de Programação Orientada a Objetos

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, 11 de setembro de 2019

BANCA EXAMINADORA

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

Dra. Claudia Pinto Pereira
Universidade Estadual de Feira de Santana

Dra. Ayla Debora Dantas de Souza Rebouças
Universidade Federal da Paraíba

Abstract

Problems related to learning programming in undergraduate computing programs contribute to lack of motivation, failure and early abandonment of these programs. The motivation of computing students in programming courses is a relevant factor that may contribute to success in learning. This work evaluates students' motivation and learning in a teaching-learning approach to object-oriented programming in the Computer Engineering program at the State University of Feira de Santana, in an integrated curricular component of Programming of the second term of this program that uses the problem-based learning approach (PBL). This integrated component was organized into three theoretical modules addressed through lectures and an integrator module that followed the PBL dynamics. Results describe the levels of motivation in terms of attention, relevance, confidence and satisfaction, and learning in terms of student performance, all measured in both the theoretical modules and the PBL integrator module. In addition, this case study has led us to important lessons learned: the acquisition of personal, interpersonal and technical skills provided by the approach; the benefits of integrating knowledge through more authentic experiences and a more disciplined practice of software production; the need for careful problem planning; the main difficulties faced by instructors to manage the course; the challenges students face to develop their skills; and and confidence as a prime motivational factor for learning. This work contributes to the knowledge in the field of computing education, more specifically in evaluating motivation and learning of computing students who learn through the PBL approach, potentially allowing to advance in the dissemination and effectiveness improvement of this approach in the field of computing.

Keywords: Object-Oriented Programming, Data Structures, Software Design, PBL, Motivation, ARCS Model, Learning.

Resumo

Problemas relacionados à aprendizagem de programação em cursos de computação contribuem para desmotivação, reprovação e abandono precoce destes cursos. A motivação de estudantes da área de computação em disciplinas de programação é um fator relevante que pode contribuir para o sucesso no aprendizado. Este trabalho avalia a motivação e a aprendizagem dos estudantes em uma abordagem de ensino-aprendizagem de programação orientada a objetos no curso de Engenharia de Computação da Universidade Estadual de Feira de Santana, em um componente curricular integrado de Programação Orientada a Objetos, Estruturas de Dados e Projeto de Sistemas do segundo semestre do curso que utiliza a metodologia de Aprendizagem Baseada em Problemas (PBL). Este componente integrado foi organizado em três módulos teóricos abordados através de aulas expositivas dialogadas e em um módulo integrador que seguiu a dinâmica da metodologia PBL. Os resultados descrevem os níveis de motivação em termos de atenção, relevância, confiança e satisfação e de aprendizagem em termos do desempenho dos estudantes, todos mensurados tanto nos módulos teóricos como no módulo integrador PBL. Além disso, a experiência levou a importantes lições aprendidas com a abordagem: a aquisição de habilidades pessoais, interpessoais e técnicas fornecidas pela abordagem; os benefícios da integração do conhecimento através de experiências mais autênticas e uma prática mais disciplinada de produção de software; a necessidade de planejamento cuidadoso de problemas; as principais dificuldades enfrentadas pelos instrutores para gerenciar o curso; os desafios enfrentados pelos alunos para desenvolver suas habilidades; e a confiança como fator motivacional primordial para a aprendizagem. Este trabalho contribui para o conhecimento na área de educação em computação, mais especificamente na avaliação da motivação e da aprendizagem de estudantes de computação que aprendem através da metodologia PBL, permitindo potencialmente avançar na disseminação e melhoria da eficácia desta metodologia na área de computação.

Palavras-chave: Programação Orientada a Objetos, Estruturas de Dados, Projeto de Sistemas, Integração curricular, PBL, Motivação, Aprendizagem, 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**.

Agradecimentos

Não pensei que fosse tão difícil agradecer. Na vida não temos o teste *post hoc* de Tukey para mensurarmos a diferença entre a importância das pessoas, e nem o "valor p"para indicar o nível de significância de cada uma delas.

Primeiramente, gostaria de agradecer a Deus por me guiar e me dar tranquilidade para seguir em frente e nunca desanimar com os obstáculos que apareceram ao longo do caminho.

Agradeço a meu filho Davi (*in memoriam*) pelo exemplo de força e resiliência. Por me ensinar que devemos prosseguir quando tudo parece perdido. Por suportar os momentos difíceis, resistindo ao cansaço, e superando as adversidades com serenidade. E por ser meu grande incentivador. Amor eterno!!! Um dia vamos nos reencontrar...

Aos meus pais Jocelino e Joice, em especial a minha mãe, que sempre esteve ao meu lado, me aconselhando e incentivando. Profissional dedicada, sempre foi o meu exemplo. Com frases encorajadoras: "Está chegando o momento filha, você vai conseguir!". Obrigada mãe pelo apoio incondicional.

Ao meu esposo amado Franclin, apoio irrestrito nessa caminhada. Obrigada por suportar as minhas ausências em vários momentos, viagens, finais de semana e por estar sempre presente. Você é um grande parceiro, meu amor!

Um agradecimento especial ao meu querido orientador, professor Roberto Bittencourt. Sempre muito paciente, cuidadoso e atento a todas as questões, àquelas que dizem respeito ao trabalho, como também às pessoais. Lembro-me quando o via circular pela universidade, já tinha profunda admiração pelo seu trabalho, e pensava: "Ah como eu queria que ele fosse meu orientador!". Muito obrigada professor! Tenho grande respeito pelo seu trabalho e também pelo ser humano que és.

Também quero agradecer a minha família amada, minha irmã Aline, a inesquecível vovó Juce (in memorian), Maurício, tios, primos. E também aos colegas, Naan, Jefferson, Luis Gustavo, Ivan, que de alguma forma contribuíram para que esse projeto fosse realizado.

Enfim, a todos, os meus sinceros agradecimentos.

Sumário

Lista de Publicações

Ribeiro, Ayala L., and Bittencourt, Roberto A. "A Case Study of an Integrated Programming Course Based on PBL.". In *2019 IEEE Frontiers in Education Conference (FIE)*. IEEE, 2019.

Ribeiro, Ayala L., and Bittencourt, Roberto A. "A PBL-Based, Integrated Learning Experience of Object-Oriented Programming, Data Structures and Software Design.". *2018 IEEE Frontiers in Education Conference (FIE)*. IEEE, 2018.

Ribeiro, Ayala L., Bittencourt, Roberto A. and Santana, Bianca L. "Análise da Motivação em um Estudo Integrado de Programação Baseado em PBL.". In *26º Workshop sobre Educação em Computação (WEI 2018)*. SBC, 2018.

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

Ao longo dos anos, desde que se identificou o potencial dos computadores e das tecnologias de comunicação como ferramenta pedagógica, vem sendo desenvolvida uma área de investigação interdisciplinar conhecida como Computação na Educação. Os grupos de pesquisa nessa área estão presentes em praticamente todas as universidades do mundo, e seus pesquisadores constituem uma massa crítica multidisciplinar, envolvendo contributos educacionais, sociológicos e tecnológicos [Martins et al. 2010].

A complexidade enfrentada por muitos estudantes nas variadas disciplinas de Computação levaram a intensas investigações com o objetivo de procurar entender as possíveis causas relacionadas. Esse movimento conhecido como Educação em Computação, do termo em inglês Computer Science Education Research, tem sua importância reconhecida por algumas instituições como a ACM *(Association for Computing Machinery)* e IEEE *(Institute of Electrical and Electronic Engineers)* que dispensam atenção aos aspectos educacionais. Uma das principais preocupações dos autores é investigar as dificuldades relacionadas a aprendizagem de programação [Pears et al. 2007].

Desmotivação, reprovação e evasão são problemas constantes enfrentados por cursos da área de computação, especialmente nas disciplinas de Algoritmos e Programação. Estas disciplinas costumam ter altos índices de evasão e reprovação, dificultando ou impedindo a continuidade dos alunos no curso [Bennedsen e Caspersen 2007]. Dentre possíveis fatores que contribuem para esta situação, pode-se elencar: i) falta de capacidade de abstração e raciocínio lógico para desenvolver soluções algorítmicas; ii) falta de motivação do estudante, que, muitas vezes, encara a disciplina como um grande obstáculo a ser superado; e iii) abordagem de ensino instrucionista, que pode não despertar o interesse do estudante [Jenkins 2002].

É relativamente comum introduzir programação através do paradigma imperativo e, posteriormente, apresentar o paradigma orientado a objetos nas disciplinas de programação nos cursos de computação. Esta transição cria um problema adicional,

pois provoca um conflito cognitivo nos aprendizes, geralmente demorado de resolver [Bittencourt et al. 2013, Jenkins 2002]. Associado aos fatores anteriores, esta organização curricular aumenta a complexidade na aquisição de habilidades de programação, gerando desmotivação e, consequentemente, dificultando a aprendizagem e a retenção do conhecimento.

Jenkins (2002) reflete sobre as possíveis razões das dificuldades enfrentadas pelos estudantes. Compara o ato de programar a uma "luta". Se os alunos lutam para aprender alguma coisa, conclui-se que há algo complexo envolvido. Se os educadores esperam ensinar de forma eficaz, devem compreender exatamente as razões que explicam porque aprender a programar é uma tarefa tão complexa e, para isso, deve-se ter uma visão mais cognitiva do processo de aprendizagem. Descreve a respeito dos possíveis fatores que esclarecem as dificuldades enfrentadas pelos estudantes na aquisição dessa habilidade, como estilo de aprendizagem, motivação, múltiplas habilidades, múltiplos processos, linguagem de programação, novidade educacional, interesse, reputação e imagem, e ritmo.

A programação orientada a objetos (POO) tornou-se, nos últimos anos, o paradigma de programação mais influente, sendo amplamente utilizada na educação e na indústria. No entanto, aprender programação orientada a objetos não é fácil [Kölling 1999b]. As dificuldades podem ser causadas pela complexidade dos conceitos a serem aprendidos em um curto período de tempo, a complexidade intrínseca destas linguagens e dos ambientes de desenvolvimento profissionais, agravados pelo uso de metodologias de aprendizagem centradas no professor.

Com objetivo de minimizar essas dificuldades, diversos estudos abordam possíveis soluções para o ensino-aprendizagem de POO. A adoção de abordagens lúdicas para facilitar o ensino e aprendizagem de programação é uma das iniciativas tomadas para solucionar esse problema. Ferramentas como Greenfoot, Alice e BlueJ são utilizadas para auxiliar no desenvolvimento do raciocínio lógico como a criação de jogos [Kölling 2010, Cooper 2010, Kölling et al. 2003]. Outra abordagem é o uso de computação com mídias. Forte e Guzdial (2005) descrevem um estudo que ocorreu em 2003 no Georgia Tech, onde foi oferecido um novo curso introdutório com foco em computação e utilizando o contexto de mídias, cujo objetivo era motivar os estudantes de outras áreas diferentes de TI. Foram introduzidos conceitos de programação e computação, através da criação de filtros em imagens semelhantes ao utilizados em programas como Photoshop, manipulando amostras para reverter e dividir sons, além de criar páginas Web e gerar animações. Este novo formato de disciplina apresentou melhores taxas de sucesso e aumentou a sensação de relevância para a carreira profissional e para as aspirações pessoais dos estudantes.

Ferramentas são importantes para auxiliar no processo de ensino e aprendizagem, mas não resolvem todos os problemas. A utilização de metodologias de aprendizagem ativa é outra alternativa proposta pela comunidade científica para atacar as dificuldades de aprendizagem de programação. Por exemplo, a aprendizagem baseada em problemas (PBL, do inglês, *Problem Based Learning*) é uma abordagem

instrucional centrada no estudante, em que parte importante do estudo ocorre em pequenos grupos, que se reúnem para resolver problemas propostos que desencadeiam e motivam o processo de aprendizagem [Santos et al. 2007]. A aprendizagem é autodirigida, baseada na reflexão e no fomento das questões envolvidas. PBL tem raízes na teoria de aprendizagem construtivista, que entende o estudante não como receptor passivo, mas como um ator ativo no processo de aprendizagem. A construção do conhecimento ocorre de forma mais colaborativa e com maior interação do estudante [Kinnunen e Malmi 2005]. O uso de PBL contribui para aquisição de habilidades como autonomia, iniciativa, comunicação, pensamento crítico, capacidade de resolver problemas, trabalho em grupo [Kay et al. 2000], além de retenção do conhecimento e aplicação do conhecimento em diferentes contextos [Prince 2004].

Vários trabalhos relatam a utilização da metodologia PBL como alternativa para o ensino de programação orientada a objetos [Kay et al. 2000, Ferreira et al. 2007, Bittencourt et al. 2013, Angelo et al. 2014]. Embora estes trabalhos relatem as experiências em detalhes, ainda não foi feita uma avaliação científica aprofundada do uso de uma abordagem com PBL no ensino de POO, combinando aspectos qualitativos e quantitativos, levando em conta questões de motivação e os resultados do aprendizado.

1.1 Objetivos e Questões de Pesquisa

O objetivo geral deste trabalho foi avaliar uma abordagem de ensino-aprendizagem de programação orientada a objetos no curso de Engenharia de Computação da Universidade Estadual de Feira de Santana que utiliza a abordagem PBL, em um componente curricular integrado de Programação Orientada a Objetos, Estruturas de Dados e Projeto de Sistemas do segundo semestre do curso. A avaliação foi realizada através de uma metodologia de pesquisa quali-quantitativa e uma abordagem de estudo de caso.

A partir deste objetivo, procurou-se responder às seguintes questões de pesquisa:

1. Como a abordagem utilizada influencia a motivação dos estudantes?
2. Quais os resultados proporcionados pela abordagem em termos de aprendizado de conhecimentos e habilidades de programação orientada a objetos?
3. Qual a relação entre motivação e aprendizagem na abordagem utilizada?

1.2 Organização do Documento

O presente trabalho está organizado da seguinte forma: o Capítulo 2 traz a revisão bibliográfica referente ao tema; o Capítulo 3 descreve a metodologia aplicada; o Capítulo 4 apresenta a nossa experiência e os resultados quantitativos para a motivação

e aprendizagem; o Capítulo 5 revela as discussões acerca dos resultados qualitativos e quantitativos das variáveis motivação e aprendizagem, as lições aprendidas e as ameaças á validade e confiabilidade desta pesquisa. Finalmente, no Capítulo 6 apresentamos as considerações finais com as nossas conclusões e trabalhos futuros.

Capítulo 2

Revisão Bibliográfica

Esta seção apresenta os conceitos essenciais para a compreensão da fundamentação teórica deste trabalho. Inicialmente, serão abordadas questões relacionadas as dificuldades encontradas pelos estudantes para aprender programação, em especial, programação orientada a objetos (Seção 2.1). Posteriormente, apresentar soluções com objetivo de reduzir esse problema, demonstrando possíveis soluções para auxiliar o ensino de POO (Seção 2.2). Ainda, será apresentada a mensuração da aprendizagem de programação através de instrumentos de avaliação (Seção 2.3). Além disso, são identificadas algumas soluções como a utilização da metodologia PBL e outros tipos de aprendizagem ativa (Seção 2.4). Em seguida, a descrição do uso da metodologia PBL e outros tipos de aprendizagem ativa em Computação (Seção 2.4.1), e especificamente o PBL na UEFS, no módulo de programação (Seção 2.4.2). Serão analisados aspectos sobre a importância da motivação e instrumentos utilizados para mensurá-lo (Seção 2.5). E por fim, a seção é encerrada com a exposição dos trabalhos relacionados (Seção 2.6).

2.1 As dificuldades para aprender programação

O ato de programar é complexo e requer a construção de uma série de habilidades, além de empenho e perseverança. Segundo Gomes (2010), a programação é muito mais do que a escrita de um conjunto de linhas de código em uma dada linguagem, é uma arte e uma ciência. Arte, porque existem diversas maneiras diferentes de codificar instruções, com alguma criatividade. Ciência, porque é construída a partir de regras previamente definidas, pois é necessária a utilização de lógica e deve seguir métodos rigorosos de programação para assegurar a eficiência, a economia e a utilidade dos programas criados.

Estudos apontam diversos elementos que contribuem para as dificuldades na aprendizagem de programação. Dijkstra et al. (1989) afirmam que a aprendizagem de programação é um processo lento e gradual. Jenkins (2002) cita várias causas do

insucesso generalizado em disciplinas de programação: i) baixo nível de abstração, gerando a falta de competências na resolução de problemas, ii) inadequação dos métodos pedagógicos aos estilos de aprendizagem dos alunos e iii) as linguagens de programação possuem sintaxes adequadas para profissionais, porém não para aprendizes iniciantes. Kolling (1999a) descreve a mudança de paradigma como outro problema, especialmente quando ocorre a transição da aprendizagem do paradigma imperativo para o orientado a objetos.

Adair e Jaeger (2011) apontam vários fatores que influenciam para que os estudantes não progridam bem e rapidamente. Dentre eles: o fato de que comumente a programação é considerada uma "novidade", muito diferente de outros assuntos universitários; a programação pode ser percebida como chata; além de poder ter uma má reputação com a imagem do "programador" que, às vezes, é percebido como inferior. Somado a isto, a programação é ensinada em uma escala de tempo curta.

Estudantes enfrentam vários problemas influenciados por aspectos cognitivos, emocionais e comportamentais, dos quais destacam-se o estilo da aprendizagem, a qualidade das bases de conhecimento e das competências acadêmicas, as estratégias de aprendizagem e os aspectos emocionais e de motivação para aprendizagem, como afirmam [Jenkins 2002, Gomes e Mendes 2015, Martins et al. 2010].

Jenkins (2002) descreve habilidades óbvias requeridas no processo de desenvolvimento de programas: a capacidade de criar o programa, compilá-lo, encontrar uma saída, além de testar e corrigir os erros encontrados. Porém, afirma, que essas são habilidades presumivelmente fáceis de identificar. Cita outras habilidades menos óbvias, chamadas de "habilidades para a vida". Descreve alguns aspectos para auxiliar na compreensão das dificuldades encontradas pelos estudantes na obtenção dessas habilidades. Para aprender a programar, é necessário o desenvolvimento de múltiplas habilidades. O estudante que enfrenta a aprendizagem, a partir de uma hierarquia de habilidades, geralmente aprenderá as habilidades de um nível mais baixo primeiro e, em seguida, avançará gradualmente para níveis superiores. A programação não é apenas mais do que uma habilidade única, envolve também mais de um processo distinto. No nível mais simples, a especificação deve ser traduzida para um algoritmo, que é então traduzido para o código do programa. A parte mais complexa e a mais importante é a primeira. Estando o algoritmo correto, os outros processos são essencialmente mecânicos. Este autor constatou que alguns estudantes que seguem as aulas teóricas podem ser proficientes em entender programas, mas são totalmente incapazes de escrever seu próprio programa, demonstrando que não dominaram todas as etapas do processo.

Robins et al. (2003) realizaram uma revisão e discussão da literatura, analisando os problemas relacionados ao ensino e aprendizagem de programação com foco em programadores novatos e experientes. Aprender a programar envolve a aquisição de conhecimentos complexos, estratégias relacionadas e habilidades práticas. Por isso, as aulas devem ser simples e o conhecimento deve evoluir de forma sistemática à medida que os estudantes iniciantes ganham experiência.

O ensino de programação orientada a objetos como segundo paradigma é muito discutido na literatura. As dificuldades podem estar associadas a aspectos como a complexidade natural destas linguagens ou a complexidade dos ambientes de desenvolvimento profissionais. Alguns defendem o ensino de forma processual iniciando pelo paradigma imperativo, preferencialmente ao orientado a objetos [Baeza-Yates 1995]. Alguns autores afirmam que a introdução de paradigmas como orientação a objetos na disciplina introdutória de programação, que é ministrada no inícios dos cursos de computação, não fornece evidências significativas de facilitar o aprendizado [Burton e Bruhn 2003]. Em contraposição, Kolling et al. (2003) sugerem que os conceitos de orientação a objetos devem ser ensinados desde o início, argumentando que as dificuldades estão nas ferramentas utilizadas para o ensino de programação e não no paradigma em si.

2.2 Soluções para o ensino em programação

A programação orientada a objetos (POO), nos últimos anos, tornou-se o mais influente paradigma de programação. É amplamente utilizado na educação e na indústria, e quase todas as universidades ensinam a orientação de objetos em algum lugar em seu currículo [Kölling 1999b]. Inúmeras soluções são propostas para promover o desenvolvimento na eficiência do ensino de programação, especialmente de programação orientada a objetos.

Robins et al. (2003) acreditam que a aprendizagem do estudante está relacionada com a aprendizagem de outros esquemas com o objetivo de produzir esquemas novos. Tratam o esquema como "uma estrutura de pedaços de conhecimento relacionados". Desse modo, a aprendizagem de programação é considerada um processo iterativo de organização de esquemas, no qual programar é uma habilidade que é aprendida através da construção de uma nova informação sobre a informação mais básica, gradualmente. Os autores fazem uma análise entre as diferenças no ensino de programação usando o paradigma orientando a objetos e o paradigma imperativo. Concluem afirmando que os novatos têm mais dificuldades ao solucionar um problema longo utilizando o paradigma orientado a objetos quando comparado ao paradigma imperativo e atribuem essas dificuldades, em parte, a uma curva de aprendizado mais longa, peculiar ao próprio paradigma orientado a objetos.

Kolling (1999a) cita alguns requisitos importantes para uma linguagem de programação para principiantes. Ela deve ser simples e de fácil compreensão, puramente orientada a objetos, segura e de alto nível. Deve permitir a transferência dos conceitos aprendidos para outras linguagens de forma intuitiva. Algumas questões, como a eficiência, que muitas vezes são consideradas extremamente importantes para linguagens de programação utilizadas na indústria, são de pouca importância para uma linguagem de ensino, sendo necessário apenas que a linguagem possa ser apoiada por um ambiente de ensino com tempo de resposta razoável. Além disso, deve

haver um ambiente de desenvolvimento adequado, permitindo aos programadores se concentrarem no que realmente é importante: a própria programação.

Alguns ambientes foram desenvolvidos para facilitar a aprendizagem de POO, minimizar as dificuldades encontradas, e assim, aumentar o nível de engajamento. Ferramentas como Greenfoot [Kölling 2010], Alice [Cooper 2010] e BlueJ [Kölling et al. 2003] são exemplos de ambientes que visam promover uma experiência lúdica, tornando a programação simples e divertida para os estudantes.

Greenfoot é uma ferramenta projetada por pesquisadores da Universidade de Kent, no Reino Unido [Kölling 2010]. Desenvolvida para ensinar programação orientada a objetos a partir da construção de cenários utilizando a linguagem Java e destinados as estudantes novatos em programação. Permite o desenvolvimento de aplicações gráficas como a criação de jogos em ambiente 2D. Através da visualização e interação com objetos, é possível criar mini-mundos, e representar graficamente seus objetos. Integra ferramentas como editor de código, compilador, máquina virtual, além de ferramentas educacionais.

Cooper (2010) apresenta a ferramenta Alice, uma plataforma inicialmente projetada para apoiar e facilitar o ensino de lógica de programação para estudantes do ensino médio, posteriormente estendida para o ensino superior. Alice é um ambiente de programação para uma primeira incursão em POO, com animação em 3D e manipulação direta dos elementos de uma linguagem de programação. A ferramenta permite que estudantes aprendam conceitos de POO criando filmes de animações e jogos. Adia a necessidade dos estudantes em escrever código e lidar com a complexidade da sintaxe, possibilitando a concentração nos conceitos de aprendizagem.

BlueJ é um ambiente de desenvolvimento integrado para desenvolvimento em linguagem Java, projetado com a intenção de facilitar o ensino e a aprendizagem de POO [Kölling et al. 2003]. Seu design difere significativamente de outros ambientes de desenvolvimento integrado, pois mostra a estrutura do programa em desenvolvimento como um diagrama UML, o que facilita a visualização da modelagem das classes. As principais ideias da pedagogia do BlueJ foram pautadas em oito diretrizes: aprender objetos primeiro, não iniciar com uma tela em branco, aprender a ler código, usar projetos "grandes", não começar com "main", não usar "Hello World", mostrar a estrutura do programa e tomar cuidados específicos com interface de usuário.

2.3 Mensuração do aprendizado de programação

Medir o aprendizado dos estudantes é fundamental para qualquer empreendimento educacional. A pesquisa de avaliação em educação em computação utiliza instrumentos válidos para medir as concepções dos alunos sobre os tópicos fundamentais, o que permite entender como a aprendizagem se desenvolve, possibilitando a inovação curricular e reformas baseadas neste conhecimento [Tew e Guzdial 2011].

Tew e Guzdial (2011) desenvolveram o FCS1 (*Foundational Assessment Instrument CS1*), o primeiro instrumento de avaliação fundamental para CS1[1] com o objetivo de avaliar e medir a compreensão dos conceitos introdutórios da ciência da computação, aplicável em uma variedade de pedagogias e linguagens de programação. O diferencial deste exame, comparado aos esforços existentes, é a possibilidade de ser amplamente adotado e utilizado em qualquer curso introdutório da ciência da computação. Os autores afirmam que a utilização desse instrumento, combinado com outros métodos de pesquisa, permitiria identificar quais dos muitos fatores em um ambiente de aprendizagem CS1 (e.g., o instrutor, linguagem de programação, ambiente de desenvolvimento integrado, a abordagem pedagógica, a motivação do aluno) são as alavancas que dirigem o domínio dos estudantes dos conceitos de computação.

Parker et al. (2016) elaboraram uma segunda avaliação para CS1 (SCS1 - *Second CS1 Assessment*), como uma versão isomórfica de uma avaliação independente de linguagem previamente validada para cursos introdutórios de ciência da computação, a FCS1. Argumentam a importância da replicação do FCS1 para uso de uma comunidade de pesquisa mais ampla. O artigo apresenta uma documentação com os processos utilizados para replicação de uma avaliação validada existente e demonstra uma replicação validada com sucesso. Sugerem que quanto mais avaliações forem feitas, replicando as avaliações previamente validadas, mais chances a comunidade tem de medir os ganhos de aprendizagem com precisão e determinar a eficácia das abordagens de ensino. Concluem afirmando que a avaliação SCS1 pode fornecer informações sobre diferenças nas abordagens de instrução, eficácia das intervenções e como os estudantes e professores, ou diferentes subconjuntos desses grupos, diferem em seu conhecimento de CS1.

Professores costuma utilizar avaliações teóricas para significar instrumentos e métodos para avaliar e documentar a natureza, a qualidade ou a capacidade dos estudantes [Libarkin e Anderson 2005]. No contexto de programação introdutória, as avaliações são usadas para medir a compreensão, aprendizado ou habilidade de um estudante dentro de uma disciplina ou trecho da disciplina[Parker et al. 2016]. As avaliações são abundantes em várias áreas da educação inclusive em disciplinas mais antigas que a computação, como física, matemática e engenharia [Libarkin e Anderson 2005]. No entanto, a educação em computação tem poucas avaliações validadas [Yadav et al. 2015].

2.4 Aprendizagem Ativa e PBL

Alvarez et al. (2005) contextualizam, em seu artigo, as transformações ocorridas nos últimos anos refletidas em todos os setores da sociedade, especialmente a educação. Paralelamente a esta situação, o uso das tecnologias da informação e da comunicação

[1] CS1 é a sigla tipicamente utilizada em universidades da América do Norte para a disciplina introdutória de programação

(TIC) foi apresentado como um elemento fundamental para ser considerado como um meio de mudanças nos processos de ensino. Para enfrentar os desafios colocados pela sociedade de hoje, são exigidos diversas competências e habilidades dos professores e estudantes. Não é suficiente ser um especialista em um assunto particular, por outro lado, os alunos devem desenvolver habilidades múltiplas ao mesmo tempo, e uma série de características e competências fundamentais, como a capacidade de resolver problemas, trabalhar em equipe, habilidades de comunicação, habilidades para aprendizagem autônoma, para a tomada de decisões, etc.

No contexto atual de acelerada transformação tecnológica, faz-se necessário novas formas de posicionamento de estudantes e professores para enfrentar os desafios propostos. A aprendizagem ativa é definida como qualquer método educacional que envolve os alunos no processo de aprendizagem [Prince 2004]. Este autor discute três tipos de abordagens de aprendizagem ativa: aprendizagem colaborativa, aprendizagem cooperativa e aprendizagem baseada em problemas. Aprendizagem colaborativa pode se referir a qualquer método instrutivo no qual os alunos e estudantes trabalham juntos em pequenos grupos em direção a um objetivo comum. A aprendizagem cooperativa pode ser definida como uma forma estruturada de trabalho em grupo onde os estudantes perseguem objetivos comuns enquanto são avaliados individualmente. Aprendizagem Baseada em Problemas (PBL, do inglês, *Problem-Based Learning*) é um método instrutivo no qual os problemas relevantes são introduzidos no início do ciclo de instrução, usados para contextualizar e motivar aprendizagem autodirigida pelos estudantes.

Aprendizagem Baseada em Problemas (PBL) é uma metodologia ativa desenvolvida pelo médico Howard Barrows que teve sua gênese entre o final da década de 1960 e início na década de 1970, na Universidade McMaster, no Canadá [Barrows e Tamblyn 1980]. Inicialmente idealizada para área de saúde, esta metodologia ganhou aceitação e está se tornando cada vez mais presente dentro de uma variedade de disciplinas no ensino superior. Consiste basicamente na resolução de problemas, nos quais os responsáveis pela construção da aprendizagem são os próprios estudantes. O professor atua como facilitador e é responsável pela escolha do problema a ser solucionado. O estudante tem autonomia para analisar e trilhar os possíveis caminhos em direção à solução do problema. Os estudantes são instigados a encontrar soluções por si mesmos e são responsáveis pela busca das fontes de informação em direção ao aprendizado. O professor prepara o ambiente, ajuda os alunos a se relacionarem com o problema, arranja uma estrutura de trabalho, aborda o problema com os alunos, reequaciona o problema, facilita a produção de um produto ou de um desempenho e estimula a autoavaliação [Delisle 1997]. Age como facilitador, provocando questionamentos, de maneira que o processo de criação do conhecimento seja de responsabilidade do estudante. O processo gera um aprendizado tanto para o aluno como para o professor [Savery e Duffy 1995].

Segundo Alvarez et al. (2005), PBL representa uma abordagem apropriada à realidade atual, pois possibilita o desenvolvimento de um trabalho com uma proposta em que os estudantes terão que buscar atividades a partir de discussões em grupo,

dentro e fora da sala de aula em uma perspectiva de aprendizagem colaborativa, além de permitir *feedback* constante entre estudantes e corpo docente, pois possibilita a realização de um tipo de avaliação continuada. Concluem, destacando o papel motivador em potencial desta abordagem metodológica, porque coloca os alunos em posição de atuar como se fossem profissionais.

O PBL permite que o estudante entenda e se aprofunde adequadamente na resposta aos problemas que são utilizados para aprender, entrando para fazer parte de suas análises de estruturas científicas, filosóficas, sociológicas, históricas e práticas. Os estudantes trabalham em colaboração em pequenos grupos, sob a supervisão de um tutor, analisam e resolvem um problema, selecionado especialmente para a realização de determinados objetivos em diferentes disciplinas. Mas o objetivo final não é a resolução do problema. O problema é usado como base para a identificação de tópicos de aprendizagem, para o estudo independente ou em grupos [Ortiz et al. 2003].

Segundo Cintra e Bittencourt (2015), em PBL, não só os estudantes são constantemente incentivados a aprender, mas também a desempenhar um papel ativo no processo de construção da aprendizagem. Um problema em PBL é um gatilho para motivar o estudo. Geralmente, o método segue um ciclo de aprendizagem, repetido enquanto durar o problema:

1. Os estudantes são apresentados a um problema semiestruturado antes de qualquer preparação ou estudo;
2. Os estudantes se reúnem em grupos para organizar ideias e recordar conhecimentos anteriores relacionados ao problema;
3. Através da discussão, os estudantes colocam questões, conhecidas como questões de aprendizagem, que tratam de aspectos do problema que não entendem;
4. Os problemas de aprendizagem são classificados em ordem de importância, e os alunos definem metas de aprendizagem para estudo independente e em grupo.

O PBL valoriza muito mais a compreensão profunda do que a memorização, embora considere que esta última também é importante para a aprendizagem, pois quanto maior for a compreensão de determinado assunto, mais fácil será a memorização e, consequentemente, a aprendizagem. Porém, o aprendizado que fica apenas no nível da memorização tem pouco valor para a vida social e profissional. Esse é um dos principais problemas decorrentes de aulas expositivas que enfatizam o conteúdo apenas no contexto em que foi aprendido. Isso não ocorre quando se utiliza PBL. Este método permite ao estudante observar e analisar atitudes e valores que, durante o método de ensino tradicional, não podem ser realizados. Como os problemas são apresentados em um contexto real, favorecem a transferência dos conhecimentos e habilidades aprendidos em sala de aula para o mundo do trabalho [Albanese e Mitchell 1993, Delisle 1997]. O PBL permite ao estudante adaptar-se às mudanças, incentiva o espírito crítico, ensina a aprender a aprender e a trabalhar e aprender em equipe [Ortiz et al. 2003].

2.4.1 PBL na Educação em Computação

Para ter sucesso no campo da Ciência da Computação, é preciso ter conhecimento e dominar uma variedade de habilidades, tais como: estar à vontade com matemática, lógica, resolução de problemas, pensamento algorítmico e programação. Infelizmente, muitos estudantes lutam para desenvolver essas habilidades, especialmente quando o assunto está relacionado à programação orientada a objetos, matemática discreta, estruturas de dados e análise de algoritmos. Com o objetivo de encontrar soluções para esses problemas, o PBL vem sendo aplicado em instituições ensino na área da computação, com o objetivo de promover a aprendizagem colaborativa e motivadora, com base na resolução de problemas [Oliveira et al. 2012].

[O'Grady 2012] realizou uma revisão sistemática com o objetivo investigar como PBL está sendo aproveitado nos currículos voltados para o ensino da computação. O estudo encontrou 63 artigos, em disciplinas da ciência da computação, engenharia de computação, sistemas de informação, tecnologia da informação e engenharia de software. Nestes cursos, há uma grande variedade de disciplinas que usaram PBL como abordagem de ensino: engenharia de software, programação de computadores, qualidade do software e sistemas operacionais, entre outras. Concluem constatando que a penetração do PBL nos currículos de computação ainda é superficial. Em muitos casos, os membros do corpo docente estão trabalhando isoladamente e enfrentam o desafio de introduzir PBL nos currículos, que permanecem essencialmente didáticos. Em alguns casos, PBL é uma medida de último recurso para melhorar a retenção ou o desempenho.

ROLEP (*Research on Learning Programming*) é um grupo de pesquisa no Laboratório de Ciências do Processamento da Informação da Universidade de Tecnologia de Helsinque, na Finlândia. O foco do grupo é fazer pesquisas sobre aprendizagem de programação: descrever o processo de aprendizagem e as questões que o afetam e descobrir maneiras de promover a aprendizagem. Estudam as experiências de aplicação de PBL em um curso de programação introdutória por vários anos. Desde 1999, os estudantes da área de computação têm estudado nas disciplinas de programação introdutória usando PBL. Os autores observaram que os estudantes dos grupos PBL estavam motivados e obtiveram bons resultados de aprendizagem [Kinnunen e Malmi 2005].

No Brasil, a utilização de PBL empregada na área de computação ainda é muito restrita. Nos trabalhos encontrados na literatura, o PBL é aplicado em seis componentes curriculares no curso de Engenharia de Software da Universidade Federal do Pampa (UNIPAMPA). Esses componentes integram, de modo interdisciplinar e transversal, diferentes conteúdos na abordagem de situação problema que se aproxima da realidade profissional [Cheiran et al. 2017]. Outra instituição que introduziu a abordagem com PBL integrada ao currículo durante todo o curso é a Universidade Estadual de Feira de Santana (UEFS), que adotou PBL no curso de Engenharia da Computação desde a sua criação [Angelo et al. 2014].

2.4.2 PBL em Computação na UEFS

O curso de Engenharia da Computação da UEFS tem adotado PBL desde sua criação em 2003. Este curso possui um currículo flexıvel, favorecendo a atualização constante dos conteúdos. O curso também é caracterizado pela integração e interdependência entre componentes curriculares que agrupam disciplinas com conteúdos relacionados em um mesmo período letivo, compartilhando trabalhos, desafios e oportunidades de aprendizado, evidenciadas particularmente pelos componentes curriculares denominados Estudos Integrados [Bittencourt e Figueiredo 2003].

O objetivo da adoção do PBL no curso de Engenharia da Computação da UEFS foi conciliar a apresentação de um volume crescente de conhecimentos técnicos e científicos à necessidade de trabalhar habilidades e atitudes necessárias ao engenheiro, tais como capacidade de aprendizagem independente e contínua, de trabalhar em grupo, bem como o respeito por opiniões diversas e a ética [Angelo et al. 2014].

O Estudo Integrado (EI) tem por objetivo ser um componente integrador sobre certo tema, e é organizado em módulos. Durante o estudo integrado, o estudante é apresentado a certo tema ou problemas abrangentes e, para compreender o tema ou resolver os problemas, torna-se necessário adquirir novos conhecimentos, os quais são agrupados em módulos. Um módulo é um recorte em determinados campos do conhecimento, organizado de forma articulada, auto-contida e coesa para acontecer o processo de ensino/aprendizagem. Os módulos de cada estudo integrado estarão, ao longo do curso, oportunizando a aprendizagem interdisciplinar, referenciados pelos componentes curriculares que compartilham do período letivo [Angelo e Bertoni 2012].

No ano de 2007, o currículo do curso contava com oito componentes de EI temáticos, distribuídos ao longo de cada semestre. Eram eles: Introdução ao Hardware (Circuitos Digitais e Introdução aos Sistemas de Computação), Sistemas Digitais (Arquitetura de Computadores e Arquitetura de Computadores Avançada), Circuitos Eletrônicos (Circuitos Elétricos e Eletrônica Geral), Programação (Algoritmos e Programação II, Estrutura de Dados e Estruturas Discretas), Concorrência e Conectividade (Sistemas Operacionais e Redes de Computadores), Engenharia de Software (Análise de Sistemas, Engenharia de Software e Banco de Dados), Sinais e Sistemas Digitais e Analógicos (Métodos Numéricos, Eletrônica Digital e Sinais e Sistemas), Linguagens de Programação (Linguagens Formais e Autômatos, Compiladores e Conceitos de Linguagens de Programação) [Santos et al. 2007]. Desde 2011, uma reformulação curricular adicionou o EI de Algoritmos, oferecido no primeiro semestre em substituição à disciplina isolada de Algoritmos e Programação, que também adota o PBL, mas como disciplina isolada.

Segundo Angelo e Bertoni (2012), os componentes curriculares que não fazem parte de nenhum Estudo Integrado, podem ou não seguir o método PBL, sendo esta uma escolha do professor. Isto ocorre porque o curso conta com a participação de

professores da instituição de outras áreas, que lecionam disciplinas em Engenharia de Computação tais como Matemática, Física, Psicologia, dentre outras.

Os conceitos de programação estão presentes em dois estudos integrados. O EI de Algoritmos é um componente curricular oferecido no primeiro semestre, que faz um estudo introdutório integrando as ideias de algoritmos, estruturas de dados básicas (arrays e registros) e programação estruturada em uma linguagem imperativa. No segundo semestre, o EI de Programação integra a programação orientada a objetos, algoritmos e estruturas de dados avançadas e projeto de sistemas [Bittencourt et al. 2013].

O objetivo geral do EI de Programação é que o estudante seja capaz de projetar e desenvolver software orientado a objetos, utilizando apropriadamente algoritmos e estruturas de dados, com domínio dos conceitos e fundamentos subjacentes às metodologias e ferramentas utilizadas. Objetivos específicos são desdobrados do objetivo geral, a partir das competências específicas desejadas (e.g., ser capaz de escrever, compilar e depurar programas orientados a objetos em Java; ser capaz de escolher e implementar estruturas de dados apropriadas a um problema de busca) [Bittencourt et al. 2013].

Santos et al. (2007) descrevem o estudo integrado como um componente que associa a metodologia baseada em problemas e projetos com o ciclo de aprendizagem-fundamentação-realização, na qual, na fase inicial, o aprendiz conhece os problemas em sua face real e visível mantendo contato com os fenômenos e objetivos motivadores do estudo. Na segunda fase, há fundamentação em termos de aquisição de estruturas teóricas necessárias à resolução dos problemas previamente conhecidos na fase inicial. Eles comparam este ciclo com o ciclo tradicional de ensino, no qual se aprende primeiro a solução do problema para posteriormente se aprender o problema. No novo ciclo, o estudante já iniciou uma reflexão crítica sobre os problemas em questão, além de estar motivado para adquirir a base teórica que lhe falta para compreendê-los melhor. Por fim, retorna-se à prática, agora renovada através de fundamentos teóricos que permitirão ao aprendiz avançar na compreensão da realidade em questão. A realização de soluções para os problemas estudados anteriormente sedimenta definitivamente a compreensão do assunto. Tais realizações vêm necessariamente na forma de projetos, atividades práticas, vivências, nos quais se espera um esforço muito maior por parte do estudante do que professor.

2.5 Motivação

As dificuldades de aprendizagem de programação não se resumem apenas à dificuldade de abstração para resolver problemas, inadequação de métodos pedagógicos, ou mesmo às dificuldades com as linguagens de programação. Há ainda fatores relacionados com a motivação dos estudantes e seu envolvimento com a disciplina, de modo que eles não a encarem como um obstáculo tão difícil a ser superado.

Keller (1987) define motivação como aquela que explica a direção e a magnitude do comportamento, ou, em outras palavras, explica quais os objetivos que as pessoas escolhem perseguir e quão ativamente ou intensamente as perseguem. Ray (1964) descreve a motivação fazendo um exame cuidadoso da palavra (motivo) e de seu uso, e conclui que a motivação deverá fazer referência a três componentes: o comportamento de um sujeito; a condição biológica interna relacionada; e a circunstância externa relacionada.

Segundo Clark e Jenkins (1999) a motivação é um conceito inerentemente abstrato e que é difícil de medir ou identificar de forma significativa. A motivação é descrita como um conceito profundamente pessoal e se os sujeitos são questionados diretamente sobre sua motivação, o questionador nunca pode estar totalmente certo de que as respostas dadas são verdades, e nunca poderá ter certeza de que as inferências feitas com base no questionamento são corretas. Mesmo que as respostas dadas sejam completamente honestas, continua a existir a necessidade de alguma especulação por parte do questionador e, portanto, alguma incerteza sempre permanece.

Fallows e Ahmet (1999) afirmam que é possível observar o comportamento de uma pessoa e disso inferir sua motivação provável, mas nunca é possível ter certeza. Algumas categorias gerais de motivação podem ser observadas e identificadas. São elas: o desejo do estudante de agradar o professor; a necessidade percebida pelo estudante quanto ao material que está sendo apresentado; o grau de interesse do estudante no assunto; os valores filosóficos e as crenças do estudante; a atitude do aprendiz sobre os materiais que estão sendo entregues; as aspirações acadêmicas e de carreira do estudante e os incentivos e recompensas que possam advir da aprendizagem.

Jenkins (2001) alega que para inspirar os estudantes através de uma instrução verdadeiramente motivacional na aprendizagem de programação, a solução é maximizar os efeitos positivos de cada um desses fatores. E, para isso, é necessário reconhecê-los. Ele menciona cinco tipos de motivação: extrínseca, intrínseca, social, de realização e nula. Na motivação extrínseca, o fator motivacional é a carreira e as recompensas associadas que resultarão da conclusão bem-sucedida do curso. Na motivação intrínseca, o principal fator motivacional é um interesse profundo na computação (ou especificamente em programação) para seu próprio bem. Na motivação social, o fator motivacional é o desejo de agradar a um terceiro cuja opinião é importante. Na motivação de realização, o fator motivacional é ”fazer bem”para a satisfação pessoal. Uma quinta categoria corresponde à ”motivação nula”, que se relaciona com casos que não se encaixam nas categorias previamente descritas.

Segundo Keller (1987), existem muitas teorias e conceitos que tentam explicar a dinâmica e os atributos da motivação e, em geral, essas teorias motivacionais podem ser agrupadas em quatro categorias com base em seus pressupostos e domínios de pesquisa. O primeiro grupo é fundamentado em fisiologia humana e neurologia e inclui estudos de genética, processos fisiológicos de excitação e processos fisiológicos de regulação. Outro grupo consiste nas abordagens comportamentais que incluem a princípios bem conhecidos de reforço positivo (condicionamento operante), condici-

onamento clássico, motivação de incentivo e influências ambientais na estimulação sensorial. O terceiro grupo, que tem recebido mais atenção nos últimos anos, consiste em teorias cognitivas, incluindo teorias de valor-expectativa, motivação social, teorias atribucionais e teorias de competências. O quarto grupo, que cresceu rapidamente em popularidade, inclui estudos de emoção e afeto.

Baseado na importância da motivação na aprendizagem, Keller (1987) desenvolveu o modelo ARCS da motivação, que se propõe a fornecer estratégias para auxiliar a reconhecer e ajudar a resolver problemas motivacionais dos alunos. O modelo analisa as necessidades motivacionais dos estudantes através de quatro categorias: Atenção, Relevância, Confiança e Satisfação. Com relação à atenção, a meta é estabelecer um equilíbrio nas atividades do aluno que permita manter sua atenção. Para isto, utilizam-se estratégias que incluem a variação de ritmo ou estilo do material pedagógico, o uso do humor ou envolvimento do estudante nas atividades.

A primeira categoria, *Atenção*, contém variáveis motivacionais relacionadas a estimular e sustentar as curiosidades e os interesses dos estudantes. O próximo passo é garantir que os estudantes acreditem que a experiência de aprendizagem tenha *Relevância* pessoal e, para isso, deve-se atender às necessidades e objetivos pessoais do estudante para obter uma atitude positiva. Mesmo que os estudantes acreditem que o conteúdo é relevante e eles estejam curiosos para aprender, eles ainda podem não estar adequadamente motivados devido a pouca ou muita confiança ou sem expectativa de sucesso. Para adquirir *Confiança*, os estudantes devem acreditar, eles precisam sentir que terão sucesso e controlarão seu sucesso. Se o professor tiver sucesso na realização desses três primeiros objetivos motivacionais (Atenção, Relevância e Confiança), os estudantes serão motivados a aprender. Em seguida, para que eles tenham um desejo contínuo de aprender, eles devem ter sentimentos de *Satisfação* com o processo ou os resultados da experiência de aprendizagem. A satisfação pode resultar de fatores extrínsecos, por exemplo, o professor demonstrar reconhecimento pelo avanço do estudante, e intrínsecos, quando o estudante percebe a relevância do conteúdo. [Keller 1987].

Existem duas ferramentas de medição que podem ser usadas em conjunto com o modelo ARCS. O Exame de Interesse do Curso (*Course Interest Survey* – CIS) e o Exame da Motivação dos Materiais Didáticos (*Instructional Materials Motivation Survey* – IMMS). O CIS foi concebido para medir as reações dos estudantes às instruções dirigidas pelo instrutor. O IMMS foi projetado para medir reações aos materiais de instrução autodirigidos. Estas são medidas de autoavaliação específicas da situação que podem ser usadas para estimar as atitudes motivacionais dos estudantes no contexto de praticamente qualquer sistema de ensino. O CIS pode ser usado na instrução de sala de aula presencial e em cursos lineares síncronos e assíncronos facilitados pelo instrutor. O IMMS pode ser usado com aprendizagem auto-dirigida baseada em impressão, instrução baseada em computador ou cursos on-line que são principalmente autodirigidos. Além disso, eles foram projetados para estar em correspondência com a base teórica representada pelos conceitos e teorias motivacionais que compõem o modelo ARCS, pois estas teorias incorporam

construções psicológicas a partir da literatura empírica sobre a motivação humana [Keller 1987].

2.6 Trabalhos Relacionados

Metodologias ativas de ensino-aprendizagem focados na proposta de aprendizagem baseada em problemas (PBL) para o ensino de programação orientada a objetos (POO) têm sido utilizadas em algumas universidades. Kay et al. (2000) relatam em seu trabalho a avaliação de uma abordagem. Eles redesenharam o currículo do curso, alterando a disciplina de introdução a programação que utilizava a linguagem de programação Pascal para uma linguagem de programação orientada a objetos, Java (facilitada pelo ambiente BlueJ), e adotaram PBL como abordagem de ensino. Concluem resumindo a experiência de três anos com algumas conquistas, inicialmente conseguiram implementar um currículo integrado para o curso, além disso, ao final, o PBL proporcionou aquisição de habilidades genéricas como a capacidade de trabalhar em grupo, de planejamento, de solucionar problemas, de comunicação, desenvolveram pensamento crítico e ainda, melhoraram a redação e a apresentação oral.

Vihavainen et al. (2011) descrevem, em seu estudo, uma avaliação de uma abordagem utilizando a metodologia ativa para o ensino de uma disciplina introdutória de programação. Neste curso, foi utilizada a linguagem de programação Java e durou um semestre. A abordagem era organizada em duas unidades: introdução a programação, na qual trabalhava conceitos básicos de programação orientada a objetos, e programação avançada, momento no qual o conhecimento aprendido na fase inicial foi aprofundado. A abordagem reduziu o número de palestras de cinco horas semanais para duas horas, para investir em aulas com enfoque em atividades práticas que apresentavam dificuldade incremental, além do apoio constante de professores experientes. Os autores concluem que a utilização do método de aprendizagem ativa, combinando assistência dada por professores experientes associada a um conjunto de valores e práticas produz resultados positivos. Os resultados apontaram uma queda nas taxas de evasão e reprovação, além do aumento na aceitação do curso.

Herala et al. (2015) relatam, em seu trabalho, o uso da metodologia da sala de aula invertida em uma disciplina de programação orientada a objetos. Essa metodologia propõe a inversão completa do modelo de ensino, propõe aulas menos expositivas, e mais produtivas e participativas. Em apoio ao método de sala de aula invertida, é disponibilizado um material contendo videoaulas, cada uma com duração de 15 a 30 minutos para que os estudantes estudem antes do encontro presencial. Além da metodologia de ensino, foram alteradas a linguagem de programação, o material das aulas, exercícios semanais, os projetos de programação e as avaliações. Trocaram a linguagem de programação C++ para Java, e as avaliações que eram escritas, passaram a ser *on-line*. Os autores concluem afirmando que após a experiência, os

estudos sugerem que a metodologia da sala de aula invertida é adequada para o ensino de programação orientada a objetos. Os exercícios incrementais, o projeto e as palestras em vídeo obtiveram *feedback* positivo dos estudantes.

Bittencourt et al. (2013) relatam uma experiência de integração das disciplinas de programação orientada a objetos, estruturas de dados e projeto de sistemas em um estudo integrado semestral com dez horas semanais, utilizando uma metodologia de aprendizagem baseada em problemas e projetos (PBL). A experiência foi aplicada em dois semestres letivos, e as principais lições aprendidas foram: a integração de conhecimentos possibilita experiências mais autênticas e práticas de produção de software mais disciplinadas; a metodologia PBL permite adquirir competências mais amplas de comunicação, trabalho em equipe e autodidatismo; problemas de manutenção de software podem reduzir a motivação por acúmulo de deficiências e devem ser evitados; a dosagem de novos conceitos nos problemas propostos deve respeitar um processo gradual e a capacidade de assimilação dos alunos.

Ferreira et al. (2007) descrevem a metodologia ativa de aprendizagem baseada em projetos em um curso de laboratório introdutório de programação orientada a objetos e a sua respectiva experiência como professor. Neste curso, os estudantes deveriam implementar um jogo de computador interativo de pequeno a médio porte em um semestre, fazendo uso da linguagem Java. Os autores concluem afirmando que a utilização desta abordagem teve um desempenho satisfatório e que aprender com base em um projeto prático poderia enriquecer a interação dos estudantes com conteúdo interdisciplinar do curso teórico e ajudá-los a perceber a relevância do curso prático para questões da vida real e sua correlação com outros cursos.

Existem alguns trabalhos que utilizam a metodologia PBL para o ensino de programação orientada a objetos, entretanto há poucos relatos na literatura que avaliam a aprendizagem e motivação. No seu trabalho, Tew e Guzdial (2011) desenvolveram um instrumento de avaliação com intuito de mensurar conceitos de programação, aplicado a uma variedade de linguagens de programação, o FCS1. Para efetuar o estudo, foram recrutados participantes de quatro cursos introdutórios diferentes, que estudavam em duas universidades por quatro membros do corpo docente separados, de modo que a definição e a compreensão do conhecimento do CS1 não estavam ligadas a um determinado membro da faculdade ou instituição. Ao final, o instrumento foi elaborado, e os autores estão explorando a aplicabilidade e validade do uso do FCS1 para medir o conhecimento dos estudantes em diferentes paradigmas e linguagens de programação.

Utting et al. (2013) utilizaram o instrumento FCS1 para fornecer informações sobre o desempenho dos estudantes em programação, de forma genérica, e posteriormente compararam com as expectativas dos professores sobre o desempenho dos alunos. A avaliação foi efetuada com o grupo de trabalho ITiCSE, convocado em 2013 para revisar e rever o grupo de trabalho McCracken da ITiCSE (2001), sobre a habilidade dos programadores novatos para resolver um problema de programação especificado. Os resultados mostraram que as expectativas dos professores não combinavam com

o desempenho dos estudantes e tendiam a ser muito otimistas em comparação com a pontuação antecipada.

Forte e Guzdial (2005) relatam uma experiência docente ocorrida no *Georgia Institute of Technology (Georgia Tech)*, em Atlanta, na qual são lecionadas disciplinas de CS1. Na primavera de 2003, foram oferecidas três abordagens diferentes para a mesma disciplina: CS1 tradicional, para estudantes de ciência da computação; Introdução a Computação, para estudantes de engenharia; e Introdução a computação com mídias, para estudantes de arquitetura, administração e artes liberais. Em seguida analisaram de que maneira os cursos personalizados e tradicionais da CS1 impactaram na motivação, as percepções dos estudantes, além de analisarem as taxas de evasão e reprovação. Os três cursos adaptados correspondentes foram projetados e oferecidos com resultados encorajadores. Os resultados iniciais sugerem que os alunos em cursos personalizados são mais propensos a "cumpri-lo", menos propensos a não gostar do material introdutório de CS1 e, no caso da computação com mídias, mais propensos a considerar a continuação da aprendizagem do CS1.

Santana et al. (2017) descrevem um estudo no qual foi investigado como uma abordagem de ensino-aprendizagem de programação para *non-majors*, em uma disciplina de Introdução a Ciência da Computação no curso de Engenharia Civil influenciou na motivação dos estudantes. Utilizaram o ambiente Scratch, da linguagem de programação Python com a biblioteca Turtle e o ambiente de desenvolvimento JES. A avaliação foi realizada a partir de um questionário adaptado do IMMS ao contexto da abordagem, além de uma avaliação qualitativa. Para avaliar as diversas dimensões da motivação, utilizou-se o modelo de motivação ARCS (Atenção, Relevância, Confiança e Satisfação). Os resultados mostraram que o ambiente Scratch proporcionou o aumento da motivação dos estudantes. Em contrapartida, com a ferramenta JES no contexto de imagens, não se conseguiu motivar os estudantes satisfatoriamente. Todavia, algumas complicações podem ter impactado no resultado final, demandando investigação posterior.

Capítulo 3

Metodologia

O presente trabalho adotou a metodologia de pesquisa de métodos mistos, incorporando elementos qualitativos e quantitativos como prática metodológica. Segundo Creswell (2002), a pesquisa de métodos mistos é um tipo de pesquisa com suposições filosóficas e também com métodos de investigação. Como uma metodologia, ela envolve suposições filosóficas que guiam a direção da coleta e da análise e a mistura das abordagens qualitativa e quantitativa em muitas fases do processo da pesquisa. Como um método, ela se concentra em coletar, analisar e misturar dados quantitativos e qualitativos em um único estudo ou uma série de estudos. Em combinação, proporciona um melhor entendimento dos problemas de pesquisa do que cada uma das abordagens isoladamente. Além disso, este trabalho adotou ainda a estratégia de estudo de caso, na qual um ou mais fenômenos são analisados em um ambiente real, delimitado em tempo e espaço, sem o controle típico de fatores como em experimentos.

A seguir, estão descritos os elementos do estudo de caso: o cenário educacional, os participantes do estudo, planejamento e avaliação do componente curricular, os procedimentos e instrumentos para a coleta e análise de dados.

3.1 O estudo de caso

O estudo de caso foi conduzido ao longo do segundo semestre de 2017. Avaliamos o Estudo Integrado (EI) de Programação do curso de Engenharia da Computação da UEFS, componente curricular formado pelos módulos teóricos de Algoritmos e Programação II, Estrutura de Dados, Projeto de Sistemas e pelo Módulo Integrador (MI) de Programação.

3.1.1 Cenário

A Universidade Estadual de Feira de Santana (UEFS) oferece o curso de Engenharia da Computação desde 2003. Este curso utiliza como estratégia de ensino a metodologia PBL e apresenta um currículo flexıvel, favorecendo a atualização constante dos conteúdos. O curso também é caracterizado pela integração e interdependência entre componentes curriculares que agrupam disciplinas com conteúdos relacionados em um mesmo período letivo, evidenciadas particularmente pelos componentes curriculares denominados Estudos Integrados [Bittencourt e Figueiredo 2003].

O Estudo Integrado (EI) objetiva ser um componente integrador sobre certo tema e é organizado em módulos. Durante o estudo integrado, o estudante é apresentado a certo tema ou problemas abrangentes e, para resolver os problemas, torna-se necessário adquirir novos conhecimentos, os quais são agrupados em módulos.

Os conceitos de programação estão presentes em dois estudos integrados desse curso. O EI de Algoritmos é um componente curricular oferecido no primeiro semestre, que faz um estudo introdutório integrando as ideias de algoritmos, estruturas de dados básicas (arrays e registros) e programação estruturada em uma linguagem imperativa. No segundo semestre, o EI de Programação integra a programação orientada a objetos, algoritmos e estruturas de dados avançadas e projeto de sistemas. Construímos este trabalho através da experiência obtida no EI de Programação.

3.1.2 Participantes

Nosso estudo ocorreu com estudantes do EI de Programação. As turmas eram heterogêneas, compostas por 46 estudantes novatos e veteranos: 44 homens e 2 mulheres. Todos apresentavam experiência inicial em programação, pois cursaram o EI de Algoritmos, componente curricular ofertado no primeiro semestre do curso.

Os módulos teóricos eram compostos por 2 professores: 1 professor para disciplina de Projeto de Sistemas e 1 professor para as disciplinas de Estrutura de Dados e Programação Orientada a Objetos. Nem todos os estudantes que fazem o módulo teórico, fazem o módulo integrador. O módulo integrador era constituído por 25 estudantes, distribuídos em quatro turmas, cada uma formada por no mínimo 6 e no máximo 10 estudantes, com um professor para cada grupo tutorial.

3.1.3 Planejamento

O objetivo geral do EI de Programação é que o estudante seja capaz de projetar e desenvolver software orientado a objetos, utilizando apropriadamente algoritmos e estruturas de dados, com domínio dos conceitos e fundamentos subjacentes às metodologias e ferramentas utilizadas. Objetivos específicos são desdobrados do objetivo geral, a partir das competências específicas desejadas (e.g., ser capaz de

escrever, compilar e depurar programas orientados a objetos em Java; ser capaz de escolher e implementar estruturas de dados apropriadas a um problema de busca) [Bittencourt et al. 2013].

O EI de programação é composto pelos módulos teóricos e o módulo integrador (MI). Os módulos teóricos apresentam 30 horas cada e são compostos pelas disciplinas Algoritmos e Programação II, Estrutura de Dados e Projeto de Sistemas. Neste módulo, as aulas são expositivas, acontecem em salas de aulas tradicionais, utilizam lousa branca, kit de pincéis, projetor multimídia e computador como materiais de apoio. Em paralelo, no MI de Programação, ocorrem duas sessões tutoriais semanais do PBL com duas horas de duração, e carga horária total de 60 horas. As sessões acontecem em salas tutoriais, onde são reunidos pequenos grupos de estudantes. Nessas sessões, são utilizados lousa branca, kit de pincéis, computadores, linguagem Java e as IDEs Eclipse ou Netbeans.

As Tabelas 3.1, 3.2 e 3.3 apresentam, respectivamente, os planejamentos do módulos teóricos: Algoritmos e Programação II, Estrutura de Dados e Projeto de Sistemas. Os módulos teóricos são organizados em três unidades. Os conceitos teóricos trabalhados em cada unidade são normalmente planejados para corroborar com aquisição dos conceitos necessários na solução dos problemas.

Tabela 3.1: Planejamento do módulo teórico Algoritmos e Programação II

Unit	**Habilidades**	**Conceitos**
I	Compreender a modelagem e programação orientada a objetos,e conceitos básicos relacionados;	1. Paradigmas de Programação 2. Orientação a Objetos 3. Objetos, Classes 4. Atributos e Estado 5. Métodos, Mensagens e Comportamentos
II	Ser capaz de escrever, compilar, depurar e testar programas na linguagem de programação Java Aplicar estruturas de dados orientadas a objetos para modelar dados simples e complexos a partir de problemas reais	6. Encapsulamento 7. Herança 8. Polimorfismo e Ligação Dinâmica 9. Relacionamentos e Composição 10. Acoplamento e Coesão 11. Implementação de Classes e Objetos 12. Classes, Atributos e Métodos Final
III	Tratar erros e exceções em programas na linguagem Java Criar interfaces gráficas simples na linguagem Java	13. Classes Abstratas e Interface 14. Atributos e Métodos Estáticos 15. Tratamento de Exceções 16. Estruturas de Dados Básicas (Collections) 17.Interface Gráfica e programação orientada a eventos 18. Manipulação de Arquivos

O planejamento do MI de Programação obedece a uma formatação em que os problemas são elaborados a partir de um cenário real e seguem um espiral de complexidade crescente. Foram propostos quatro problemas, distribuídos ao longo de um semestre. Para cada problema, é necessária a aquisição de conceitos e a prática de habilidades de programação, conforme apresentado na Tabela 3.4.

Tabela 3.2: Planejamento do módulo teórico Estrutura de Dados

Unit	Habilidades	Conceitos
I	Saber computar a complexidade de algoritmos básicos Conhecer estruturas de dados básicas como lista, fila e pilha	1. Complexidade de algoritmos 2. Alocação de memória 3. Array (tabelas) 4. Lista simples e encadeada 5. Fila 6. Pilha 7. Busca linear
II	Conhecer principais métodos de busca e ordenação	8. Busca binária 9. Métodos de Odenação (BubbleSort, SelectionSort, MergeSort, InsertionSort HeapSort, QuickSort) 10. Àrvore binária em array 11. Heap 12. Fila com prioridade 13. Bucket Sort
III	Conhecer estruturas de dados avançadas como árvores e hash Conhecer o funcionamento de grafos e saber desenvolver algoritmos de busca em grafos orientados e não orientados	14. Árvores 15. Árvore binária balanceada (AVL) 16. Árvores B 17. Hashing 18. Tabela Hash 19. Grafos

Tabela 3.3: Planejamento do módulo teórico Projeto de Sistemas

Unit	Habilidades	Conceitos
I	Conhecer, compreender e aplicar as metodologias, técnicas e ferramentas de projeto de sistemas de software OO	1. UML: Diagrama de classes 2. Modelo conceitual: (conceitos, associações e atributos 3. Construção de modelos conceituais 4. Diagrama de classes de projeto 5. Operações 6. Atributos de associações 7. Coleções 8. Navegabilidade de relacionamentos 9. Design de relacionamentos 10. Mapeamento de projeto para código
II	Produzir documentação de projeto baseada em diagramas essenciais da linguagem de modelagem unificada (UML) Conhecer e utilizar artefatos de software de requisitos, análise conceitual, código-fonte e testes de modo a projetar sistemas de software numa visão integrada do ciclo de vida do software	11. Teste de Unidade 12. Framework JUnit 13. Generalização e especialização 14. Supertipos e subtipos 15. Polimorfismo 16. Tipos abstrato: classes abstratas e interfaces 17. Polimorfismo com tipos abstratos 18. Herança múltipla 19. Testes de sistemas 20. Testes de aceitação
III	Escolher padrões de atribuições de responsabilidades e padrões de projeto para resolver problemas de projeto de software OO	21. Diagramas de interação em UML 22. Diagrama de comunicação 23. Padrões para atribuição de responsabilidades (GRASP) 24. Padrão Observer 25. Padrão MVC

Tabela 3.4: Planejamento do Módulo Integrador

Problema	**Tema**	**Habilidades**	**Conceitos**
01	Sistema de Cadastramento Biométrico	Aplicar as conceitos aprendidos de programação orientada a objetos.	1. Classes e objetos. 2. Atributos, métodos e construtores. 3. Entrada e saída em Java através do console. 4. Interfaces em Java. 5. Padrão MVC. 6. Modelo conceitual. 7. Diagrama de classes de projeto. 8. Tipos de dados abstratos em estruturas de dados. 9. Lista encadeada: implementação de operações. 10. Padrão de projeto Iterator: implementação e uso.
02	Ferramenta para leilão eletrônico	Capacidade de modelar, projetar, codificar e testar pequenos sistemas de software com base nos conceitos aprendidos nas disciplinas teóricas có-requisitos do módulo integrador	1. Construtores. 2. Sobrecarga de métodos. 3. Composição de objetos. 4. Herança simples. 5. Padrão MVC. 6. Modelo conceitual. 7. Diagrama de classes de projeto. 8. Testes de unidade. 9. Pilhas, filas, filas com prioridades. 10. Algoritmos de ordenação.
03	Gerenciador de carteira de ações	Capacidade de modelar, projetar, codificar e testar pequenos sistemas de software com base nos conceitos aprendidos nas disciplinas teóricas có-requisitos do módulo integrador	1. Arquivos de texto. 2. Arquivos binários e serialização. 3. Tratamento de exceções. 4. Padrão Façade. 5. Padrão MVC. 6. Testes de unidade. 7. Testes de aceitação. 8. Diagrama de classes de projeto. 9. Árvores binárias: representação e algoritmos. 10. Balanceamento de árvores binárias. 11. Documentação com Javadoc.
04	Gerenciamento de viagens em uma app pessoal	Capacidade de modelar, projetar, codificar e testar pequenos sistemas de software com base nos conceitos aprendidos nas disciplinas teóricas có-requisitos do módulo integrador.	1. Componentes da interface gráfica de usuário em Java. 2. Tratamento de eventos de interface em Java. 3. Uso de coleções da biblioteca padrão de Java. 4. Classes abstratas e herança polimórfica. 5. Padrão MVC. 6. Testes de unidade. 7. Testes de aceitação. 8. Diagrama de classes de projeto. 9. Tabelas hash: representação e algoritmos. 10. Grafos: representação e percurso. 11. Algoritmo de caminho mínimo para grafos.

3.1.4 Avaliação do Estudo Integrado

O segundo semestre letivo de 2017 durou 4 meses. Nos módulos teóricos ocorreram 3 avaliações teóricas, que correspondem as notas da I, II e III unidade. As avaliações teóricas eram compostas por questões objetivas, geralmente de múltipla escolha, e questões discursivas. O peso de cada avaliação é 10,0 (dez), e ao final das três unidades é efetuada a média aritmética. Os estudantes são aprovados obtendo média igual ou superior a 7,0 (sete).

No módulo integrador, são avaliados o produto de software e o desempenho dos estudantes nas sessões tutoriais, para cada um dos quatro problemas propostos. O desempenho é mensurado por assiduidade, pontualidade, participação e contribuição efetiva. O produto é avaliado por baremas, e considera o uso de conceitos de projeto de sistemas, programação orientada a objetos e estruturas de dados. A nota do produto inclui também o relatório escrito. O produto entregue corresponde a 70% da nota e o desempenho nos tutoriais corresponde a 30% da nota. Ao final das três unidades, é calculada a média aritmética. Para obter aprovação no componente curricular, a média aritmética deverá ser igual ou maior que 7,0 (sete).

3.1.5 Coleta de Dados

O estudo de caso foi conduzido ao longo do segundo semestre de 2017. Avaliamos o Estudo Integrado (EI) de Programação do curso de Engenharia da Computação da UEFS, componente curricular formado pelos módulos: Algoritmos e Programação II, Estrutura de Dados, Projeto de Sistemas e o Módulo Integrador (MI) de Programação. As turmas eram heterogêneas, compostas por 25 estudantes novatos e veteranos.

Inicialmente, para atender às questões éticas e preservando o direito ao anonimato dos participantes da pesquisa, apresentamos um Termo de Consentimento Livre e Esclarecido (TCLE), que está disponível no Apêndice A, que foi assinado pelos estudantes que desejaram participar. Este documento é a proteção legal e moral do pesquisador e dos participantes, posto que é a manifestação clara de concordância ou não em relação à participação na pesquisa. Os estudantes que não assinaram o termo não participaram da pesquisa.

O estudo de caso foi iniciado com a aplicação de um questionário pré-intervenção para identificar os dados demográficos dos estudantes, bem como suas impressões em relação à computação e, mais especificamente, à programação em si, conforme disponível no ApêndiceB. No MI de Programação, onde ocorrem as sessões tutoriais de PBL, um questionário foi aplicado ao final de cada um dos quatro problemas propostos para avaliar a motivação dos estudantes e suas percepções sobre a aprendizagem; a seção de motivação do questionário replica o instrumento IMMS – *Instructional Materials Motivation Survey* [Keller 2010], presentes no Apêndice C. IMMS é um instrumento projetado para medir reações a materiais de instrução autodirigidos,

embora possa ser adaptado para situações presenciais com foco nos materiais instrucionais. Das 36 questões originais do IMMS, três questões da categoria atenção explicitamente relacionadas a materiais instrucionais puramente textuais foram suprimidas por tratarem de aspectos não utilizados nesta abordagem.

Em cada Módulo Teórico, foi aplicado, ao final do semestre letivo, o questionário *Course Interest Survey* (CIS), proposto por Keller (2010), disponível no Apêndice D. O objetivo do CIS é mensurar a motivação dos estudantes em relação a um curso presencial específico. Assim como o IMMS, o CIS está embasado teoricamente com os conceitos motivacionais e teorias relacionadas ao Modelo ARCS. A utilização do CIS no nosso estudo de caso teve a finalidade de avaliar a motivação dos estudantes, considerando as categorias cognitivas do modelo ARCS, em relação às disciplinas de Estruturas de Dados, Projeto de Sistemas e Algoritmos e Programação II.

Além da motivação, também avaliamos a aprendizagem dos estudantes. Para isto, foi aplicado o questionário de habilidades de programação *Second CS1 Assessment* (SCS1). O SCS1, elaborado por Parker, Guzdial e Engleman (2016), é uma segunda versão isomórfica validada a partir do questionário *Foundational* CS1 (FCS1) desenvolvido por Tew e Guzdial (2011). Este instrumento avalia a compreensão dos estudantes em relação a conceitos introdutórios da ciência da computação e é aplicável em uma variedade de pedagogias e linguagens de programação. Aplicamos o questionário de habilidades de programação (SCS1) no início e no final do semestre, com o objetivo de mensurar os ganhos de aprendizagem em relação aos conceitos e habilidades de programação.

Mensuramos aprendizagem também através das notas dos estudantes. Para o MI de Programação, recuperamos as notas dos projetos em cada problema. Concomitantemente, para os Módulos Teóricos, recuperamos as médias finais de cada um deles.

Em relação aos dados qualitativos, foram efetuadas observações em sala de aula e entrevistas semiestruturadas. As observações foram realizadas em todas as aulas, com exceção de duas aulas das sessões tutorias, tanto nos módulos teóricos como nas sessões tutoriais. Observamos o comportamento dos estudantes, como a atenção, motivação e participação nas aulas ministradas pelos professores, bem como a discussão entre os estudantes nas sessões tutoriais. Além disso, foram realizadas entrevistas com quatro professores e seis estudantes no final do período letivo e no mês posterior ao período em questão, conforme disponível nos Apêndices E, F e G. Para reduzir o viés, foram utilizados o critério de desempenho de notas dos alunos entrevistados. Foram escolhidos dois alunos com desempenho excelente, mediano e fraco. Empregamos os códigos E para os estudantes, e P para os professores nos extratos das falas citadas na pesquisa. Ao final, realizamos a triangulação entre os dados qualitativos e quantitativos na tentativa de assegurar uma compreensão em profundidade dos resultados e, assim, apoiar a interpretação dos resultados.

3.1.6 Análise de Dados

Os dados qualitativos de entrevistas e observações contabilizaram 42 arquivos de texto. Sobre este material, inicialmente realizamos a codificação aberta através de análise de conteúdo, chegando a 143 códigos. Em seguida, abstraímos as categorias e recodificamos os dados axialmente a partir de dez categorias emergentes. São elas: Avaliação, Conceitos, Dinâmica do MI, Habilidades, Módulo Teórico, Motivação, Organização do Estudo Integrado, Paradigma de Programação, Método PBL vesus Método Tradicional e Problemas. Finalmente, construímos memorandos de cinco categorias que consideramos relevantes encontradas na codificação com o propósito de responder às questões de pesquisa. As cinco categorias foram descritas como: Conceitos, Habilidades, Organização do Estudo Integrado, Problemas e Soft Skills. Utilizamos a ferramenta NVIVO para auxiliar em todas as etapas desde a codificação até a geração de memorandos.

Para a análise quantitativa dos dados, formados pelos questionários do IMMS e do CIS, os dados foram tabulados e foram geradas estatísticas descritivas e gráficos de barras empilhadas de cada construto dos questionários. Utilizamos os gráficos box-plots para descrever a concentração e a dispersão dos dados das categorias de motivação em cada uma dos problemas, no IMMS, e módulos teóricos, no CIS. Em seguida, realizamos o teste de aderência de Kolmogorov-Sminorv para verificar se os conjuntos de dados aderiam à distribuição normal, o que se confirmou. Utilizamos o teste ANOVA para fazer comparações e verificar se houve diferença entre as categorias motivacionais tanto entre os problemas (IMMS) quanto entre os módulos teóricos (CIS). Havendo diferença entre as categorias motivacionais, utilizamos o teste *post-hoc* de Tukey para determinar os pares de problemas (no caso do IMMS) e os pares de módulos teóricos (no caso do CIS) que apresentaram diferenças significativas entre si. A significância dos testes foi determinada considerando o nível de significância de $0,05$.

Além da motivação, analisamos as estatísticas para as variáveis de aprendizagem. Para tal, recuperamos as médias finais dos estudantes nas avaliações dos módulos teóricos, do módulo integrador, e das provas do SCS1 pré- e pós-intervenção. Computamos estatísticas descritivas para estas variáveis, obtendo a média e o desvio padrão para cada uma delas. Também utilizamos box-plots e diagramas de barras de erros para descrever a concentração e a dispersão destas variáveis. Para o SCS1 pré- e pós-intervenção, efetuamos o teste de hipótese t de Student para amostras emparelhadas para avaliar se houve diferença significativa entre as médias das amostras.

Ainda, para medir as relações entre as variáveis de interesse de motivação e aprendizagem, computamos correlações entre elas. Inicialmente realizamos o teste de aderência Kolmogorov-Sminorv para verificar se os conjuntos de dados aderiam à distribuição normal. Aplicamos correlação de Pearson para as situações em que os dados seguem uma distribuição normal, e correlação de Spearman para situações

contrárias, ou seja, que não seguem uma distribuição normal. Apenas as notas dos problemas do MI de programação não apresentaram distribuição normal.

Empregamos a correlação de Pearson entre as notas finais dos módulos teóricos, do módulo integrador e do SCS1 pós-intervenção. Do mesmo modo, em relação as notas finais dos módulos teóricos e do módulo integrador, e das notas finais dos módulos teóricos entre si. Usamos também a correlação de Pearson para mensurar a relação entre motivação e aprendizagem nos Módulos Teóricos. Entretanto, em relação à motivação e à aprendizagem no MI, aplicamos a correlação de Spearman, pois os dados não aderiam à distribuição normal.

Finalmente, ainda avaliando a aprendizagem, analisamos a evolução das notas dos estudantes nos problemas do MI. Para esse fim, produzimos estatísticas descritivas de mediana e amplitude interquartis e o gráfico box-plot para cada problema. Como as notas dos problemas não apresentaram normalidade, utilizamos a análise de variância de Friedman para testar se houve diferenças significativas entre as notas finais dos problemas no MI. Havendo diferença, utilizamos o teste não paramétrico de Wilcoxon com correção do valor-p para identificar em quais pares de problemas esta diferença foi significativa.

3.1.7 Validade e Confiabilidade

Toda pesquisa se preocupa em produzir conhecimento válido e confiável de maneira ética. Independente do tipo de pesquisa, a validade e a confiabilidade são preocupações que podem ser abordadas através de uma atenção cuidadosa à conceituação de um estudo e à maneira como os dados são coletados, analisados e interpretados, e também na forma como os resultados são apresentados [Merriam 2009]. Logo, esta seção objetiva discutir a confiabilidade e as possíveis ameaças à validade em todas etapas deste trabalho. Utilizamos os procedimentos metodológicos baseados nas orientações de Merriam (2009) e Creswell (2002).

A confiabilidade em relação aos dados qualitativos foi obtida seguindo um procedimento cuidadoso de codificação proposto por Creswell (2002). Inicialmente, fizemos a coleta dos dados através das entrevistas e observações. Em seguida, realizamos codificação aberta e sintetizamos os conceitos pré-definidos. Após esta etapa, tentamos encontrar elementos observando onde as coisas se repetiam e chegamos às categorias mais abstratas. Finalmente, escrevemos os memorandos. Neste processo sistemático, procuramos seguir um rigor metodológico com a finalidade de evitar desvios na definição de códigos ou possíveis erros na interpretação dos dados.

Capítulo 4

Resultados

Neste capítulo são apresentados os resultados da pesquisa. Inicialmente, relatamos a nossa experiência nas sessões tutoriais e nos módulos teóricos. A seguir, apresentamos os resultados dos níveis motivacionais a partir dos dados coletados através dos questionários IMMS e CIS, para as quatro categorias do Modelo ARCS. Posteriormente, reunimos os resultados de aprendizagem utilizando as médias finais dos estudantes nos módulos teóricos, no módulo integrador e das provas do SCS1. Por fim, demonstramos as relações entre motivação e aprendizagem nos módulos teóricos e no módulo integrador.

4.1 Nossa Experiência

No MI, reunimos os resultados das experiências para cada um dos quatro problemas propostos. Ao final dos problemas os estudantes deveriam entregar o relatório e o código implementado.

As sessões tutoriais seguem regras pré-estabelecidas obedecendo a dinâmica da metodologia PBL. No início da sessão, os estudantes escolhem entre eles, o coordenador, o secretário da mesa e o secretário do quadro. O coordenador da sessão tem a função de liderar o grupo, estimular os participantes, manter a dinâmica, direcionar e orientar a discussão sobre as possíveis soluções para os problemas. O secretário da mesa faz anotações sobre os assuntos discutidos, ajuda a ordenar as ideias em relatório. O secretário do quadro faz anotações na lousa contendo as ideias, fatos, questões e metas de aprendizagem levantados pelos estudantes. Posteriormente, os problemas são apresentados aos estudantes, que leem e interpretam o problema em conjunto, levantam ideias e fatos mais relevantes para solucionar o problema. Após essa fase, elaboram questões com vistas a solucionar o problema, e estabelecem um plano de ação que permita resolver as questões levantadas, criando as metas de aprendizagem. Ao final das sessões tutoriais, o secretário da mesa gera e compartilha com o grupo o relatório concebido com os resultados da sessão.

O Problema 1 foi apresentado aos estudantes na primeira aula do MI de Programação. Nesse problema, os estudantes projetaram e implementaram um sistema para efetuar o cadastramento biométrico eleitoral. Para isso, o problema apresentava a situação-problema, com *users stories* e um modelo conceitual a partir do qual os estudantes deveriam projetar e desenvolver uma solução em três semanas. Para solucionar o problema, era necessária a aquisição de conceitos introdutórios de orientação a objetos, listas encadeadas, padrões *Iterador* e MVC.

Observamos que, apesar da grande quantidade de novos conceitos existentes no Problema 1, a maioria dos estudantes conseguiu entregar o projeto em tempo hábil. Compreender o modelo conceitual e o diagrama de classes pareceu relativamente simples, porém, projetar os sistemas utilizando esses instrumentos revelou-se mais difícil. Os conceitos trabalhados nos módulos teóricos ocorreram concomitantemente com o desenrolar das discussões sobre os problemas nas sessões tutoriais, colaborando para a elucidação das dúvidas dos estudantes. Contudo, observamos que em alguns momentos os estudantes sentiam necessidade de buscar outros materiais para alcançar a aprendizagem.

O Problema 2 foi apresentado aos estudantes na aula que seu deu após finalização do prazo de entrega do Problema 1. Nesse problema, os alunos desenvolveram uma ferramenta para leilão eletrônico. A partir da situação problema, *users stories* e diagrama de classes, os estudantes projetaram e implementaram um sistema. Para isso, era necessário utilizar novos conceitos de estruturas de dados e de orientação a objetos, como herança, além de filas com prioridades, métodos de ordenação e testes de unidade.

Os problemas são elaborados com complexidade crescente. Nesse problema, os estudantes deveriam construir seus próprios testes de unidade, elaborar um novo modelo conceitual, utilizar filas com prioridades e escolher um método de ordenação e a partir disso, escrever o código-fonte. Notamos que os estudantes apresentaram dificuldades no levantamento e análise de requisitos e na elaboração do diagrama de classes, necessitando a intervenção do tutor em alguns momentos, no sentido de redirecionar a discussão sem desviar do eixo proposto pelo problema.

Observamos dificuldades na escolha do algoritmo de ordenação e na elaboração dos testes unitários. Por outro lado, percebemos que os alunos captaram a relevância dos conceitos trabalhados, principalmente em relação ao conceito de herança e a possibilidade de reuso do código, e a elaboração dos testes unitários, que permitem a identificação de falhas no fluxo da informação do produto que está sendo elaborado, trazendo confiança para os estudantes. A maioria dos estudantes entregou o projeto no prazo estipulado de quatro semanas.

No Problema 3, os estudantes projetaram e implementaram um gerenciador de carteira de ações. Este problema apresentava apenas especificações e os *users stories*. Introduziu conceitos de árvores balanceadas, leitura e escrita de arquivos, documentação com *Javadoc*, interfaces e padrão *Façade*. Diferentemente dos outros problemas, os estudantes deveriam implementar todos os testes de unidade para as classes

utilizadas, com exceção da *façade*. Todas as classes e os testes deveriam estar documentados utilizando o padrão *javadoc*. Além do código, os estudantes deveriam apresentar ainda o diagrama de classes de projeto do sistema.

Nesse problema, observamos que os estudantes apresentaram muita dificuldade na sua interpretação. Atribuímos, principalmente, pela dificuldade na compreensão do domínio. A falta de conhecimento sobre o mercado de ações e a bolsa de valores foram os principais motivos, além das dificuldades naturais relacionados aos conceitos de orientação a objetos, estrutura de dados e projetos de sistemas. A maioria dos estudantes tiveram dificuldades na implementação do balanceamento da árvore binária AVL. Porém, a maior dificuldade encontrada foi a compreensão e utilização do padrão *façade*.

Boa parte dos estudantes não concluiu o Problema 3 em tempo hábil ou entregou o projeto incompleto. Uma das possíveis explicações é a complexidade elevada e o domínio desconhecido.

No Problema 4, os estudantes projetaram e implementaram um aplicativo para gerenciamento de viagens. Diferentemente dos outros problemas, esse continha apenas a situação-problema e suas especificações. Para solucionar esse problema, era necessário a utilização de uma interface gráfica, tabelas *hash* e conceitos de grafos. Além da implementação: dos testes de unidade para todas as classes, testes de aceitação e a documentação em *Javadoc*. Ao final, os estudantes deveriam elaborar os *users stories* e o diagrama de classes de projeto do sistema. Essas informações deveriam constar em relatório, além do código fonte.

Finalmente, no quarto problema, obtivemos os melhores resultados. Os estudantes não apresentaram dificuldades com conceitos de orientação a objetos e estrutura de dados. Notamos dificuldades pontuais em projetos de sistemas. Os estudantes demonstraram dificuldades na construção do diagrama de classes de projetos, principalmente com as associações e navegabilidade entre classes. Por outro lado, o Problema 4 revelou-se uma experiência estimulante para os estudantes. Uma das possíveis explicações seria a utilização da interface gráfica, permitindo o *feedback* visual imediato, e o domínio escolhido. O domínio do problema tratava do desenvolvimento de um aplicativo similar a um já existente no mercado, que permitia o planejamento de viagens.

Nos módulos teóricos, observamos a dinâmica e o comportamento dos estudantes nas aulas de Algoritmos e Programação II, Estruturas de Dados e Projetos de Sistemas. Foram abordados os conteúdos planejados anteriormente com o objetivo de introduzir conceitos que seriam tratados também nos problemas do MI de Programação.

Em Algoritmos e Programação II, o professor utilizou uma metodologia tradicional de ensino, ministrando aulas em forma de aulas expositivas dialogadas. Ao final das unidades, era aplicado uma avaliação escrita com objetivo de quantificar a aprendizagem do estudantes. As avaliações sempre eram corrigidas em sala de aula, onde o professor e alunos discutiam as possíveis soluções para as questões. Na Unidade I,

foram introduzidos conceitos iniciais sobre o paradigma de programação orientado a objetos, conceitos como classes, objetos, atributos e métodos. Em certos momentos, os estudantes tiravam dúvidas sobre alguns conceitos que não haviam sido trabalhados até o momento. Observamos que isto se revelou uma prática comum durante as aulas e geralmente acontecia devido à demanda de conhecimentos necessários para a resolução dos problemas do MI. Na Unidade II foram estudados conceitos como encapsulamento, herança, polimorfismo, relacionamentos, composição, dentre outros. Na Unidade III foram apresentados conceitos de classes abstratas, interface gráfica, atributos e métodos estáticos, estrutura de dados básicas, dentre outros. O professor sempre explicava os conceitos com exemplos práticos, que eram codificados na sala de aula para que os estudantes observassem o comportamento e esclarecessem suas dúvidas. De modo geral, os estudantes demonstraram facilidade no entendimento desses conceitos.

Em Estrutura de Dados, a metodologia utilizada foi semelhante à de Algoritmos e Programação II, pois era o mesmo professor que lecionava os dois módulos teóricos. Aqui também foram aplicadas avaliações ao final da cada unidade, posteriormente corrigidas em sala de aula. Na Unidade I, foram introduzidos conceitos iniciais como listas, filas, pilhas, dentre outros. Na Unidade II, foram apresentados conceitos como árvores, árvores binárias, balanceamento de árvores, além de métodos de busca e ordenação. Na Unidade III, conceitos mais avançados de estruturas de dados como grafos e tabelas *hash*. Nesse módulo teórico observamos que os estudantes participavam muito das aulas, sempre esclarecendo dúvidas com o professor. As aulas tornavam-se mais dinâmicas com discussões interessantes acerca dos assuntos. Apesar da grande demanda de conteúdos, em pouco tempo, percebemos através do comportamento e de relatos dos estudantes, um maior engajamento com esse módulo teórico.

Em Projeto de Sistemas o professor utilizou uma metodologia tradicional, com aulas expositivas. Na Unidade I foram ministradas aulas com alguns conceitos relacionados a aplicação de técnicas e ferramentas de projetos de software orientado a objetos. Na Unidade II foram trabalhados principalmente conceitos relacionados à interpretação e implementação de testes unitários, atendendo ao pedidos dos estudantes no sentido de apoiá-los na resolução do Problema 2 do MI. Na Unidade III, a mais comprometida em termos de desenvolvimento do conteúdo programático por causa do grande número de paralisações e feriados que ocorreram nos dias das aulas, os estudantes fizeram seminários sobre conceitos relacionados a padrões de projeto. De modo geral, neste módulo teórico, observamos alguns problemas pontuais como estudantes dispersos, um grande número de ausências, além de queixas em relação ao entendimento dos conceitos. A maioria dos estudantes achava o conteúdo muito abstrato e sentiam-se inseguros na resolução das atividades e das avaliações.

4.2 Motivação

Reunimos os resultados dos questionários IMMS para cada um dos quatro problemas propostos a partir das categorias de Atenção, Relevância, Confiança e Satisfação do modelo ARCS e os apresentamos a seguir. As respostas aos questionários são baseadas em uma escala de Likert de cinco níveis. Para facilitar a interpretação, consideramos que houve concordância quando as respostas foram Concordo Parcialmente ou Totalmente e discordância quando as respostas foram Discordo Parcialmente ou Totalmente. Também agrupamos os resultados CIS para cada um dos módulos teóricos baseados nas categorias Atenção, Relevância, Confiança e Satisfação. Foram utilizadas a escala de Likert á partir das respostas aos questionários. Neste caso utilizamos uma escala diferente, com as opções Não é verdade, Um pouco verdadeiro, Moderadamente verdadeiro, Principalmente verdadeiro e Muito verdadeiro.

Para cada categoria do ARCS no IMMS, produzimos um escore a partir da soma dos valores das respostas de cada questão, convertendo a escala nominal para uma escala numérica variando de 1 (Discordo Totalmente) a 5 (Concordo Totalmente), e tratando as questões fraseadas na forma negativa com uma escala invertida de 5 a 1. No CIS, para cada categoria do ARCS, utilizamos a escala de 1 (Não é verdade) a 5 (Muito verdadeiro). Os escores foram normalizados entre 1 e 5, dividindo cada soma pelo número de questões de cada categoria. Confirmamos a normalidade de todos os escores através do teste de aderência de Kolmogorov-Smirnov.

4.2.1 Atenção

Esta subseção descreve os resultados da categoria Atenção tanto no módulo integrador como nos módulos teóricos.

Atenção no Módulo Integrador

A Figura 4.1 apresenta os resultados sobre a categoria Atenção nos problemas do MI de Programação. Em todos os problemas, os estudantes afirmaram que aprenderam coisas surpreendentes ou inesperadas (A8 – concordância de 65% em P1, 65% em P2, 50% em P3, 79% em P4). A maior parte deles concordou que havia coisas interessantes que chamavam atenção (A1 – 74% em P1, 65% em P2, 89% em P4), e achou o conteúdo dos problemas atraente (A2 – 52% em P1, 52% em P2, 84% em P4). Entretanto, a maioria dos estudantes considerou o Problema 3 pouco atraente (A4 – 68%), tampouco gostou da maneira como a informação foi organizada (A5 – 65%), além de considerar que o conteúdo foi abstrato, dificultando a atenção nas sessões tutoriais (A3 — concordância de 64%). O Problema 4 obteve os melhores resultados. Uma parcela significativa de estudantes concordou que havia coisas que estimulavam a criatividade (A6 – 84%), não achou o conteúdo abstrato (A3

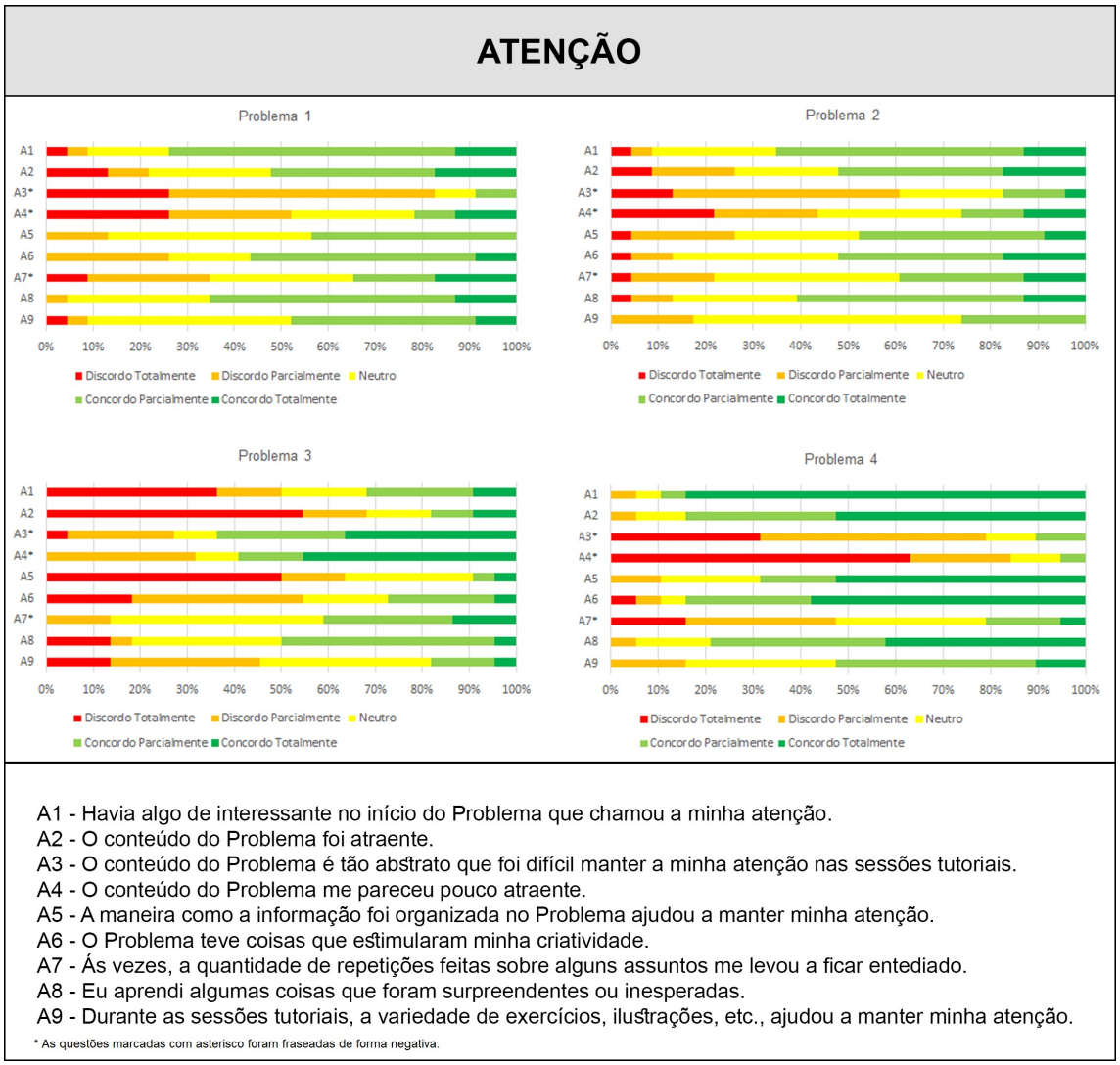


Figura 4.1: Resultados para a categoria Atenção nos Problemas

– discordância de 79%), e julgou que a maneira como a informação foi organizada contribuiu para manter a atenção (A5 – concordância de 69%).

A Figura 4.2 traz os box-plots do escore da categoria Atenção nos problemas. Observamos que a mediana da atenção obteve resultados positivos nos Problemas 1 e 2 (escores entre 3 e 4), bastante positivos no Problema 4 (escore acima de 4), e negativos no Problema 3 (entre 2 e 3). Sobre a dispersão dos dados, nos Problemas 1 e 2, notamos uma dispersão menor que a do Problema 3 e maior que a do Problema 4. Assim, os escores sofreram menos variações nos Problemas 1 e 2, o que é ainda mais evidente no Problema 4. A dispersão dos escores é maior no Problema 3.

Utilizamos a análise de variância (ANOVA) para testar se o escore de Atenção varia significativamente entre os Problemas, o que foi confirmado ($F = 18,01$, $p < 0,001$). O resultado evidencia que a distribuição de pelo menos um dos grupos difere

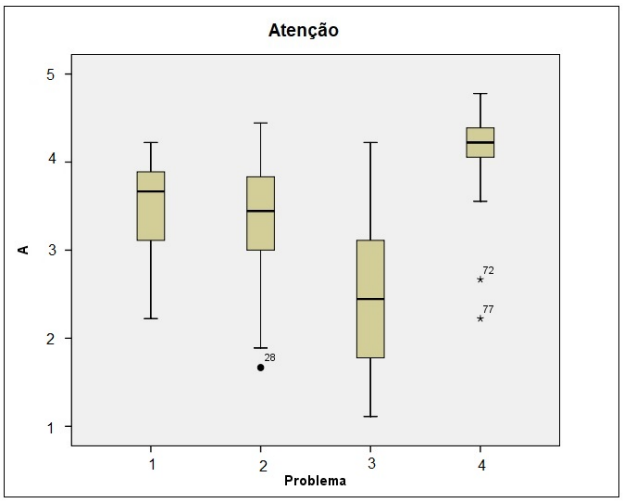


Figura 4.2: Box-Plot dos escores de Atenção nos Problemas

das demais, mas não indica entre quais grupos a diferença é significativa. Para se determinar quais pares de escores são diferentes entre si, adotamos o Teste *post-hoc* de Tukey. Houve diferenças estatisticamente significativas (valor-p $< 0{,}05$) entre os Problemas: a) 1 e 3; b) 1 e 4; c) 2 e 3; d) 2 e 4; e) 3 e 4. Os níveis de atenção foram mais altos no Problema 4, seguidos pelos Problemas 1 e 2 e mais baixos no Problema 3.

Atenção nos Módulos Teóricos

A Figura 4.3 apresenta os resultados sobre a categoria Atenção nos módulos teóricos. Em Estrutura de Dados e Algoritmos e Programação II, a maioria dos estudantes sentiu-se entusiasmado (A1 – concordância de 80% em Algoritmos e Programação II, 90% em Estrutura de Dados), e achou atraente os assuntos estudados nessas disciplinas (A2 – 75% em Algoritmos e Programação II, 84% em Estrutura de Dados). Além disso, a maior parte deles demonstrou curiosidade estimulados pelas questões trabalhadas em sala (A8 – em Algoritmos e Programação II 60%, e 80% em Estrutura de Dados). Entretanto, em Projeto de Sistemas uma parcela significativa dos estudantes não estava entusiasmado (A1 – 50%), nem curioso em relação aos conteúdos (A4 – 65%) e as questões ou problemas aplicados em sala (A8 – 50%). Ainda afirmaram ficarem distraídos durante as aulas (A7 – 60% em Projeto de Sistemas).

A Figura 4.4 apresenta os box-plots do escore da categoria Atenção nos módulos teóricos. Observamos que a mediana da atenção alcançou resultados positivos nas disciplinas de Estrutura de Dados e Algoritmos e Programação II (escores entre 3 e 5), e negativos em Projeto de Sistemas (escores entre 1 e 3). Em relação à dispersão dos dados, em Algoritmos e Programação II e Projeto de Sistemas observamos uma dispersão maior que em Estrutura de Dados, onde os escores sofreram menos variações.

Utilizamos ANOVA para testar se a Atenção varia significativamente entre os módulos teóricos, o que foi confirmado com $F = 22,98$, $p < 0,001$. O Teste *post-hoc* de Tukey aponta que os resultados apresentaram diferenças significativas entres os módulos teóricos: a) Algoritmos e Programação II e Projeto de Sistemas; b)Estrutura

Figura 4.3: Resultados para a categoria Atenção nos Módulos Teóricos

de Dados e Projeto de Sistemas. Os níveis de atenção foram mais altos em Algoritmos e Programação II e Estrutura de Dados e mais baixos em Projeto de Sistemas.

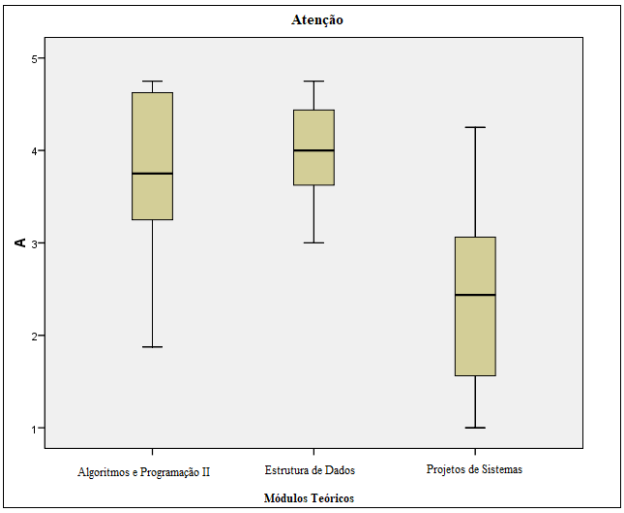


Figura 4.4: Box-Plot dos escores de Atenção nos Módulos Teóricos

4.2.2 Relevância

Esta subseção descreve os resultados da categoria Relevância tanto no módulo integrador como nos módulos teóricos.

Relevância no Módulo Integrador

Na Figura 4.5, a categoria Relevância nos problemas é ilustrada. Estudantes reconheceram a utilidade do conteúdo adotado em todos os problemas (R9 – concordância de 87% em P1, 88% em P2, 64% em P3, 95% em P4). Constataram a importância dos conteúdos, principalmente para aqueles que estão aprendendo a programar (R4 – 78% em P1, 88% em P2, 89% em P4). A percepção da relevância fica menos evidente no Problema 3, pois a maioria não relacionou os conteúdos aprendidos com coisas que já foram feitas, vistas ou pensadas (R8 – discordância de 68% em P3) e considerou que o conteúdo não foi relevante para atender suas necessidades (R7 – 86% em P3). Em contrapartida, no Problema 4, a percepção da relevância é mais significativa pois uma parcela expressiva dos estudantes reconheceu os benefícios de aprender os conceitos com o conteúdo e o estilo usados (R6 – concordância de 89% em P4), e concordaram (R8 – 84% em P4) ao relacionar os conteúdos aprendidos com coisas que já foram vistas, feitas ou pensadas.

A Figura 4.6 exibe o box-plot do escore da categoria Relevância nos problemas. A mediana da relevância é maior nos problemas 1, 2 e 4, menor no problema 3, porém positiva em todos os problemas.

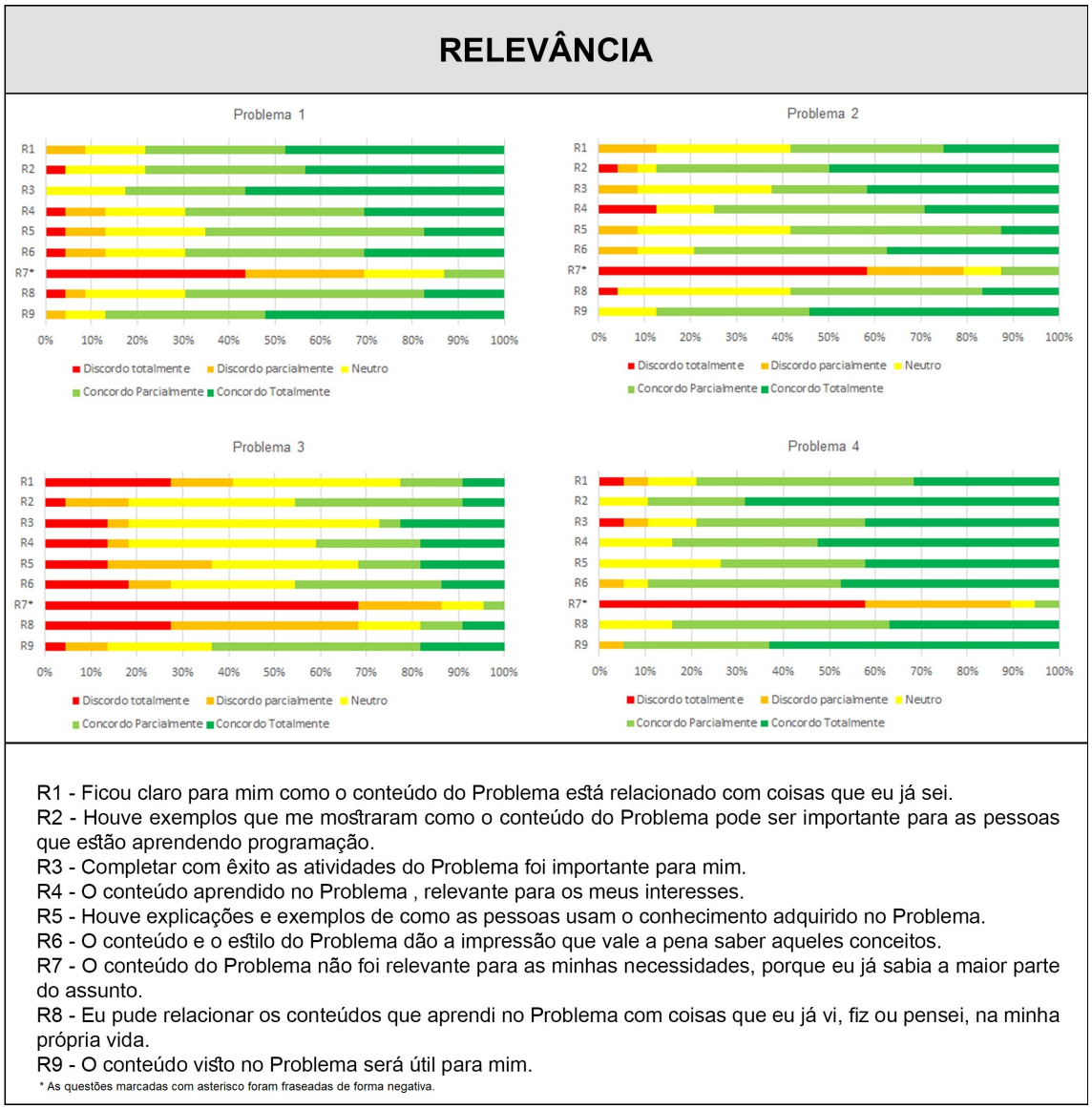


Figura 4.5: Resultados para a categoria Relevância nos Problemas

Utilizamos ANOVA para testar se a Relevância sofre mudanças significativas entre os Problemas, confirmado com $F = 13,67$, $p < 0,001$. Utilizamos o Teste *post hoc* de Tukey para identificação das diferenças específicas entre os pares de escores. O resultado demonstrou diferenças significativas entre os Problemas: a) 1 e 3; b) 2 e 3; c) 3 e 4. Os estudantes entenderam que todos os problemas foram relevantes, porém, consideraram mais relevantes os Problemas 1, 2 e 4.

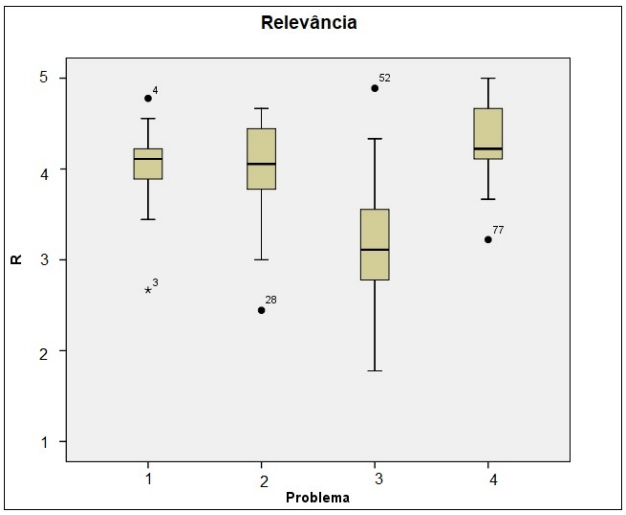


Figura 4.6: Box-Plot dos escores de Relevância nos Problemas

Relevância nos Módulos Teóricos

A Figura 4.7 demonstra os resultados sobre a categoria Relevância nos módulos teóricos. Estudantes reconheceram a utilidade dos conteúdos aprendidos em todos os módulos teóricos (R1 – concordância de 95% em Algoritmos e Programação II, 100% em Estrutura de Dados, 75% em Projetos de Sistemas). Perceberam com clareza os benefícios (R9 – 90% em Algoritmos e Programação II, 95% em Estrutura de Dados , 55% em Projeto de Sistemas). A percepção da relevância é muito evidente em Algoritmos e Programação II e Estrutura de Dados, a maioria dos estudantes reconheceram a importância das disciplinas para a realização dos seus objetivos (R7 – 90% em Algoritmos e Programação II, 100% em Estrutura de Dados). Porém, em Projeto de Sistemas isso não se revela com a mesma intensidade, uma pequena parcela dos estudantes reconheceu a importância da disciplina (R7 – 35%). Ainda, uma parcela significativa dos estudantes afirmou que os alunos não participavam ativamente das aulas (R6 – 50% em Projeto de Sistemas).

A Figura 4.8 apresenta os box-plots do escore da categoria Relevância nos módulos teóricos. A mediana da relevância é maior em Algoritmos e Programação II e Estrutura de Dados, menor em Projeto de Sistemas, todavia positiva em todas as disciplinas. Os escores sofreram variações consideráveis em Algoritmos e Programação II e Projeto de Sistemas. Diferentemente de Algoritmos e Programação II, em que os escores variaram positivamente acima da neutralidade, Projeto de Sistemas apresentou variações com desvios para abaixo da neutralidade.

Utilizamos ANOVA para testar se a Relevância sofre mudanças significativas entre os módulos teóricos, confirmado com $F = 26,43$, $p < 0,001$. O Teste *post-hoc* de Tukey foi utilizado para detectar quais pares de escores são diferentes entre si. Encontramos diferenças estatisticamente significativas entre os módulos teóricos: a)

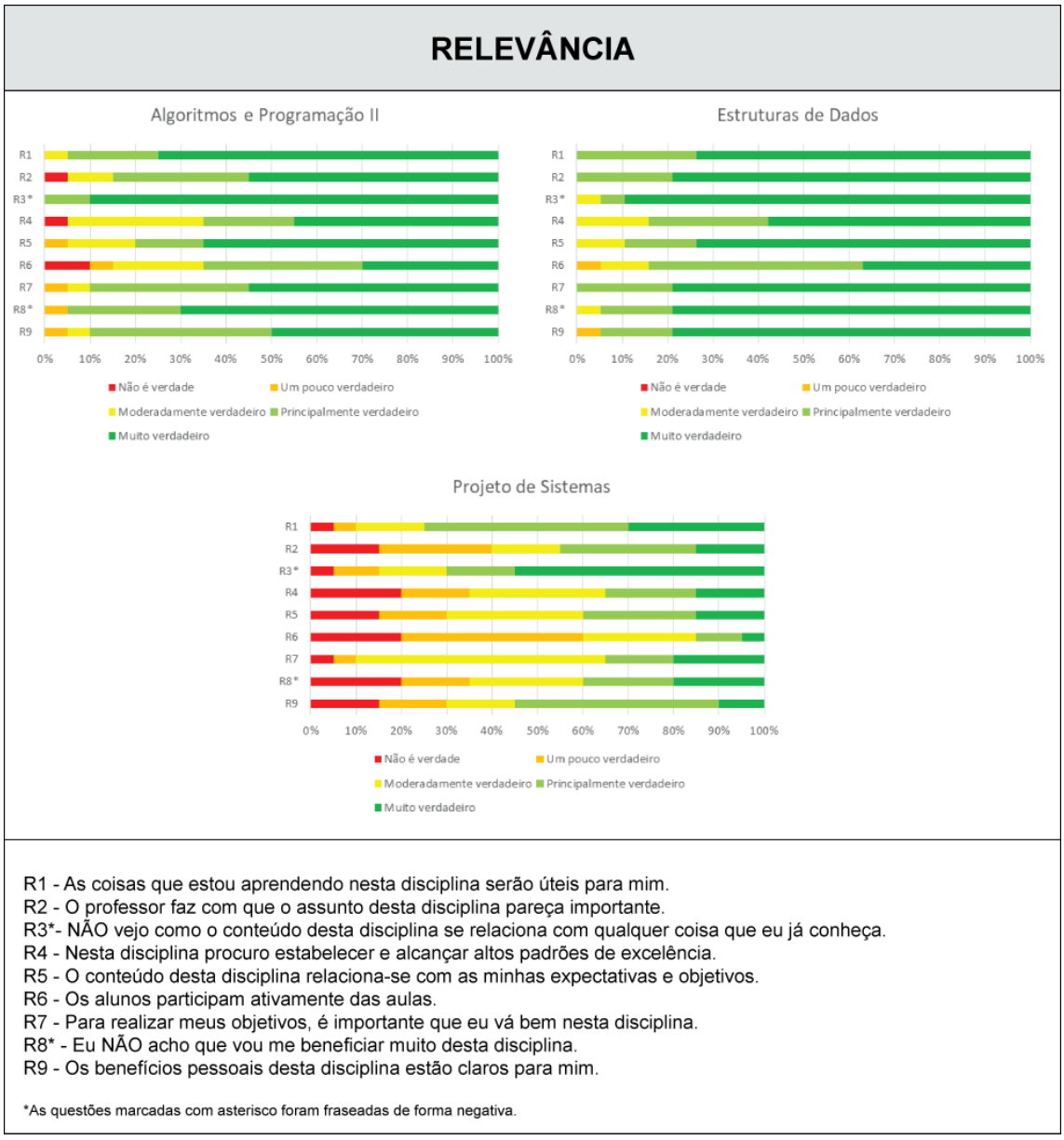


Figura 4.7: Resultados para a categoria Relevância nos Módulos Teóricos

Algoritmos e Programação II e Projeto de Sistemas; b) Estrutura de Dados e Projeto de Sistemas. Os estudantes acharam relevantes todos os módulos teóricos, porém esta percepção é mais evidente em Algoritmos e Programação II e Estrutura de Dados.

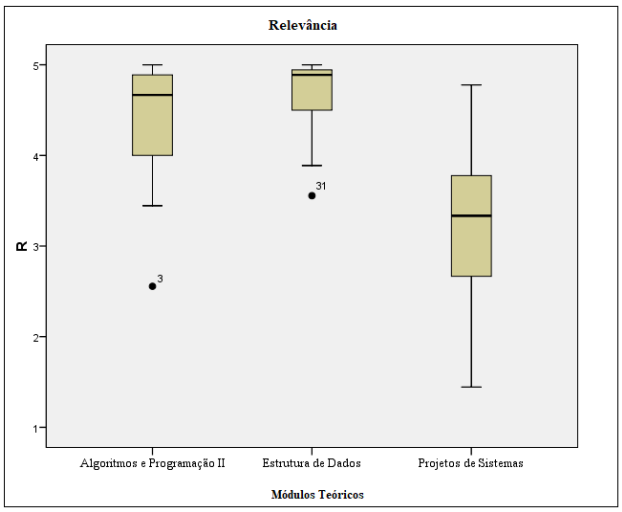


Figura 4.8: Box-Plot dos escores da Relevância nos Módulos Teóricos

4.2.3 Confiança

Esta subseção descreve os resultados da categoria Confiança tanto no módulo integrador como nos módulos teóricos.

Confiança no Módulo Integrador

A Figura 4.9 apresenta os resultados sobre a categoria Confiança nos problemas do MI de Programação. Os estudantes concordaram que após as primeiras sessões tutoriais sentiram-se mais confiantes do que deviam aprender (C3 – 74% em P1, 71% em P2, 63% em P4) e perceberam que eram capazes de passar na avaliação (C7 – 74% em P1, 74% em P2, 69%e em P4). Observamos que o Problema 3 não despertou a confiança dos estudantes da mesma maneira que os outros problemas, visto que a maioria qualificou os assuntos como mais difíceis do que gostariam que fossem (C2 – 91%), não conseguiram compreender como algumas coisas eram feitas (C8 – 86%), bem como não se sentiram confiantes do que realmente deveriam aprender (C3 – discordância de 77%).

Na Figura 4.10, está representado o box-plot do escore da dimensão Confiança nos problemas. Observamos que a mediana da confiança alcançou resultados positivos nos problemas 1, 2 e 4. O problema 3 apresentou mediana negativa (abaixo de 3). A dispersão dos dados é homogênea nos Problemas 1 e 2. Os Problemas 3 e 4 apresentam dispersão maior, indicando uma maior variabilidade nos escores.

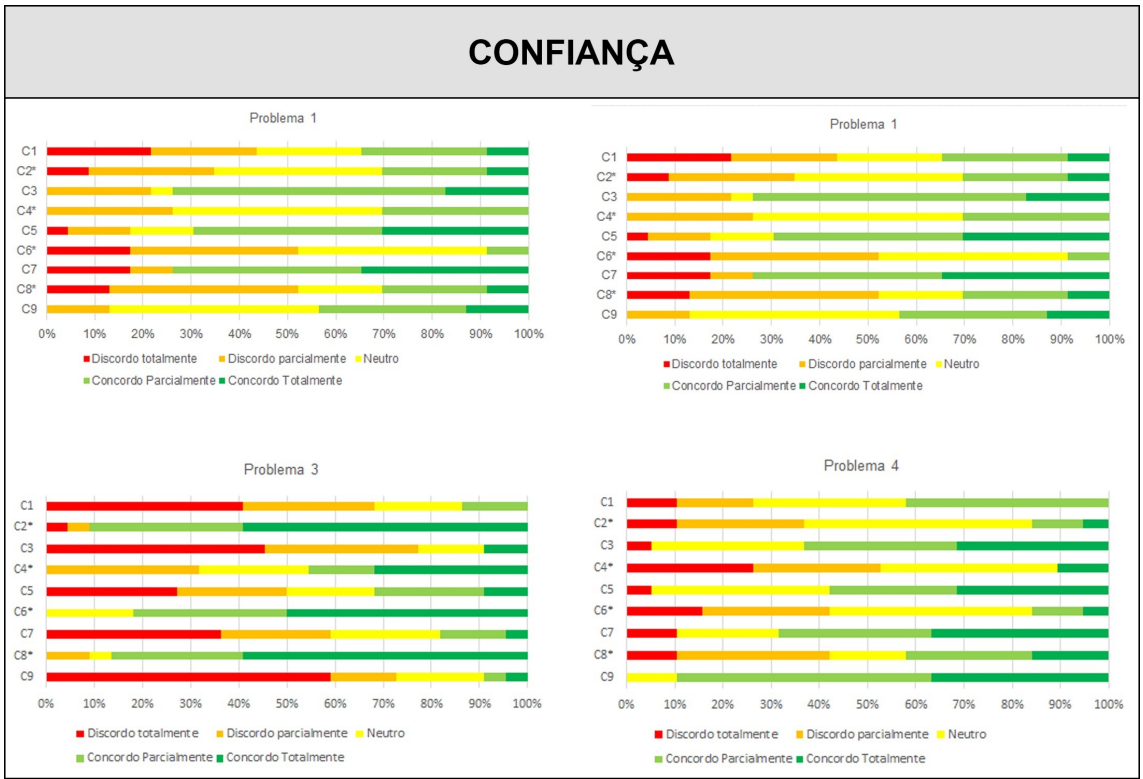


Figura 4.9: Resultados para a categoria Confiança nos Problemas

Usamos ANOVA para testar se a Confiança varia significativamente entre os Problemas, o que foi confirmado com $F = 16,84$, $p < 0,001$. O Teste *post-hoc* de Tukey revela que os resultados apresentaram diferenças significativas entres os grupos, constatado entre os Problemas: a) 1 e 3; b) 2 e 3; c) 3 e 4. Os estudantes permaneceram confiantes nos Problemas 1, 2 e 4, contudo não se mantiveram assim no Problema 3.

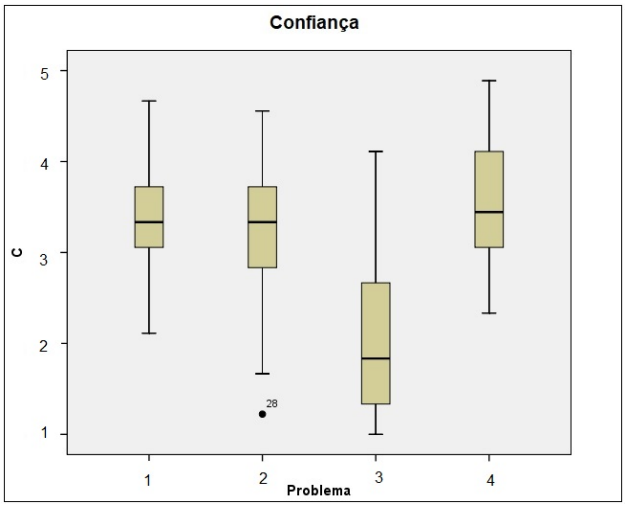


Figura 4.10: Box-Plot dos escores de Confiança nos Problemas

Confiança nos Módulos Teóricos

Na Figura 4.11, está ilustrada a categoria Confiança nos módulos teóricos. A maioria dos estudantes entende que o sucesso nas disciplinas teóricas depende deles (C3 – 100% em Algoritmos e Programação II, 95% em Estrutura de Dados, 60% em Projeto de Sistemas). Os estudantes se sentem confiantes (C1 – 65% em Algoritmos e Programação II, 79% em Estrutura de Dados) e acreditam que seu esforço será recompensado com o sucesso nas disciplinas teóricas (C6 – 85% em Algoritmos e Programação II, 89% em Estrutura de Dados). Em contrapartida, em Projeto de Sistemas, eles não se sentem tão confiantes assim (C1 – 20%) e tampouco acreditam que seu esforço será recompensado a partir da energia empregada (C6 – 25%).

A Figura 4.12 exibe o box-plot do escore da categoria Confiança nos módulos teóricos. A mediana da confiança é bastante alta em Algoritmos e Programação II e Estrutura de Dados. Por outro lado, Projeto de Sistemas apresentou mediana negativa. A dispersão dos dados é homogênea em Algoritmos e Programação II e Estrutura de Dados. Projeto de Sistemas apresentou uma maior variabilidade dos escores transitando entre valores positivos e negativos.

Utilizamos ANOVA para testar se a Confiança varia significativamente entre os módulos teóricos. Foi confirmado com $F = 17,75$, $p < 0,001$. O Teste *post-hoc* de Tukey demonstra que os resultados apresentaram diferenças significativas entre os módulos teóricos: a) Algoritmos e Programação II e Projeto de Sistemas; b) Estrutura de Dados e Projeto de Sistemas. Os estudantes estavam bastante confiantes em Algoritmos e Programação II e Estrutura de Dados, porém, em Projeto de Sistemas, a maioria dos estudantes não se sentia confiante.

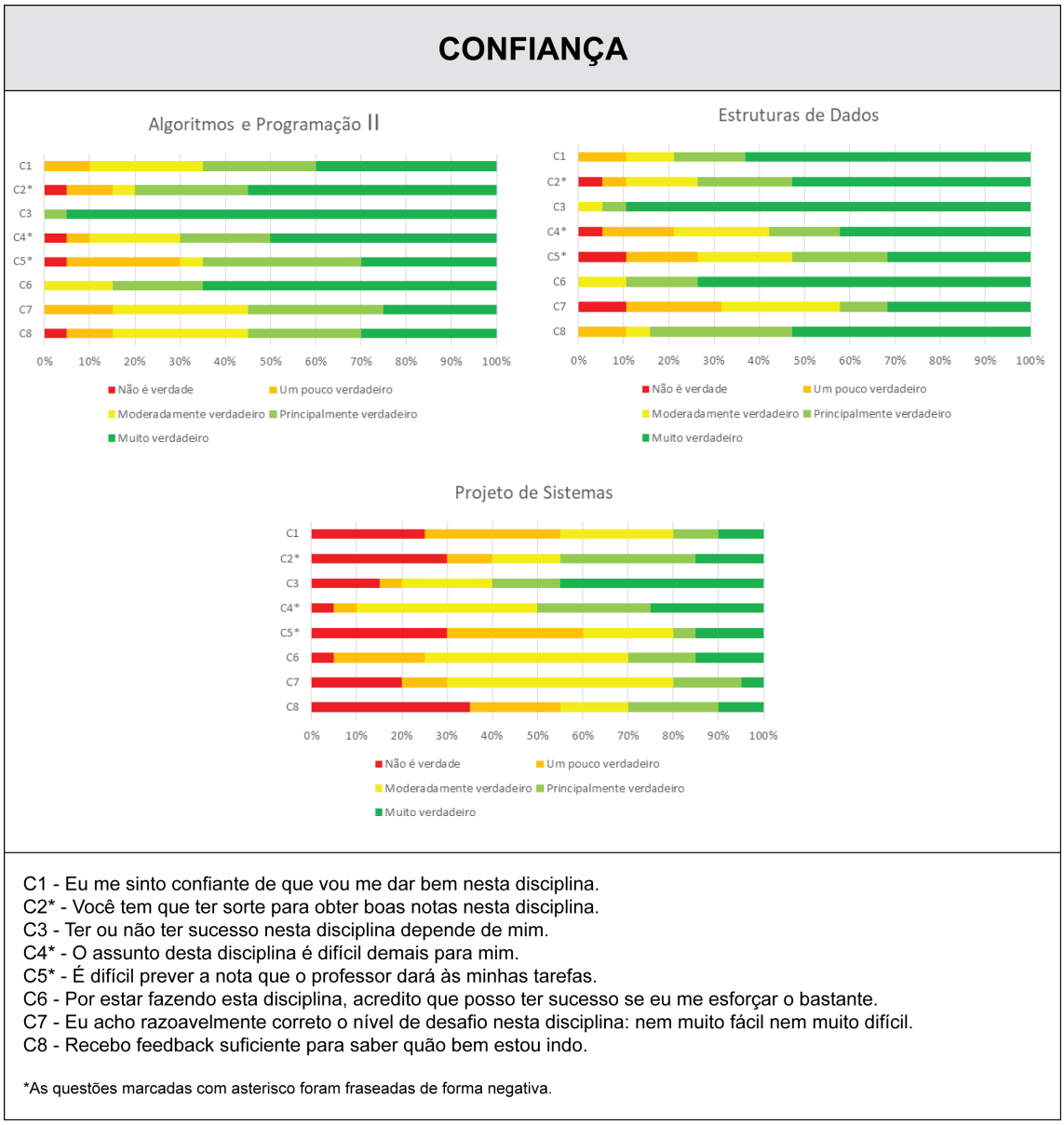


Figura 4.11: Resultados para a categoria Confiança nos Módulos Teóricos

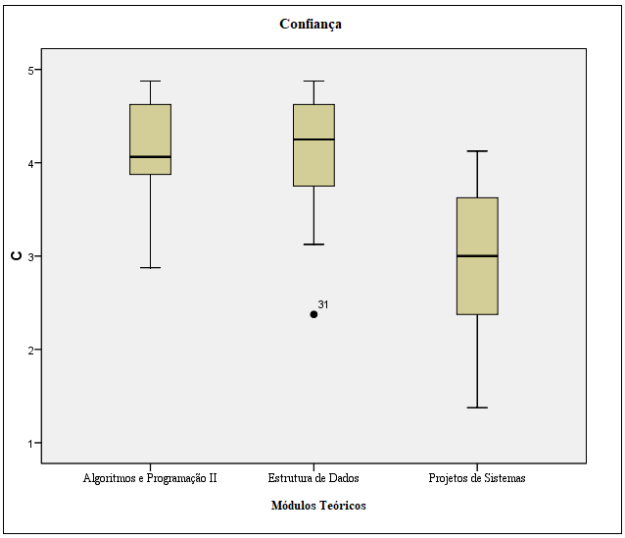


Figura 4.12: Box-Plot dos escores da Confiança nos Módulos Teóricos

4.2.4 Satisfação

Esta subseção descreve os resultados da categoria Satisfação tanto no módulo integrador como nos módulos teóricos.

Satisfação no Módulo Integrador

Na Figura 4.13, exibimos os resultados sobre a categoria Satisfação nos Problemas do MI de Programação. A maioria dos estudantes afirmou que ao completar os exercícios sentiram-se uma sensação gratificante de realização (S1 – 74% em P1, 71% em P2, 84% em P4), e que se sentiram bem ao concluir o desafio com êxito (S5 – 61% em P1, 71% em P2, 63% em P4). Todavia, verificamos que a satisfação no Problema 3 demonstrou resultados insatisfatórios. Boa parte dos estudantes discordou que foi um prazer estudar a metodologia utilizada (S6 – 64%), e também discordou que gostava de estudar programação (S3 – 59%), assim como de saber mais sobre o conteúdo (S2 – 55%). Por outro lado, o Problema 4 se destacou no quesito satisfação, quando os estudantes relataram o desejo de saber mais sobre os conteúdos abordados (S2 – 79%), e afirmaram que realmente gostaram de estudar programação (S3 – 79%).

A Figura 4.14 apresenta o box-plot do escore da categoria Satisfação nos problemas. Verificamos que a mediana da satisfação obteve resultados bastante positivos nos Problemas 1, 2 e 4. O Problema 3 apresentou tendência negativa. Em relação à dispersão dos dados, nos Problemas 1, 2 e 3 verificamos uma dispersão maior que a do Problema 4. Assim, os escores sofreram mais variações nos Problemas 1, 2 e 3. A dispersão dos escores é menor no Problema 4, logo, os dados são mais homogêneos.

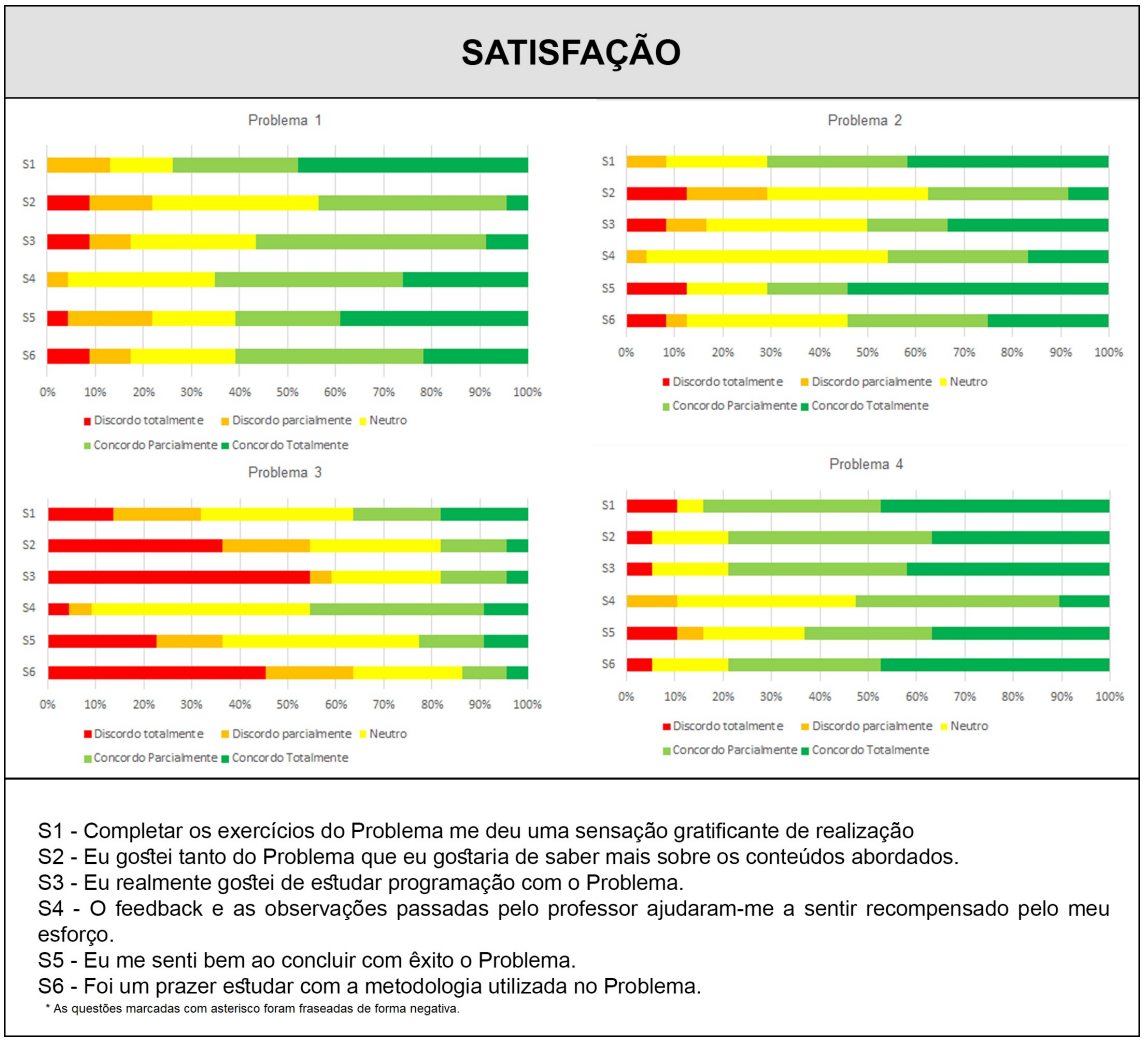


Figura 4.13: Resultados para a categoria Satisfação nos Problemas

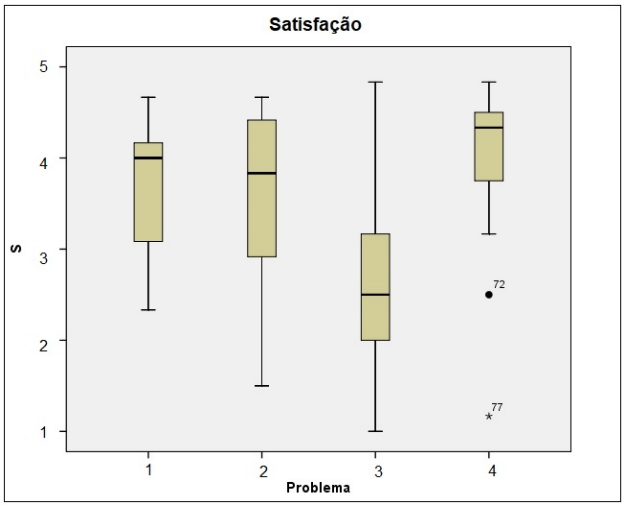


Figura 4.14: Box-Plot dos escores de Satisfação nos Problemas

Utilizamos ANOVA para testar se a Satisfação difere significativamente entre os Problemas, o que foi confirmado com $F = 9,533$, $p < 0,001$. Utilizamos o Teste *post hoc* de Tukey para identificação das diferenças específicas entre os pares de escores. O resultado demonstrou diferenças significativas entre os Problemas: a) 1 e 3; b) 2 e 3; c) 3 e 4. Os estudantes alcançaram níveis de satisfação mais altos nos Problemas 1, 2 e 4 e menores no Problema 3.

Satisfação nos Módulos Teóricos

Na Figura 4.15 apresentamos os resultados sobre a categoria Satisfação nos módulos teóricos. A maioria dos estudantes gostaram de estudar para as disciplinas teóricas (S4 – 65% em Algoritmos e Programação II, 79% em Estrutura de Dados), e sentiam satisfação ao cursá-las (S2 – 50% em Algoritmos e Programação II, 74% em Estrutura de Dados). Ainda, consideraram justas as notas recebidas quando comparadas com outros alunos (S3 – 90% em Algoritmos e Programação II, 89% em Estrutura de Dados). Em contrapartida, a maior parte dos estudantes afirmou que não gosta de estudar para a disciplina de Projeto de Sistemas (S4 – 58%) e não estão satisfeitos em cursá-la (S2 – 60%). Além disso, a maioria dos estudantes estavam insatisfeitos com as avaliações feitas pelo professor (S5 – 58%).

Na Figura 4.16, está representado o box-plot do escore da categoria Satisfação nos módulos teóricos. Observamos que a mediana da satisfação alcançou resultados bastante positivos nos módulos teóricos de Algoritmos e Programação II e Estruturas de Dados. Projeto de Sistemas apresentou mediana negativa. A dispersão dos dados é homogênea em Algoritmos e Programação II e Estrutura de Dados. Projeto de Sistemas, por sua vez, demonstrou uma maior variabilidade de escores.

Utilizamos ANOVA para testar se a Satisfação varia significativamente entre os módulos teóricos, o que foi confirmado com $F = 27,50$, $p < 0,001$. O teste *post hoc* de Tukey revela que os resultados apresentaram diferenças significativas entre os módulos teóricos: a) Algoritmos e Programação II e Projeto de Sistemas; b) Estrutura de Dados e Projeto de Sistemas. Os estudantes estavam satisfeitos ao cursar as disciplinas Algoritmos e Programação II e Estrutura de Dados, porém, em Projeto de Sistemas a maioria dos estudantes demonstraram insatisfação ao cursar este módulo.

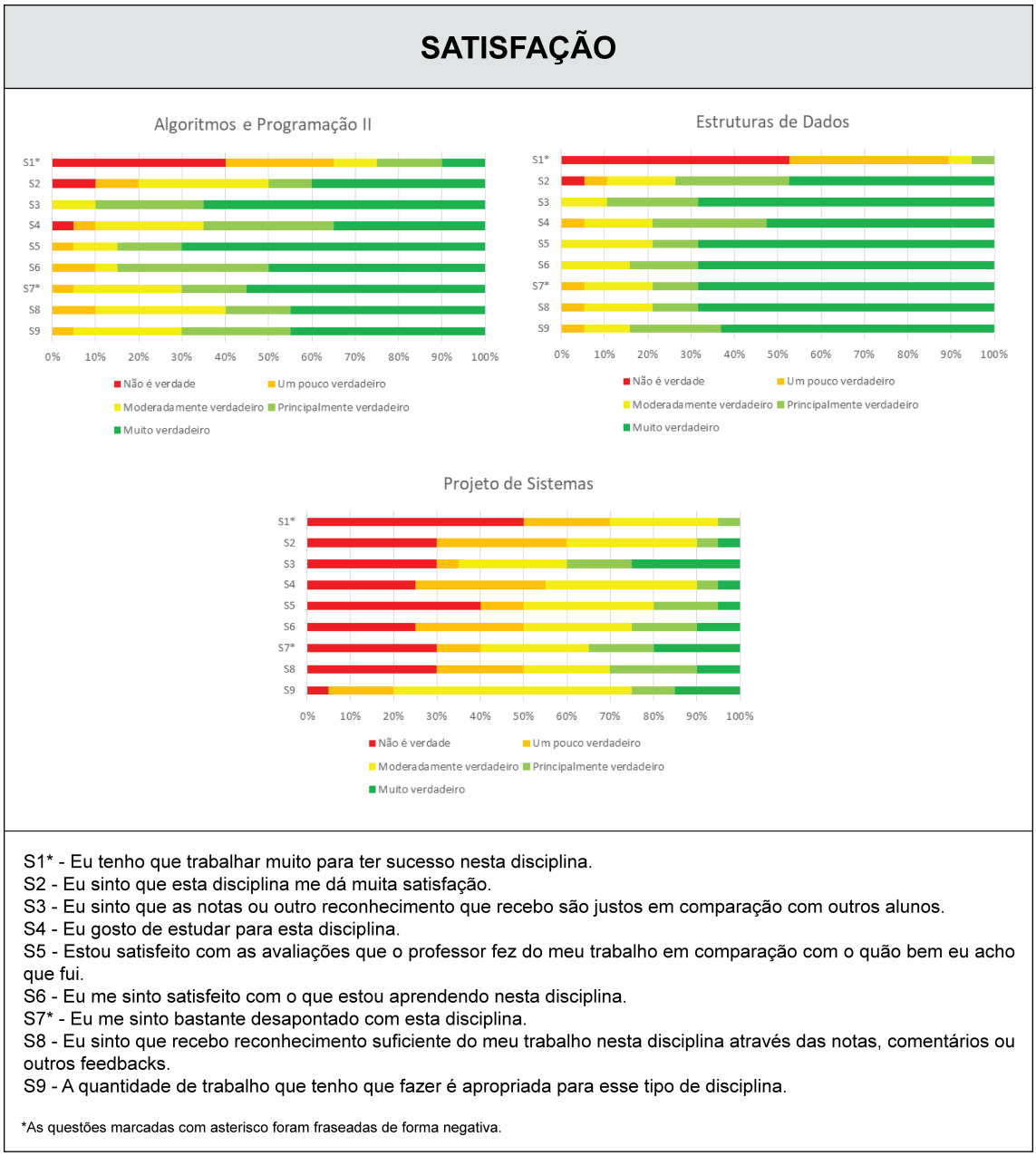


Figura 4.15: Resultados para a categoria Satisfação nos Módulos Teóricos

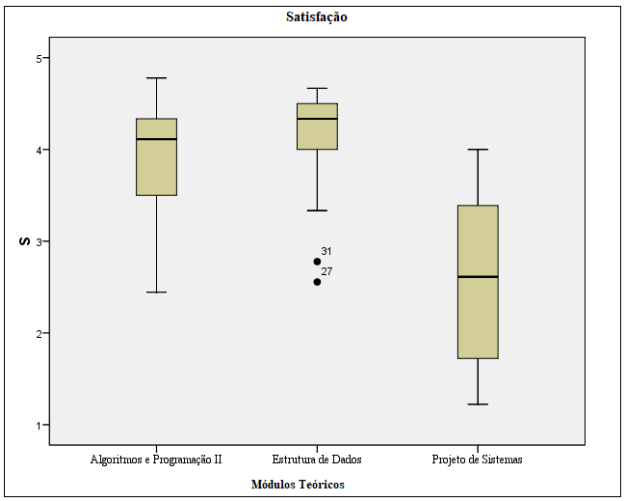


Figura 4.16: Box-Plot dos escores da Satisfação nos Módulos Teóricos

4.3 Aprendizagem

Agrupamos os resultados da aprendizagem considerando as médias finais dos estudantes nas avaliações dos módulos teóricos, do módulo integrador, e das provas SCS1 pré- e pós-intervenção.

A Tabela 4.1 apresenta a análise estatística descritiva com a quantidade de estudantes que realizaram as avaliações, a média das notas e o desvio padrão nas avaliações dos módulos teóricos, do módulo integrador e nas provas SCS1 pré- e pós-intervenção. Em relação às médias das notas, observamos que o MI de Programação ($\bar{x} = 8{,}10$, $\sigma = 1{,}41$) obteve os melhores resultados, seguidos dos módulos teóricos Algoritmos e Programação II ($\bar{x} = 6{,}51$, $\sigma = 2{,}95$) e Estrutura de Dados ($\bar{x} = 6{,}28$, $\sigma = 3{,}23$). Na disciplina de Projetos de Sistemas ($\bar{x} = 5{,}03$, $\sigma = 2{,}36$) foram obtidas as menores médias. Conforme esperado, o SCS1 pós-intervenção ($\bar{x} = 10{,}88$, $\sigma = 3{,}87$) obteve média maior que SCS1 pré-intervenção ($\bar{x} = 9{,}54$, $\sigma = 4{,}13$), embora o resultado não tenha sido muito diferente.

Tabela 4.1: Resultados da análise estatística descritiva.

Avaliação	**N**	**Média**	**Desvio-Padrão**
Algoritmos e Programação II	22	6,51	2,95
Estrutura de Dados	25	6,28	3,23
Projeto de Sistemas	25	5,03	2,36
MI de Programação	20	8,10	1,41
SCS1 Pré-Intervenção	22	9,54	4,13
SCS1 Pós-Intervenção	17	10,88	3,87

A Figura 4.17 apresenta o box-plot com as médias das avaliações dos módulos teóricos, módulo integrador, SCS1 pré- e pós-intervenção, à esquerda, e o diagrama de barra de erros, à direita. Nos gráficos, as notas do SCS1 foram normalizadas entre 0 e 10, para facilitar a comparação com as demais variáveis de aprendizagem nos mesmos gráficos. Em relação ao módulos teóricos e integrador, os dados são mais concentrados no MI de Programação, com mediana de 8,50, mais dispersos em Estrutura de Dados e Algoritmos e Programação II, com medianas, respectivamente, de 7,70 e 7,95 seguidos por Projeto de Sistemas, com mediana 5,00. O SCS1 pré-intervenção normalizado apresentou mediana 3,33, enquanto o pós-intervenção obteve 4,07. As medianas do SCS1 pré- e pós-intervenção mudaram pouco, mas observamos que houve uma concentração maior das notas no SCS1 pós-intervenção.

Testamos os dados das avaliações de aprendizagem para normalidade, o que se confirmou. Realizamos o Teste t para amostras emparelhadas para verificar se houve variação na aquisição de habilidades de programação a partir dos resultados do SCS1, o que não se confirmou, com resultados $t = -1,60$ e $p = 0,12$.

Apresentamos, a seguir, as correlações entre as notas finais dos módulos teóricos e módulo integrador com as notas da avaliação do SCS1 que foi aplicado ao final

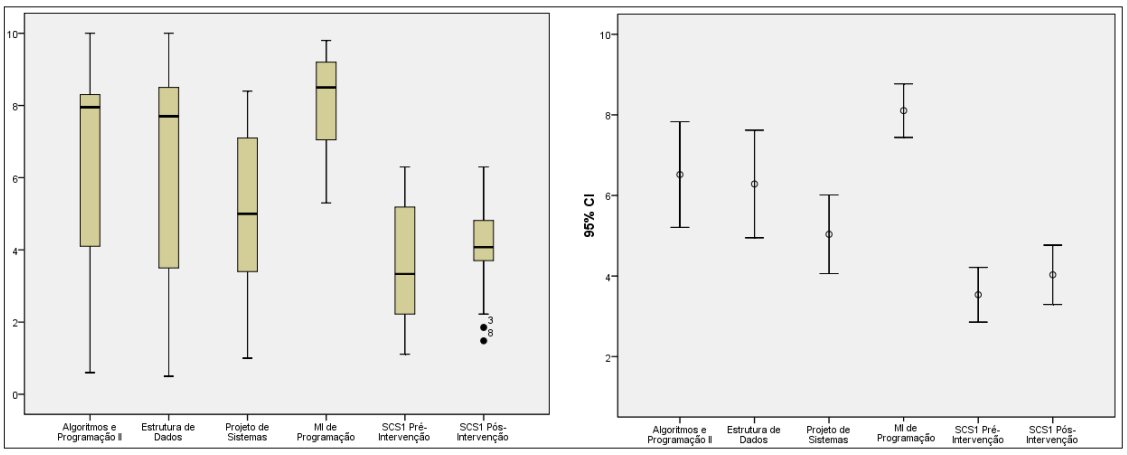


Figura 4.17: (a) Box-plot das médias das avaliações; (b) Diagrama de barra de erros das médias das avaliações.

da intervenção. Além disso, fizemos a correlação entre as notas finais dos módulos teóricos e do módulo integrador, e das notas finais dos módulos teóricos entre si.

A Tabela 4.2 ilustra as correlações calculadas. Em relação ao SCS1 Pós-Intervenção e os módulos teóricos e integrador, observamos correlação forte com Projetos de Sistemas, e correlação moderada com Algoritmos e Programação II e Estruturas de Dados. Com o MI de Programação, não houve correlação. Com referência aos módulos teóricos e o módulo integrador, verificou-se que o MI tem correlação muito forte com Algoritmos e Programação II e Estruturas de Dados e correlação forte com Projeto de Sistemas. Em relação aos módulos teóricos entre si, constatamos que Algoritmos e Programação II possui correlação muito forte com Estruturas de Dados e correlação forte com Projetos de Sistemas.

Tabela 4.2: Correlações entre os Módulos Teóricos, Módulo Integrador e prova SCS1 Pós-Intervenção

Categorias	SCS1 Pós-Intervenção	MI de Programação	Projeto de Sistemas	Estrutura de Dados	Algoritmos e Programação II
Algoritmos e Programação II	.384 (.158)	.724 (.000)**	.575 (.005)**	.850 (.000)**	1 (.000)
Estrutura de Dados	.423 (.090)	.762 (.000)**	.572 (.003)**	1 (.000)	
Projeto de Sistemas	.624 (.007)**	.544 (.013)**	1 (.000)		
MI de Programação	-.034 (.901)	1 (.000)			
SCS1 Pós-Intervenção	1 (.000)				

A Figura 4.18 representa o box-plot com as notas finais dos Problemas no MI. Observamos que a notas foram diminuindo no decorrer dos problemas. Constatamos que, nos Problemas 1 e 2, as notas estavam mais concentradas e apresentaram as maiores medianas, respectivamente, 9,30 e 8,80. No Problema 3 houve uma dispersão maior das notas, e mediana 6,60. O Problema 4 apresentou mediana 6,90, maior que no Problema 3, entretanto a média foi mais baixa. O Problema 4 apresentou a maior dispersão das notas. Os valores das medianas e amplitude interquartis estão descritas na Tabela 4.3. Os estudantes obtiveram melhor desempenho de notas no Problema 1 e 2, seguidos do Problema 4, e pior desempenho no Problema 3.

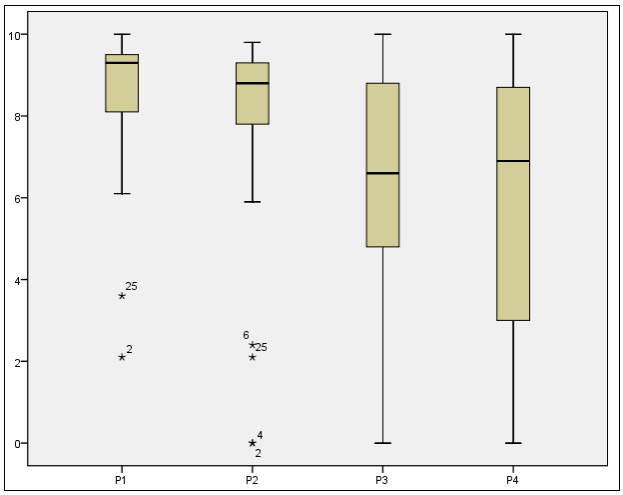


Figura 4.18: Box-plot das notas finais dos Problemas no MI.

Aplicamos o teste de aderência de Kolmogorov-Sminorv para verificarmos se os dados seguem uma distribuição normal. Através desse teste, obtivemos os resultados para os escores do Problema 1, $D(25) = 1,43$, $p < 0,05$, para o Problema 2, $D(25) = 1,60$, $p < 0,05$, Problema 3, $D(25) = 0,81$, $p > 0,05$ e o Problema 4 com, $D(25) = 0,84$, $p > 0,05$. Com a hipótese nula rejeitada, consideramos que os dados não obedecem a uma distribuição normal.

Tabela 4.3: Valores das Medianas e Amplitude Interquartis das notas finais dos Problemas no MI

Problemas	**Mediana**	**Amplitude**
P1	9,30	7,90
P2	8,80	9,80
P3	6,60	10,00
P4	6,90	10,00

Utilizamos a análise de variância de Friedman para testar se o escore das notas finais dos problemas variam significativamente durante as unidades, confirmado com ($qui-quadrado = 29,836$, $p < 0,05$). Para identificar quais pares de escores possuem diferenças significativas usamos o Teste de Wilcoxon. Aplicamos o teste de Wilcoxon entre os seis pares e consideramos o valor $p < 0,0083$ para identificar significância com a correção para múltiplos testes. Houve diferenças significativas entre as notas nos Problemas: a) 3 e 1; b) 4 e 1; c) 3 e 2; d) 4 e 2. As medianas baixaram de forma significativa do Problema 1 para o 3 e do Problema 1 para o 4. Não houve diferença significativa nas medianas do Problema 1 para o Problema 2. As medianas também baixaram de forma significativa do Problema 2 para o Problema 3, e do Problema 2 para o Problema 4. Não houve diferença significativa do Problema 3 para o Problema 4, apesar da mediana do Problema 4 ser maior que no Problema 3.

4.4 Relação entre Motivação e Aprendizagem

Esta subseção apresenta as relações entre motivação e aprendizagem tanto no módulo integrador como nos módulos teóricos.

4.4.1 Relação entre Motivação e Aprendizagem no Módulo Integrador

Apresentamos, logo adiante, as correlações entre as notas finais da aprendizagem nos quatro problemas do MI com as notas das categorias do ARCS obtidas através do IMMS.

Tabela 4.4: Correlações entre Motivação e Aprendizagem no Módulo Integrador

Problema	**Atenção**	**Relevância**	**Confiança**	**Satisfação**
P1	-.005(.980)	.235(.281)	.513(.102)*	-.240(.270)
P2	.447(.028)*	.172(.423)	.563(.004)**	.496(.014)*
P3	-.160(.488)	-.105(.650)	-.107(.646)	-.245(.284)
P4	.784(.000)**	.389(.100)	.602(.003)**	.777(.000)**

A Tabela 4.4 demonstra essas correlações. Sobre a correlação das notas do Problema 1 com as categorias do ARCS, observamos que não houve correlação com Atenção. Apresentou correlação fraca com Relevância e Satisfação e forte com Confiança. No Problema 2, verificamos correlação fraca com Relevância, moderada com Atenção e Satisfação e forte com Confiança. O Problema 3 apresentou correlação fraca e negativa com Atenção, Relevância, Confiança e Satisfação. Já no Problema 4, constatamos correlação moderada com Relevância, forte com Confiança e muito forte com Atenção e Satisfação.

4.4.2 Relação entre Motivação e Aprendizagem nos Módulos Teóricos

Nesta subseção apresentamos as correlações entre as notas finais da aprendizagem nos Módulos Teóricos com as notas das categorias do ARCS obtidas através do CIS.

Tabela 4.5: Correlações entre Motivação e Aprendizagem nos Módulos Teóricos

Módulos Teóricos	**Atenção**	**Relevância**	**Confiança**	**Satisfação**
Algoritmos e Programação II	.057(.818)	.241(.321)	568(.011)*	.464 (.045)*
Estrutura de Dados	.471(.042)*	.828(.000)**	.661(.002)**	.843(.000)*
Projeto de Sistemas	-.245(.298)	-.097(.683)	-.313(.179)	-.321(.168)

Na Tabela 4.5, estão descritas essas correlações. Em relação a Algoritmos e Programação II e as categorias motivacionais do ARCS, identificamos que não houve correlação com Atenção, mas houve correlação fraca com Relevância. A correlação foi forte com Confiança e moderada com Satisfação. Em Estrutura de Dados observamos correlação moderada com Atenção, forte com Confiança e muito forte com Relevância e Satisfação. Entretanto, o módulo teórico Projetos de Sistemas não apresentou correlação com Relevância, correlação fraca e negativa com Atenção, moderada e fraca com Confiança e Satisfação, respectivamente.

Capítulo 5

Discussão

Neste capítulo, finalizamos a discussão acerca dos resultados quantitativos e qualitativos das variáveis motivação e aprendizagem. Analisamos as possíveis relações entre a concepção dos problemas, os resultados de motivação e aprendizagem dos estudantes, além disso, estudamos os efeitos que a integração curricular têm nos resultados de motivação e aprendizagem.

Ainda, discutimos a motivação a partir dos problemas realizados no MI de Programação e dos Módulos Teóricos, baseados nas categorias do modelo ARCS. Analisamos também os conceitos, habilidades adquiridas e as relações entre a motivação e aprendizagem, verificando separadamente cada uma das categorias do modelo ARCS. Finalmente, sintetizamos as lições aprendidas desta pesquisa.

5.1 Concepção dos Problemas

Nesta seção analisamos a influência da concepção dos problemas e os possíveis reflexos na motivação e na aprendizagem.

5.1.1 Concepção dos Problemas e Motivação

As categorias Atenção, Confiança, Relevância e Satisfação, de modo geral, apresentaram resultados razoáveis nos Problemas 1 e 2, negativos no Problema 3, e mais positivos no Problema 4, conforme visto nas Figuras 4.2, ... As hipóteses que os justificam são baseadas em evidências coletadas qualitativamente a partir das nossas observações, percepções e do *feedback* dos estudantes.

As principais características da motivação intrínseca são: a satisfação, o interesse, o desafio, a curiosidade e a novidade [Keller 2010]. Nossos resultados sugerem que a novidade influenciou em melhores níveis de Atenção, Relevância, Confiança e Satisfação nos Problemas 1, 2 e 4, e piores no Problema 3. Observamos que no Problema

1, houve a introdução de conceitos de POO, gerando uma novidade. *" No Problema 1, o desafio maior era você utilizar uma linguagem que nunca tinha visto antes. O que me manteve motivado foi justamente a aquisição do conhecimento da nova linguagem. O Problema 1, de verdade, foi o que mais me motivou."* (E3). No Problema 2, inseriu-se o conceito de testes, outra novidade considerada interessante pelos estudantes, pois agora teriam uma forma diferente de trabalhar. O Problema 4 trouxe o conceito de interface gráfica, proporcionando aos estudantes uma experiência estimulante, permitindo o *feedback* visual imediato. *"O Problema 4 foi o que mais me motivou pela questão da interface gráfica, porque eu curti muito aprender como fazer em Java."* (E2). O Problema 3 não apresentava muitos conceitos novos, com exceção do conceito de árvore binária, os conceitos de POO eram similares e a dificuldade do problema era similar à dos anteriores.

Por outro lado, o domínio do problema, com a descrição de um cenário que apresente um contexto problematizador, escolhido a partir de um contexto real, torna o problema mais atraente e motivador. O domínio nos Problemas 1, 2 e 4 proporcionou uma identificação imediata dos estudantes gerando relevância, principalmente no Problema 4, no qual o domínio do problema tratava do desenvolvimento de um aplicativo similar a um já existente no mercado, que permitia o planejamento de viagens. Todavia, o domínio do Problema 3 não obteve a mesma aceitação. O domínio era desconhecido para grande maioria dos estudantes, tratando de um sistema gerenciador de carteira de ações. *"O tema do Problema 3, Bolsa de Valores, foi um tema complicado. Quando eu recebi o problema eu fiquei viajando, por que eu não sabia nada sobre o assunto."* (E1). A maioria dos estudantes não considerou o problema relevante, gerando uma grande dificuldade motivacional.

Finalmente, o problema deve ter a complexidade ideal. O problema não pode ser complexo demais, que impeça o entendimento dos conceitos, nem simples demais que impossibilite a reflexão e a discussão acerca do que deve ser aprendido. O problema deve ter a clareza e o tamanho necessários para incentivar os estudantes no desenvolvimento da solução do problema.

As observações sugerem que a complexidade existente no Problema 3 influenciou nos baixos índices motivacionais em todas as dimensões do ARCS. Os estudantes consideraram o problema abstrato e confuso. *"O Problema 3, eu achei o pior porque estava difícil de entender o que é que o problema queria. Não era nem a questão do código em si, quer dizer tinha questão do código também, mas a interpretação do problema estava complicada."* (E1). O Problema 4 obteve a melhor avaliação dentre os problemas. Os estudantes acharam-no bem estruturado, com informações claras e objetivo bem definido, além de o considerarem mais intuitivo, por fazer alusão a uma aplicação existente.

5.1.2 Concepção dos Problemas e Aprendizagem

Os problemas propostos aos estudantes nas sessões tutoriais do PBL são elaborados pelos professores do MI de Programação em um processo colaborativo. Para cada problema, é determinado um tema, objetivos de aprendizagem, descrição do cenário, um cronograma dos tutoriais e aulas, bem como os prazos para entrega dos produtos e os respectivos relatórios. São desenvolvidos projetos com a elaboração de produtos, que geralmente são os programas contendo a solução de software orientada a objetos e o relatório, com entrega individual ou em dupla.

A elaboração dos problemas está entre os maiores desafios, pois pode interferir na motivação e, consequentemente, no aprendizado dos estudantes. A qualidade dos problemas está relacionada diretamente com questões a ser consideradas. A primeira questão é, o problema deve ser solucionável ao mesmo tempo que é aberto. Além disso, a descrição do cenário deve ter o tamanho ideal e nível de complexidade que permita a discussão em grupo. Finalmente, é recomendado o cuidado na escolha dos conceitos e do domínio do problema.

Duch et al. (2001) afirmam que um bom problema, primeiramente, deve envolver o interesse dos alunos com objetivo de motivá-los a buscar uma investigação mais profunda dos conceitos que são introduzidos. Além disso, deve se relacionar com assuntos do mundo real. As questões iniciais do problema devem ser abertas, não se limitando a uma única resposta correta. Ainda, exigem que os alunos tomem decisões ou façam julgamentos baseados em fatos, informações, lógica e, ou, racionalização. A duração e a complexidade do problema deve ser controlado, entre outros fatores.

Os problemas devem ser alcançáveis e abertos, de maneira que tenham mais de uma solução viável. *"Eu tenho uma opinião assim sobre os problemas, e cada professor tem uma visão diferente, eu sou favorável a um problema de texto mais curto, mais aberto, alguns professores fazem o texto mais longo, muito detalhado."* (P1). Além disso, como os problemas são trabalhados em várias sessões tutoriais do PBL, devem envolver conteúdos mais abrangentes.

Segundo relato dos professores, a descrição do cenário não deve ser muito extensa nem muito curta, deve ter o tamanho ideal. *"O problema mais enxuto na leitura, e tal, dá mais margem de liberdade de decisão ao aluno, eu prefiro assim."* (P1). Deve ter um nível apropriado de dificuldade para que haja a possibilidade de discussão em grupo na busca das possíveis soluções. *"Eu pessoalmente, assim, eu não gosto muito de problemas complexos, por que amarra o problema, o aluno fica muito preocupado não só entender e cumprir uma série de requisitos."* (P1). Além disso, a escolha e inserção dos conceitos a serem trabalhados em cada problema é uma questão sensível e deve obedecer um nível crescente de complexidade. *"No começo, nos primeiros anos, eu aqui, a gente colocava a interface gráfica logo no início, eles apanhavam muito, como se diz, mas praticavam bastante."* (P1). De modo geral, os Problemas 1 e 2 obtiveram os melhores resultados de desempenho de notas, seguidos dos Problemas 3 e 4. Uma das possíveis explicações é que os Problemas 1 e 2, eram

mais fáceis, havia menos conteúdos, além da complexidade menor do que a exigida nos Problemas 3 e 4.

Outra questão a ser considerada é a escolha do domínio do problema. O domínio do problema deve estar contextualizado com o mundo real, permitindo a correlação do conteúdo retratado em questão com elementos ligados à realidade do estudante. *"O que eu observo é o seguinte. Quando você consegue elaborar um problema mais próximo da vivência dos alunos, melhor. Então, por exemplo, problemas que mais se assemelham a experiências vivenciadas dentro da internet, eles são mais bem recebidos."* (P1). No Problema 3 tivemos os piores índices motivacionais e de aprendizagem pois, além da complexidade dos conceitos, os estudantes tiveram dificuldade com o domínio e o aprendizado de alguns conceitos como, por exemplo, o balanceamento de árvores. *"O Problema 3, da Bolsa de Valores, me deixou confuso porque eu não conhecia nada sobre o mercado de ações, eu não entendia como funcionava[...] Além disso, era muita classe uma dentro da outra. Era uma ação que estava dentro de uma carteira, que estava dentro de um cliente, que estava dentro de uma empresa. E tinha também a árvore binária, a gente tinha que implementar uma árvore e depois balancear. Inclusive, eu não fiz o balanceamento porque eu não consegui."* (E4).

A escolha de domínios contendo temas lúdicos proporciona um maior engajamento dos estudantes na resolução do problemas aumentando o nível motivacional, facilitando a aprendizagem. *"(...) Então já existe também no mercado e eles poderiam associar, então os que motivam mais são aqueles que estão mais próximos da vida dos alunos. Se você faz uma coisa envolvendo compra, viagens, música, jogo, ele se envolve. Quando tem um problema de jogo, ele se interessa, então isso é um indicador interessante para pensar depois, né?"* (P2). No Problema 4, obtivemos os melhores índices motivacionais. Um dos fatores contribuidores possíveis foi ter sido considerado o problema mais interessante pela maioria dos estudantes. No entanto, isso não se confirmou no resultados das notas. Acreditamos que o desempenho não foi tão bom porque o Problema 4 sofreu influência de outros fatores. Dentre eles, o o fato de ser o problema mais difícil, ser aplicado no final de semestre quando os estudantes estão mais cansados, além disso, muitos desistem por outras razões que não necessariamente, Atenção, Relevância, Confiança e Satisfação.

Quanto mais desconhecido é o domínio do problema para o estudante, menos motivador. *"Então, quando você faz um problema distante da realidade deles, isso desperta menos interesse por causa dessas coisas. Talvez, se fosse em outro país onde a bolsa de valores fosse uma coisa mais buscada, e tal, como investimento, talvez esse tipo de aplicativo de problema despertasse mais interesse. Acho que realmente foi uma coisa muito estranha."* (P1). Por outro lado, apesar da dificuldade, o domínio estranho permite a aquisição de informações, antes desconhecidas pelos estudantes. *"Como que eu faço uma carteira de ações? Aquela coisa toda era um problema desconhecido. Era uma coisa estranha para eles, mas foi interessante na questão da informação para eles se familiarizarem."* (P1).

5.2 Integração Curricular, Motivação e Aprendizagem

A organização do EI de Programação segue um planejamento que obedece a um espiral de complexidade crescente, em que habilidades e conhecimentos são exercitados permitindo a aquisição de competências. Para tal, nos Módulos Teóricos são apresentados conteúdos que auxiliam os estudantes na resolução dos problemas propostos no MI de Programação. Concomitantemente, no MI de Programação são apresentados problemas práticos baseados em cenários do mundo real, envolvendo conceitos dos módulos teóricos.

Essa organização curricular permite essa integração, na qual os módulos teóricos e o módulo integrador se apoiam mutuamente. Porém, isto gera algumas dificuldades. Uma delas é o excesso de conceitos a serem aprendidos em um curto período de tempo, principalmente pela pequena carga horária dos módulos teóricos. Esta dificuldade pode provocar uma sobrecarga de atividades para os estudantes.

O que é aprendido nos módulos teóricos pode ajudar no MI. Em paralelo ao módulo integrador, os módulos teóricos apresentam e discutem assuntos específicos, normalmente encadeados com a teoria necessária ao desenrolar dos problemas, mas também envolvendo outros aspectos que eventualmente não sejam abordados nos problemas. Estes assuntos abordados nos módulos teóricos auxiliam os estudantes na resolução dos problemas propostos no MI de Programação. *"O módulo teórico auxilia muito na introdução de conceitos, os alunos conseguem ter uma perspectiva muito melhor de onde eles vão atacar. Então, no MI, eles vão focar duas ou três hipóteses para solucionar. No teórico, ele abre horizontes, eu vi aquilo na sala de aula, então, eu posso praticar isso no MI..."* (P2). Isto ficou mais evidente na relação entre o MI e os módulos teóricos de Estruturas de Dados e Algoritmos e Programação II, em relação à motivação, pois os problemas do MI trabalhavam conceitos relacionados aos MTs, gerando aumento na relevância. Além disso, os resultados do MI tiveram correlação muito forte e significativa com as notas de Algoritmos e Programação II e Estruturas de Dados, e forte com Projeto de Sistemas.

Isso também se dá no sentido contrário: o módulo integrador pode auxiliar na aprendizagem dos módulos teóricos. Os conteúdos que ainda não foram trabalhados nos módulos teóricos, mas que são requeridos nos problemas do MI de Programação, permitem que os estudantes busquem o aprendizado destes conceitos. *"A gente começou a ver diagrama de classe na aula teórica de Projeto de Sistemas no começo do Problema 2. Só que no Problema 1, a gente já tinha que entregar o diagrama. Então, quando vimos o assunto em Projeto de Sistemas, a gente já tinha uma noção."* (E4).

Essa organização pedagógica em que a teoria e prática caminham juntas oferece vantagens notáveis, tanto do ponto de vista do enriquecimento da estrutura cognitiva do estudante, como também, da lembrança posterior dos conhecimentos aprendidos,

tornando a aprendizagem uma experiência significativa. *"E aqui no MI, eles ficam meio que naquela questão tentativa e erro, tentativa e erro, tentativa e erro, sabe? Essa coisa dele fazer errar, fazer errar, ajuda também no processo. Porque quando ele vê a solução adequada, ele não esquece mais. Então, isso ajuda bastante."* (P2).

Ainda, observamos que a integração curricular influencia na interação dos Módulos Teóricos entre si, refletindo nos resultados de aprendizagem e motivação. Mais especificamente, percebemos uma sincronia principalmente entre Algoritmos e Programação II e Estruturas de Dados. Isso se deve ao fato de serem disciplinas que naturalmente se complementam, ou seja, o estudo da programação passa efetivamente pelo estudo das estruturas de dados. Além disso, era o mesmo professor para os dois MTs, com a mesma didática, mesmo estilo de avaliação. Esta constatação foi confirmada na análise da motivação, pois, de modo geral, os estudantes estavam mais motivados para todas as categorias do ARCS nos módulos teóricos de Algoritmos e Programação II e Estruturas de Dados. *"Eu gostei muito da didática do professor de Estrutura de Dados e POO, ajudava bastante. Mesmo que você não conseguisse implementar, você tinha a ideia. Ele conseguia passar como a coisa funcionava de forma muito clara. Então se você dedicasse um pouquinho de tempo para implementação, você conseguiria. Porque a teoria passada pelo professor era muito boa mesmo."* (E3). Este comportamento foi visto também nos resultados de aprendizagem. Os módulos teóricos de Algoritmos e Programação II e Estruturas de Dados apresentam resultados de desempenho de notas similares, além de forte correlação entre si.

A organização curricular adotada também gerou dificuldades. Uma delas foi o excesso de conceitos trabalhados nos módulos teóricos em um curto período de tempo. *"E também, com prazos curtos, é um dilema que eles enfrentam, ter que aprender em um curto período de tempo é um ritmo difícil."* (P1). De maneira geral, os módulos teóricos possuem uma quantidade de conteúdos extensa para uma carga horária relativamente pequena. *"A primeira coisa que é importante, Algoritmos e Programação II possui uma carga horária de 30 horas, por isso, ela fica muito condensada na parte teórica. Então assim, como o conteúdo é muito extenso, eu tenho que focar muito na parte teórica, não tenho como pedir muito exercício prático. Se eu fosse fazer isso, eu não tinha como cobrir a parte teórica."* (P3).

Esse problema fica ainda mais evidenciado no módulo teórico de Estruturas de Dados, que possui uma carga ainda maior de conteúdos quando comparada aos outros módulos teóricos. *" Algoritmos e Programação II, eu já acho mais suave, né, porque as coisas acabam meio que se repetindo. Mas lá a gente trabalha técnicas de programação, né, que facilitam. A carga horária das disciplinas, sobretudo a disciplina de Estruturas de Dados, acaba fazendo com que eles fiquem bastante sobrecarregados, eu acho que lá sobrecarrega o aluno ainda mais do que o próprio PBL. Porque a quantidade de assuntos que eles tem que trabalhar é ainda maior"* (P2). Além disso, ocasionalmente, alguns conceitos que estão no conteúdo programático dos módulos teóricos não são aprimorados como deveriam por falta de tempo. *"O diagrama de interação, o diagrama de comunicação, o diagrama de sequência, eu não consigo*

trabalhar muito profundamente esses diagramas, que são bastante importantes pra eles, por falta de tempo." (P2)

Essas dificuldades associadas geram uma sobrecarga de trabalho para os estudantes. *"O ponto negativo é porque a gente foca tanto no PBL que ás vezes, as outras disciplinas ficam prejudicadas. Acho que é mais isso, mas isso aí, é questão de equilíbrio também."* (E1).

5.3 Diferenças entre as dimensões do Modelo ARCS

Em respeito à Atenção, Keller (2010) argumenta que mesmo os melhores programas não obterão resultados positivos caso os estudantes não estejam motivados para aprender. Para isso, é necessário estabelecer um equilíbrio nas atividades do aprendiz que permita manter sua atenção. Portanto, é importante que se utilizem estratégias que incluam a variação de ritmo ou estilo do material pedagógico, o uso do humor ou o envolvimento do estudante nas atividades. Os resultados sugerem que a atenção tem níveis relativamente altos nos Problemas 1, 2 e 4, e baixos no Problema 3. Uma possível explicação para esse resultado no Problema 3, é que a maioria dos estudantes não possuíam nenhum conhecimento prévio sobre Bolsa de Valores, ou seja, o domínio era estranho para eles. Além disso, eles o consideraram complexo. *"O Problema 3 foi o pior dos problemas. Ele me deixou mais nervoso e frustrado. Porque além de não entender nada sobre mercado de ações e ficar totalmente perdido com o contexto do problema, acho que também foi difícil entender o que estava sendo exigido da gente ali. Foi muito frustrante!"* (E6).

Keller (1987) afirma que a Relevância consiste em fazer os estudantes perceberem a importância do que está sendo ensinado, e a utilização imediata destes ensinamentos devem fornecer respostas aos seus motivos e valores. As concepções de cada estudante certamente determinam os rumos através dos quais seus objetivos podem ser alcançados. O fato de constatar relevância no assunto ensinado traz para o estudante a confiança de que ele está no caminho certo e que deve continuar a lutar para alcançar seus objetivos. A percepção da relevância foi relativamente alta nos Problemas 1, 2 e 4, e mais baixa no Problema 3. Porém, ainda sendo baixa no Problema 3, foi mais alta quando comparada às outras dimensões. Uma possível explicação para este grau de relevância é que os estudantes fazem uma disciplina de programação em um curso de computação, que é central neste curso e muito relacionada com a realidade profissional deles.

Na dimensão Confiança, foram obtidos os menores níveis em todos os problemas, quando comparadas às outras dimensões. Nos Problemas 1, 2 e 4, os resultados são mais neutros, e mais baixos no Problema 3. As prováveis justificativas seriam a complexidade dos conteúdos, o excesso de conceitos e as diversas competências

que devem ser adquiridas para solucionar os problemas. Os estudantes enfrentam dificuldades ao lidar com a novidade dos conceitos de POO, Estruturas de Dados e Projeto de Sistemas.

Na dimensão Satisfação, foram alcançados níveis altos nos Problemas 1 e 2, baixos no Problema 3, e mais altos no Problema 4. A satisfação, segundo Keller (1987), é o resultado da avaliação cognitiva dos estudantes da equidade entre o esforço investido e os resultados percebidos ao final do processo de aprendizagem, a partir da interação com um dado objeto educacional. Deste modo, os resultados com dispersão ampla sugerem que nem todos vivenciam o sucesso ou a motivação da mesma maneira, nem gostam igualmente das suas experiências. Porém, os estudantes avaliaram que vale a pena continuar investindo seu esforço, confirmado nos Problemas 1, 2 e 4.

Analisamos também os resultados dos questionários CIS nos módulos teóricos para cada uma das categorias do modelo ARCS. Os resultados de atenção foram mais altos em Algoritmos e Programação II e Estruturas de Dados e mais baixos em Projeto de Sistemas. Uma justificativa possível é a importância percebida pelos estudantes que os conceitos aprendidos em Algoritmos e Programação II e em Estruturas de Dados têm para o desenvolvimento das soluções de software. Principalmente em Estruturas de Dados, pois os alunos reconheciam que a escolha das estruturas apropriadas para cada problema poderia interferir no bom desempenho dos programas. *"A disciplina de Estruturas de Dados afetou positivamente a minha motivação, porque na hora de fazer um sistema, por exemplo, eu tenho agora, como escolher a melhor maneira de desenvolver, armazenar e trabalhar melhor com meus dados. Posso analisar os tempos de execução. E isto ajuda bastante no projeto de um sistema posteriormente."* (E3).

No que diz respeito à percepção da relevância, de maneira geral, os estudantes consideraram relevantes todos os módulos teóricos, talvez pelo fato de serem *majors* em computação e entenderem a importância dos conteúdos trabalhados nas disciplinas teóricas para o desenvolvimento de software.

Entretanto, em relação à confiança, os resultados não foram positivos para o módulo teórico de Projeto de Sistemas. Os estudantes estavam muito confiantes em Algoritmos e Programação II e Estrutura de Dados, e não estavam confiantes em Projeto de Sistemas. Uma possível explicação é que os conceitos exigidos em Algoritmos e Programação II e Estruturas de Dados são os mais exigidos para solucionar os problemas do MI. Ainda, em geral os estudantes gostavam da dinâmica das aulas e da maneira como eram avaliados. *"Em POO, era o mesmo professor de Estruturas de Dados. Acabava que era muito bom também. Porque POO era mais teoria que Estrutura de Dados. Estrutura de Dados era mais fazer o código mesmo, até mesmo simular. Em POO, assim como Estruturas de Dados, era mesma metodologia, mesma didática. O professor fazia exemplos na sala de aula, que eram um paralelo com a vida real e com os assuntos. Eu gostei muito."* (E3).

Essa tendência também se confirmou em relação à satisfação. Em geral, os estudantes estavam satisfeitos ao cursar as disciplinas de Algoritmos e Programação II e

Estruturas de Dados e insatisfeitos com a disciplina de Projeto de Sistemas. Acreditamos que isso aconteceu devido a um somatório de fatores. Primeiramente, como já citado anteriormente, a dificuldade com a subjetividade e a capacidade de abstração do conteúdo, além da dificuldade com didática utilizada pelo professor nas aulas e a forma de avaliar. E também fatores externos, como por exemplo, o número de feriados e paralisações que ocorreram nos dias das aulas. Esse problema gerou uma interrupção na linha do raciocínio dos conteúdos aplicados gerando desmotivação nos estudantes. *"Em Projeto de Sistemas, os assuntos dados em sala de aula não eram muito diretos, ou seja, você tem diversas maneiras por exemplo de fazer um diagrama de classe. Isso me deixava confuso. Fora isso, nas avaliações, tinha que fazer um diagrama de classe que para mim estava certo, mas para o professor não estava tão certo. Então acabava que nas avaliações dele, em especial, ele ia tirando pontos para aqueles erros, e muitos se desmotivaram com a avaliação dele."* (E3). Percebe-se que a natureza dúbia do projeto de software afeta a percepção dos estudantes sobre o que é certo ou errado.

5.4 Conceitos e Habilidades Aprendidos

Aqui, apresentamos uma discussão sobre os conceitos e habilidades aprendidos.

5.4.1 Conceitos aprendidos

O objetivo do EI de Programação é capacitar os estudantes para projetar e desenvolver software utilizando o paradigma orientado a objetos. Para isso, é necessário a compreensão dos conceitos, técnicas e sua aplicação em projetos de desenvolvimento de sistemas.

Os resultados sugerem que, em relação a aprendizagem dos conceitos de programação orientada a objetos nos módulos teóricos e no MI de Programação, o exercício experimentado na tentativa de solucionar os problemas propostos nas sessões tutoriais propiciaram um avanço no aprendizado dos estudantes. Ao final do semestre, a maioria deles conseguiu criar classes para representar objetos do mundo real, os seus atributos e métodos associados, apesar da dificuldade inicial com a abstração de alguns conceitos. Demonstraram dominar razoavelmente bem os conceitos de encapsulamento, herança e polimorfismo, ainda que, em alguns contextos, apresentavam embaraço ao empregar o conceito de polimorfismo. A princípio, mostraram problemas também com o conceito de exceções, que foram reparados no decorrer da execução das atividades.

O ensino e apresentação dos conceitos acontece de maneira gradual em ordem crescente de complexidade. Inicialmente, os estudantes apresentaram certa dificuldade na assimilação dos conceitos. Porém percebemos que a prática vivenciada na busca

pela solução dos problemas no MI de Programação, associada às aulas teóricas, proporciona uma evolução na aprendizagem. *"Então a tendência é que eles comecem com uma assimilação dos assuntos, assim, do nível um pouco baixo, com o passar do tempo, essa curva de aprendizagem, ela vai melhorando consideravelmente [...] se você não entende os conceitos básicos de orientação a objetos, fica difícil você entender um padrão, por exemplo, porque é baseado nos conceitos de interface, de classe abstrata, nos conceitos de polimorfismo, por exemplo, são super importantes[...]"* (P2).

Em relação à aprendizagem dos conceitos de programação orientada a objetos nos módulos teóricos e no MI de Programação, a maioria dos estudantes demonstraram dominar relativamente bem a sintaxe da linguagem Java. *"Eu aprendi realmente como funciona a programação orientada a objetos."* (E2). Além disso, compreenderam o conceito de classes, que são utilizadas para representar objetos do mundo real, e de atributos e métodos, que representam respectivamente as características e o comportamento dos objetos.

Contudo, apresentaram uma dificuldade no início do semestre letivo. Como estudaram programação estruturada no semestre anterior, demoraram um pouco para compreender como um código orientado a objetos é organizado. *"A questão da dificuldade do primeiro problema, foi só por ser o primeiro contato com POO. Eu fiquei um pouco perdido por causa dos pacotes. Eu não sabia como organizar o código direito, as classes... A questão do padrão MVC foi que pegou mais. Eu fiquei meio perdido no início."* (E6).

Ainda, aprenderam relativamente bem os conceitos de encapsulamento, herança e polimorfismo. *"Eu aprendi todos esses conceitos aí... classe, polimorfismo, herança, exceção, métodos, acho que eu aprendi todos os conceitos. Eu não aprendi muito conceito de interface."* (E1). Porém, apesar de entenderem o conceito de polimorfismo, percebemos que boa parte dos estudantes tem dificuldade em saber quando empregá-lo e as vantagens da sua utilização. *"Um assunto que eu abordo bastante mas eu acho que é difícil para os alunos entenderem o conceito de polimorfismo e saber quando usar. Acho que eles sempre têm dificuldade, entendem o que é, mas não sabe quando usar. Eles não sabem quando usar aquilo... às vezes até entende o que é. Mas não sabe porquê que eu uso isso? Qual é a vantagem disso?"* (P3).

Outra questão importante, os estudantes sabem quando utilizar uma classe anônima ou quando utilizar uma classe nomeada. *"Declaração de variáveis eu acho que eles aprendem bem, entender quando utilizar uma classe anônima ou uma classe mais geral, eles conseguem perceber esses conceitos."* (P3).

Para Bruner (2009), o conceito de aprendizagem em espiral pode enunciar-se da seguinte forma: qualquer ciência pode ser ensinada, pelo menos nas suas formas mais simples, a alunos de todas as idades, uma vez que os mesmos tópicos serão, posteriormente, retomados e aprofundados mais tarde.

Deste modo, inicialmente os estudantes demonstraram uma certa dificuldade em

relação aos erros imprevistos durante a execução dos programas, as exceções. Entretanto, a partir do segundo problema, a maioria já sabia tratar as exceções. De maneira geral, muitos conceitos de programação orientada a objetos aplicados nos problemas das sessões tutoriais do PBL repetiram-se para os quatro problemas propostos durante o semestre, com exceção do conceito de interface gráfica, permitindo que os estudantes aprimorassem suas habilidades e competências técnicas. *"Eu aprendi os conceitos padrão em POO, os pilares da orientação a objetos. Como o que é e quando utilizar classes, métodos e atributos. POO é bem batido porque a gente vem aplicando os conceitos nos problemas... Não teve nenhum conceito que eu não tenha aprendido em POO."* (E2).

Também apresentaram algumas dificuldades relacionadas a qualidade de software e no uso de boas práticas, como por exemplo, definir atributos privados nas declarações de variáveis de instância. Os estudantes têm dificuldade em compreender que tal ação tem como efeito ajudar no encapsulamento dos dados, preservando ainda mais a segurança e a aplicação de programação orientada a objetos. *"Todos esses conceitos, agora aí vem um bocado, relacionados à qualidade de software. Os alunos não conseguem perceber com as aulas de programação orientada a objetos. Eles entendem até o conceito, mas às vezes não sabem o porquê que eu tenho que declarar uma variável privada. Bem ele sabe o porquê, mas talvez não consigam entender o impacto daquilo como um todo, e às vezes reclamam. Eu tô fazendo mas podia ser public."* (P3).

Em relação ao módulo teórico de Estruturas de Dados, assim como nos outros módulos teóricos, os conceitos mais vivenciados na prática dos problemas do MI de Programação são mais bem assimilados pelos estudantes. De modo geral, os alunos aprenderam as estruturas de dados básicas, como filas, pilhas e árvores, assim como o conceito de grafos e os algoritmos de ordenação mais simples. Porém, apresentaram dificuldades com algoritmos de ordenação basicamente recursivos, como por exemplo o *merge sort*.

É importante ressaltar, que os estudantes cursaram um módulo de programação com o paradigma imperativo no semestre anterior e, por isso, trazem uma bagagem prévia de conceitos adquiridos relacionados com esse contexto. Podemos citar como exemplo, o conceito de listas encadeadas. *"Eu aprendi listas encadeadas antes, mesmo sendo em um paradigma diferente do que eu já conhecia, era somente modificar para o paradigma orientado a objetos."* (E3).

De modo geral, os alunos aprenderam as estruturas de dados básicas, como filas, pilhas, árvores. *"Como eu fiz todos os problemas, eu aprendi todas as estruturas que a gente estava aplicando nos problemas, desde de listas, filas, pilhas até grafos e árvores. A gente conseguiu aprender aquilo que estava sendo proposto."* (E2). Aprenderam criar uma árvore, entretanto, a maioria deles teve dificuldade com o conceito de deleção e balanceamento de árvores. *"Eu tive dificuldade com o balanceamento da árvore, porque na hora de remover você tem que balancear de novo. Aí tem que verificar todos os arredores da árvore e balancear de novo. Inserção de um*

nó é mais fácil, inserção eu consigo fazer. Mais a remoção atrapalha mais." (E1).

Os resultados sugerem que, assim como em POO, aqueles conteúdos exercitados nas soluções dos problemas do MI de Programação, são mais bem assimilados pelos estudantes. Podemos citar como exemplo o conceitos de grafos. No último problema os estudantes utilizaram este conceito e também o algoritmo de Dijkstra para encontrar o caminho mínimo. *"Eu aprendi: grafos, lista, Dijkstra. Eu gostei muito de Estruturas de Dados, foi a melhor disciplina do semestre. Eu consigo tanto utilizar as estruturas que já tem pronta, como também fazer na mão."* (E1).

A maioria dos estudantes compreendeu os algoritmos de ordenação mais simples como o *bubble sort*, *insertion sort* e *selection sort*. Ainda, apesar da complexidade dos detalhes e da recursividade, boa parte dos estudantes aprenderam o método de ordenação *quick sort*. *"Tudo que foi cobrado de Estruturas de Dados, eu entendi. Mas tive um pouco de dificuldade com alguns algoritmos de ordenação, alguns específicos. O que eu tive mais dificuldade foi com o Quick Sort e também Bucket Sort. Inclusive meu desempenho na segunda prova não foi tão bom."* (E2). Por outro lado, apresentaram dificuldades maiores no aprendizado do *merge sort* e *heap sort*. *"No problema 2, o que mais influenciou foi a questão de ordenação. Heap sort foi o que eu dei mais prioridade para estudar, e também, foi o que mais eu tive dificuldade."* (E4).

No módulo teórico de Projeto de Sistemas, os resultados mostraram que os estudantes demonstraram muita dificuldade na interpretação dos problemas. Dividimos estas dificuldades mais gerais em três categorias: abstração do problema, as regras de negócio e utilização dos artefatos da UML. A abstração do problema está relacionada com a dificuldade de entender o domínio dos problemas, identificar todas as funcionalidades do sistema e documentá-lo. As regras de negócio compreendem identificar as regras de negócio e as restrições e/ou premissas para resolver os problemas. Finalmente, há também a dificuldade na utilização dos artefatos da UML, como os diagramas e suas relações.

O primeiro obstáculo encontrado pelos estudantes é lidar com a subjetividade na interpretação dos problemas. Geralmente ficam totalmente condicionados à programação em si. Entretanto, a metodologia do módulo teórico de projetos de sistemas requer uma habilidade extra, que é a abstrair o problema que simule um cenário real e posteriormente, representá-lo através de um projeto orientado a objetos utilizando uma notação comum.

Ainda, confrontam-se com questões ligadas às dificuldades no entendimento e utilização dos artefatos da UML. Boa parte dos estudantes apresentaram problemas em identificar e descrever fluxos e regras e, consequentemente, na criação de diagramas. *"Em Projeto de Sistemas, eu tive dificuldade na hora de resolver os problemas. Ao ler o problema, eu não conseguia identificar a princípio todos os objetos e classes que o problema tinha. Acabava que ficava faltando algum dado, alguma lacuna no diagrama de classe."* (E3).

5.4.2 Habilidades aprendidas

Uma das questões de pesquisa deste estudo busca compreender quais são os resultados proporcionados pela abordagem em termos de aprendizado de conhecimentos e habilidades de programação orientada a objetos. A abordagem utilizada proporcionou a aquisição de habilidades técnicas e habilidades comportamentais.

As habilidades técnicas compreendem um conjunto de capacidades e competências diretamente relacionadas ao exercício técnico, envolvendo conhecimentos, processos, métodos e procedimentos específicos. A abordagem de ensino-aprendizagem usada permitiu a aquisição de habilidades e competências relacionadas nos módulos teóricos e no módulo integrador.

Os resultados revelam que os estudantes são capazes de compreender a práxis da programação orientada a objetos e também os conceitos básicos relacionados. *"Pensar no programa orientado a objetos, eu acho que ele sabe, por exemplo, eles vêm de um conceito que todo programa é feito em um único arquivo. Quando a gente vai para a programação orientada a objetos, a gente começa a separar os arquivos em classes e depois tenta agrupar essas classes."* (P3).

Ainda, foram instruídos a aplicar estruturas de dados orientadas a objetos para modelar dados simples e complexos a partir de problemas reais. Da mesma forma, acreditamos que estão preparados para manipular arquivos através da serialização. *"Eu já sei que consigo projetar um sistema que seja solicitado com orientação a objetos. Aplicar, digamos assim, fazer da melhor forma, sabe? Tipo, utilizar uma estrutura de dados necessária pra ficar mais eficaz a minha solução."* (E2). Também sabem tratar erros e exceções. *"[...] entender o que é uma exceção, quando utilizar essa exceção, tanto na parte de hierarquia Exception e tudo que está relacionado ao tratamento de exceções."* (P3). Além disso, são capazes de criar interfaces gráficas simples na linguagem Java. *" Eles sabem que a programação pra interface tem que utilizar alguns layouts e, a depender do delta, eu consigo fazer uma interface melhor do que outra."* (P3). Ademais, percebemos que os estudantes conhecem o funcionamento das principais estruturas de dados, sabendo quando cada uma deve ser aplicada em um problema do mundo real. *"Depois de ver visto Estruturas de Dados, eu aprendi algo muito importante que é identificar a melhor estrutura para criar, tipo, uma solução eficaz."* (E2). Inclusive, baseada na experiência vivenciada, acreditamos que estão aptos para estimar a complexidade de algoritmos básicos. *"Eles sabem entender um pouco da complexidade de algoritmos, não muito, mas eles conseguem olhar um algoritmo e ver se ele vai ser lento ou rápido, eles vêem vários exemplos com complexidade de algoritmos."* (P3).

Percebemos também que são capazes de reconhecer os principais métodos de busca e ordenação, bem como sua aplicação para cada situação específica. *" Eles sabem fazer alguns algoritmos de ordenação como ordenar uma grande coleção de dados, eles entendem a importância de ter algoritmos bons para resolver um problema, o algoritmo certo para um problema certo, ou seja, não é qualquer algoritmo que resolve*

um problema." (P3). Conhecem e manipulam estruturas de dados avançadas como árvores e tabelas *hash*. Além disso, compreendem e manipulam grafos e conseguem desenvolver algoritmos de busca em grafos orientados e não orientados. *"Eu consigo fazer sistemas de cadastro, algoritmos de menor caminho. Por exemplo, o aplicativo de viagem que precisava calcular a rota. Isso aí eu consigo fazer. Eu me sinto capaz de solucionar um problema que requeira a programação orientada a objetos."* (E1).

Além do mais, através da prática exigida na resolução dos problemas, acreditamos que são capazes de identificar e buscar soluções existentes para adequar a cada problema em particular. *"Saber olhar para o problema de uma forma diferente e saber como buscar soluções prontas, confiáveis e já testadas para aquele problema que ele está enfrentado ali, para que ele não tenha que ter o retrabalho de poder pensar em uma solução eficiente para aquilo."* (P2). E, finalmente, no módulo integrador, os estudantes adquirem habilidades relacionadas à própria vivência no exercício prático de algumas etapas do desenvolvimento do software: levantamento de requisitos, projeto, implementação e testes. *"O MI auxilia na prática, no MI é a oportunidade que eles têm de praticar, de fazer o código, de escrever, de testar a recursão, de ver um problema maior, real."* (P3).

Aplicamos a prova SCS1 no início e no final do semestre com o objetivo de identificar se houve ganho de conceitos e habilidades de programação no EI de Programação. Os resultados estatísticos computados sugeriram que não houve ganho significativo ao final da intervenção. O motivo que possivelmente explique esse resultado é que a prova do SCS1 é baseada em conceitos e habilidades geralmente adquiridos na primeira disciplina de programação que utiliza o paradigma imperativo. Todavia, o EI de Programação, objeto do nosso estudo, utiliza o paradigma orientado a objetos como padrão de desenvolvimento, o qual não está presente em questões do SCS1.

Além das habilidades técnicas adquiridas, a utilização da metodologia de aprendizagem baseada em problemas e projetos permitiu adquirir uma série de competências comportamentais. Dentre elas, destacam-se a **autonomia, autodidatismo, proatividade, comunicação e relacionamento interpessoal**. Essas características comportamentais geram benefícios para os estudantes, ampliando as chances de sucesso na vida profissional e acadêmica, visto que, estão habituados a solucionar problemas e encarar novos desafios. Esses fatores associados, preparam melhor o estudante para o perfil exigido pelos padrões da sociedade atual.

A **autonomia** é desenvolvida na metodologia PBL pois o estudante é o principal responsável pelo seu aprendizado, o que é o oposto da metodologia tradicional, na qual o estudante exerce um papel passivo no seu processo de aprendizagem, dependendo muito do professor, que desempenha um papel ativo na mediação do conhecimento. *"O melhor do PBL realmente, é a questão de praticar. Porque no método tradicional, é aquela coisa de ver o professor fazendo você ver, você entende, mas na hora que você for fazer, você tem dificuldade. E o PBL força a gente praticar. Descobre aquela dificuldade vai tirar dúvida com professor, depois que treinou."* (E1).

O **autodidatismo** é estimulado pois o estudante é sempre incentivado a aprender

algo, sem ter um professor ou mestre lhe ensinando ou ministrando aulas. *"Antigamente eu tinha o costume de esperar o professor dar aula para aprender os conteúdos. Apesar de no MI de programação, em algumas partes terem sido assim. A gente ia para sessão, discutia e não sabia o que fazer. Ia pra aula do professor já tinha noção do que fazer, do que estudar. Então, o PBL influencia muito a gente a pesquisar por si próprio. Então, por exemplo: eu quero fazer tal coisa, eu vou, jogo no google como fazer. Aparece um livro, um artigo, um site explicando, uma videoaula. A gente vai, começa a assistir, começa a fazer. A pessoa se torna mais autodidata, digamos assim. Porque o PBL influencia, né! A pessoa querer pesquisar, querer fazer. De certa forma, força a pessoa aprender sozinha também."* (E4).

Ainda, há o desenvolvimento da **comunicação oral** proporcionada pela própria dinâmica das sessões tutoriais, na qual os estudantes precisam dialogar e discutir as possíveis soluções para os problemas. *"Então, alunos que têm dificuldade de comunicação às vezes são retraídos, tímidos e tal, tem ali uma oportunidade de desenvolver essas habilidades, melhorar pelo menos essas qualidades importantes da comunicação de diferentes formas da atitude de participar de uma reunião, de expor suas ideias e suas opiniões[...]."* (P1).

Finalmente, a metodologia PBL aprimora as **relações interpessoais**. As discussões sobre as possíveis soluções para os problemas ocorrem dentro das sessões tutoriais e também extra-classe. Os estudantes se ajudam, de forma colaborativa, na construção de novos conhecimentos, aprendem a se expressar e também a ouvir, a aceitar opinião do colega, e com isso, desenvolvem o respeito mútuo. *" Outra vantagem do PBL, é a questão das sessões também, da discussão. Por que a discussão[...] Bem, quando você está fazendo sozinha, às vezes você vê uma coisa e você fica perdido, aí uma hora ou outra você vai acabar recorrendo só ao professor. E quando está na sessão, você pode recorrer ao professor, pode recorrer aos colegas. E aí, os colegas, às vezes, mesmo que eles não saibam, mas tem alguma dica. Aí você vai atrás daquela dica para descobrir o assunto. Então, é mais fácil de estudar. É como se fosse um grupo de estudo, é mais fácil do que estudar só."* (E1).

5.5 Relação entre Motivação e Aprendizagem

Nesta pesquisa, investigamos motivação e aprendizagem e as possíveis correlações entre essas variáveis nos Módulos Teóricos e no MI de Programação. Computamos as correlações entre os problemas do MI de Programação com as categorias motivacionais do modelo ARCS, obtidos através do IMMS. Da mesma forma, computamos as correlações dos Módulos Teóricos com as categorias motivacionais do modelo ARCS, obtidos através do CIS. A partir destes resultados, fizemos a análise a seguir.

A categoria atenção está relacionada com a manutenção do interesse dos alunos durante a aprendizagem. A correlação entre atenção e aprendizagem apresentou diferentes resultados nos diferentes problemas do MI de Programação. No Problema

1, não houve correlação entre a atenção e o desempenho. Ou seja, independente da atenção, os alunos tinham bom desempenho de notas. Uma possível explicação para isso é que o Problema 1 era fácil e os conceitos exigidos eram simples. *"O Problema 1 foi bom porque a gente estava aprendendo o básico ainda. Então, o problema foi bem simples. Que foi muito bom por isso, para aprender o básico da programação mesmo."* (E1).

Por outro lado, nos Problemas 2 e 4, a atenção teve correlação com o desempenho, sendo mais alta no Problema 4. No caso do Problema 4, acreditamos que isto pode estar relacionado com o aprendizado do conceito de interface gráfica. Os estudantes sentiram-se atraídos pela possibilidade de interação com usuário onde os programas são apresentados em modo gráfico. *"O Problema 4, foi muito bom. A gente aprendeu realmente. Porque quando você fazia um programa antes, a gente fazia só aquela leitura pelo console. Não tinha interface, não tinha nada. E aí a gente aprendeu como interagir realmente com o usuário. E isso é muito legal."* (E1). Outra questão a ser considerada, é que apesar da complexidade do problema, boa parte dos estudantes o consideraram interessante e desafiador. Keller (2010) afirma que a atenção está associada com a captura do interesse dos alunos e, para isso, é necessário introduzir novas atividades que sejam desafiadoras. E descreve a importância da utilização de aspectos visuais concretos para estimular a curiosidade.

De modo geral, a atenção não apresentou correlação significativa com os resultados de notas dos módulos teóricos. O resultado mais positivo foi em Estrutura de Dados onde houve correlação moderada. Uma explicação possível é que os conceitos de Estruturas de Dados eram novidade e além disso, considerados importantes para resolução dos problemas. No entanto, em Algoritmos e Programação II e Projeto de Sistemas não houve correlação com o desempenho. Talvez isto se justifique pela própria metodologia PBL que desenvolve nos estudantes a autonomia e leva-os a não necessitar tanta atenção nas aulas teóricas. Nesta metodologia, o aluno está sempre em busca do conhecimento para tentar solucionar os problemas. Do mesmo modo, se tem algo que não ficou claro nas aulas teóricas, eles buscam em outras fontes de pesquisa. *"O PBL influencia a pessoa pesquisar, a querer fazer. E de certa forma, força a pessoa a aprender sozinha. Inclusive levamos essa experiência para os módulos teóricos. É tanto, que no começo desse semestre foi meio complicado porque eu tive alguns problemas de saúde, e eu faltei algumas aulas. Só que não influenciou tanto porque quando eu chego em casa dá para estudar e pegar o conteúdo. E é algo que a gente pega com a prática do PBL."* (E4).

No Problema 3, excepcionalmente, foram baixas todas as correlações entre as categorias motivacionais do ARCS e os resultados de notas. Ou seja, houve um deslocamento total da motivação dos estudantes em relação ao desempenho. Vale ressaltar é que mesmo estudantes menos motivados tiraram boas notas. Uma possível explicação para a baixa motivação é que a maioria dos alunos consideraram este problema complexo. Sobre o desempenho positivo, acreditamos que ocorreu porque eles sabem que a disciplina é importante e, por isso, se esforçam para ter bons resultados.

A relevância sempre foi alta em todos os problemas do MI de Programação e também nos módulos teóricos. Contudo, não há correlação entre a relevância com desempenho de notas dos problemas. Isto é, neste caso, a relevância é um fator motivacional importante mas não necessariamente se refletiu em resultados. Uma justificativa provável é que os estudantes são *majors*, gostam das disciplinas e sabem que programação de computadores é essencial para suas vidas profissionais. Assim, achar os assuntos dos módulos relevantes não significa obrigatoriamente que terão sucesso. Talvez a relevância tenha mais importância para *non-majors* do que para um estudante de graduação em Computação, como já foi percebido em outro trabalho [Santana et al. 2017]. Neste caso, pode não ser tão eficaz o professor do MI de Programação produzir conteúdos de aprendizagem relevantes para os objetivos e estilos de aprendizagem dos estudantes, pois isto pode não interferir nos resultados. Entretanto, o professor pode ter mais chances de sucesso ao fazer algo relacionado a atenção, confiança e satisfação.

Em Estruturas de Dados, diferente dos outros módulos teóricos, houve correlação muito forte e significativa entre a relevância e o resultado de notas. A relevância provavelmente teve impacto no bom desempenho dos estudantes. A maioria dos estudantes considerava os conceitos de estruturas de dados relevantes para suas vidas profissionais. Acreditamos também que a metodologia utilizada nas aulas pelo professor pode ter influenciado, aumentando esta percepção. Eram implementados exemplos práticos de aplicações reais, e isto talvez tenha contribuído para este resultado. *"Estruturas de Dados afetou totalmente positivamente a minha motivação. Era algo de novo. As aulas eram muito boas."* (E2). Keller (2010), afirma que a relevância se divide em subcategorias, dentre elas a orientação ao objetivo. A orientação ao objetivo é um componente chave da relevância por estar relacionado com o benefício que o aprendizado daquele conhecimento ou a aquisição de certa habilidade poderá trazer no futuro como, por exemplo, conseguir um emprego. Esse tipo de motivação utilitarista é provavelmente o fator de relevância mais influente.

A categoria motivacional confiança apresentou correlação forte com resultado de notas em todos os problemas, exceto no Problema 3. Este é o construto do ARCS que mais está relacionado com desempenho dos alunos. Portanto, o que provavelmente mais contribuiu para o sucesso. A partir dos nossos resultados, temos uma forte suspeita que quanto mais confiantes, mais os estudantes aprendem. Por outro lado, se o problema é mal elaborado, provavelmente não importa a confiança.

Apesar da confiança não ter sido muito alta, principalmente nos Problemas 1 e 2, quando existiu confiança, houve bons resultados. Uma possível explicação é que, no início, os estudantes se sentem inseguros porque há muitas novidades. Porém, no decorrer do semestre, a confiança vai evoluindo à medida em que os projetos são concretizados. *"O método PBL primeiramente me dá muito nervoso, porque eu fico achando que nunca vou conseguir resolver os problemas. Mas depois que o problema passa e que eu consigo encontrar a solução, me dá uma sensação de que eu consigo fazer muito mais as coisas, sabe? Me mostra que eu consegui me superar, isso é uma das coisas que eu acho mais legal. Porque a gente sempre se*

desespera quando pega o problema de começo. Mas quando consegue entregar, e ver o programa funcionando, é uma sensação de realização muito grande." (E6). Segundo Keller (2010), a confiança possui diferentes componentes: interesse (preferências e atenção num determinado contexto); relevância (utilidade percebida e objetivos de uma determinada atividade); expectativas (perspectivas de sucesso do próprio indivíduo); e resultados (valor de reforço dos resultados obtidos, aqui, dos ganhos alcançados com a entrega dos programas funcionando).

Houve correlação entre a confiança e os resultados em Algoritmos e Programação II e Estruturas de Dados. Entretanto, em Projeto de Sistemas, não houve esta correlação. Na verdade, não houve correlação significativa em nenhuma das categorias motivacionais do ARCS e o resultados neste módulo teórico. Neste caso, os alunos não estavam motivados e foi o módulo teórico com pior desempenho. Uma justificativa possível, é que a maior parte dos estudantes achou os conceitos trabalhados muito subjetivos e ambíguos, dificultando o entendimento e, por isso, foram aumentadas as incertezas no momento da resolução dos problemas e das avaliações teóricas. *"Em relação às avaliações, tinha a questão da interpretação, porque tinha o problema, a gente interpretava de uma forma que, para a gente, estava certo, mas aí o professor corrigia e dizia que estava errado [...] E aí nós ficávamos inseguros."* (E1).

A satisfação também traz muitos benefícios. Isto pode ser visto no caso do Problema 4. A realização dos estudantes é tão grande que a satisfação casa adequadamente com os resultados. Por isso, é importante pensar em desenvolver atividades que, ao final, os estudantes tenham satisfação. *"O Problema 4, eu achei muito interessante porque a gente podia se basear numa ferramenta que a gente poderia ver como que estava funcionando. E isso poderia auxiliar o nosso processo de desenvolvimento. Além do que, no Problema 4, foi exigido da gente a utilização de interface gráfica. E isso era uma coisa nova, que a gente tinha que aprender. Acho que acabou motivando também a apresentar um bom resultado no final. A partir do exemplo de uma aplicação real que muita gente utiliza."* (E6).

Ainda, a correlação da satisfação com os resultados também podem ser percebidos em Algoritmos e Programação II e Estruturas de Dados. De modo geral, os estudantes estavam satisfeitos com as aulas, com o professor e também com a maneira como eram avaliados. Além disso, os conhecimentos teóricos aprendidos eram aplicados nos problemas. Esta aprendizagem através do experienciamento traz satisfação para o aprendiz, o que é confirmado por Keller (2010), quando diz que é uma experiência muito satisfatória para um aluno ser capaz de realizar com sucesso uma tarefa desafiadora, ao final de uma aula, que ele não poderia fazer no começo. Um dos resultados mais recompensadores da instrução orientada para o desempenho é usar as habilidades ou conhecimentos recém-adquiridos.

5.6 Lições aprendidas

Os benefícios gerados com a integração dos conhecimentos, o cuidado no planejamento do problema, a aquisição de competências pessoais, interpessoais e técnicas, e as dificuldades enfrentadas por professores e estudantes, e a importância da confiança como construto motivacional, são algumas das lições aprendidas nesse trabalho.

A metodologia PBL caracteriza uma estratégia de formação através da qual os estudantes são confrontados com problemas contextualizados e estruturados em que se empenham em encontrar soluções significativas. O caminho trilhado em direção à solução permite a **aquisição de competências pessoais, interpessoais e técnicas**. As competências pessoais estão relacionados com a autonomia, capacidade de solucionar problemas e tomar decisões. Competências interpessoais consistem na capacidade de trabalhar em grupo, liderança e comunicação. *"O que tem de melhor no método PBL para o aluno é a construção do seu conhecimento, ele próprio tentar apender por conta própria, o estímulo, a criatividade, a autonomia, o desenvolvimento das habilidades interpessoais, que isso não existe no método tradicional, o lado da comunicação é totalmente estimulada no método PBL"* (P1). As competências técnicas estão relacionadas com a própria dinâmica que preconiza a metodologia PBL. *"O método PBL é bom porque faz com que a gente aprenda na prática."* (E1). A experiência permite o aprimoramento de habilidades técnicas adquiridas com a vivência prática proporcionada na resolução dos problemas.

Os benefícios gerados pela **integração de conhecimentos** são uma das principais contribuições ao aprendizado do desenvolvimento de software. Os estudantes, ao longo de um semestre, experimentaram a programação orientada a objetos, estruturas de dados e projeto de sistemas em uma organização pedagógica onde teoria e prática caminhavam juntas. *"O MI auxilia na prática, no MI é a oportunidade que eles têm de praticar, de fazer o código, de escrever, de testar a recursão, de ver um problema maior, real. Eu consigo perceber que eles estão sendo estimulados no MI e que, consequentemente, leva ao aprendizado na disciplina teórica."* (P2). Isto permitiu que os estudantes realizassem projetos mais autênticos adquirindo experiência nas várias etapas do ciclo de vida do software em um modo de trabalho mais disciplinado, o que será útil no desenvolvimento de seus cursos e em suas vidas profissionais. *"O método PBL é desafiador e se aproxima da vida real, parece que sai de algo estritamente acadêmico e se aproxima muito do mundo real."* (E2).

O **planejamento dos problemas** tem um impacto profundo na aprendizagem. Um problema bem elaborado pode elevar os níveis motivacionais, assim como, quando mal planejado, pode levar a resultados catastróficos. A qualidade dos problemas PBL está entre os maiores desafios. O excesso de conceitos, a escolha do domínio do problema e a complexidade exigida são aspectos que devem ser considerados. Os conceitos nos problemas propostos devem seguir um processo gradual, de modo que permita capacidade de assimilação por parte dos estudantes. A escolha do domínio do problema deve introduzir fundamentos próximos à realidade dos estudantes,

sendo uma estratégia para tornar o problema mais atraente e motivador. A complexidade do problema deve garantir que, com cooperação, os estudantes consigam solucionar o problema. *"O que eu observo é o seguinte, quando você consegue elaborar um problema mais próximo da vivência dos alunos melhor, então por exemplo, o problema do aplicativo de viagens para smartphones que simulam trajetos turísticos foi o mais bem recebidos pelos estudantes"*(P1).

A **inserção de novidades** é essencial para manter os níveis de motivação. Keller (1987) afirma que não importa o quão bem sucedidas sejam algumas das estratégias de ensino, elas não permanecerão assim para sempre. Um erro cometido por professores e instrutores é que quando eles encontram uma estratégia que é altamente bem-sucedida, eles se exaltam e tendem a abusar dela. Cada nova estratégia tem um efeito inovador junto com qualquer nível mais profundo de conexão motivacional. O simples fato de ser novo pode estimular uma certa quantidade de interesse. Mas quando a novidade passar, a estratégia continuará a ser motivadora apenas se tiver uma conexão substancial e significativa com as exigências motivacionais dos alunos. Mesmo assim, eles se cansarão disso, porque o desejo de novidade é um aspecto da motivação humana.

As **dificuldades enfrentadas pelos professores** estão relacionadas com as intervenções nas sessões tutoriais e o discernimento ao lidar com as relações interpessoais e a avaliação dos estudantes. O professor desempenha o papel de facilitador e é responsável por orientar e monitorar o trabalho em grupo. Ele deve ter a capacidade de desempenhar diversos papéis nas diversas fases do processo. A princípio deve ser capaz de imaginar como uma pessoa com conhecimento limitado sobre determinado conteúdo se comporta. A partir disso, adotar estratégias nas quais a sua colaboração seja relevante para o processo. Em contrapartida, as suas intervenções não podem ser exageradas, por que assim, fugiria do escopo proposto pela metodologia PBL. A intervenção requer uma habilidade empírica, em que o professor deve ser sensível ao processo do desenvolvimento do conhecimento. Além disso, o professor deve saber lidar com as relações interpessoais. Saber gerenciar conflitos, ter flexibilidade, verificar as inconsistências, considerar as alternativas, e sempre conduzir para obter maior produtividade e coesão entre o grupo.

A avaliação no PBL é uma tarefa difícil. Uma das principais dificuldades encontra-se em avaliar o desempenho dos estudantes, verificar a autenticidade do produto e montar um barema apropriado de avaliação. O desempenho dos estudantes é medido através da sua participação nas sessões tutoriais do PBL. A participação dos estudantes na sessão é extremamente importante para a troca de ideias em busca de soluções. Porém, há uma diversidade de perfis psicológicos: muitos estudantes são tímidos, introvertidos, calados; outros são participativos, extrovertidos, comunicativos. Não podemos afirmar que o estudante calado ou tímido não está estudando ou participando. Às vezes, uma participação curta, mas precisa, pode ser significativa. Entender as diversas subjetividades dos perfis dos estudantes e encontrar o equilíbrio ao avaliar é um dos grandes desafios do professor. *" No método PBL, alunos que têm dificuldade de comunicação, às vezes, são retraídos, tímidos e tal, têm ali uma*

oportunidade de desenvolver essas habilidades. Melhorar pelo menos essas qualidades importantes da comunicação, de diferentes formas da atitude, como participar de uma reunião, expor suas ideias e suas opiniões. Além de saber lidar com as diferenças, e entendendo que aquilo ali é uma simulação da realidade. Se ele estiver no trabalho, certamente ele vai ter que trabalhar com um grupo de pessoas. Com a experiência nas sessões, ele já vai estar muito mais bem preparado para colaborar." (P1).

Garantir a autenticidade do produto, é outra dificuldade encontrada pelos professores. Esse é um problema típico de trabalhos extra-classe. Assegurar que não houve plágio não é uma tarefa fácil. Os relatórios individuais tentam resolver esse problema, mas nem sempre são suficientes. Além disso, ao avaliar os produtos, os professores devem elaborar um barema para avaliação. Para montar o barema, é necessário o *feedback* dos professores, e nem sempre isso acontece. A maior dificuldade está em encontrar o barema apropriado para cada problema.

Os **estudantes enfrentaram dificuldades** na transição do paradigma de programação, capacidade de abstração, no desenvolvimento da autonomia. A transição do paradigma estruturado para o paradigma orientado a objetos traz um conflito cognitivo para os estudantes, geralmente demorado de resolver. Por outro lado, aprender programação orientada a objetos não é fácil. As dificuldades podem ser causadas pela complexidade dos conceitos a serem aprendidos em um curto período de tempo, pela complexidade intrínseca destas linguagens e também dos ambientes de desenvolvimento profissionais. *"No início, tem o choque da mudança da linguagem estruturada para linguagem orientada a objetos, mas quando você aprende, percebe que é bem melhor"* (E1). Além disso, a metodologia PBL naturalmente desenvolve a autonomia nos estudantes. Aqueles que não possuem essa característica enfrentam problemas. *"O MI de programação tem uma capacidade muito grande de envolver os alunos, mas não todos, alunos que têm dificuldade com a programação, com o perfil do curso ou tem dificuldades de personalidade, talvez não fiquem muito motivados."* (P1).

A **confiança** é um fator motivacional fundamental para aprendizagem. Este é construto mais importante dentre todos os outros. Confiança significa: "Eu sou capaz de fazer"; "Eu sou capaz de dirigir o meu próprio estudo"; "Eu sou capaz de dirigir o meu próprio resultado". Segundo Keller, o desejo de se sentir competente é uma motivação humana básica e o grau com que alguém se sente competente em determinada situação reflete-se em seus sentimentos de confiança. A confiança é frequentemente associada a percepções de controle pessoal sobre ser capaz de ter sucesso em uma tarefa e os resultados que se seguem até alcançar o sucesso. No entanto, em um cenário de aprendizado, o controle, muitas vezes, está nas mãos de um professor. Para aumentar a motivação, a influência controladora do professor deve ser focada nas áreas de liderar a experiência e aderir aos padrões esperados. Isso proporciona um ambiente de aprendizado estável, onde o aluno deve ter o máximo de controle pessoal possível sobre a experiência real de aprendizagem.

5.7 Validade e Confiabilidade

Uma possível ameaça à validade em relação aos dados qualitativos é se realmente estamos capturando os fenômenos de motivação e aprendizagem nas entrevistas e observações, bem como, as prováveis relações entre estes fenômenos. A estratégia para tentar minimizar essas ameaças foi a utilização de diversas fontes de pesquisa. Inciamos realizando observações prolongadas das aulas do MI de Programação e dos Módulos Teóricos. Efetuamos entrevistas com professores de todos os módulos teóricos e um professor do MI de Programação, totalizando 4 entrevistas. Além disso, fizemos entrevistas com 3 alunos de dois grupos tutoriais do MI de Programação, totalizando 6 entrevistas. Ainda, coletamos os resultados finais da aprendizagem nos MTs e no MI. Reunimos esses dados e realizamos triangulações para o entendimento das concordâncias e contradições existentes na pesquisa. Merriam (2009) argumenta que o uso de perspectivas tanto qualitativas quanto quantitativas de abordagem de pesquisa, mesmo que divergentes na condução de processos perceptivos do fenômeno, podem enriquecer sua compreensão de um dado fenômeno.

Uma possível ameaça a confiabilidade dos dados quantitativos é a avaliação das notas dos alunos realizada pelos professores. Por outro lado, tentamos minorar esse problema utilizando diferentes notas de todas os módulos que compõem o EI de Programação. Além disto, estes módulos se repetem há vários anos, o que provoca um aumento no grau de confiabilidade. Em relação à motivação, essa ameaça é bem menor pois o IMMS e o CIS são instrumentos testados e validados, além de utilizados largamente.

A validade de construto refere-se à demonstração de que o instrumento realmente mede aquilo a que se propõe medir [Creswell 2002]. No que diz respeito aos dados quantitativos, existem ameaças à validade de construtos nos questionários que utilizamos para mensurar motivação e aprendizagem. Entretanto, a validade de construto é muito bem definida nesses instrumentos porque já foram testados e avaliados. Em relação às notas dos alunos, temos o problema da subjetividade no processo avaliativo. Entretanto, essa maneira de avaliar é popular em toda a comunidade de educação.

Ameaças à validade interna são procedimentos, tratamentos ou experiências dos participantes que ameaçam a capacidade dos pesquisadores de fazer inferências corretas a partir dos dados em um experimento [Creswell 2002]. De modo geral, a validade interna relacionada aos dados quantitativos é pouco ameaçada porque usamos instrumentos testados e validados. Além disso, em alguns casos, como nos problemas do MI, medimos várias vezes em diversas etapas do curso.

Ainda, como estamos analisando motivação e aprendizagem e suas relações em potencial, uma possível ameaça à validade interna podia ser a influência do "professor carismático"nos resultados de motivação e aprendizagem. Entretanto, estamos interessados em avaliar como a abordagem que utiliza a abordagem PBL influencia na motivação e na aprendizagem dos alunos independente do professor. Desta ma-

neira, acreditamos que isto é minorado pela própria dinâmica da abordagem. Nesta abordagem, os estudantes exercitam autonomia e independência. Eles conduzem as sessões tutoriais, por exemplo, e o professor intervém eventualmente, quando necessário. Então, a ameaça à validade interna é minimizada porque não há tanta influência dos professores.

Existem ameaças à validade externa, porém não temos pretensão de generalizar as nossas descobertas para quaisquer cenários. A nossa preocupação é analisar motivação e aprendizagem dentro do nosso contexto. É analisar em profundidade a nossa experiência pois acreditamos que estudos de caso analisados em profundidade têm contribuições importantes, principalmente, para profissionais de educação e computação.

Capítulo 6

Considerações Finais

Neste capítulo, estão descritas as conclusões deste trabalho e sugestões de trabalhos futuros.

6.1 Conclusões

Este trabalho apresentou um estudo de caso de uma abordagem integrada para ensino e aprendizagem de programação orientada a objetos, estruturas de dados e projeto de sistemas baseada em PBL e realizada durante o segundo semestre letivo de 2017 em um curso de Engenharia da Computação.

Apresentamos o cenário, os participantes, a avaliação, o planejamento dos módulos teóricos e integrador, além dos resultados, discussão e lições aprendidas. Foram avaliados os níveis de motivação dos estudantes utilizando o Modelo ARCS de Keller. A aprendizagem foi mensurada a partir das notas dos estudantes nos módulos teóricos e integrador. Os resultados descrevem em detalhes as variáveis de motivação e aprendizagem e como a abordagem PBL as afeta, além da relação específica entre motivação e aprendizagem.

Em relação aos resultados do MI de Programação e às quatro dimensões de motivação do ARCS, observamos que níveis de relevância foram relativamente altos em todos os problemas, possivelmente explicados por tratar-se de estudantes da área de TI. Os níveis de confiança foram um pouco mais baixos pelas prováveis dificuldades que os estudantes têm com os conceitos de programação orientada a objetos e as diversas competências a serem adquiridas. Os níveis de satisfação e atenção foram relativamente altos em três dos quatro problemas.

No que diz respeito aos resultados dos Módulos Teóricos e às quatro categorias de motivação do ARCS, percebemos que todos os módulos eram relevantes para os estudantes. Atribuímos isto à importância dada aos conceitos teóricos aprendidos

nestes módulos para o desenvolvimento de software. Além disso, os níveis de confiança foram mais altos quando os assuntos se relacionavam com a prática vivenciada nos problemas. Sobre a atenção, observamos Projeto de Sistemas teve resultados mais baixos que os outros dois módulos, o que pode ser explicado pela percepção dos estudantes de aplicação potencial dos conceitos dos módulos na resolução de problemas de software. A satisfação foi menor quando o módulo apresentava os conceitos de modo mais subjetivo e abstrato, como em Projeto de Sistemas.

Percebe-se que a aprendizagem de habilidades técnicas e comportamentais é potencializada pela abordagem PBL, que permite aos estudantes adotarem uma postura mais ativa no processo de aprendizagem. A aquisição de conceitos, por outro lado, é potencializada pela integração curricular, que permite que conceitos trabalhados na prática realimentem a teoria e vice-versa. Por outro lado, deve haver um cuidado adicional com a elaboração dos problemas, principalmente com a dosagem adequada de conceitos, a complexidade mantida sob controle e a escolha de domínios familiares aos estudantes.

Durante o trabalho, foram evidenciadas as relações entre a abordagem PBL e a motivação dos estudantes, entre a abordagem PBL e a aprendizagem de conceitos e habilidades técnicos e comportamentais, e entre a motivação dos estudantes e os resultados de aprendizagem, discutidas detalhadamente em uma discussão pormenorizada.

Algumas lições foram aprendidas nesta pesquisa. Acreditamos que o percurso trilhado pelos estudantes para solucionar os problemas permite a aquisição de competências pessoais, interpessoais e técnicas. Além disso, os professores enfrentam dificuldades para avaliar o desempenho dos estudantes, como por exemplo em relação a avaliação e a identificação de plágios. Estudantes também enfrentam dificuldades, entre elas, as relacionadas com a transição do paradigma estruturado para o paradigma orientado a objetos. Ainda, o cuidado na elaboração do problema é um fator sensível, questões como domínio do problema, complexidade dos conteúdos, a quantidade de conceitos e a introdução de novidades são fatores essenciais. Finalmente, a confiança é um fator motivacional que contribui para o sucesso na aprendizagem.

Com este trabalho, contribuímos na avaliação da motivação e do processo de ensino-aprendizagem de estudantes de computação que aprendem através da abordagem PBL. Este conhecimento permitirá potencialmente avançar no processo de disseminação e na eficácia desta abordagem na área de computação.

6.2 Trabalhos Futuros

Possíveis trabalhos futuros envolvem o impacto da motivação no processo de aprendizagem através de modelos quantitativos mais completos como técnicas de regressão múltipla ou logística.

Motivação é um dos aspectos não-cognitivos relevantes para a aprendizagem de computação, mas outros fatores como engajamento, atitudes e disposições dos estudantes podem ser agregados a trabalhos desta natureza. Além disso, aspectos cognitivos como conhecimentos prévios e aptidões também merecem ser adicionados a uma análise mais completa.

Finalmente, é possível replicar este trabalho em um contexto onde se utilize uma abordagem instrucionista tradicional ou mesmo outras abordagens ativas. Também é possível, de posse destes dados, fazer comparações com os resultados encontrados neste trabalho.

Referências Bibliográficas

[Adair e Jaeger 2011] Adair, D. e Jaeger, M. (2011). Difficulties in Teaching and Learning the Java Programming Language. (October 2015).

[Albanese e Mitchell 1993] Albanese, M. A. e Mitchell, S. (1993). Problem-based learning: a review of literature on its outcomes and implementation issues. Academic Medicine : Journal of the Association of American Medical Colleges, 68(1):52–81.

[Álvarez et al. 2005] Álvarez, I., Ayuste, A., Gros Salvat, B., Guerra, V., e Romañá, T. (2005). Construir conocimiento con soporte tecnológico para un aprendizaje colaborativo. Revista Iberoamericana de Educación (OEI), 2005, num. 36/1, pp. 1–15.

[Angelo e Bertoni 2012] Angelo, M. F. e Bertoni, F. C. (2012). Análise da aplicação do método PBL no processo de ensino e aprendizagem em um curso de engenharia de computação. Revista de Ensino de Engenharia, 30(2):35–42.

[Angelo et al. 2014] Angelo, M. F., Loula, A. C., Bertoni, F. C., e Santos, J. A. M. (2014). Aplicação e avaliação do método PBL em um componente curricular integrado de programação de computadores. Revista de Ensino de Engenharia, 33(2):31–43.

[Baeza-Yates 1995] Baeza-Yates, R. A. (1995). Teaching algorithms. ACM SIGACT News, 26(4):51–59.

[Barrows e Tamblyn 1980] Barrows, H. e Tamblyn, R. (1980). Problem based-learning: An approach to medical education, volume 1. Springer Publishing Company.

[Bennedsen e Caspersen 2007] Bennedsen, J. e Caspersen, M. E. (2007). Failure rates in introductory programming. SIGCSE Bull., 39(2):32–36.

[Bittencourt e Figueiredo 2003] Bittencourt, R. A. e Figueiredo, O. A. (2003). O Currículo do Curso de Engenharia de Computação da UEFS: Flexibilização e Integração Curricular. In Anais do XXIII Congresso da Sociedade Brasileira de Computação, pp. 171—-182, Campinas, São Paulo. SBC.

[Bittencourt et al. 2013] Bittencourt, R. A., Rodrigues, C. A., e Cruz, D. S. S. (2013). Uma Experiência Integrada de Programação Orientada a Objetos, Es-

truturas de Dados e Projeto de Sistemas com PBL. Anais do XXXIII Congresso da Sociedade Brasileira de Computação - XXI Workshop sobre Educação em Computação., pp. 591–600.

[Bruner 2009] Bruner, J. S. (2009). The process of education. Harvard University Press.

[Burton e Bruhn 2003] Burton, P. J. e Bruhn, R. E. (2003). Teaching programming in the oop era. ACM SIGCSE Bulletin, 35(2):111–114.

[Cheiran et al. 2017] Cheiran, J. F. P., de M Rodrigues, E., de S Carvalho, E. L., e da Silva, J. P. S. (2017). Problem-based learning to align theory and practice in software testing teaching. In Proceedings of the 31st Brazilian Symposium on Software Engineering, pp. 328–337. ACM.

[Cintra e Bittencourt 2015] Cintra, C. d. S. e Bittencourt, R. A. (2015). Being a PBL Teacher in Computer Engineering : An Interpretative Phenomenological Analysis. In Frontiers in Education Conference (FIE), 2015 IEEE, pp. 1–8. IEEE.

[Clark e Jenkins 1999] Clark, M. e Jenkins, T. (1999). The expectations and intentions of new information systems undergraduates. In Proceedings of UKAIS Annual Conference, York.

[Cooper 2010] Cooper, S. (2010). The design of alice. ACM Transactions on Computing Education (TOCE), 10(4):15.

[Creswell 2002] Creswell, J. W. (2002). Educational research: Planning, conducting, and evaluating quantitative. Prentice Hall Upper Saddle River, NJ.

[Delisle 1997] Delisle, R. (1997). How to use problem-based learning in the classroom. Ascd.

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

[Duch et al. 2001] Duch, B. J., Groh, S. E., e Allen, D. E. (2001). The power of problem-based learning: a practical"how to"for teaching undergraduate courses in any discipline. Stylus Publishing, LLC.

[Fallows e Ahmet 1999] Fallows, S. J. e Ahmet, K. (1999). Inspiring students: Case studies in motivating the learner. Psychology Press.

[Ferreira et al. 2007] Ferreira, G. M., Nascimento, M. Z., Assis, K. D. R., e Ramos, R. P. (2007). Teaching object oriented programming computer languages: Learning based on projects. In Proceedings of the International Conference on Software Engineering Advances, ICSEA '07, pp. 81–, Washington, DC, USA. IEEE Computer Society.

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

[Gomes 2010] Gomes, A. d. J. (2010). Dificuldades de aprendizagem de programação de computadores: contributos para a sua compreensão e resolução. PhD thesis.

[Gomes e Mendes 2015] Gomes, A. J. e Mendes, A. J. (2015). À procura de um contexto para apoiar a aprendizagem inicial de programação. Educação, Formação & Tecnologias-ISSN 1646-933X, 8(1):13–27.

[Herala et al. 2015] Herala, A., Vanhala, E., e Nikula, U. (2015). Object-oriented programming course revisited. In Proceedings of the 15th Koli Calling Conference on Computing Education Research, Koli Calling '15, pp. 23–32, New York, NY, USA. ACM.

[Jenkins 2001] Jenkins, T. (2001). The motivation of students of programming. In ACM SIGCSE Bulletin, volume 33, pp. 53–56. ACM.

[Jenkins 2002] Jenkins, T. (2002). 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, pp. 53–58.

[Kay et al. 2000] Kay, J., Barg, M., Fekete, A., Greening, T., Hollands, O., Kingston, J. H., e Crawford, K. (2000). Problem-Based Learning for Foundation Computer Science Courses. Computer Science Education, 10(2):109–128.

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

[Keller 2010] Keller, J. M. (2010). Motivational Design for Learning and Performance: The ARCS Model Approach. Springer US.

[Kinnunen e Malmi 2005] Kinnunen, P. e Malmi, L. (2005). Problems in problem-based learning-experiences, analysis and lessons learned on an introductory programming course. Informatics in Education, 4(2):193.

[Kölling 1999a] Kölling, M. (1999a). The problem of teaching object-oriented programming. 11(9):6–12.

[Kölling 1999b] Kölling, M. (1999b). The Problem of Teaching Object-Oriented Programming, Part 1: Languages. 11(8):8–15.

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

[Kölling et al. 2003] Kölling, M., Quig, B., Patterson, A., e Rosenberg, J. (2003). The BlueJ System and its Pedagogy. Computer Science Education, 13(4):1–12.

[Libarkin e Anderson 2005] Libarkin, J. C. e Anderson, S. W. (2005). Assessment of learning in entry-level geoscience courses: Results from the geoscience concept inventory. Journal of Geoscience Education, 53(4):394–401.

[Martins et al. 2010] Martins, S. W., Mendes, A. J., e Figueiredo, A. D. (2010). A strategy to improve student's motivation levels in programming courses. In Frontiers in Education Conference (FIE), 2010 IEEE, pp. F4F–1. IEEE.

[Merriam 2009] Merriam, S. B. (2009). Qualitative Research: A Guide to Design and Implementation. Revised and expanded from Qualitative research and case study applications in education. John Wiley & Sons, 2009.

[O'Grady 2012] O'Grady, M. J. (2012). Practical problem-based learning in computing education. Trans. Comput. Educ., 12(3):10:1–10:16.

[Oliveira et al. 2012] Oliveira, A., Rodrigues, R., e Garcia, V. (2012). Um Mapeamento Sistemático para Problem Based Learning aplicado à Ciência da Computação. In Anais do Congresso Brasileiro de Informática na Educação – Workshop de Informática na Escola.

[Ortiz et al. 2003] Ortiz, J. A. M., González, A. G., Marcos, A. P., Victoria, M., e Nardiz, A. (2003). Aprendizaje basado en problemas: una alternativa al método tradicional. Revista de Docencia Universitaria, 3(2).

[Parker et al. 2016] Parker, M. C., Guzdial, M., e Engleman, S. (2016). Replication, validation, and use of a language independent cs1 knowledge assessment. In Proceedings of the 2016 ACM Conference on International Computing Education Research, ICER '16, pp. 93–101, New York, NY, USA. ACM.

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

[Prince 2004] Prince, M. (2004). Does Active Learning Work? A Review of the Research. Journal of Engineering Education, 93(3):223–231.

[Ray 1964] Ray, W. S. (1964). The Science of psychology: an introduction. Macmillan.

[Robins et al. 2003] Robins, A., Rountree, J., e Rountree, N. (2003). Learning and teaching programming: A review and discussion. Computer Science Education, 13(2):137–172.

[Santana et al. 2017] Santana, B., Figuerêdo, J., e Bittencourt, R. (2017). Motivaçao de estudantes non-majors em uma disciplina de programaçao. In XXV WEI–XXV Workshop sobre Educaç ao em Computaç ao, Sao Paulo. Anais do XXXVII Congresso da Sociedade Brasileira de Computaçao.

[Santos et al. 2007] Santos, D. M. B., Ribeiro, G., Rezende, P., Pinto, C., Sena, P., Bertoni, F. C., Bittencourt, R. A., Estadual, U., Km, E., Universit, C., e Brasil, N. (2007). Aplicação do método de aprendizagem baseada em problemas no curso de engenharia de computação da universidade estadual de feira de santana. XXXV Congresso Brasileiro de Educação em Engenharia, pp. 2A07–1–2A07–14.

[Savery e Duffy 1995] Savery, J. R. e Duffy, T. M. (1995). Problem based learning: An instructional model and its constructivist framework. Educational technology, 35(5):31–38.

[Tew e Guzdial 2011] Tew, A. E. e Guzdial, M. (2011). The fcs1: A language independent assessment of cs1 knowledge. In Proceedings of the 42Nd ACM Technical Symposium on Computer Science Education, SIGCSE '11, pp. 111–116, New York, NY, USA. ACM.

[Utting et al. 2013] Utting, I., Tew, A. E., McCracken, M., Thomas, L., Bouvier, D., Frye, R., Paterson, J., Caspersen, M., Kolikant, Y. B.-D., Sorva, J., e Wilusz, T. (2013). A fresh look at novice programmers' performance and their teachers' expectations. In Proceedings of the ITiCSE Working Group Reports Conference on Innovation and Technology in Computer Science Education-working Group Reports, ITiCSE -WGR '13, pp. 15–32, New York, NY, USA. ACM.

[Vihavainen et al. 2011] Vihavainen, A., Paksula, M., e Luukkainen, M. (2011). Extreme apprenticeship method in teaching programming for beginners. In Proceedings of the 42Nd ACM Technical Symposium on Computer Science Education, SIGCSE '11, pp. 93–98, New York, NY, USA. ACM.

[Yadav et al. 2015] Yadav, A., Burkhart, D., Moix, D., Snow, E., Bandaru, P., e Clayborn, L. (2015). Sowing the Seeds : A Landscape Study on Assessment in Secondary Computer Science Education.

Apêndice A

TCLE – Termo de Consentimento Livre e Esclarecido

UEFS
universidade estadual de feira de santana

TERMO DE CONSENTIMENTO LIVRE E ESCLARECIDO

Título do projeto: Análise de Implementação da Metodologia de Aprendizagem Baseada em Problemas e Projetos no Curso de Engenharia de Computação da UEFS
Pesquisador responsável: Carlos Alberto Rodrigues
Pesquisadores colaboradores: Armando Sanca Sanca, Elisangela Oliveira Carneiro, Roberto Almeida Bittencourt, Thiago Cerqueira de Jesus, Tiago Amador Coelho, Fernanda Castelo Branco Santana e Ayala Lemos Ribeiro

Convidamos você para participar desta pesquisa cujo objetivo é avaliar a metodologia PBL (Aprendizagem Baseada em Problemas) no curso de Engenharia de Computação da UEFS. Dada a complexidade e importância da metodologia no curso, a avaliação de elementos de tal metodologia é de extrema relevância para mensurar os seus resultados. O levantamento de informações será através da aplicação de questionários, entrevistas e observações *in loco*. Para analisar os dados coletados nos 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 o de contribuir com a aplicação da metodologia PBL e a identificação de aspectos que necessitam de aprimoramento. Um possível risco seria se, por algum motivo, você se sentir constrangido ao responder os questionários ou entrevistas, ou ser observado durante as aulas. Porém, você 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. Além disso, garantiremos que o seu anonimato será mantido, respeitando sua integridade intelectual, social e cultural.
Não haverá remuneração ou qualquer custo com a participação na pesquisa e, se porventura houver algum custo, eles serão de inteira responsabilidade dos pesquisadores. 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 por um período de 5 anos, sendo destruídos logo após. 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.

Feira de Santana, _____ de _____ de _____.

_____ _____
Assinatura do participante Assinatura do pesquisador responsável

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

Apêndice B

Questionário Pré-Intervenção

Questionário 1 (Pré-Intervenção) Cód:_____

Instruções: Para cada questão objetiva, assinale apenas uma alternativa dentre 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

Sobre o período atual:

5. Você já cursou o MI de Programação antes?
( ) Sim : _____ vezes ( ) Não

6. Você já cursou Algoritmos e Programação II antes?
( ) Sim : _____ vezes ( ) Não

7. Você já cursou Estruturas de Dados antes?
( ) Sim : _____ vezes ( ) Não

8. Você já cursou Projeto de Sistemas antes?
( ) Sim : _____ vezes ( ) Não

Sobre o primeiro semestre do curso:

9. Quantas vezes você cursou o MI de Algoritmos até a sua aprovação?
_____ vezes

10. Quem foi o seu(sua) tutor(a) no MI de Algoritmos na última vez que você cursou?
( ) Ricardo ( ) Cláudia ( ) Matheus ( ) Danilo ( ) Fernanda
( ) Michele ( ) Jamylle ( ) Aécio ( ) David ( ) Thiago Mariano ( ) Outro:_____

11. Quantas vezes você cursou Algoritmos e Programação I até a sua aprovação?
_____ vezes

12. Quem foi o seu(sua) professor(a) em Algoritmos e Programação I na última vez que você cursou?
( ) Ricardo ( ) Cláudia ( ) Michele ( ) Fernanda ( ) Outro:_____

13. Caso tenha cursado Algoritmos e Programação I com Ricardo no período anterior (2017.1), marque um X.
( ) Cursei Algoritmos e Programação I com Ricardo no período anterior.

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 IMMS – Instructional Materials Motivation Survey

QUESTIONÁRIO – MI - PROGRAMAÇÃO
PROBLEMA 1

Sobre o Problema 1, marque a opção que mais se adequa ao seu pensamento.	Discordo totalmente	Discordo parcialmente	Neutro	Concordo parcialmente	Concordo totalmente
1. Quando eu vi os assuntos do Problema 1 pela primeira vez, eu tive a impressão de que seriam fáceis para mim.					
2. Havia algo de interessante no início do Problema 1 que chamou a minha atenção.					
3. Os assuntos do Problema 1 foram mais difíceis de entender do que eu gostaria que fossem.					
4. Depois das primeiras sessões tutoriais do Problema 1, senti-me confiante de que eu sabia o que eu devia aprender.					
5. Completar os exercícios do Problema 1 me deu uma sensação gratificante de realização.					
6. Ficou claro para mim como o conteúdo do Problema 1 está relacionado com coisas que eu já sei.					
7. Várias sessões do Problema 1 tinham tanta informação que era difícil escolher e lembrar quais os pontos importantes.					
8. O conteúdo do Problema 1 foi atraente.					
9. Houve exemplos que me mostraram como o conteúdo do Problema 1 pode ser importante para as pessoas que estão aprendendo programação.					
10. Completar com êxito as atividades do Problema 1 foi importante para mim.					
11. O conteúdo do Problema 1 é tão abstrato que foi difícil manter a minha atenção nas sessões tutoriais.					
12. Enquanto eu participava das sessões tutoriais do Problema 1, eu estava confiante de que eu poderia aprender o conteúdo.					
13. Eu gostei tanto do Problema 1 que eu gostaria de saber mais sobre os conteúdos abordados.					
14. O conteúdo do Problema 1 me pareceu pouco atraente.					
15. O conteúdo aprendido no Problema 1 é relevante para os meus interesses.					
16. A maneira como a informação foi organizada no Problema 1 ajudou a manter minha atenção.					
17. Houve explicações e exemplos de como as pessoas usam o conhecimento adquirido no Problema 1.					
18. Os exercícios no Problema 1 foram muito difíceis.					
19. O Problema 1 teve coisas que estimularam minha criatividade.					
20. Eu realmente gostei de estudar programação com o Problema 1.					
21. Às vezes, a quantidade de repetições feitas sobre alguns assuntos me levou a ficar entediado.					
22. O conteúdo e o estilo do Problema 1 dão a impressão que vale a pena saber aqueles conceitos.					
23. Eu aprendi algumas coisas que foram surpreendentes ou inesperadas.					
24a. Após trabalhar no Problema 1 por um tempo, senti-me confiante de que eu seria capaz de passar na avaliação deste problema.					
24b. O conteúdo do Problema 1 não foi relevante para as minhas necessidades, porque eu já sabia a maior parte do assunto.					
25. O feedback e as observações passadas pelo professor ajudaram-me a sentir recompensado pelo meu esforço.					

26. Durante as sessões tutoriais, a variedade de exercícios, ilustrações, etc., ajudou a manter minha atenção.					
27. Eu pude relacionar os conteúdos que aprendi no Problema 1 com coisas que eu já vi, fiz ou pensei, na minha própria vida.					
28. Eu me senti bem ao concluir com êxito o Problema 1.					
29. O conteúdo visto no Problema 1 será útil para mim.					
30. Eu não pude compreender como algumas coisas eram feitas no Problema 1.					
31. A boa organização do conteúdo no Problema 1 ajudou-me a sentir confiança que eu aprenderia o assunto.					
32. Foi um prazer estudar com a metodologia utilizada no Problema 1.					

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

	Muito Difícil	Difícil	Regular	Fácil	Muito Fácil
33. Classes e objetos.					
34. Atributos, métodos e construtores.					
35. Entrada e saída em Java através do console.					
36. Interfaces em Java.					
37. Padrão MVC.					
38. Modelo conceitual.					
39. Diagrama de classes de projeto.					
40. Tipos de dados abstratos em estruturas de dados.					
41. Lista encadeada: implementação de operações.					
42. Padrão de projeto Iterator: implementação e uso.					

40. De um modo geral, como você classifica a metodologia das sessões tutoriais do Problema 1 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**

QUESTIONÁRIO – MI – PROGRAMAÇÃO
PROBLEMA 2

Cód:_____

Sobre o Problema 2, marque a opção que mais se adequa ao seu pensamento.	Discordo totalmente	Discordo parcialmente	Neutro	Concordo parcialmente	Concordo totalmente
1. Quando eu vi os assuntos do Problema 2 pela primeira vez, eu tive a impressão de que seriam fáceis para mim.					
2. Havia algo de interessante no início do Problema 2 que chamou a minha atenção.					
3. Os assuntos do Problema 2 foram mais difíceis de entender do que eu gostaria que fossem.					
4. Depois das primeiras sessões tutoriais do Problema 2, senti-me confiante de que eu sabia o que eu devia aprender.					
5. Completar os exercícios do Problema 2 me deu uma sensação gratificante de realização.					
6. Ficou claro para mim como o conteúdo do Problema 2 está relacionado com coisas que eu já sei.					
7. Várias sessões do Problema 2 tinham tanta informação que era difícil escolher e lembrar quais os pontos importantes.					
8. O conteúdo do Problema 2 foi atraente.					
9. Houve exemplos que me mostraram como o conteúdo do Problema 2 pode ser importante para as pessoas que estão aprendendo programação.					
10. Completar com êxito as atividades do Problema 2 foi importante para mim.					
11. O conteúdo do Problema 2 é tão abstrato que foi difícil manter a minha atenção nas sessões tutoriais.					
12. Enquanto eu participava das sessões tutoriais do Problema 2, eu estava confiante de que eu poderia aprender o conteúdo.					
13. Eu gostei tanto do Problema 2 que eu gostaria de saber mais sobre os conteúdos abordados.					
14. O conteúdo do Problema 2 me pareceu pouco atraente.					
15. O conteúdo aprendido no Problema 2 é relevante para os meus interesses.					
16. A maneira como a informação foi organizada no Problema 2 ajudou a manter minha atenção.					
17. Houve explicações e exemplos de como as pessoas usam o conhecimento adquirido no Problema 2.					
18. Os exercícios no Problema 2 foram muito difíceis.					
19. O Problema 2 teve coisas que estimularam minha criatividade.					
20. Eu realmente gostei de estudar programação com o Problema 2.					
21. Às vezes, a quantidade de repetições feitas sobre alguns assuntos me levou a ficar entediado.					
22. O conteúdo e o estilo do Problema 2 dão a impressão que vale a pena saber aqueles conceitos.					
23. Eu aprendi algumas coisas que foram surpreendentes ou inesperadas.					
24a. Após trabalhar no Problema 2 por um tempo, senti-me confiante de que eu seria capaz de passar na avaliação deste problema.					
24b. O conteúdo do Problema 2 não foi relevante para as minhas necessidades, porque eu já sabia a maior parte do assunto.					
25. O feedback e as observações passadas pelo professor ajudaram-me a sentir recompensado pelo meu esforço.					

26. Durante as sessões tutoriais, a variedade de exercícios, ilustrações, etc., ajudou a manter minha atenção.					
27. Eu pude relacionar os conteúdos que aprendi no Problema 2 com coisas que eu já vi, fiz ou pensei, na minha própria vida.					
28. Eu me senti bem ao concluir com êxito o Problema 2.					
29. O conteúdo visto no Problema 2 será útil para mim.					
30. Eu não pude compreender como algumas coisas eram feitas no Problema 2.					
31. A boa organização do conteúdo no Problema 2 ajudou-me a sentir confiança que eu aprenderia o assunto.					
32. Foi um prazer estudar com a metodologia utilizada no Problema 2.					

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

	Muito Difícil	Difícil	Regular	Fácil	Muito Fácil
33. Construtores.					
34. Sobrecarga de métodos.					
35. Composição de objetos.					
36. Herança simples.					
37. Padrão MVC.					
38. Modelo conceitual.					
39. Diagrama de classes de projeto.					
40. Testes de unidade.					
41. Pilhas, filas, filas com prioridades.					
42. Algoritmos de ordenação.					

40. De um modo geral, como você classifica a metodologia das sessões tutoriais do Problema 2 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**

QUESTIONÁRIO – MI – PROGRAMAÇÃO
PROBLEMA 3

Cód:_____

Sobre o Problema 3, marque a opção que mais se adequa ao seu pensamento.	Discordo totalmente	Discordo parcialmente	Neutro	Concordo parcialmente	Concordo totalmente
1. Quando eu vi os assuntos do Problema 3 pela primeira vez, eu tive a impressão de que seriam fáceis para mim.					
2. Havia algo de interessante no início do Problema 3 que chamou a minha atenção.					
3. Os assuntos do Problema 3 foram mais difíceis de entender do que eu gostaria que fossem.					
4. Depois das primeiras sessões tutoriais do Problema 3, senti-me confiante de que eu sabia o que eu devia aprender.					
5. Completar os exercícios do Problema 3 me deu uma sensação gratificante de realização.					
6. Ficou claro para mim como o conteúdo do Problema 3 está relacionado com coisas que eu já sei.					
7. Várias sessões do Problema 3 tinham tanta informação que era difícil escolher e lembrar quais os pontos importantes.					
8. O conteúdo do Problema 3 foi atraente.					
9. Houve exemplos que me mostraram como o conteúdo do Problema 3 pode ser importante para as pessoas que estão aprendendo programação.					
10. Completar com êxito as atividades do Problema 3 foi importante para mim.					
11. O conteúdo do Problema 3 é tão abstrato que foi difícil manter a minha atenção nas sessões tutoriais.					
12. Enquanto eu participava das sessões tutoriais do Problema 3, eu estava confiante de que eu poderia aprender o conteúdo.					
13. Eu gostei tanto do Problema 3 que eu gostaria de saber mais sobre os conteúdos abordados.					
14. O conteúdo do Problema 3 me pareceu pouco atraente.					
15. O conteúdo aprendido no Problema 3 é relevante para os meus interesses.					
16. A maneira como a informação foi organizada no Problema 3 ajudou a manter minha atenção.					
17. Houve explicações e exemplos de como as pessoas usam o conhecimento adquirido no Problema 3.					
18. Os exercícios no Problema 3 foram muito difíceis.					
19. O Problema 3 teve coisas que estimularam minha criatividade.					
20. Eu realmente gostei de estudar programação com o Problema 3.					
21. Às vezes, a quantidade de repetições feitas sobre alguns assuntos me levou a ficar entediado.					
22. O conteúdo e o estilo do Problema 3 dão a impressão que vale a pena saber aqueles conceitos.					
23. Eu aprendi algumas coisas que foram surpreendentes ou inesperadas.					
24a. Após trabalhar no Problema 3 por um tempo, senti-me confiante de que eu seria capaz de passar na avaliação deste problema.					
24b. O conteúdo do Problema 3 não foi relevante para as minhas necessidades, porque eu já sabia a maior parte do assunto.					
25. O feedback e as observações passadas pelo professor ajudaram-me a sentir recompensado pelo meu esforço.					

26. Durante as sessões tutoriais, a variedade de exercícios, ilustrações, etc., ajudou a manter minha atenção.					
27. Eu pude relacionar os conteúdos que aprendi no Problema 3 com coisas que eu já vi, fiz ou pensei, na minha própria vida.					
28. Eu me senti bem ao concluir com êxito o Problema 3.					
29. O conteúdo visto no Problema 3 será útil para mim.					
30. Eu não pude compreender como algumas coisas eram feitas no Problema 3.					
31. A boa organização do conteúdo no Problema 3 ajudou-me a sentir confiança que eu aprenderia o assunto.					
32. Foi um prazer estudar com a metodologia utilizada no Problema 3.					

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

	Muito Difícil	Difícil	Regular	Fácil	Muito Fácil
33. Arquivos de texto.					
34. Arquivos binários e serialização.					
35. Tratamento de exceções.					
36. Padrão Façade.					
37. Padrão MVC.					
38. Testes de unidade.					
39. Testes de aceitação.					
40. Diagrama de classes de projeto.					
41. Árvores binárias: representação e algoritmos.					
42. Balanceamento de árvores binárias.					
43. Documentação com Javadoc.					

43. De um modo geral, como você classifica a metodologia das sessões tutoriais do Problema 3 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**

QUESTIONÁRIO – MI – PROGRAMAÇÃO PROBLEMA 4

Cód:_____

Sobre o Problema 4, marque a opção que mais se adequa ao seu pensamento.	Discordo totalmente	Discordo parcialmente	Neutro	Concordo parcialmente	Concordo totalmente
1. Quando eu vi os assuntos do Problema 4 pela primeira vez, eu tive a impressão de que seriam fáceis para mim.					
2. Havia algo de interessante no início do Problema 4 que chamou a minha atenção.					
3. Os assuntos do Problema 4 foram mais difíceis de entender do que eu gostaria que fossem.					
4. Depois das primeiras sessões tutoriais do Problema 4, senti-me confiante de que eu sabia o que eu devia aprender.					
5. Completar os exercícios do Problema 4 me deu uma sensação gratificante de realização.					
6. Ficou claro para mim como o conteúdo do Problema 4 está relacionado com coisas que eu já sei.					
7. Várias sessões do Problema 4 tinham tanta informação que era difícil escolher e lembrar quais os pontos importantes.					
8. O conteúdo do Problema 4 foi atraente.					
9. Houve exemplos que me mostraram como o conteúdo do Problema 4 pode ser importante para as pessoas que estão aprendendo programação.					
10. Completar com êxito as atividades do Problema 4 foi importante para mim.					
11. O conteúdo do Problema 4 é tão abstrato que foi difícil manter a minha atenção nas sessões tutoriais.					
12. Enquanto eu participava das sessões tutoriais do Problema 4, eu estava confiante de que eu poderia aprender o conteúdo.					
13. Eu gostei tanto do Problema 4 que eu gostaria de saber mais sobre os conteúdos abordados.					
14. O conteúdo do Problema 4 me pareceu pouco atraente.					
15. O conteúdo aprendido no Problema 4 é relevante para os meus interesses.					
16. A maneira como a informação foi organizada no Problema 4 ajudou a manter minha atenção.					
17. Houve explicações e exemplos de como as pessoas usam o conhecimento adquirido no Problema 4.					
18. Os exercícios no Problema 4 foram muito difíceis.					
19. O Problema 4 teve coisas que estimularam minha criatividade.					
20. Eu realmente gostei de estudar programação com o Problema 4.					
21. Às vezes, a quantidade de repetições feitas sobre alguns assuntos me levou a ficar entediado.					
22. O conteúdo e o estilo do Problema 4 dão a impressão que vale a pena saber aqueles conceitos.					
23. Eu aprendi algumas coisas que foram surpreendentes ou inesperadas.					
24a. Após trabalhar no Problema 4 por um tempo, senti-me confiante de que eu seria capaz de passar na avaliação deste problema.					
24b. O conteúdo do Problema 4 não foi relevante para as minhas necessidades, porque eu já sabia a maior parte do assunto.					
25. O feedback e as observações passadas pelo professor ajudaram-me a sentir recompensado pelo meu esforço.					

26. Durante as sessões tutoriais, a variedade de exercícios, ilustrações, etc., ajudou a manter minha atenção.					
27. Eu pude relacionar os conteúdos que aprendi no Problema 4 com coisas que eu já vi, fiz ou pensei, na minha própria vida.					
28. Eu me senti bem ao concluir com êxito o Problema 4.					
29. O conteúdo visto no Problema 4 será útil para mim.					
30. Eu não pude compreender como algumas coisas eram feitas no Problema 4.					
31. A boa organização do conteúdo no Problema 4 ajudou-me a sentir confiança que eu aprenderia o assunto.					
32. Foi um prazer estudar com a metodologia utilizada no Problema 4.					

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

	Muito Difícil	Difícil	Regular	Fácil	Muito Fácil
33. Componentes da interface gráfica de usuário em Java.					
34. Tratamento de eventos de interface em Java.					
35. Uso de coleções da biblioteca padrão de Java.					
36. Classes abstratas e herança polimórfica.					
37. Padrão MVC.					
38. Testes de unidade.					
39. Testes de aceitação.					
40. Diagrama de classes de projeto.					
41. Tabelas hash: representação e algoritmos.					
42. Grafos: representação e percurso.					
43. Algoritmo de caminho mínimo para grafos.					

43. De um modo geral, como você classifica a metodologia das sessões tutoriais do Problema 4 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 CIS – Course Interest Survey

Course Interest Survey – Nome da Disciplina 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 E

Guia de Entrevista do Professor do Módulo Integrador – Final do Período

Objetivos e Questões de Pesquisa

O objetivo geral deste trabalho é avaliar uma abordagem de ensino-aprendizagem de programação orientada a objetos no curso de Engenharia de Computação da Universidade Estadual de Feira de Santana, num componente curricular de Programação do segundo semestre do curso.

A partir deste objetivo, pretende-se responder às seguintes questões de pesquisa:

1. Como a abordagem utilizada influencia a motivação dos estudantes?
2. Quais os resultados proporcionados pela abordagem em termos de aprendizado de conhecimentos e habilidades de programação orientada a objetos?
3. Como o ciclo de aprendizagem PBL influencia na qualidade do estudo dos estudantes?

Guia de Entrevista

1. Há quanto tempo você ensina o MI de Programação?
2. Você gosta de ensinar no MI? Por quê?
3. De que maneira você acha que o método PBL afetou a motivação dos estudantes no MI de Programação?
4. Para cada problema, perguntar: De que maneira o problema afetou a motivação dos estudantes?
    a. Probes: Você pode comparar os problemas em relação a como afetaram a motivação dos estudantes? Por uma análise preliminar dos dados quantitativos, parece que o problema X motivou mais os estudantes. Você concorda? Por quê? O mesmo para o problema que motivou menos.
5. De que maneira as avaliações do MI afetaram a motivação dos estudantes
6. Que assuntos você acha que estudantes aprenderam mais no MI de programação? E os que você acha que eles aprenderam menos?
7. O que você acha que os alunos que fizeram o MI de Programação sabem fazer depois de terem terminado o módulo? E o que você acha que eles deveriam saber fazer após o MI, mas que ainda não se sentem capazes?
8. No MI, quais as principais dificuldades que os alunos tiveram? E as facilidades (ou pontos positivos)?
9. Sobre o método PBL, como você acha que ele afeta o estudo individual dos alunos? E o estudo em grupo deles?
10. Comparando o método PBL com o tradicional, quais aspectos, em sua opinião, são melhores ou piores? Por quê?

Apêndice F

Guia de Entrevista do Professor do Módulo Teórico – Final do Período

Objetivos e Questões de Pesquisa

O objetivo geral deste trabalho é avaliar uma abordagem de ensino-aprendizagem de programação orientada a objetos no curso de Engenharia de Computação da Universidade Estadual de Feira de Santana, num componente curricular de Programação do segundo semestre do curso.

A partir deste objetivo, pretende-se responder às seguintes questões de pesquisa:

1. Como a abordagem utilizada influencia a motivação dos estudantes?
2. Quais os resultados proporcionados pela abordagem em termos de aprendizado de conhecimentos e habilidades de programação orientada a objetos?
3. Como o ciclo de aprendizagem PBL influencia na qualidade do estudo dos estudantes?

Guia de Entrevista

1. Há quanto tempo você ensina este módulo teórico?
2. Você gosta de ensinar este módulo teórico? Por quê?
3. De que maneira você acha que a sua abordagem de ensino afetou a motivação dos estudantes?
4. De que maneira você acha que os materiais usados nas aulas afetou a motivação dos estudantes?
5. De que maneira as suas avaliações afetaram a motivação dos estudantes?
6. Que assuntos você acha que estudantes aprenderam mais neste módulo teórico? E os que você acha que eles aprenderam menos?
7. O que você acha que os alunos que fizeram este módulo teórico sabem fazer depois de terem terminado o módulo? E o que você acha que eles deveriam saber fazer após este módulo teórico, mas que ainda não se sentem capazes?
8. Neste módulo teórico, quais as principais dificuldades que os alunos tiveram? E as facilidades (ou pontos positivos)?
9. Qual a sua percepção da relação entre o módulo teórico e o MI? De que maneira você acha que o módulo teórico auxilia no aprendizado no MI? E no sentido oposto, de que maneira você acha que o MI auxilia no aprendizado do módulo teórico?

Apêndice G

Guia de Entrevista do Estudante – Final do período

Objetivos e Questões de Pesquisa

O objetivo geral deste trabalho é avaliar uma abordagem de ensino-aprendizagem de programação orientada a objetos no curso de Engenharia de Computação da Universidade Estadual de Feira de Santana, num componente curricular de Programação do segundo semestre do curso.

A partir deste objetivo, pretende-se responder às seguintes questões de pesquisa:

1. Como a abordagem utilizada influencia a motivação dos estudantes?
2. Quais os resultados proporcionados pela abordagem em termos de aprendizado de conhecimentos e habilidades de programação orientada a objetos?
3. Como o ciclo de aprendizagem PBL influencia na qualidade do estudo dos estudantes?

Guia de Entrevista

1. Em 2017.2, foi a sua primeira vez a cursar o MI de Programação? E os módulos teóricos de Algoritmos e Programação II, Estruturas de Dados e Projeto de Sistemas?
2. No final de 2017.2, você foi aprovado no MI de Programação? A que você atribui este resultado?
3. E nos módulos teóricos? A que você atribui este resultado?
4. De que maneira o método PBL afetou a sua motivação no MI de Programação?
5. Para cada problema, perguntar: De que maneira o problema afetou a sua motivação?
    a. Probes: Você pode comparar os problemas em relação a como afetaram a sua motivação? Por uma análise preliminar dos dados quantitativos, parece que o problema X motivou mais os estudantes. Você concorda? Por quê? O mesmo para o problema que motivou menos.
6. Para cada módulo teórico, perguntar: De que maneira o módulo teórico afetou a sua motivação?
    a. Probes: sobre o professor, sobre as aulas, sobre as avaliações.
7. Que assuntos você acha que aprendeu no MI de programação? E os que você acha que não aprendeu?
8. O que você acha que sabe fazer depois de ter feito o MI de Programação? E o que você acha que deveria saber fazer após o MI, mas não se sente ainda capaz de fazer?
9. Para cada módulo teórico, perguntar: Que assuntos você acha que aprendeu? E os que você acha que não aprendeu?
10. Para cada módulo teórico, perguntar: O que você acha que sabe fazer depois de ter feito o módulo teórico? E o que você acha que deveria saber fazer após o módulo teórico, mas não se sente ainda capaz de fazer?
11. No MI, quais as principais dificuldades que você teve? E as facilidades (ou pontos positivos)?
12. Para cada módulo teórico, perguntar: Quais as principais dificuldades que você teve no módulo teórico? E as facilidades (ou pontos positivos)?
13. Sobre o método PBL, como você acha que ele afeta o seu estudo individual? E o estudo em grupo com seus colegas?
14. Comparando o método PBL com o tradicional, quais aspectos são melhores ou piores? Por quê?

Apêndice H

PROBLEMA 1

Problema 1: BIOCAD – **Sistema para o Recadastramento Biométrico**

O **Recadastramento Biométrico** é o processo de digitalização das informações biométricas do eleitor. No recadastramento, são colhidas eletronicamente a assinatura, a foto (com medidas da face) e as impressões digitais do cidadão. Ele é realizado para dar mais segurança à identificação do eleitor no momento da votação. Estão obrigados ao cadastramento todos os eleitores convocados ou não pela Zona Eleitoral que esteja executando esse procedimento, inclusive aqueles cujo voto é facultativo e já possuem título (analfabetos, maiores de 16 e menores de 18 anos e os maiores de 70 anos de idade). A principal vantagem do sistema biométrico é a segurança do voto, além da atualização do cadastro. Com a identificação biométrica não haverá a possibilidade de um eleitor votar no lugar de outro, tornando ainda mais seguro o sistema de votação eletrônico (http://www.tre-ba.jus.br).

Desde fevereiro deste ano, quando abriu o prazo oficial para a realização do recadastramento biométrico em mais de 50 municípios baianos, o Tribunal Regional Eleitoral da Bahia (TRE-BA), vem buscando soluções para aproximar a Justiça Eleitoral dos cidadãos e ampliar a divulgação quanto aos benefícios do procedimento. Várias parcerias têm sido realizadas para agilizar o atendimento ao público. Hoje existem 5 guichês para atender os cidadãos que agendaram o cadastramento e 4 guichês para atendimento sem agendamento por ordem de chegada, com atendimento diário(exceto aos sábados e domingos), das 8 h às 17 h (1 hora para almoço)

Para informatizar o atendimento, os estudantes do segundo semestre do curso de Engenharia da Computação da UEFS foram convocados pelo TRE-BA para desenvolver um software. Um analista de sistemas enviou alguns requisitos necessários para este desenvolvimento. Ele criou rapidamente as *User Stories* e o modelo conceitual e entregou para os estudantes. Disse que estava muito atarefado com os equipamentos da biometria e que eles aprenderiam rapidamente o processo de agendamento. Você fica muito confuso com tantas informações e resolve ir pessoalmente ao TRE coletar requisitos necessários para o funcionamento do sistema. Você percebe que existem dois tipos de atendimento: um agendado, com horário marcado a cada 15 minutos em um dos 5 guichês e outro atendimento por ordem de chegada, também de 15 em 15 minutos. Os agendamentos são realizados todo dia 20 de cada mês, preenchendo as vagas do dia 1 até o ultimo dia do mês seguinte. Os funcionários trabalham 8 horas por dia (32 atendimentos realizados por dia e por guichê).

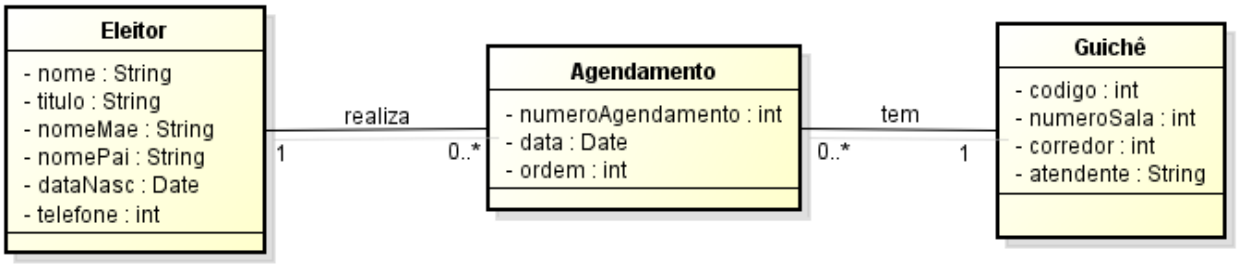
Ao chegar, você percebe que o secretário do TRE adiciona os guichês disponíveis através de um código, o número da sala que acontecerá o atendimento, corredor em que se encontra e nome do atendente. Ele também cadastra os dados do eleitor: nome, número do título, telefone para contato, data de nascimento, nome do pai, nome da mãe e telefone para contato. O agendamento do atendimento é realizado informando o código do guichê, o nome do eleitor, data e hora. O software precisa verificar se o guichê e o eleitor estão cadastrados e realizar o controle de horário.

Você retorna para a UEFS todo perdido, porém muito empolgado para entender o que são as tais das *User Stories* e o modelo conceitual. João, analista de sistemas do TRE, liga e te deixa ainda mais preocupado. “Cara...você ainda não começou? Temos menos de um mês para o desenvolvimento... Faz o seguinte, estuda com calma o modelo e as *User Stories*. Vou deixar no site do projeto um diretório src que vai adiantar bastante a tua vida. Lá tem também uma classe Console para te ajudar com a interface.”

Sem muita paciência para lhe explicar muito mais, ele diz que tem “outras coisas para resolver. Tem livros de Java e de programação orientada a objetos na biblioteca, você aprende rápido”. Não lhe restando muito mais a fazer, você resolver encarar o problema com toda a vontade. Afinal, “Depois deste projeto, eu vou ser um programador profissional.”, você pensa.

User Story	Título	Breve Descrição
1	Adicionar Guichê de Atendimento	Recepção adiciona novo guichê de atendimento, cadastrando número da sala, corredor e atendente. Um código é gerado para o guichê automaticamente.
2	Listar guichês	Recepção lista os guichês cadastrados no TRE.
3	Buscar guichês	Recepção busca o guichê pelo código.
4	Adicionar eleitor	Recepção cadastra eleitor no software, com nome, título, data de nascimento e telefone para contato, nome do pai e nome da mãe.
5	Verificar guichê disponível	O sistema deve verificar se a data informada e hora para determinado guichê está livre.
6	Agendar horário para eleitor	Recepção informa o número do guichê, a data, ordem de atendimento (horário1, 2, 3 ...32), título do eleitor.
7	Listar eleitores agendados no dia por guichê	Recepção lista eleitores agendados de um guichê de um determinado dia.

Modelo Conceitual:



/Produto:

Você deve enviar um e-mail com o produto final para o seu tutor até às 12 horas (meio-dia) do dia 06/10/2017 (sexta-feira), anexando o arquivo compactado com o código fonte e o diagrama de classes do projeto. Você deve entregar o código fonte em um só arquivo compactado, com todas as classes que você desenvolveu para este projeto, junto com os demais arquivos que você recebeu (estão todos no arquivo Biocad.zip, que está localizado na página do tutorial, em http://sites.ecomp.uefs.br/mip-20172/). Todas as classes devem estar compilando e implementando as funcionalidades adequadamente. O software deverá estar funcionando de acordo com o problema e as *User Stories*.

Avaliação:

O produto entregue corresponde a 60% da nota e o desempenho nos tutoriais corresponde a 40% da nota.

Cronograma:

Aula	Data	Atividade
1	19/09	Apresentação do Problema 1
2	22/09	Sessão Tutorial
3	26/09	Sessão Tutorial
4	29/09	Sessão Tutorial
5	03/10	Sessão Tutorial
6	06/10	Entrega do Produto/ Problema 2

Apêndice I

PROBLEMA 2

Problema 2: AuctionTool – Ferramenta para leilão eletrônico.

Assim como no leilão convencional, o leilão eletrônico consiste numa modalidade de venda por concorrência, cujo o vendedor define um valor mínimo para os itens que serão vendidos e a partir daí os prováveis compradores oferecem um valor superior ao preço inicial definido. Adquire o item, o comprador que oferecer o maior valor. Porém, todas as transações acontecem através de softwares. A famosa empresa de vendas na internet LOX resolveu incluir essa modalidade em seu negócio. Para tanto, ela solicitou aos alunos de MI-Programação, da UEFS, o desenvolvimento de um software responsável por gerenciar seus leilões eletrônicos. O programa deve ser capaz de inserir, alterar e excluir categorias de itens. As categorias devem possuir um código, gerado automaticamente pelo sistema, um nome que represente a categoria e a data quando a categoria foi criada. Uma categoria pode possuir um conjunto de itens. Uma categoria não pode ser excluída, caso possua itens associados à ela. O sistema também deve permitir o gerenciamento (inclusão, alteração e exclusão) dos itens que serão leiloados. Cada item deve ter um código, gerado automaticamente pelo sistema, um nome, uma categoria, um preço de lance inicial, uma pessoa (atual dono do item), o STATUS do item (CADASTRADO, EM LEILAO, VENDIDO, DESISTENCIA) e a data que o item foi cadastrado. Três tipos especiais de itens podem ser cadastrados no sistema: imóveis, remédios e alimentos. No primeiro tipo deve ser registrado também a informação se o imóvel está ocupado ou não. No segundo, deve ser registrada a data de validade do medicamento, e a informação se o medicamento precisa ou não de receita. No terceiro tipo também deve ser registrada a data de validade do alimento. Só podem ser feitas alterações nos dados dos itens que estejam com STATUS igual a CADASTRADO. Um item também só pode ser excluído, caso ele esteja com STATUS igual a CADASTRADO. Os campos STATUS e data de cadastrado são configurados automaticamente pelo sistema. Deve existir uma funcionalidade que permita a mudança do STATUS de um item de CADASTRADO para EM LEILAO e uma outra funcionalidade para alterar o STATUS de EM LEILAO para VENDIDO. Todas as informações dos itens são obrigatórias. Não é possível colocar um item com STATUS de VENDIDO, sem que haja pelo menos um lance para o item. O sistema precisa registrar todos os lances ocorridos para um determinado item. Para registrar o lance, é preciso guardar a data e hora que o lance aconteceu, a pessoa que efetuou o lance e o valor do lance. Não existe limite de lances para um item por uma determinada pessoa. O sistema precisa registrar todos os lances de todas as pessoas. Não deve ser permitido que uma pessoa efetue um lance com um valor inferior ao preço mínimo do item. Também não deve ser permitido que uma pessoa dê um lance em um item com data de validade vencida. Só pode ser efetuado um lance para um item que esteja com STATUS igual a EM LEILAO. Cada lance dado deve ser incluído em uma fila dentro do item ao qual o lance se refere, tendo prioridade o maior lance. Isso significa dizer que cada item deve possuir uma única fila com seus respectivos lances. O sistema também deve ser capaz de gerenciar (incluir, alterar e remover) os dados de uma pessoa (cpf, nome, e-mail, telefone e pontuacao). Todos os campos são obrigatórios. Só podem ser removidas pessoas que não tenham efetuado nenhum lance e que não seja dona de nenhum item. A pontuação corresponde ao somatório de vezes que uma pessoa conseguiu vencer um leilão, mais a quantidade de vezes que essa mesma pessoa conseguiu vender um item em um leilão. É preciso disponibilizar uma funcionalidade para alterar o STATUS do item de VENDIDO para DESISTENCIA. Quando isso acontecer, o lance que havia sido considerado vencedor deve ser invalidado. Nesse caso, o segundo lance mais alto, caso exista, será declarado vencedor. O sistema deve ser capaz de exibir os dados do vencedor de um leilão a partir do código do item. O sistema também deve ser capaz de listar todos os itens de uma determinada categoria, todas as pessoas cadastradas no sistema ordenadas de maneira eficiente (O(n log n)) por suas pontuações em ordem decrescente e todos os lances para um determinado item. Foi solicitado que os alunos façam o código passar nos testes de unidade, fornecidos no arquivo AuctionTool.zip, e também façam testes de unidade adicionais de modo a contemplar todas as user stories, e entreguem o diagrama de classes que represente o sistema desenvolvido. O padrão de desenvolvimento adotado pelos desenvolvedores precisa estar adequado ao padrão MVC. Para facilitar o desenvolvimento do sistema, a empresa entregou para os desenvolvedores uma lista das funcionalidades que precisam ser implementadas e o modelo conceitual do sistema como segue abaixo:

User Stories:

User Story	Título	Descrição
1	Inserir Categoria	Registrar uma categoria no sistema.
2	Alterar Categoria	Alterar os dados de uma categoria no sistema.
3	Remover Categoria	Remover uma categoria que não tenha produtos associados do sistema.
4	Inserir Item	Inserir um item para leilão.
5	Alterar Item	Alterar os dados de um item do leilão, cujo o leilão ainda não tenha sido aberto.
6	Remover Item	Remover um item, cujo o leilão ainda não tenha sido aberto.
7	Inserir Pessoa	Inserir dados de uma pessoa no sistema.
8	Alterar Pessoa	Alterar os dados de uma pessoa no sistema.

9	Remover Pessoa	Remover uma pessoa que ainda não tenha dado lance ou que não seja dona de nenhum item do sistema.
10	Dar Lance	Permitir que uma pessoa consiga efetuar um lance em um item que esteja com STATUS EM LEILAO.
11	Colocar Item em Leilão	Disponibilizar um item já cadastrado para leilão.
12	Encerrar Leilão	Encerrar o leilão de um item que já tenha recebido pelo menos um lance.
13	Desistir do Item	Permitir que uma pessoa que tenha ganhado o lance desista do item.
14	Exibir Vencedor	Exibir os dados do vencedor de um item, cujo leilão tenha sido encerrado, a partir do código do item.
15	Listar Itens por Categoria	Listar todos os dados de todos os itens de uma determinada categoria.
16	Listar Pessoas	Listar todos os dados de todas as pessoas cadastradas no sistema ordenadas em ordem decrescente pela pontução da pessoa.
17	Listar Lances	Listar todos dados dos lances de um determinado item a partir do seu código.

Produto:

Você deve enviar um e-mail com o produto final para o seu tutor até às 12 horas (meio-dia) do dia 27/10/2017 (sexta-feira), anexando o arquivo compactado com o código fonte, o diagrama de classes do projeto e o relatório (usar o modelo de artigo da SBC para o relatório). Você deve entregar o código-fonte em um só arquivo compactado, com todas as classes que você desenvolveu para este projeto, junto com os demais arquivos que você recebeu (estão todos no arquivo auctionTool.zip, que está localizado na página do tutorial, em http://sites.ecomp.uefs.br/mip-20172/. Todas as classes devem estar compilando e implementando as funcionalidades adequadamente.

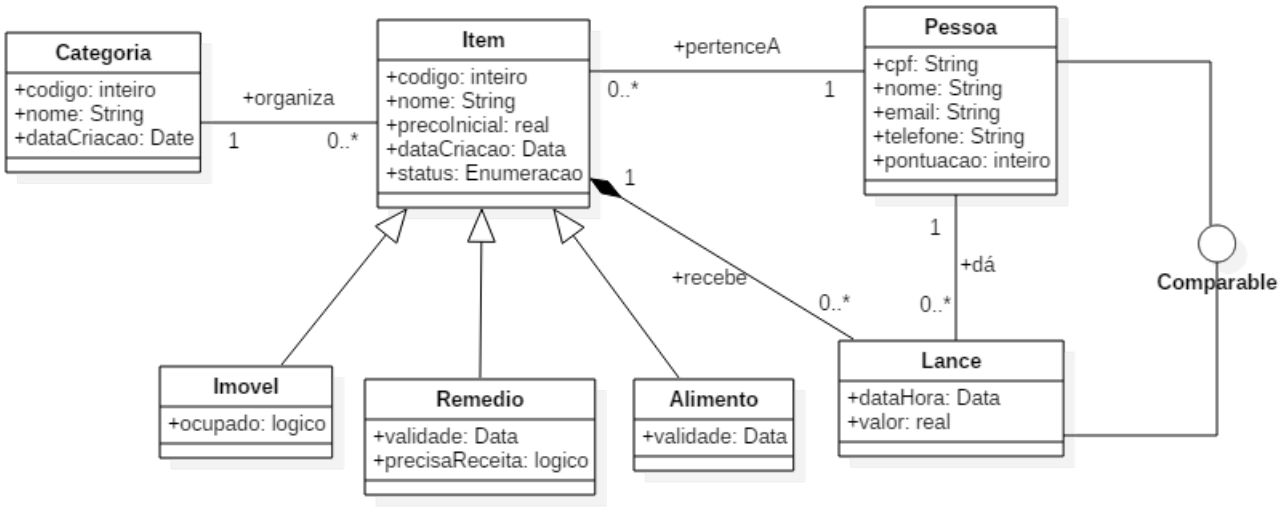
Avaliação:

O produto entregue (relatório, diagrama e código) corresponde a 70% da nota da unidade, e o desempenho nos tutoriais corresponde a 30% da nota.

Cronograma

Aula	Data	Atividade
1	06/10	Apresentação do Problema 2
2	10/10	Sessão Tutorial
-	13/10	Ponto Facultativo. Não haverá aula.
3	17/10	Sessão Tutorial
4	20/10	Sessão Tutorial
5	24/10	Sessão Tutorial
6	27/10	Sessão Tutorial
7	31/10	Sessão Tutorial
-	03/11	Ponto Facultativo. Não haverá aula.
8	07/11	Sessão Tutorial
-	10/11	Entrega do Problema 2

Modelo Conceitual:



Apêndice J

PROBLEMA 3

Problema 3 – TreeStock – Gerenciador de carteira de ações

Descrição

Uma corretora do mercado de ações chamada **TreeStock** necessita de um *software* que auxilie nas suas atividades administrativas. A corretora trabalha no mercado acionário e gerencia a carteira de ações dos seus clientes.

O mercado de ações é bem volátil e, por este motivo, é importante o acesso rápido às informações do sistema. A corretora **TreeStock** precisa agir de forma rápida no gerenciamento das ações de seus clientes.

Cada cliente possui uma quantidade variável de lotes (cada lote corresponde a cem ações), pertencentes a diferentes empresas do mercado de ações da bolsa de valores de São Paulo e uma conta na corretora, onde estão registrados os valores equivalentes das ações e o restante em dinheiro.

O agrupamento das ações de diferentes empresas, juntamente com as respectivas quantidades que cada cliente pode possuir, chama-se “carteira”. O valor das carteiras podem variar dependendo da cotação das ações durante o dia. É importante que a empresa possa monitorar o montante da conta (soma do valor da carteira e o dinheiro) dos seus clientes de maneira rápida.

Existem basicamente dois tipos de ações, ordinárias (ON) e preferenciais (PN). As ordinárias dão direito a voto nas assembleias da empresa a as preferenciais concedem o direito ao acionista de obter dividendos trimestrais de acordo com o número de ações PN que ele tiver. Quando o cliente recebe os dividendos, este valor é somado ao saldo na conta.

Diante deste contexto, sua equipe foi contratada para modelar e construir um modelo de um sistema que satisfaça às necessidades da empresa em um prazo de 25 dias. O sistema deve ser implementado em linguagem de programação Java.

Uma estrutura de dados deve ser criada para que as ações possam ser contabilizadas para cada cliente. Os clientes devem ser ordenados pelo seu CPF a fim de facilitar a busca. A empresa também deseja saber a listagem de um determinado número de clientes, ordenados pelas maiores contas, para tratá-los de maneira VIP (*Very Important People*). A situação do sistema poderá ser salva a qualquer momento ou carregada em arquivo. As cotações das ações e dividendos também serão atualizadas através de um arquivo elaborado especificamente para isso. As contas serão atualizadas de acordo com a variação das carteiras pelas cotações das ações e pela inclusão de dividendos de acordo com a quantidade de ações PN que o acionista tiver.

Produto

Na estrutura de dados será possível realizar as ações: busca, inclusão, remoção, alteração das quantidades das ações e dividendos dos clientes, assim como busca, inserção e remoção de clientes.

A lista das ações que poderão ser negociadas, suas respectivas cotações e inclusão de dividendos serão atualizadas através do carregamento de arquivos para o gerenciamento de todas as carteiras. É prioritário que se tratando das buscas, inserção ou remoção dos clientes e ordenação das contas a complexidade média seja de O(log n) (onde n é o número de elementos da estrutura).

A corretora também deseja atender exclusivamente um seleto número de clientes com maior valor em conta. Para isso deve existir a possibilidade de se escolher um determinado número TOP-K de clientes com maiores valores de conta, ordenados pelos maiores valores.

Diferente dos sistemas anteriores, sua equipe é quem deverá implementar todos os testes de unidade para as classes que utilizar, com exceção da *facade*. Os métodos também deverão suportar tratamento de exceções na *view*. A interface *Collection* do Java poderá ser utilizada, com exceção da estrutura utilizada para ordenação.

Você deve entregar o código-fonte em um só arquivo compactado, com todos os arquivos do projeto e as classes desenvolvidas para este projeto.

Todas as classes devem estar compilando e implementando as funcionalidades adequadamente. Todas as classes e os testes devem estar documentados utilizando o padrão *javadoc*.

Um arquivo de projeto em NetBeans contendo os testes de aceitação será disponibilizado no site da disciplina e um modelo de arquivo de carregamento das cotações e dividendos. Você deve fazer todas as implementações do sistema dentro deste projeto.

O software terá interface por linha de comando. A correção será realizada analisando o funcionamento, o código produzido, a execução dos testes de unidade, testes de aceitação e a documentação *javadoc*.

Modelo do arquivo de cotações e dividendos

1ª linha: Cabeçalho contendo a data das cotações

2ª linha em diante: Nome da ação, tipo da ação, valor da cotação de fechamento no dia, valor dos dividendo por ação (quando houver)

Obs.: Para o caso das cotações e dividendos, os dois últimos algarismos são duas casas decimais.

Avaliação

A solução do problema e o diagrama de classes são individuais.

Calendário

#	Data	Atividade
1	10/11	Apresentação do Problema 3 / Sessão 1
2	14/11	Sessão 2
3	17/11	Sessão 3
4	21/11	Sessão 4
5	24/11	Sessão 5
6	28/11	Sessão 6
7	01/12	Sessão 7
-	05/12	Entrega do Problema 3 (até às 12:00)

User Stories

Nº	Título	Descrição
1	Manutenção de clientes.	Faz a inclusão, remoção e busca de clientes.
2	Manutenção das ações na carteira do cliente.	Faz a inclusão, alteração e exclusão das ações dos clientes e suas quantidades.
3	Inserir ação no software	Insere uma determinada ação de uma empresa e sua cotação.
4	Carregar arquivo das cotações e dividendos	Atualiza os valores das ações, carteiras e conta.
5	Exclui uma ação do software	Exclui uma determinada ação caso não pertença a nenhuma carteira
6	Listar TOP-K de contas	Mostra um determinado número de contas dos clientes por ordem dos valores maiores para os menores
7	Ordenar por clientes	Ordena os clientes pelo CPF e informa o valor de sua conta
8	Listar a carteira de um cliente	Lista as ações/quantidades da carteira de um determinado cliente e o valor correspondente em dinheiro
9	Salvar arquivo do sistema	Salva todas as informações dos dados do sistema
10	Carregar o arquivo do sistema	Recupera todos os dados salvos da última sessão do sistema

Apêndice K

PROBLEMA 4

Problema 4: MINHA APP *ROAD TRIPS*

Tema

Gerenciamento de viagens em uma app pessoal.

Problema

Após alguns estágios, você decide partir para um negócio só seu. Você sempre teve vontade de abrir um negócio e acha que a área de computação é ideal para isso. Inspirado pelo Google Trips (https://get.google.com/trips/) que está fazendo o maior sucesso entre os viajantes no mundo, lembrando todas as viagens de carro que você já fez na vida e sendo um fã das quatro rodas, você decide fazer a sua própria app de viagens rodoviárias.

Para levantar os requisitos que serão desenvolvidos, você instala e abre o Google Trips e percebe que ele mantém diversas informações (dados) sobre cada cidade como nome, latitude, longitude, área, população, descrição da cidade, fotos, coisas a fazer, locais para comer e locais para dormir. Você pensa: "Existem dois tipos de informação: os dados essenciais da cidade e os dados associados à cidade." Acho que eu posso começar com os dados essenciais, adicionando os associados à medida que eu vá evoluindo minha solução.

Depois, você lembra do Google Maps e de que cada cidade tem várias rotas para permitir transitar entre ela e outras cidades. E pior, entre uma cidade e outra existem vários caminhos possíveis, estradas que bifurcam, junções que ligam duas ou mais rodovias. Para simplificar tudo, você resolve criar um Ponto como entidade principal, estendendo Ponto para duas outras classes: Intersecao e Cidade. Uma interseção seria um ponto que conecta duas ou mais estradas. Uma cidade também pode conectar duas ou mais cidades e, além disso, possui todos aquelas características adicionais que interessam aos viajantes: locais para comer, locais para dormir, coisas a fazer. Para não complicar sua vida, você resolve trabalhar apenas com locais para comer, deixando o resto para depois.

Você pensa um pouco mais. Ponto é uma coisa abstrata, que não existe de verdade. Já interseção e cidade é algo que existe pra valer. Não faz sentido pedir a um ponto a distância para um ponto vizinho, mas isso faz sentido para Intersecao e para Cidade, é algo que pode existir abstratamente para Ponto, mas para fazer funcionar, só mesmo em Intersecao ou Cidade. Do mesmo modo, há algumas coisas que só faz sentido perguntar sobre cidades (população, por exemplo), enquanto que Intersecao pode ter atributos adicionais sobre o tipo de interseção (rótula, cruzamento, semáforo) que não fazem sentido para uma cidade. Sua mente dá um nó e você resolve ler mais sobre abstração em POO.

Em seguida, você se inspira mais uma vez em um outro projeto, o EComp Maps, projeto aberto de mapas que pretende mapear todas as ruas e estradas em todas as cidades do mundo. As ideias de lá lhe ajudam a criar a sua app antes de você resolver pagar ao Google para usar os mapas deles. Claro que trabalhar com um número mais reduzido de cidades (talvez umas 15) e trechos de estradas (talvez uns 100) já dá para começar a testar seu protótipo. Reparando nos dados, você vê que há dois arquivos de texto: um com uma linha por cidade/interseção com o código do local, latitude e longitude, e outro com uma linha por trecho de estrada, com o código do local de origem, o código do local de destino e a distância em quilômetros entre origem e destino.

Uma coisa que você reparou é que o administrador do sistema é quem vai cadastrar os dados de cada cidade, necessitando de uma interface gráfica para isso, além de carregar os dados dos mapas e persistir os dados de todo o sistema. E os usuários, por sua vez, vão apenas criar as suas viagens, fazendo um roteiro bem simples como no Google Trips: escolhem a primeira cidade, data de chegada e data de

partida, adicionam a próxima cidade e as datas de chegada e partida para esta cidade e assim sucessivamente até fechar a viagem com a última cidade de destino. Um login para os usuários seria uma boa saída, incluindo um *hash* da senha, para não guardar a senha aberta nos arquivos do sistema.

Você pensa que seria muito bacana que a app oferecesse o caminho mais curto entre cada duas cidades do roteiro. Pesquisa um pouco e descobre que alguém já pensou nisso, ainda no século XX, inventando uns algoritmos maneiros para grafos. (Falando em grafos, você acabou de ler um artigo numa rede social mostrando que uma solução extremamente eficiente para representar grafos é usando tabelas hash para encontrar os elementos adjacentes.)

Após as pesquisas no Google Trips para entender melhor os requisitos, você decide criar algumas *user stories* e um modelo conceitual para auxiliar no desenvolvimento da sua app. Além disso, antes de iniciar o desenvolvimento da interface gráfica, você resolve criar a fachada do sistema e preparar os testes de aceitação, para garantir que dê tudo certo ao colocar a interface gráfica acessando os controladores do sistema.

Como sua app não precisa estar limitada à web, você decide fazer o primeiro protótipo de interface gráfica usando uma das bibliotecas de Java para construção de interfaces gráficas. Você lembra das várias opções (AWT, Swing, SWT) e decide qual vai ficar mais bonita e independente da plataforma.

Produto

Você deve entregar, individualmente ou em dupla, o código-fonte em um só arquivo compactado, com todas as classes que você desenvolveu para este projeto. Você também deve implementar documentação em Javadoc, testes de unidade para todas as classes que você criar (com exceção das classes da view) e os testes de aceitação do projeto. É uma boa ideia rodar os seus testes antes de entregar o seu projeto.

Além disso, você deve escrever um relatório técnico **individual** (entre cinco e sete páginas), no formato de artigo da SBC (no site do curso http://sites.ecomp.uefs.br/mip/home/tutorial você pode encontrar os *templates* do relatório) contendo uma introdução com a apresentação dos problemas e dificuldades que você enfrentou, a descrição detalhada da sua solução, incluindo um **diagrama de classes de projeto do sistema**. No relatório, você ainda deve detalhar as decisões que você teve de tomar (fundamentando-as, sempre que possível), apontando os pontos positivos e negativos deste trabalho, e encerrando com as suas conclusões. Lembrando que todas as fontes de pesquisa utilizadas devem estar citadas no relatório, sendo desnecessária sua reprodução.

Você deve enviar um e-mail para o seu tutor até às 12 horas da tarde do dia 08/02/2018 (quinta-feira), anexando o arquivo compactado com o código-fonte, javadoc e os testes. Além disso, você deve enviar o seu relatório individual do problema até às 12 horas da tarde do dia 09/02/2018 (em formato *pdf*).

Cronograma

Aula	Data	Atividade
1	08/12	Apresentação do Problema 4 / Sessão Tutorial
2	12/12	Sessão Tutorial
3	15/12	Sessão Tutorial
4	19/12	Sessão Tutorial
5	22/12	Sessão Tutorial
6	02/02	Sessão Tutorial
7	06/02	Sessão Tutorial
	08/02	Entrega do Produto individual ou em dupla (até meio-dia)
	09/02	Entrega do Relatório individual (até meio-dia)