UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO APLICADA

CLÁUDIO MARCIO FELISBINO

FERRAMENTA PARA O APOIO ENSINO-APRENDIZAGEM DO MODELO ORIENTADO A OBJETOS DURANTE A CONSTRUÇÃO DO DIAGRAMA DE CLASSES

DISSERTAÇÃO DE MESTRADO

CURITIBA
2017

CLÁUDIO MARCIO FELISBINO

FERRAMENTA PARA O APOIO ENSINO-APRENDIZAGEM DO MODELO ORIENTADO A OBJETOS DURANTE A CONSTRUÇÃO DO DIAGRAMA DE CLASSES

Dissertação submetida ao Programa de Pós-Graduação em Computação Aplicada da Universidade Tecnológica Federal do Paraná como requisito parcial para a obtenção do título de Mestre em Computação Aplicada

Orientador: Prof. Dr. Laudelino Cordeiro Bastos
Coorientador: Prof. Dr. Adolfo Gustavo Serra Seca Neto

CURITIBA

2017

Dados Internacionais de Catalogação na Publicação

F315f 2017

Felisbino, Cláudio Marcio
Ferramenta para o apoio ensino-aprendizagem do modelo orientado a objetos durante a construção do diagrama de classes / Cláudio Marcio Felisbino.-- 2017.
116 f.: il.; 30 cm.

Disponível também via World Wide Web. Texto em português, com resumo em inglês.
Dissertação (Mestrado) - Universidade Tecnológica Federal do Paraná. Programa de Pós-graduação em Computação Aplicada, Curitiba, 2017
Bibliografia: f. 111-116.

1. Métodos orientados a objetos (Computação). 2. Programação orientada a objetos (Computação). 3. UML (Computação). 4. Prática de ensino. 5. Aprendizagem. 6. Software – Desenvolvimento. 7. Simulação (Computadores). 8. Computação – Dissertações. I. Bastos, Laudelino Cordeiro, orient. II. Seca Neto, Adolfo Gustavo Serra, coorient. III. Universidade Tecnológica Federal do Paraná. Programa de Pós-graduação em Computação Aplicada. IV. Título.

CDD: Ed. 22 -- 621.39

Bibliotecária: Luiza Aquemi Matsumoto CRB-9/794

ATA DE DEFESA DE DISSERTAÇÃO DE MESTRADO Nº 57

Aos 28 dias do mês de agosto de 2017, realizou-se na sala C301 a sessão pública de Defesa da Dissertação de Mestrado intitulada **"Ferramenta para apoio ensino-aprendizagem do modelo orientado a objetos durante a construção do diagrama de classes"**, apresentado pelo aluno **Cláudio Marcio Felisbino** como requisito parcial para a obtenção do título de Mestre em Computação Aplicada, na área de concentração "Engenhara de sistemas computacionais", linha de pesquisa "Engenharia de software".

Constituição da Banca Examinadora:

Laudelino Cordeiro Bastos (Orientador e Presidente) – UTFPR _____

Flávio Bortolozzi - Unicesumar _____

Robinson Vida Noronha - UTFPR _____

Maria Cláudia Figueiredo Pereira Emer – UTFPR _____

Em conformidade com os regulamentos do Programa de Pós-Graduação em Computação aplicada e da Universidade Tecnológica Federal do Paraná, o trabalho apresentado foi considerado _____ (aprovado/reprovado) pela banca examinadora. No caso de aprovação, a mesma está condicionada ao cumprimento integral das exigências da banca examinadora, registradas no verso desta ata, da entrega da versão final da dissertação em conformidade com as normas da UTFPR e da entrega da documentação necessária à elaboração do diploma, em até_____dias desta data.

Ciente (assinatura do aluno): _____

(para uso da coordenação)

A Coordenação do PPGCA/UTFPR declara que foram cumpridos todos os requisitos exigidos pelo programa para a obtenção do título de Mestre.

Curitiba PR,_____/_____/_____ _____

"A Ata de Defesa original está arquivada na Secretaria do PPGCA".

AGRADECIMENTOS

Dedico esta dissertação à minha querida esposa Karina, grande incentivadora deste trabalho e ao meu amado filho Miguel, fonte das minhas inspirações, que me apoiaram em todos os momentos durante o tempo que estive desenvolvendo este trabalho.

Também dedico ao meu estimado orientador, Professor Laudelino, que se dedicou sem restrições para que este trabalho fosse concluído.

RESUMO

O ensino do modelo orientado a objetos tornou-se uma prática muito comum na introdução dos cursos de computação e as dificuldades no processo de ensino-aprendizagem são bem conhecidas. Grande parte dos estudantes apresenta dificuldades em assimilar os conceitos e abstrações envolvidas neste paradigma. Este trabalho descreve o desenvolvimento, a implementação e testes de uma ferramenta para apoiar o processo de ensino-aprendizagem do modelo orientado a objetos, durante a construção de Diagramas de Classe. Com a ferramenta desenvolvida, foram coletados dados durante a construção de Diagramas de Classe UML, em experimentos realizados com alunos de uma Universidade situada na cidade de Curitiba. A ferramenta desenvolvida permitiu revelar comportamentos dos alunos durante a construção dos diagramas de classes. Os resultados demonstram que as intervenções realizadas por alunos em itens do Diagrama de Classes, como atributos, métodos e associações, durante a criação do mesmo, influenciam diretamente no conceito obtido. Além disso, a ferramenta oferece informações específicas sobre cada aluno, que podem ser utilizadas pelo professor para incentivar cada um a estudar determinados temas, a fim de melhorar o desempenho dos mesmos no entendimento do modelo orientado a objetos e na criação de Diagramas de Classes. A ferramenta para o apoio ao ensino-aprendizagem do modelo orientado a objetos durante a construção do Diagrama de Classes fornece dados negligenciados pelas ferramentas de construção de diagramas, podendo ser um importante método para aprimorar o processo de ensino-aprendizagem do modelo orientado a objetos.

Palavras chaves: Orientação a Objetos, Diagrama de Classes, Ensino-Aprendizagem, Educação.

ABSTRACT

The teaching of the object-oriented model has become a very common practice in the introduction of computer courses and difficulties in the teaching-learning process are well known. Most of the students present difficulties in assimilating the concepts and abstractions involved in this paradigm.

This study describes the development, implementation and testing of a tool to support the teaching-learning process of the object-oriented model during the construction of Class Diagrams. With the tool developed, data were collected during the construction of UML Class Diagrams, in experiments carried out with students of a University located in the city of Curitiba. The developed tool allowed to reveal the behaviors of the students during the construction of the class diagrams. The results show that the interventions made by students in items of the Class Diagram, such as attributes, methods and associations, during the creation of the same, directly influence the concept obtained. In addition, the tool offers specific information about each student that can be used by the teacher to encourage each one to study certain themes in order to improve their performance in understanding the object-oriented model and creating Class Diagrams. The teaching-learning tool of the object-oriented model during the construction of the Class Diagram provides data neglected by the diagrams-building tools, and can be an important method to improve the teaching-learning process of the model oriented to Objects.

Keywords: Object Orientation, Class Diagram, Teaching-Learning, Education

LISTA DE FUGURAS

FIGURA 1 – NOTAÇÃO COMUM DO DIAGRAMA DE CLASSES UML 21
FIGURA 2 – ESTRATÉGIAS DE ENSINO DO MODELO ORIENTADO A OBJETOS .. 43
FIGURA 3 – CLASSIFICAÇÃO DOS ARTIGOS ENCONTRADOS COM A REVISÃO SISTEMÁTICA 51
FIGURA 4 – ARQUITETURA DA FERRAMENTA PARA O APOIO AO ENSINO-APRENDIZAGEM DO MODELO ORIENTADO A OBJETOS DURANTE A CONSTRUÇÃO DO DIAGRAMA DE CLASSES 55
FIGURA 5 – FLUXO DE EXECUÇÃO DOS EXPERIMENTOS 56
FIGURA 6 – INTERAÇÃO DO MICROSOFT VISIO COM A AÇÃO EXECUTADA PELO ALUNO 61
FIGURA 7 – RESULTADO GERADO PELO AVALIADOR DE LOGS 79
FIGURA 8 – DIAGRAMA DE CLASSES DO EXPERIMENTO 1 85
FIGURA 9 – DIAGRAMA DE CLASSES DOS EXPERIMENTOS 2, 3 E 4 89
FIGURA 10 – CORRELAÇÃO ENTRE INTERVENÇÃO SOBRE ATRIBUTOS X COMPETÊNCIA ATINGIDA NO EXPERIMENTO 1 99
FIGURA 11 - CORRELAÇÃO ENTRE INTERVENÇÃO SOBRE MÉTODOS X COMPETÊNCIA ATINGIDA NO EXPERIMENTO 1 101
FIGURA 12 - CORRELAÇÃO ENTRE INTERVENÇÃO SOBRE RELACIONAMENTOS DE ASSOCIAÇÃO SIMPLES X COMPETÊNCIA ATINGIDA NO EXPERIMENTO 2 103
FIGURA 13 - CORRELAÇÃO ENTRE INTERVENÇÃO SOBRE RELACIONAMENTOS DE AGREGAÇÃO X COMPETÊNCIA ATINGIDA NO EXPERIMENTO 4 104
FIGURA 14 - CORRELAÇÃO ENTRE INTERVENÇÃO SOBRE RELACIONAMENTOS DE COMPOSIÇÃO X COMPETÊNCIA ATINGIDA NO EXPERIMENTO 4 105

LISTA DE TABELAS E QUADROS

QUADRO 1 – ANÁLISE DE FERRAMENTAS QUE AUXILIAM O PROCESSO DE ENSINO-APRENDIZAGEM DO MODELO ORIENTADO A OBJETOS . 47
QUADRO 2 – FERRAMENTAS INDICADAS PARA AS DIFICULDADES APRESENTADAS DO PARADIGMA ORIENTADO A OBJETOS 47
QUADRO 3 – DIFICULDADES DOS CONCEITOS MAIS FUNDAMENTAIS DE ORIENTAÇÃO A OBJETOS 47
QUADRO 4 –CRITÉRIOS DE INCLUSÃO E EXCLUSÃO DE ARTIGOS 49
QUADRO 5 –RESULTADO QUANTITATIVO DA BUSCA 50
QUADRO 6 –ARTIGOS CONSIDERADOS PARA A REVISÃO SISTEMÁTICA 51
QUADRO 7 – INFORMAÇÕES GERADAS NO ARQUIVO DE LOG 56
QUADRO 8 – INSTRUÇÕES GERADAS PELO GERADOR DE LOGS 57
QUADRO 9 – FLUXO DA INSTRUÇÃO CREATE CLASS 62
QUADRO 10 – FLUXO DA INSTRUÇÃO RENAME CLASS 63
QUADRO 11 – FLUXO DA INSTRUÇÃO REMOVE CLASS 64
QUADRO 12 – FLUXO DA INSTRUÇÃO CREATE ATTRIBUTE 65
QUADRO 13 – FLUXO DA INSTRUÇÃO RENAME ATTRIBUTE 66
QUADRO 14 – FLUXO DA INSTRUÇÃO REMOVE ATTRIBUTE 67
QUADRO 15 – FLUXO DA INSTRUÇÃO CREATE OPERATION 68
QUADRO 16 – FLUXO DA INSTRUÇÃO RENAME OPERATION 69
QUADRO 17 – FLUXO DA INSTRUÇÃO REMOVE OPERATION 70
QUADRO 18 – FLUXO DA INSTRUÇÃO CREATE INHERITANCE 71
QUADRO 19 – FLUXO DA INSTRUÇÃO REMOVE INHERITANCE 72
QUADRO 20 – FLUXO DA INSTRUÇÃO CREATE ASSOCIOATION 73
QUADRO 21 – FLUXO DA INSTRUÇÃO REMOVE ASSOCIOATION 74
QUADRO 22 – FLUXO DA INSTRUÇÃO CREATE AGGREGATION 75
QUADRO 23 – FLUXO DA INSTRUÇÃO REMOVE AGGREGATION 76
QUADRO 24 – FLUXO DA INSTRUÇÃO CREATE COMPOSITION 77
QUADRO 25 – FLUXO DA INSTRUÇÃO REMOVE COMPOSITION 78

QUADRO 26 – INSTRUÇÕES GERADAS PELO AVALIADOR DE LOGS 80
QUADRO 27 – RESUMO DOS EXPERIMENTOS REALIZADOS 83
QUADRO 28 – CENÁRIO PARA O EXPERIMENTO 1 85
QUADRO 29 – RESULTADOS OBTIDOS DO EXPERIMENTO 1 86
QUADRO 30 – CENÁRIO PARA OS EXPERIMENTOS 2, 3 E 4 88
QUADRO 31 – RESULTADOS OBTIDOS DO EXPERIMENTO 2 90
QUADRO 32 – RESULTADOS OBTIDOS DO EXPERIMENTO 3 92
QUADRO 33 – RESULTADOS DO EXPERIMENTO 4 95
QUADRO 34 - INTERPRETAÇÃO DOS RESULTADOS DO CÁLCULO DO COEFICIENTE DE CORRELAÇÃO DE PEARSON 97
QUADRO 36 - INTERVENÇÕES SOBRE OS MÉTODOS DO DIAGRAMA DE CLASSES NO EXPERIMENTO 1 100
QUADRO 37 - INTERVENÇÕES SOBRE AS ASSOCIAÇÕES SIMPLES DO DIAGRAMA DE CLASSES NO EXPERIMENTO 2 102
QUADRO 38 – ANÁLISE DO ITEM “ATRIBUTOS” OBTIDOS COM O EXPERIMENTO 2 107
QUADRO 39 - ANÁLISE DO ITEM “MÉTODOS” OBTIDOS COM O EXPERIMENTO 2 108
QUADRO 40 - ANÁLISE DOS ITENS “ASSOCIAÇÕES SIMPLES OU COMUNS”, “HERANÇA”, “COMPOSIÇÃO” E “AGREGAÇÃO” OBTIDOS COM O EXPERIMENTO 2 109

LISTA DE ABREVIATURAS E SIGLAS

UML: Unified Modeling Language.

ARCS: Attention, Relevance, Confidence and Satisfaction.

LCM: Learning Cycle Mode.

XNA: XNA's Not Acronymed.

GUI: - Graphical User Interface.

WEBL: Web enhanced BlueJ learning.

VBA: Visual Basic for Applications.

SUMÁRIO

1 INTRODUÇÃO 14
1.1 OBJETIVO GERAL E OBJETIVOS ESPECÍFICOS 16
1.2. ESTRUTURA DO TRABALHO 16
2 REVISÃO BIBLIOGRÁFICA 17
2.1 ENSINO-APRENDIZAGEM 17
2.2 ORIENTAÇÃO A OBJETOS 18
2.2.1 Diagrama de Classes 20
2.3 ENSINO-APRENDIZAGEM EM ORIENTAÇÃO A OBJETOS 22
2.4 ABORDAGENS PEDAGÓGICAS 22
2.5 FERRAMENTAS DE APOIO AO ENSINO 29
2.6 DIFICULDADES NO PROCESSO DE ENSINO-APRENDIZAGEM DO MODELO ORIENTADO A OBJETOS 38
2.7 CONSIDERAÇÕES 41
3 MATERIAIS E MÉTODOS 42
3.1 QUESTÕES DE PESQUISA 42
3.1.1 Estratégias no processo de ensino-aprendizagem do modelo orientado a objetos 42
3.1.2 Ferramentas computacionais que analisam o processo de ensino-aprendizagem do modelo orientado a objetos, durante o uso da ferramenta 44
3.1.3 Categorias 44
3.1.4 Dificuldades no processo de ensino-aprendizagem do modelo orientado a objetos 45
3.2 ESTRATÉGIAS DE PESQUISA DE ARTIGOS DA LITERATURA 46
3.2.1 Escopo da pesquisa 46
3.2.2 Palavras-chave 46
3.2.3 Strings de busca 46
3.2.4 Condução da revisão sistemática 47

3.2.5 IEEE Xplore 47
3.2.6 ACM Digital Library 47
3.2.7 Springer Link 48
3.2.8 ScienceDirect 48
3.2.9 Critérios de seleção 48
3.2.10 Extração de dados e classificação 49
3.2.11 Resultados obtidos 50
3.3 FERRAMENTA PARA O APOIO ENSINO-APRENDIZAGEM DO MODELO ORIENTADO A OBJETOS DURANTE A CONSTRUÇÃO DO DIAGRAMA DE CLASSES 54
3.4 EXPERIMENTOS REALIZADOS 82
3.4.1 Primeiro experimento 84
3.4.2 Segundo experimento 87
3.4.3 Terceiro experimento 90
3.4.4 Quarto experimento 93
3.5. RESULTADOS, ANÁLISE E DISCUSSÃO 96
4 CONCLUSÃO 111
REFERÊNCIAS 113

1 INTRODUÇÃO

Com a constante evolução das empresas, as regras de negócio também evoluem. Para conservar o padrão de qualidade na produção de software, é necessário a adoção de métodos de desenvolvimento que atendam a essa dinâmica de mercado. Existem diferentes tipos de modelos que podem ser utilizados, seguindo diferentes abordagens. A orientação a objetos é uma das abordagens mais utilizadas.

No contexto acadêmico, o ensino de programação e modelagem orientada a objetos é uma prática comum na introdução da Ciência da Computação (Anquan et al. 2010). A programação orientada a objetos, é uma maneira de pensar sobre o processo de decompor o problema e desenvolver soluções de programação (Li e Xu 2010).

O processo de ensino-aprendizagem do modelo orientado a objetos, para o desenvolvimento de sistemas de software, sempre foi alvo de discussões e preocupação por parte de professores e alunos, pois nota-se que os alunos possuem muitas dificuldades no aprendizado deste modelo. A orientação a objetos é um paradigma muito difundido na academia e no mercado e seus conceitos norteiam a modelagem de sistemas de software e a codificação em linguagens de programação orientadas a objetos. Para os cursos de computação, a Sociedade Brasileira de Computação evidencia a importância da orientação a objetos como um dos principais tópicos das disciplinas de Linguagens de Programação e de Engenharia de Software. Assim, os conceitos de orientação a objetos devem ser ensinados de forma completa, correta e consistente, pois a falta ou a má compreensão destes conceitos poderá trazer consequências prejudiciais na formação dos alunos.

A justificativa para realizar um estudo sobre o processo de ensino-aprendizagem do modelo orientado a objetos se dá pelo fato de que algumas pesquisas indicam que iniciantes em programação têm dificuldades na compreensão e aplicação de conceitos e abstrações presentes nesse paradigma (Xinogalos et al. 2006). Muitas vezes os alunos enfrentam dificuldades em aprender programação e os conceitos de orientação a objetos e isso tem contribuído para a geração de um alto índice de evasão e reprovação nas disciplinas que requerem conhecimentos de programação orientada a objetos (Hinterholz 2009). O estudo realizado por Henrique e Rebouças (2015),

apresenta os índices de reprovação na disciplina de programação orientada a objetos da UFPB (Universidade Federal da Paraíba) - Campus IV - durante oito semestres, que se mostraram, na maioria dos casos, superiores a 50%. Para os autores, a complexidade do paradigma orientado a objetos pode ser um dos fatores que leva a esses números, e esta complexidade já é discutida por vários trabalhos da área de ensino de computação e informática. Por isso, se faz necessário criar alternativas de ensino para auxiliar os alunos durante a disciplina de programação orientada a objetos e motivá-los mais, já que os índices de abandono (trancamentos ou reprovações por falta) na disciplina são bem altos.

Vários estudos têm apresentado métodos de avaliação para o diagnóstico de problemas de aprendizagem com o objetivo de melhorar o ensino da programação de computadores nos cursos de Ciências da Computação (Krause 2014). Desta forma, várias pesquisas têm sido realizadas com o objetivo de auxiliar professores e alunos no processo de ensino-aprendizagem do modelo orientado a objetos. Para Yulia e Adipranata (2010), aprender projeto e programação orientada a objetos é um desafio para os novatos, especialmente para estudantes que já têm familiaridade com o paradigma procedural. A mudança do paradigma procedural para o orientado a objetos é uma tarefa difícil. Assim, muitos alunos têm dificuldades que não podem ser totalmente resolvidas por professores durante o processo de ensino-aprendizagem.

Ensinar a resolução de problemas e programação orientada a objetos é uma tarefa muito complexa (Westin e Nordstrom 2004). A maneira como os cursos de programação introdutórios estão organizados não funcionam bem para o paradigma orientado a objetos, pois os alunos são avaliados geralmente ao final do processo de aprendizagem.

Considerando a existência de estudos que discutem o processo de ensino-aprendizagem do modelo orientado a objetos, o objeto de estudo deste trabalho é analisar o comportamento dos alunos durante o aprendizado do modelo orientado a objetos, por meio da modelagem do Diagrama de Classes, não avaliando somente o modelo final construído pelos alunos, mas todo o percurso para obter este modelo, identificando assim possíveis fraquezas e dificuldades que os alunos apresentam para o entendimento e construção de soluções usando o modelo orientado a objetos. Para

isso, foram desenvolvidos e implementados dois algoritmos: o primeiro para registrar as atividades dos alunos durante a modelagem do Diagrama de Classes e o segundo para consolidar e permitir análises dos dados gerados durante a construção do Diagrama de Classes, por meio da realização de experimentos.

1.1 OBJETIVO GERAL E OBJETIVOS ESPECÍFICOS

O objetivo geral deste trabalho é conceber e avaliar um método que permita avaliar os resultados do processo de ensino-aprendizagem do modelo orientado a objetos, durante a construção de Diagramas de Classe, com o uso de uma ferramenta computacional.

Para buscar o objetivo principal, os objetivos específicos são:

- Desenvolver e implementar um algoritmo para registrar as atividades dos alunos durante a modelagem do Diagrama de Classes.
- Desenvolver e implementar um algoritmo para consolidar e permitir análises dos dados gerados durante a construção do Diagrama de Classes.
- Coletar dados que são gerados durante a criação do Diagrama de Classes, por meio de experimentos.
- Investigar possíveis correlações entre os dados gerados pelos experimentos e o conceito obtido pelos alunos.
- Investigar as possíveis dificuldades que os alunos apresentam no processo de ensino-aprendizagem do modelo orientado a objetos, durante a criação do Diagrama de Classes.

1.2. ESTRUTURA DO TRABALHO

Este documento está estruturado da seguinte forma: o Capítulo 2 apresenta a revisão bibliográfica, o Capítulo 3 descreve a metodologia adotada, o Capítulo 4 apresenta os resultados, análises e discussões do trabalho e finalmente o Capítulo 5 apresenta as conclusões obtidas durante a realização dos experimentos e análise dos resultados.

2 REVISÃO BIBLIOGRÁFICA

Neste capitulo serão contextualizados os principais conceitos tratados neste trabalho e que justificam a pesquisa: Ensino-Aprendizagem e Orientação a Objetos. Além disso, serão apresentados trabalhos sobre o processo de ensino-aprendizagem do paradigma orientado a objetos, que foram selecionados por meio de uma revisão sistemática da literatura.

2.1 ENSINO-APRENDIZAGEM

O ensino e a aprendizagem são dois aspectos de um fenômeno conhecido como ensino-aprendizagem, onde a natureza do ensinar tem como compromisso assegurar que todos aprendam (Moreira e Masini 1982).

O modelo de ensino tradicional sinaliza a ocorrência de um ensino centrado na figura do professor, que detêm a autonomia do conhecimento, gerando estratégias repetitivas, geralmente com aulas expositivas, e consequentemente criando um fluxo unilateral de comunicação, dificultando o desenvolvimento do pensamento crítico por parte do aprendiz, que na maioria das vezes assimila o que lhe é imposto, sem muitos questionamentos. Esta conduta caracteriza o ensino tradicional, supondo que o indivíduo que aprende é incapaz de ter controle de si mesmo, devendo ser conduzido por pessoas que sabem mais que do ele. Este tipo de educação frequentemente impede a criatividade, a iniciativa, a auto responsabilidade e a auto direção. Esta prática é denominada de educação bancária, onde o papel do aluno é limitado a receber depósitos, guardar e arquivar, preocupando-se basicamente com a transmissão do conhecimento e com a experiência do professor, sem atentar para os aprendizes enquanto pessoas que fazem parte de um contexto maior (Haddad 1993). Assim, gera-se um aluno passivo, memorizador de conceitos abstratos e sem preparo para resolver questões práticas, fundamentadas na realidade em que vive. Haddad (1993) considera que ensinar é facilitar a aprendizagem, criando condições para que o outro, a partir dele próprio aprenda e cresça. O autor acrescenta que, na modalidade de ensino tradicional, o indivíduo é o centro da aprendizagem que se processa em função do desenvolvimento e interesse do aluno. Coloca-se assim uma ênfase nas relações interpessoais e no crescimento pessoal que delas resultam.

Os estudos mais recentes sugerem colocar o aluno como sujeito ativo no processo ensino-aprendizagem. Este é o principal objetivo das abordagens ativas de ensino-aprendizagem, ou metodologias ativas. Segundo Moura e van Hattum-Janssen (2011), capacitar o aluno a assumir a responsabilidade pelo aprendizado e reconhecer os pontos de vista construtivista da aprendizagem, sugerem que o aluno é o único que pode ser responsável pelo aprendizado. Os professores podem fazer muito para facilitar a aprendizagem, mas eles não podem ser os únicos responsáveis pela aprendizagem do aluno.

2.2 ORIENTAÇÃO A OBJETOS

A modelagem de informações, popularizada nos anos 80, é a precursora mais próxima da análise e do projeto orientados a objetos. O primeiro material publicado sobre análise orientada a objetos foi de Sally Shlaer e Stephen Mellor (Winbland 1993). A técnica, começa pela definição de objetos, classes e atributos. Diferente da decomposição funcional, a técnica é caracterizada pela definição de objetos, classes e atributos, resultando em pouca codificação dirigida aos dados.

Fundamentada na forma natural do ser humano pensar o mundo real, a modelagem orientada a objetos, apresenta-se, também, como uma forma vantajosa de proceder-se a análise, projeto e construção de modelos matemáticos que, implementados em computador, busquem representar as principais características que se deseja abstrair desse mundo. Observe-se que essa capacidade de representação de sistemas reais de alta complexidade foi o que originou esse paradigma, concebido inicialmente na Noruega, no início dos anos 60, aplicado às linguagens de simulação. Isso vem a ser de singular importância para a modelagem de problemas que representem o mundo físico, seu entorno e as diferentes interações entre ele e as necessidades humanas, como acontece com os problemas de engenharia. Para Coad e Yourdon (1992), a modelagem orientada a objetos decorre, na verdade, da observação de três processos de organização mental utilizados pelo ser humano para entender o mundo em que vive:

- Diferenciação: condição de distinguir objetos entre si, por meio de seus atributos e da capacidade que apresentam de reagirem de modo diferenciado a estímulos externos;
- Distinção entre Todo e Parte: condição de perceber um objeto como constituído de partes componentes que interligadas formam um todo;
- Percepção de Classes Distintas: condição de reunir objetos com características semelhantes em classes.

Assim, a modelagem orientada a objetos é uma metodologia com grande potencial de servir como extensão para a capacidade humana de perceber, compreender e interagir como o mundo, por intermédio da utilização de mecanismos compatíveis com esses processos mentais, e não apenas uma técnica de modelagem computacional. Essa é a razão pela qual lhe é dada, aqui, de modo enfático, a denominação de modelagem orientada a objetos ao invés de programação orientada a objetos, costumeiramente utilizada no âmbito da computação.

A orientação a objetos é um paradigma de análise, projeto e programação de sistemas de informação, baseado na composição e na interação entre diversas unidades do sistema, chamadas de objetos. A análise e projeto orientados a objetos têm como meta identificar o melhor conjunto de objetos para descrever um sistema. O funcionamento deste sistema se dá por meio do relacionamento e troca de mensagens entre estes objetos (Stein Júnio, Malucelli e Bastos 2009).

O paradigma de orientação a objetos é considerado por muitos profissionais um tanto complexo e de difícil compreensão (Guedes 2009). Por ser um paradigma que trabalha muito com a abstração e a classificação dos objetos, nem sempre fica claro por parte dos alunos esse conceito, embora seja fundamental para a aprendizagem da disciplina de modelagem de software, em cuja ementa, o paradigma de orientação a objetos é parte constitutiva e essencial. Dessa forma, criar mecanismos para facilitar essa aprendizagem é necessário, a fim de que ajudem tanto o professor quanto o aluno para que as aulas sejam mais interativas e dinâmicas, quebrando o paradigma atual das aulas expositivas, que são executadas partindo de um modelo pronto, tornando-se mecanizada.

Projetar um software orientado a objetos é como organizar uma comunidade de indivíduos, cujas tarefas coletivas alcançam a metas maiores da comunidade. Encontrar abstrações bem definidas significa identificar os indivíduos como classes de objetos, cujos comportamentos engrenam bem com os outros. Cada objeto deve ficar sozinho e agregar valor, mas os objetos não são projetados isoladamente (Wirfs-Brock 2006).

A programação requer um entendimento correto de conceitos abstratos e a teoria da programação orientada a objetos é baseada em uma representação do mundo real que emprega princípios abstratos e operações abstratas básicas (Oliveira, Conte e Riso 1998). No entanto, a abstração é um conceito muito complexo para ser ensinado, por isso muitos estudantes encontram dificuldade em programação, especialmente no modelo orientado a objetos. Isso pode levar a altas taxas de insucesso ou de abandono (Esteves et al 2011).

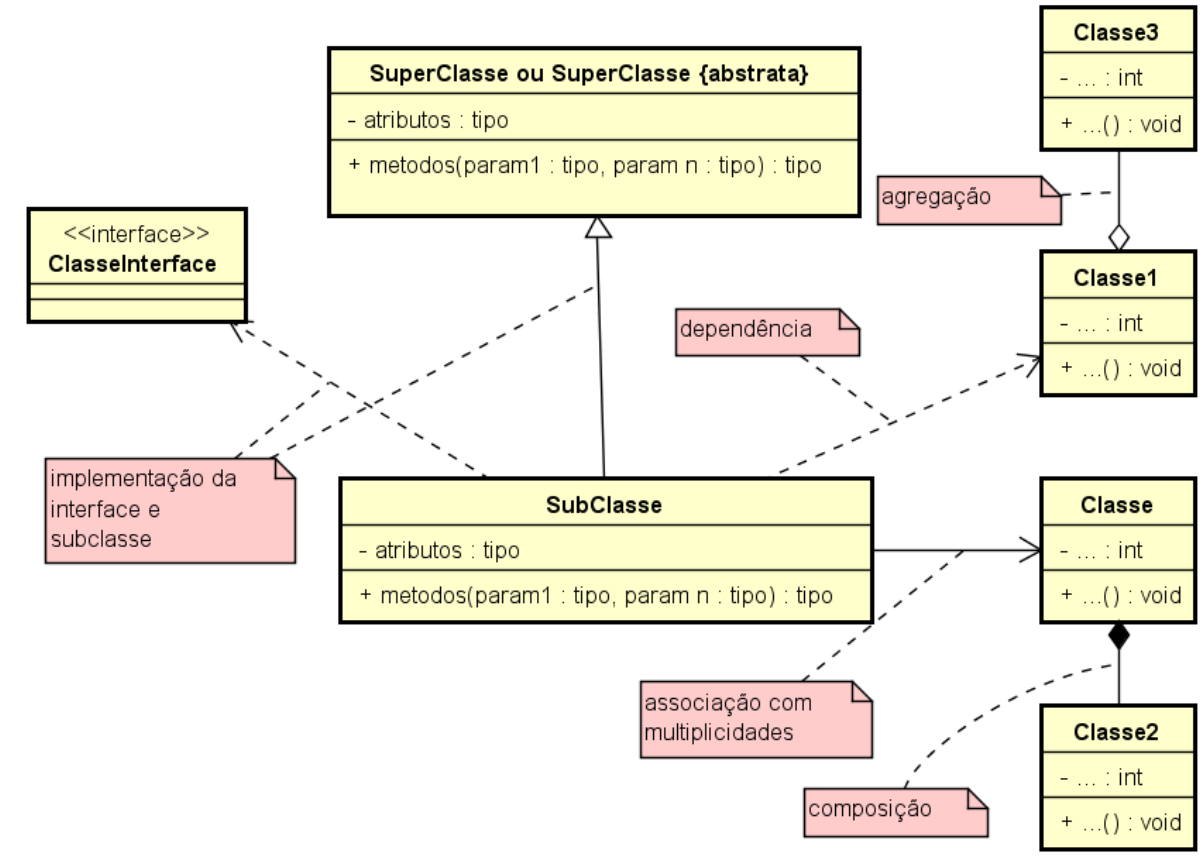
2.2.1 Diagrama de Classes

O Diagrama de Classes é um artefato da UML (Unified Modeling Language), sendo o diagrama central da modelagem orientada a objetos. A UML é uma notação padrão de diagramação. Embora seja útil aprender a diagramação, há questões mais cruciais orientadas a objetos para aprender; especificamente, como pensar em objetos (Larman 2007).

Para Larman (2007), o Diagrama de Classes serve para ilustrar classes, interfaces e suas associações. Eles são usados para modelagem estática de objetos. O Diagrama de Classes é considerado por muitos autores como o mais importante e o mais utilizado diagrama da UML. Seu principal enfoque está em permitir a visualização das classes que irão compor o sistema com seus respectivos atributos e métodos, bem como em demonstrar como as classes do sistema se relacionam, se complementam e transmitem informações entre si. Este diagrama apresenta uma visão estática de como as classes estão organizadas, preocupando-se em definir a estrutura lógica das mesmas. O Diagrama de Classes serve como base para a construção da maior parte dos demais diagramas da UML. A Figura 1 apresenta a notação comum do Diagrama de Classes UML. A maioria dos elementos da Figura 1 é opcional. Modeladores os

desenham, mostram ou ocultam dependendo do contexto e das necessidades do leitor ou ferramenta UML.

FIGURA 1 – NOTAÇÃO COMUM DO DIAGRAMA DE CLASSES UML



FONTE: Adaptado de Larman, 2007.

É importante ressaltar que a UML não é análise e projeto orientado a objetos ou um método, é apenas uma notação de diagramação. Assim, não adianta aprender diagramação UML e não ser capaz de criar um excelente projeto orientado a objetos, ou avaliar e melhorar um existente. Essa é a habilidade mais difícil e de maior importância.

2.3 ENSINO-APRENDIZAGEM EM ORIENTAÇÃO A OBJETOS

Com o objetivo de investigar trabalhos que auxiliam no processo de ensino-aprendizagem para o modelo orientado a objetos, foi realizada uma revisão sistemática da literatura, seguindo o processo de condução de revisões sistemáticas definido por Kitchenham (2004). Revisões sistemáticas são baseadas em uma estratégia de pesquisa bem definida, que visa detectar o máximo possível de material bibliográfico relevante. Antes de iniciar a busca dos estudos primários, deve-se definir um protocolo de revisão que especifica a questão central da pesquisa e os métodos que serão utilizados para executar a revisão. O protocolo deve indicar os critérios de inclusão e exclusão explícitos para acessar cada estudo primário potencial e documentar a estratégia de busca utilizada, de forma a permitir que leitores possam conhecer seu grau de rigor e completude (Esteves et al 2011).

Nas seções 2.4 e 2.5 serão apresentados os principais trabalhos obtidos com o processo da revisão sistemática, classificados em categorias. Não há uma classificação padrão de categorização dos estudos relacionados ao processo de ensino-aprendizagem do modelo orientado a objetos. A operação de categorização está detalhada na seção 3.1.3 deste trabalho. Assim, os trabalhos encontrados na revisão sistemática serão classificados nas seguintes categorias: Abordagens pedagógicas e Ferramentas de apoio ao ensino.

A seção 2.6 apresenta os estudos que apontam as principais dificuldades encontradas no processo de ensino-aprendizagem do modelo orientado a objetos.

2.4 ABORDAGENS PEDAGÓGICAS

Nesta seção, serão apresentados os estudos que utilizam metodologias, práticas e abordagens pedagógicas que apoiam o ensino-aprendizagem do modelo orientado a objetos.

A pesquisa realizada por Martín, Lázaro e Hernán-Losada (2010) apresenta um estudo de caso de aprendizagem ativa no curso de Engenharia de Telecomunicações da Universidade Rey Juan Carlos. Especificamente, este estudo apresenta a experiência de diferentes atividades de aprendizagem ativa em programação orientada a objetos, onde os alunos desenvolveram a sua iniciativa e pensamento crítico.

Primeiramente, o professor explica os conceitos e os diferentes tipos de diagramas UML. Em seguida, o professor propõe um conjunto de exercícios práticos relacionados com o modelo orientado a objetos. Após isso, os alunos enviam as suas soluções para um sistema, que pode ser acessado por outros alunos. Eles devem corrigir as soluções de dois colegas de classe, enviando os seus feedbacks. Desta forma, os professores despertam o trabalho colaborativo, o que promove a aprendizagem ativa e autônoma. Depois que o método começou a ser aplicado, 14 alunos não alcançaram aprovação na disciplina de orientação a objetos em 2006-2007. Já em 2007-2008, 12 alunos não foram promovidos na disciplina e em 2008-2009 6 alunos não conseguiram aprovação. Observou-se que este método consegue uma diminuição de reprovação significativa desses estudantes. Apenas 23% do total de alunos não passou no período de 2008-2009 em comparação a 54% no período de 2006-2007 e 48% no período de 2007-2008 ano. Estes resultados globais mostram uma clara melhoria nos escores dos alunos.

Na tentativa de melhorar o desempenho dos alunos, a metodologia ativa Flipped *Classroom* (Sala de Aula Invertida) é aplicada por Jonsson (2015). Ao invés de assistir às aulas tradicionais, os alunos aprendem o conteúdo do curso por conta própria, assistindo a vídeos gravados, enquanto as atividades consideradas tradicionais, como a resolução de exercícios e trabalhos, são realizadas em sala aula. O autor adiciona também a discussão em pares, para que os assuntos vistos pelos alunos possam ser discutidos entre eles. Os resultados mostram que 81% dos alunos que participaram do experimento foram aprovados, em comparação com alunos que receberam o mesmo conteúdo no modelo tradicional, onde o resultado foi de 60%. Além disso, a porcentagem de alunos que foram aprovados com boas notas foi de 58%, contra 32% no modelo tradicional.

Historicamente, na abordagem tradicional do ensino orientado a objetos, os professores iniciam com os princípios básicos da programação estruturada. Os alunos são introduzidos com assuntos relacionados a variáveis, tipos de dados, estruturas de controle e funções ou métodos, para daí sim aprenderem os conceitos de orientação a objetos. No estudo realizado por Janke, Brune e Wagner (2015), é adotado o método Outside-In, cuja essência é iniciar com as propriedades que causam mais impacto fora do elemento e seguir até chegar ao centro do mesmo, nos detalhes mais internos.

Desta forma, o estudo utiliza o método Outside-In para iniciar com os conceitos de orientação a objetos. Os resultados mostram que os estudantes que aprenderam orientação a objetos com o método Outside-In, predominantemente compreendem os conceitos e são capazes de explicar partes do mesmo. Em sete perguntas relacionadas a conceitos de orientação a objetos, os alunos atingiram 77,4% dos pontos alcançáveis. Em cinco, das sete perguntas os alunos atingiram mais de 75% dos pontos. Em exceções de tempo de execução e manipulação de exceções parecia ser mais difícil, aqui 68,3% dos pontos possíveis foram alcançados. Em relação a polimorfismo os alunos alcançaram 57,2% de todos os pontos possíveis, que ainda é mais do que a metade. Para os autores, em geral, o método Outside-In pode ser considerado bem-sucedido.

Objetos de aprendizagem também são utilizados como motivação no processo de ensino-aprendizagem do modelo orientado a objetos. Um objeto de aprendizagem é definido como qualquer entidade, digital ou não digital, que pode ser utilizado para a aprendizagem, educação ou formação. Para Narasimhamurthy e Al Shawkani (2009), um objeto de aprendizagem é uma unidade de recurso digital que pode ser compartilhado para apoiar o ensino e aprendizagem. Sempre que um objeto de aprendizagem é desenvolvido, o professor deve envolver os alunos no conteúdo, fortalecendo o envolvimento cognitivo entre o aluno e o conteúdo. Nesse sentido, os autores adotam a teoria educacional de ARCS (Attention, Relevance, Confidence and Satisfaction) e as estratégias pedagógicas significativas para projetar um modelo de objeto de aprendizagem construtivo. A partir da observação e estudo de vários modelos, os autores propõem a criação de um ambiente simulado que compila um objeto de aprendizagem para qualquer tópico da linguagem de programação Java. Neste modelo, a compreensão dos conceitos é apresentada como páginas flash estáticas. Em seguida, o aluno tem a liberdade de fazer o link com o programa Jeliot, que permite a entrada e execução de qualquer código em um ambiente on-line virtual. Este é o lugar onde o dinamismo deste modelo reside. Durante o processo de ensino-aprendizagem, os professores tentam atrair a atenção dos alunos, fornecendo exemplos do cotidiano na utilização do conceito a ser aprendido. Os alunos compreendem a relevância do conceito no contexto da linguagem de programação. O

próximo passo é construir gradativamente o conhecimento adquirido em um novo programa e isso vai elevar a confiança em aprender novos conceitos. Durante a resolução das questões, no ambiente interativo, o aluno demonstra mais satisfação. Assim, o modelo alcança o objetivo, devidamente alinhado com a teoria educacional ARCS.

Algumas pesquisas sugerem que as dificuldades e a falta de motivação dos alunos em aprender o modelo orientado a objetos tem como causa o uso de materiais que não estão relacionados com as suas vidas e o seu cotidiano. No estudo realizado por Zhu (2011), uma abordagem é desenvolvida para integrar a educação financeira em um curso de programação orientada a objetos. A maioria dos alunos não têm conhecimento financeiro necessário para gerir as suas dívidas. Essa abordagem é eficaz para motivar os alunos e melhorar os resultados de aprendizagem. Para ilustrar o ensino de tipos de dados, operadores, estruturas de controle, classes, objetos, herança e polimorfismo, os alunos recebem uma atribuição para calcular e gerar uma saída do balanço mensal, juros e juros de uma conta bancária que vence a uma determinada taxa. De acordo com a teoria construtivista, a aprendizagem é um processo de construção do conhecimento pelo aluno, e o processo funciona melhor quando ele está ativamente envolvido na aquisição, produção, análise e estruturação da informação. Com essa abordagem, os estudantes estão ativamente engajados no processo de aprendizagem, abordando um dos maiores problemas que enfrentam.

Colocar o aluno como sujeito ativo no processo de ensino-aprendizagem é um dos objetivos do LCM (Learning Cycle Mode), que é uma abordagem proposta por Liu et al. (2009). Nesta abordagem, primeiramente os alunos são desafiados a coletar informações relevantes sobre um determinado assunto ou conceito, o que de fato interessa no início da disciplina e, posteriormente, eles se dedicam à aprendizagem baseada na investigação de materiais relacionados ao assunto. Depois disso, os estudantes criam uma aplicação que está em paralelo aos materiais de aprendizagem investigados. Com a abordagem LCM, o entusiasmo para a solução de problemas é despertado naturalmente. Os alunos da disciplina de orientação a objetos devem desempenhar um papel mais ativo para resolver os problemas. Neste artigo, foi

discutido o método de ensino LCM na implementação do curso de programação orientada a objetos. O LCM é segmentado em 3 etapas:

- Solicitação da consulta ou pesquisa.
- Investigação.
- Aplicação.

Por fim, é apresentada a avaliação de aplicação no ensino LCM. Os autores propõem também, o ganho no ensino em cultivar talentos, estimulando o pensamento ativo e motivar inovação. Neste LCM, as etapas são:

- Construção da estrutura de conhecimento.
- O percurso de investigação ativa.
- Melhorar a sua capacidade prática.

Ensinar a resolução de problemas e programação orientada a objetos é uma tarefa muito complexa. A maneira como os cursos de programação introdutórios estão organizados não funcionam bem para o paradigma orientado a objetos. Programas de estudo tradicionais para programação introdutória tendem a se concentrar em detalhes sintáticos e, portanto, a maioria dos estudantes têm dificuldades para tirar proveito dos conceitos de orientação a objetos. Ser fiel ao paradigma orientado a objetos no início de um curso de computação não é fácil quando, ao mesmo tempo, os alunos têm que se familiarizar com a programação. Nesse sentido, Westin e Nordstrom (2004) afirmam que os objetos devem ser ensinados desde o início com um foco inicial no projeto. O ensino de conceitos de orientação a objetos gerais deve ser favorecido em detrimento dos detalhes sintáticos de uma ou outra linguagem de programação. Os autores propõem um projeto chamado FataOO, em que objetivo geral foi desenvolver uma abordagem holística *Object-Centric* para o ensino de orientação a objetos. Ao olhar para as dificuldades em ensinar o modelo orientado a objetos, três grandes problemas envolvidos foram detectados:

- Falta de pedagogias adequadas e comprovadas para ensinar os de orientação a objetos e programação.

- A falta de livros didáticos adequados. Quase todos os livros didáticos atuais afirmam seguir uma abordagem particularmente adequada para a programação a orientada a objetos, como abordar objetos em primeiro lugar.
- Falta de ambientes de programação adequados. A maioria dos ambientes são demasiadamente complexos para iniciantes que ainda lutam com a programação.

Estas ferramentas acrescentam complexidade em vez de simplificar a tarefa. O grande objetivo deste estudo é proporcionar aos alunos um bom começo em seus estudos universitários e subsídios para a resolução de problemas e programação orientada a objetos.

O estudo realizado por Giordano e Maiorana (2013) sugere que o desenvolvimento de jogos estimula os alunos e os deixam mais motivados para aprender os conceitos de orientação a objetos. Os autores apresentam uma experiência de ensino com foco na aprendizagem de projetos e programação orientada a objetos usando o Microsoft XNA para construir jogos. Os principais elementos da abordagem são:

- Um projeto e investigação com base em um método pedagógico.
- Uma abordagem Object-First, com a construção de Diagramas de Classe e diagramas de sequência UML.
- A criação de um local compartilhado para as soluções dos alunos, promovendo a autorreflexão e pensamento crítico.

Com isso, os autores elaboram um plano de ensino com as seguintes características:

- Uma abordagem pedagógica baseada em inquérito, assemelhando-se a pedagogia de processo orientado Guided-Inquiry. Uma primeira abordagem a objetos que leva à concepção de um Diagrama de Classes UML aplicada a um projeto de design nas duas primeiras semanas do curso, seguido pelo Diagrama de Sequência UML, primeiro simulado por dramatização e, em seguida, projetado em UML nas semanas seguintes.

- Um projeto de aprendizagem baseado na atividade em grupo. A atividade em grupo no desenvolvimento do projeto foi realizada com uma atribuição específica de responsabilidade para cada aluno do grupo, que teve que implementar primeiro uma única classe com o teste associado e, em seguida, um número crescente de classes e programas de teste para cada classe.
- Um blog para apoiar as atividades. O blog é usado como um meio para os alunos publicarem as suas soluções, que são revisadas pelos outros alunos e formativamente avaliadas pelo professor.

Técnicas de aprendizagem cooperativa ou colaborativa se baseiam na integração dos alunos em busca do conhecimento e visam a realização de um objetivo comum. Para Portillo e Campos (2009), este tipo de técnica gera conhecimento mais significativo e duradouro. Além disso, o aluno tem de enfrentar o desafio de não só aprender individualmente, mas também explicar aos seus colegas os conceitos que eles não compreendem bem, o que implica uma prática de suas habilidades de comunicação, argumentação e discussão. O trabalho apresenta um projeto baseado em aprendizagem cooperativa para ensinar orientação a objetos, utilizando a técnica *Jigsaw*. *Jigsaw* é uma técnica de aprendizagem ativa e cooperativa em que os alunos podem aprender melhor fazendo o trabalho uns com os outros. Na prática, essa técnica consiste em dividir o material de aprendizagem em tarefas parciais. Cada aluno em uma equipe terá que executar uma dessas tarefas parciais, o que eventualmente vai acabar integrado por todos os membros da equipe. Os resultados dos testes realizados no início e no final de uma sessão comprovou que os alunos que tinham algumas dúvidas no início, eventualmente, melhorando seu desempenho quando o trabalho cooperativo foi finalizado. De acordo com as respostas dadas em um questionário anônimo, pode-se dizer que os alunos consideram que o trabalho realizado ajudou a melhorar a sua compreensão e aprendizagem do tema. Do total de alunos, 38% melhorou as suas competências de alguma forma e 26% mantêm os seus conhecimentos. Outros 6% regrediram e 30% mantiverem seu conhecimento com erros.

O trabalho proposto por Giordano e Maiorana (2014), apresenta uma experiência de ensino onde foi usada a abordagem *Design First* para ensinar orientação a objetos intercalando com a programação orientada a objetos que visa implementar o software

modelado por meio de atividades de laboratório. Com essa abordagem, o objetivo é fornecer, primeiramente, uma base sólida em design, usando os Diagramas de Classe e sequência da UML. Com isso, o objetivo é melhorar a abstração e as habilidades dos alunos para resolver problemas, que são considerados mais importantes do que regra sintática. Com foco inicial na modelagem, seguido por uma fase de codificação, os professores devem ter atenção para a implementação do projeto, para consolidar e praticamente aplicar os conceitos mais abstratos. As duas fases têm que ser desenvolvidas simultaneamente, sempre que possível. Segundo os autores, a abordagem, tal como avaliada pelas qualidades dos projetos dos alunos e por melhorias no tipo e número de erros nas provas escritas formais, é promissora.

A aprendizagem pela descoberta é muito desejável, porque prepara os alunos para a aprendizagem ao longo da vida. Muitas vezes, os alunos adquirirem conceitos de orientação a objetos, aprendendo a programar. Para ser capaz de modelar e avaliar soluções próprias, os alunos devem adquirir muitos detalhes de uma linguagem de programação e aplicar em ambientes de desenvolvimento de software complexos. A pesquisa apresentada por Brinda (2006), reflete a aprendizagem por exploração. A ideia básica é que, antes de desenvolver soluções próprias, os alunos explorem soluções de médio porte por meio de sistemas computacionais.

2.5 FERRAMENTAS DE APOIO AO ENSINO

Nesta seção serão apresentados os estudos que utilizam softwares que apoiam o ensino-aprendizagem do modelo orientado a objetos.

De modo geral, ambientes de aprendizagem, ensino do modelo orientado a objetos baseado em jogos e softwares que facilitam o aprendizado do modelo orientado a objetos, são utilizados como facilitadores no processo de ensino-aprendizagem do modelo orientado a objetos. De acordo com Hosanee e Panchoo (2015), muitas pesquisas são realizadas e ferramentas educacionais são desenvolvidas para ajudar os professores a proporcionar padrões de qualidade às aulas, com o objetivo de motivar os alunos a não desistir das aulas de programação. Estas ferramentas de software foram classificadas em 5 categorias:

- *Narrative tools*: Permitir que o usuário siga uma história através de tutoriais. No final, o usuário deve ser capaz de construir a história no software.
- *Visual programming tool*: As ferramentas de programação visual podem ser estáticas ou dinâmicas. As ferramentas estáticas não nos permitem ver o estado e comportamento das classes e as dinâmicas nos permitem ver o estado e o comportamento dos objetos.
- *Flow-model tools*: São ferramentas que permitem ao usuário conectar elementos do programa.
- *Especialized output realizations*: São ferramentas especializadas que fornecem comentários não-textuais que orienta o usuário mais adequadamente.
- *Tiered language tools*: Permite que o usuário trabalhe apenas com as características que ele tem permissão em função das suas habilidades de proficiência. Por exemplo, um usuário iniciante terá menos recursos que aparecem em sua tela, em comparação com um usuário avançado.

O Quadro 1 apresenta a análise de 9 ferramentas que auxiliam o processo de ensino-aprendizagem do modelo orientado a objetos.

QUADRO 1- ANÁLISE DE FERRAMENTAS QUE AUXILIAM O PROCESSO DE ENSINO-APRENDIZAGEM DO MODELO ORIENTADO A OBJETOS

Software	História		Visual	Diagrama	Saída Especializada	Interface Simples
Alice	X		X			
BlueJ			X	Diagrama de classes		X
JPIE			X			
Greenfoot			X	Diagrama de classes		
Jeeroo	X		X			
Scratch	X		X			
Baltie			X	UML	X	
Jhave			X			
Raptor			X	Flowchart		

FONTE: Adaptado de Hosanee e Panchoo, 2015.

Muitas ferramentas são concebidas com o objetivo de ajudar os alunos com dificuldades de aprendizagem, especialmente em linguagens de programação orientada a objetos. No trabalho realizado por Fournier (2005), é usada a ActiveTutor, que é uma ferramenta pedagógica para ajudar os professores que ensinam programação orientada a objetos. A ferramenta mostra como os especialistas constroem software e o que acontece enquanto eles estão em execução. É independente de linguagem de programação, conhece os fundamentos e é capaz de mostra-los, não importando se a linguagem usada para escrever algoritmos é C++, C#, Java ou simplesmente uma linguagem formal. Durante a execução e animação de um algoritmo, ele mostra os objetos construídos pelo algoritmo em execução e as ligações entre eles. Se os objetos estão fazendo referência a outros objetos ou apontando para outros objetos, a ferramenta desenha setas de cores diferentes para mostrar os diferentes links. Cada

objeto tem uma localização na tela e é destacado ou não de acordo com o seu papel e importância em um dado momento.

Adiar a aprendizagem de uma linguagem de programação é o objetivo proposto por Livovský e Porubän (2014). Professores que iniciam o curso de programação explicando o paradigma de orientação a objetos por meio de pequenos exemplos de código enfrentam um problema: os alunos não sabem a linguagem de programação ainda e com isso, eles precisam lidar com sintaxes e características da linguagem ao mesmo tempo que aprendem os conceitos e princípios da orientação a objetos. Nesta abordagem, os autores ensinam os princípios básicos, tais como classe, objetos e relações entre os objetos no início do curso de programação orientada a objetos. Os autores também aproveitam a atratividade de jogos de computador para aumentar a motivação de alunos, para conduzir a um melhor desempenho de aprendizagem dos alunos. Uma das abordagens mais populares usadas ultimamente é um método chamado *Object-First*. Esta abordagem entende que, primeiramente, os alunos devem ser colocados em exposição com os conceitos fundamentais do paradigma orientado a objetos. Assim que o aluno entende o conceito de objetos, ele vai para a definição de classes. O foco está no uso antes da implementação.

Fazendo uso da estratégia *Object-First*, Xinogalos, Satratzemi e Dagdilelis (2007) propõem a remodelagem de um curso para o ensino do modelo orientado a objetos usando a ferramenta BlueJ. O objetivo do curso é para que os alunos sejam capazes de usar conceitos básicos da orientação a objetos. Mais especificamente para:

- Compreender e utilizar as bibliotecas básicas de classe.
- Analisar e estender as classes existentes.
- Projetar aplicações simples usando a técnica orientada a objetos e programação.

Usando a ferramenta Alice e a estratégia *Object-First*, Dwarika e De Villiers (2015) realizaram um experimento e concluíram que 69% dos participantes sentiram que tinham uma boa compreensão dos objetos e que seria mais fácil aprender orientação a objetos durante o primeiro ano de estudo, para depois aprender as estruturas convencionais de uma determinada linguagem de programação, enquanto 27% discordam. Uma elevada porcentagem de alunos, 91%, afirmou que Alice ajudou a

compreender os princípios de orientação a objetos. Em relação a harmonia entre o sistema e o mundo real, 35% dos alunos afirmaram que Alice melhorou a sua compreensão dos conceitos de orientação a objetos, tais como métodos, herança, propriedades, parâmetros, classes, objetos, instanciação e polimorfismo.

O Quadro 2 apresenta as ferramentas indicadas para cada dificuldade apresentada, no *survey* realizado por Henrique e Rebouças (2015).

QUADRO 2 - FERRAMENTAS INDICADAS PARA AS DIFICULDADES APRESENTADAS DO PARADIGMA ORIENTADO A OBJETOS

Conteúdo	Percentual de alunos que consideram esse conteúdo difícil	Ferramenta Selecionada
Exceções	51/74 (68,9%)	-
Classes Abstratas	35/74 (47,3%)	BlueJ
Polimorfismo	23/74 (43,2%)	BlueJ, Object Editor, Páginas WEB e Raptor
Interface	23/74 (31,1%)	Páginas WEB e Raptor
Construtores e Instanciação	18/74 (24,3%)	Object Editor e BlueJ
Herança	12/74 (16,21%)	BlueJ e Raptor
Criação e Invocação de Métodos	10/74 (13,51%)	BlueJ, Jeliot, Object Editor, Alice, ivProg e Javatool
Objetos e Classes	5/74 (6,75%)	Greenfoot, BlueJ, AFM, CRC-Cards, Smaltalk Card, ivProg, Alice, PROGTUTOR, OntoRepProv e Javatool

FONTE: Adaptado de Henrique e Rebouças, 2015.

O estudo realizado por Yan (2009) foi proposto com o objetivo de adotar uma abordagem baseada em GUI (Graphical User Interface) para ensinar conceitos de orientação a objetos. Nesta abordagem, a visão é que antes de deixar os alunos criarem um aplicativo simples, fazer uso de um jogo é importante para garantir que os alunos realmente entendam os conceitos básicos de orientação a objetos e consigam pensar na maneira orientada a objetos. Nesse sentido, o autor adota o uso de um

ambiente de software de apoio chamado Greenfoot. O sistema Greenfoot é um framework e um ambiente para criar aplicativos de simulação interativos em um plano bidimensional. Greenfoot permite a aplicação e interação com objetos no contexto de cenários. Uma vez que os objetos são criados no sistema, eles podem ser colocados de forma interativa em um mundo Greenfoot, e os alunos podem jogar diretamente com esses objetos, invocando seus métodos. A ferramenta também possui uma IDE (|ntegrated Development Environment), ou ambiente de desenvolvimento integrado. Contém um editor, um compilador e um depurador. Segundo o autor, a abordagem proporciona uma atmosfera positiva e favorável para que os alunos possam aprender os princípios da programação orientada a objetos, mantendo-os interessados e reforçando esses princípios de programação.

Fazer uso de vários ambientes de aprendizagem é o que propõe Kuljis (2004). Numa abordagem chamada WEBL (Web enhanced BlueJ learning), o autor conduz o processo de ensino-aprendizagem do modelo orientado a objetos usando WebCT para facilitar o progresso dos alunos em aprender princípios orientados a objetos com BlueJ.

O aumento na confiança do aluno, conforto e satisfação, foram benefícios observados por Dorn e Sanders (2003), ao usar a ferramenta pedagógica Jeroo para o ensino dos conceitos de orientação a objetos. A linguagem Jeroo foi projetada para se concentrar em três itens principais. Objetos, métodos e estruturas de controle. A classe Jeroo é a única que está disponível no ambiente. Não há tipos de dados e não há variáveis que não sejam referências a objetos Jeroo. No entanto, existem oito constantes predefinidas. A classe Jeroo tem vários métodos pré-definidos que podem ser divididos em duas categorias:

- Métodos de sensores.
- Métodos de ação, que permitem que um objeto Jeroo analise e possa interagir com o seu entorno imediato.

Os métodos de sensores são essencialmente métodos booleanos que fornecem informações sobre imediações. O aluno pode definir métodos de ação adicionais para estender o comportamento do Jeroos. Os métodos definidos pelo usuário não possuem argumentos, mas eles podem chamar outros métodos. O módulo de tempo de execução, que é acessado a partir de menu ou um botão na barra de ferramentas,

fornece tanto tradução quanto execução visuais de um programa Jeroo. Este módulo ilustra a ligação entre o código-fonte e comportamento do Jeroo. Um programa pode ser executado por passos ou continuamente em uma das cinco velocidades. Em qualquer dos modos, o código fonte é destacado e informa como o programa está sendo executado, e as ações dos Jeroos são animados simultaneamente. A interface do usuário tem uma única janela na qual tudo é sempre visível. O código-fonte realçado, as animações e um painel de status atualizado, compõem um rico ambiente de ensino e aprendizagem.

Os conceitos de orientação a objetos são usados em quase todas as aplicações, como: Reserva ferroviária, Reserva aérea, Sistemas bancários, Gestão hospitalar, Gestão de biblioteca, Sistema de telefonia, etc. Em uma disciplina que aborda a orientação a objetos, tudo é um objeto, como números, arrays, registros, campos, arquivos, formulários de fatura etc. Normalmente, os alunos sentem dificuldades para reconhecer os objetos do mundo real. Analisar o papel do objeto e mapear objetos do mundo real são questões importantes no desenvolvimento de software. Para ensinar a disciplina de orientação a objetos para alunos de pós-graduação, o estudo realizado por Kanakaraddi, Naragund e Chikaraddi (2013) fez uso das seguintes ferramentas pedagógicas: Demonstração de ferramentas de código aberto e apresentação de artigos e atividades em laboratório utilizando a ferramenta Rational Rose. Ferramentas pedagógicas são utilizadas para ensinar de forma eficaz.

Para fazer face às dificuldades persistentes impostas no processo ensino-aprendizagem do modelo orientado a objetos, Yulia e Adipranata (2010) propõem a abordagem da aprendizagem cooperativa baseada em game design com ambiente de programação visual para ensinar orientação a objetos. Segundo os autores, é mais fácil ensinar orientação a objetos usando game design, porque em um jogo de computador tudo é um objeto; os monstros, paredes, moedas, bônus, armas e munições. Pensar na criação de jogos, significa pensar sobre objetos e como eles interagem entre si. Assim, o criador do jogo raciocina, naturalmente, em torno de objetos.

No estudo realizado por Satratzemi, Xinogalos e Dagdilelis (2003), os autores propõem um novo ambiente de programação integrada, o objectKarel, para o ensino do paradigma de programação a objetos. O objectKarel usa a metáfora de um mundo de

robôs. Os atores do micromundo são robôs (objetos) que são atribuídos a várias tarefas em um mundo que consiste em: cruzar ruas horizontais e avenidas verticais formando intervalos de um bloco, paredes entre as ruas ou avenidas, usadas para representar obstáculos (barreiras, montanhas, labirintos etc), e apitos (pequenos cones de plástico que emitem um sinal sonoro com baixo ruído) colocados nas esquinas. Os alunos escrevem programas que instruem os robôs a executar suas tarefas. As principais características de ensino de objectKarel, são os seguintes:

- Um tipo especial de editor.
- Execução e animação de programas.
- Detecção de erros em tempo de execução, com mensagens de erro compreensíveis.
- e-Lessons. Segundo os autores, as conclusões mais importantes extraídas a partir da análise dos dados são: Os alunos tornaram-se familiarizados com o ambiente de programação e encontraram soluções interessantes para os problemas a serem resolvidos.

O estudo dos programas produzidos pelos alunos revelou que, em geral, eles não encontram dificuldades particulares com os conceitos ensinados. No entanto, muitos tiveram dificuldades em diferenciar os conceitos de polimorfismo e sobrescrita de métodos. Em geral, o uso piloto do ambiente de programação objectKarel mostrou que é possível ensinar programação orientada a objetos num curto período de tempo, com ambientes educacionais especialmente concebidos. Normalmente essa tarefa é mais complexa com os ambientes profissionais de programação.

A estratégia adotada por Gálvez, Guzmán e Conejo (2009) consiste em fornecer aos alunos (para além das aulas presenciais) uma ferramenta de aprendizagem chamada OOPS e acesso a um sistema de testes chamado SIETTE, que é um sistema de avaliação baseado na web, no qual os alunos podem fazer testes e os professores podem corrigi-los. OOPS é um ambiente de resolução de problemas em que os alunos podem resolver exercícios de orientação a objetos. O sistema diagnostica o nível de conhecimento do aluno, mas também gera um feedback e dicas para ajuda-los a entender e superar os seus equívocos.

Os jogos são fortemente considerados e indicados como sendo grandes facilitadores para a aprendizagem baseada em jogos, pois tornaram-se uma cultura. A razão pela qual os jogos são utilizados como meio de ensino, no processo de ensino-aprendizagem, é porque os alunos possuem grande facilidade e sentem-se a vontade em usar esse tipo de ferramenta. Portanto, é essencial apresentar aos alunos, uma boa ferramenta, afim de atraí-los, em especial, os novatos na aprendizagem dos conceitos orientados a objetos. A singularidade de uma abordagem baseada em jogos, é o envolvimento e a emoção da realização de um jogo. Esse fator torna os jogos como ferramentas adequadas para atrair os alunos e deixá-los mais interessados em aprender e compreender os conceitos de orientação a objetos. Nesse sentido, Rais et al. (2011) investigam aspectos para apoiar a compreensão dos conceitos de orientação a objetos através de jogos. Os autores selecionaram um grupo de estudantes para participar de um experimento com o objetivo de explorar os jogos escolhidos e para responder a um conjunto de perguntas. O experimento consiste em três partes:

- Questionário.
- Usabilidade.
- Compreensão dos conceitos de orientação a objetos.

A terceira seção do estudo abrange a compreensão dos conceitos de orientação a objetos, examinando qual o entendimento dos alunos sobre conceitos orientados a objeto. As perguntas testam a compreensão dos alunos depois de explorar os métodos de aprendizagem. As competências orientadas a objetos e que mapeiam as perguntas em cada método de aprendizagem são:

- Características gerais da orientação a objetos.
- Objetos.
- Objetos, classes e métodos.
- Herança.
- Polimorfismo.
- Encapsulamento.

Cada questão tem a sua própria pontuação baseada nas respostas dos alunos e a pontuação mostra o nível de compreensão dos alunos em cada conceito. Por fim, o próximo passo é a identificação dos pontos fracos de cada conceito orientado a objetos.

Os resultados mostram que os alunos ainda não compreendem completamente os conceitos de herança, abstração e polimorfismo.

2.6 DIFICULDADES NO PROCESSO DE ENSINO-APRENDIZAGEM DO MODELO ORIENTADO A OBJETOS

Aprender orientação a objetos é uma tarefa muito difícil para alunos iniciantes. Vários estudos têm sido realizados com o objetivo de verificar as dificuldades e equívocos dos alunos, quando são ensinados os conceitos de orientação a objetos.

Na pesquisa realizada por Xinogalos (2015), são apresentados os resultados de estudos que investigam as dificuldades dos conceitos mais fundamentais de orientação a objetos, como mostra Quadro 3. Os seguintes elementos foram observados:

- Combinação de objetos e classes: Os alunos acham difícil distinguir classes e suas instâncias, ou seja, objetos.
- Combinação de Objetos e Variáveis: Os objetos são vistos como meros invólucros para variáveis.
- Combinação de Objetos e Registros: Os alunos enxergam todos os objetos apenas como registros de banco de dados.
- Combinação de Classes e Coleções: Uma classe é tratada como uma coleção de objetos, ao invés de um modelo para criar objetos.
- Classes e Objetos como conjunto e subconjunto: Os alunos tendem a confundir a relação entre uma classe e um objeto com a de um conjunto e um subconjunto, que em orientação a objetos é de fato um relacionamento entre uma superclasse e uma subclasse.
- Classes e objetos como todo parte: Os alunos confundem a relação classe-objeto com a relação entre o todo e suas partes.
- Múltiplas classes: Escrever programas que incluem mais de uma classe é difícil para os alunos.
- Classes compostas: Os alunos demonstram dificuldades em definir uma classe que contém atributos de outras classes.
- Problemas com modelagem: Os alunos não conseguem perceber que modelos de classe representam algum domínio do mundo real.

- Visão Estática do Objeto: Dificuldade dos alunos em compreender a natureza estática de uma classe, da natureza dinâmica dos objetos.

QUADRO 3 - DIFICULDADES DOS CONCEITOS MAIS FUNDAMENTAIS DE ORIENTAÇÃO A OBJETOS

	Holland et al. [1997]	Ragonis e Ben-Ari [2005a]	Eckerdal e Thuné [2006]	Thomasson et al. [2006]	Teif e Hazzan [2006]	Sanders e Thomas [2007]	Sanders et al. [2008]
Combinação de Objetos e Classes	SIM	SIM		NÃO		SIM	SIM
Combinação de Objetos e Variáveis	SIM			NÃO		NÃO	
Combinação de Objetos e Registros	SIM					NÃO	NÃO
Combinação de Classes e Coleções		SIM		SIM	SIM	SIM	
Classes e Objetos como conjunto e subconjunto					SIM		
Classes e objetos como todo parte					SIM		
Classes compostas		SIM					
Problemas com modelagem			SIM	SIM		SIM	SIM
Visão estática do objeto		SIM					SIM

FONTE: Adaptado de Xinogalos, 2015.

O estudo realizado por Hosanee e Panchoo (2015), afirma que conceitos como polimorfismo, herança e associação são considerados complexos por alunos iniciantes. O conceito de associação é visto como sendo mais difícil de aprender, em comparação à herança, e menos difícil em comparação ao polimorfismo, que é visto como sendo um tema muito difícil de ser aprendido. Para Thomasson, Ratcliffe e Thomas (2006), ao criar Diagramas de Classe, as falhas mais comuns são as classes não referenciadas.

Isso aponta para o não entendimento de como as classes se associam em um diagrama. Na pesquisa realizada por Kayama et al. (2014), foram identificados os seguintes problemas na criação de modelos orientados a objetos:

- Dificuldades no entendimento de multiplicidade (erro relacionado a associação).
- O uso do mesmo atributo em várias classes (erro relacionado a atributos).
- Descrição do método ou do comportamento da classe como um atributo (erro relacionado a atributos).
- Baixo nível de conhecimento dos atributos adequados e relações (erro relacionado a atributos e associações).

Em um experimento realizado por Dwarika e De Villiers (2015), 25% dos alunos expressaram dificuldades, e até mesmo incapacidades, na resolução de problemas e aplicação de conceitos de orientação a objetos. Além disso, 20% não compreendem instanciação (ou seja, a criação de uma instância de um objeto). Ainda, 13% relataram dificuldades em compreender a lógica de métodos e 11% tiveram dificuldades com a herança. O *survey* realizado por Henrique e Rebouças (2015) aponta que os alunos consideram os seguintes conteúdos como sendo difíceis de aprender:

- Exceções – 68,9%.
- Classes abstratas: 47,3%.
- Polimorfismo: 43,2%.
- Interface: 31,1%.
- Construtores e instanciação: 24,3%
- Herança: 16,21%.
- Criação e invocação de métodos: 13,51%.
- Objetos e classes: 6,75%.

Para Fournier (2005), os alunos demonstram grandes dificuldades em aprender os conceitos de orientação a objetos ao mesmo tempo que aprendem uma determinada linguagem de programação. Yulia e Adipranata (2010), afirmam que os alunos enfrentam problemas na transição do paradigma procedural/estruturado para o paradigma orientado a objetos.

2.7 CONSIDERAÇÕES

Neste capitulo foram apresentados os principais conceitos tratados neste trabalho e que justificam a pesquisa: Ensino-Aprendizagem e Orientação a Objetos. Além disso, foram apresentados trabalhos sobre o processo de ensino-aprendizagem do paradigma orientado a objetos, que foram selecionados por meio de uma revisão sistemática da literatura. Também foram apresentados trabalhos encontrados na literatura que expõem as dificuldades encontradas no processo de ensino-aprendizagem do modelo orientado a objetos.

Nota-se que são utilizadas várias abordagens e estratégias de ensino para minimizar as dificuldades dos alunos no aprendizado do modelo orientado a objetos. A quantidade de estudos nesta área é considerável, mas poucos conseguem efetivamente apontar como se deve proceder para melhorar o processo de ensino-aprendizagem. Não foram identificados trabalhos que utilizam ferramentas computacionais que permitem analisar o processo de ensino-aprendizagem do modelo orientado a objetos, durante o uso da ferramenta. Portanto, como já mencionado na seção 1.1 deste trabalho, o objetivo geral é conceber e avaliar um método que permita avaliar os resultados do processo de ensino-aprendizagem do modelo orientado a objetos, durante a criação de Diagramas de Classe, com o uso de uma ferramenta computacional.

3 MATERIAIS E MÉTODOS

Neste capitulo serão apresentados os procedimentos realizados no desenvolvimento desta pesquisa. Com o objetivo de investigar trabalhos que auxiliam no processo de ensino-aprendizagem para o modelo orientado a objetos, foi realizada uma revisão sistemática da literatura, seguindo o processo de condução de revisões sistemáticas definido por Kitchenham (2004). Com a condução da revisão sistemática, pretendeu-se identificar materiais relevantes sobre o processo de ensino-aprendizagem para o modelo orientado a objetos. A seguir serão apresentados os componentes do protocolo desenvolvido.

3.1 QUESTÕES DE PESQUISA

Ao realizar qualquer revisão sistemática da literatura, é necessária a definição de questões de pesquisa para dirigir a metodologia de pesquisa como um todo Kitchenham (2004).

As questões de pesquisa que foram respondidas na revisão sistemática são as seguintes:

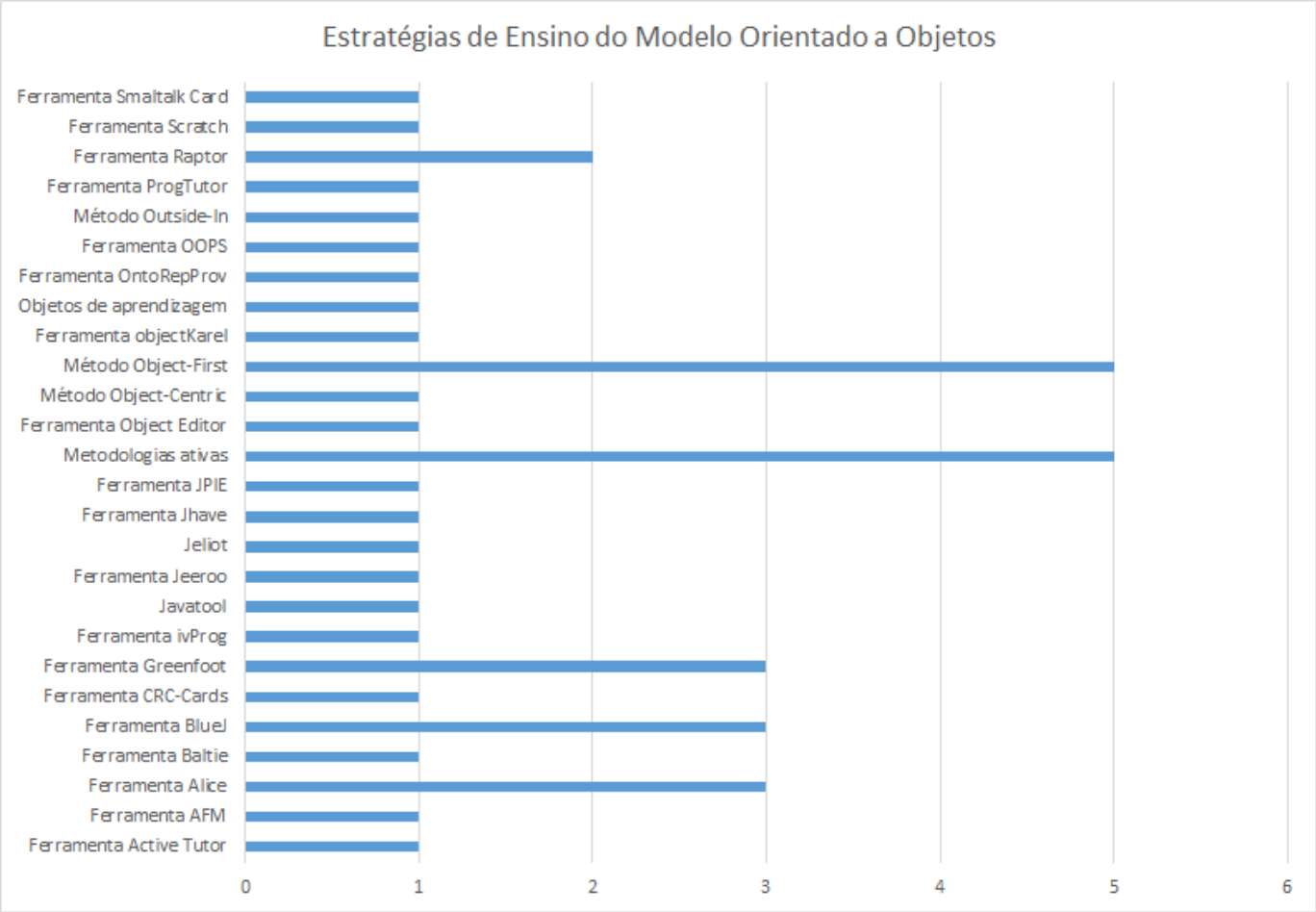
- *QP1:* Que estratégias são utilizadas no processo de ensino-aprendizagem do modelo orientado a objetos?
- *QP2:* Há ferramentas computacionais que analisam o processo de ensino-aprendizagem do modelo orientado a objetos, durante o uso da ferramenta?
- *QP3:* Existe um padrão de categorias para classificar os estudos do processo de ensino-aprendizagem do modelo orientado a objetos?
- *QP4:* Quais as principais dificuldades encontradas no processo de ensino-aprendizagem do modelo orientado a objetos?

3.1.1 Estratégias no processo de ensino-aprendizagem do modelo orientado a objetos

Aprender orientação a objetos é uma tarefa muito difícil para alunos iniciantes. Muitos trabalhos têm apresentado diversas práticas e técnicas de ensino-aprendizagem para melhorar o ensino do modelo orientado a objetos. Nas seções 2.4 e 2.5 deste trabalho, são apresentados os principais resultados encontrados na revisão sistemática, respondendo a primeira pergunta de pesquisa, que faz o seguinte questionamento:

QP1: Que estratégias são utilizadas no processo de ensino-aprendizagem do modelo orientado a objetos? Nota-se que nos trabalhos apresentados são utilizadas várias estratégias, como mostra a Figura 2, dentro do proposto que é verificar a existência de abordagens pedagógicas e ferramentas computacionais que contribuem no processo de ensino-aprendizagem do modelo orientado a objetos.

FIGURA 2 – ESTRATÉGIAS DE ENSINO DO MODELO ORIENTADO A OBJETOS



FONTE: O autor (2017).

3.1.2 Ferramentas computacionais que analisam o processo de ensino-aprendizagem do modelo orientado a objetos, durante o uso da ferramenta

Na seção 2.5 deste trabalho, são apresentados os principais resultados de autores que utilizam ferramentas computacionais para auxiliar o processo de ensino-aprendizagem do modelo orientado a objetos. Com a revisão sistemática, não foram identificados trabalhos que utilizam ferramentas computacionais que permitem analisar o processo de ensino-aprendizagem do modelo orientado a objetos, durante o uso da ferramenta, respondendo a segunda pergunta de pesquisa, que faz o seguinte questionamento: *QP2:* Há ferramentas computacionais que analisam o processo de ensino-aprendizagem do modelo orientado a objetos, durante o uso da ferramenta?

3.1.3 Categorias

A categorização é uma operação de classificação de elementos constituídos de um conjunto, por diferenciação seguida de um reagrupamento baseado em analogias, a partir de critérios definidos (Franco 2007). Classificar elementos em categorias impõe a investigação do que cada um deles tem em comum com outros. O que vai permitir o seu agrupamento é a parte comum existente entre eles (Bardin 2007).

No estudo realizado por Boticki, Katic e Martin (2013), os autores afirmam que as pesquisas existentes para auxiliar os alunos a aprenderem programação orientada a objetos podem ser classificadas nas seguintes categorias:

- Abordagens que focam a concepção e projeto dos cursos.
- Abordagens metodológicas.
- Ferramentas para ambientes de aprendizagem.
- Estudos exploratórios para atrair estudantes para a programação orientada a objetos.

Em outro estudo, realizado por Pears et al. (2007), que contempla um survey da literatura sobre o ensino introdutório de programação, as categorias definidas são:

- Currículos dos cursos.
- Pedagogia.
- Escolha da linguagem de programação.
- Ferramentas para o ensino.

Pode-se notar que não há uma classificação padrão de categorização dos estudos relacionados ao processo de ensino-aprendizagem do modelo orientado a objetos, respondendo a terceira pergunta de pesquisa, que faz o seguinte questionamento: *QP3:* Existe um padrão de categorias para classificar os estudos do processo de ensino-aprendizagem do modelo orientado a objetos? Assim, os estudos encontrados nesta revisão sistemática foram classificados nas seguintes categorias:

- *Abordagens pedagógicas:* Nesta categoria, serão agrupados estudos que utilizam metodologias e práticas pedagógicas que apoiam o ensino-aprendizagem do modelo orientado a objetos.
- *Ferramentas de apoio ao ensino:* Nesta categoria serão agrupados estudos que utilizam softwares de apoio ao ensino para o modelo orientado a objetos. De modo geral, são utilizados ambientes de aprendizagem, ensino do modelo orientado a objetos baseado em jogos e softwares que facilitam o aprendizado do modelo orientado a objetos.

3.1.4 Dificuldades no processo de ensino-aprendizagem do modelo orientado a objetos

Na seção 2.6 deste trabalho, são apresentados estudos que relatam as principais dificuldades no processo de ensino aprendizagem do modelo orientado a objetos, respondendo a quarta pergunta de pesquisa, que faz o seguinte questionamento: *QP4:* Quais as principais dificuldades encontradas no processo de ensino-aprendizagem do modelo orientado a objetos?

3.2 ESTRATÉGIAS DE PESQUISA DE ARTIGOS DA LITERATURA

Esta seção descreve onde e como foram realizadas as pesquisas de artigos da literatura, bem como as palavras-chave que geraram as *strings* de busca.

3.2.1 Escopo da pesquisa

Para as pesquisas de artigos da literatura foram usadas bases de dados eletrônicas, máquinas de busca eletrônica e bibliotecas digitais. Algumas bases de dados de artigos podem influenciar o conjunto de artigos encontrados por beneficiar um formato, um método de estudo ou uma subárea de pesquisa. Para diminuir esta possível influência, quatro bases de dados foram selecionadas:

- IEEE Xplore
- ACM Digital Library
- Springer Link
- ScienceDirect

3.2.2 Palavras-chave

Para as palavras-chave usadas nas pesquisas de artigos da literatura nas bases de dados indicadas na subseção 3.2.1 deste trabalho, foram utilizados apenas termos em inglês, por ser o idioma utilizado nas bases indicadas. São elas:

- Learning
- Teaching
- Education
- Object-Oriented
- Diagnosis

3.2.3 Strings de busca

As *strings* de busca foram geradas a partir da combinação das palavras-chave definidas na subseção 3.2.2 deste trabalho. Após realizar alguns testes com as *strings*, algumas bases de dados limitaram muito o resultado de busca. Portanto, as composições das *strings* de busca são diferentes para as bases de dados indicadas,

esperando-se obter um melhor resultado. A seguir, são relacionadas as *strings* de busca nas respectivas bases de dados indicadas:

- *IEEE Xplore:* ((((learning) AND teaching) AND education) AND object-oriented).
- *ACM Digital Library:* (((((learning) AND teaching) AND education) AND object-oriented) AND diagnosis).
- *Springer Link:* learning AND teaching AND education.
- *ScienceDirect:* (((((learning) AND teaching) AND education) AND object-oriented) AND diagnosis).

3.2.4 Condução da revisão sistemática

Após a definição das palavras-chave e composição das strings de busca, as mesmas foram executadas nas suas respectivas bases de dados. As subseções 3.2.5, 3.2.6, 3.2.7 e 3.2.8 apresentam como foram conduzidas as pesquisas das strings de busca nas bases de dados indicadas na subseção 3.2.2 deste trabalho.

3.2.5 IEEE Xplore

Para a pesquisa na base de dados da IEEE Xplore, foi utilizado o mecanismo de busca avançado. Aqui, foi excluída a palavra-chave *diagnosis*, por limitar muito o resultado esperado. O período de busca foi estimado em 16 anos, ou seja, de Janeiro de 2000 a Janeiro de 2016. A base de dados da IEEE Xplore está disponível em http://ieeexplore.ieee.org/Xplore/home.jsp.

3.2.6 ACM Digital Library

Para a pesquisa na base de dados da ACM Digital Library, foi utilizada uma string de busca baseada na string gerada pela IEEE Xplore, acrescentada da palavra-chave diagnosis. O período de busca foi estimado em 16 anos, ou seja, de Janeiro de 2000 a Janeiro de 2016. A base de dados da ACM Digital Library está disponível em http://dl.acm.org/.

3.2.7 Springer Link

Para a pesquisa na base de dados da Springer Link, foi utilizado o mecanismo de busca avançado. Para um melhor resultado dos artigos retornados pela busca, foi definido que o termo object-oriented estivesse no título do artigo. O período de busca foi estimado em 16 anos, ou seja, de Janeiro de 2000 a Janeiro de 2016. A base de dados da Springer Link está disponível em https://link.springer.com/.

3.2.8 ScienceDirect

Para a pesquisa na base de dados da ScienceDirect, foi utilizado o mecanismo de busca avançado. O período de busca foi estimado em 16 anos, ou seja, de Janeiro 2000 a Janeiro de 2016. A base de dados da ScienceDirect está disponível em http://www.sciencedirect.com/.

3.2.9 Critérios de seleção

Os critérios de seleção indicam os estudos que devem ser incluídos e excluídos no processo da revisão sistemática. O Quadro 4 apresenta os critérios de inclusão e exclusão para selecionar os trabalhos.

QUADRO 4 –CRITÉRIOS DE INCLUSÃO E EXCLUSÃO DE ARTIGOS

Critérios de Inclusão	Critérios de Exclusão
Os trabalhos devem estar no contexto educacional.	Os trabalhos não estão no contexto educacional.
Os trabalhos devem conter, no título ou resumo, alguma relação com o processo de ensino-aprendizagem do modelo orientado a objetos.	Os trabalhos não tratam sobre o ensino do modelo orientado a objetos.
Os trabalhos devem relatar o uso de ferramentas de apoio no processo de ensino-aprendizagem do modelo orientado a objetos.	
Os trabalhos devem relatar o uso de metodologias e práticas pedagógicas no processo de ensino-aprendizagem do modelo orientado a objetos.	
O texto completo deve estar disponível.	

FONTE: O autor (2017).

3.2.10 Extração de dados e classificação

Considerando os critérios de inclusão e exclusão dos estudos, definidos na subseção 3.2.9 deste trabalho, a seleção dos trabalhos foi conduzida pelas seguintes estratégias:

- Leitura do título.
- Leitura do resumo.
- Leitura completa dos trabalhos.
- Extração dos dados.

Ao ser selecionado, os dados dos artigos foram extraídos e organizados em uma planilha contendo: título, referência bibliográfica, resumo, categoria, ano e base de

dados. Durante a fase de leitura completa dos trabalhos, alguns foram retirados da revisão sistemática, obedecendo os critérios de inclusão e exclusão. Por fim, os artigos foram classificados nas suas respectivas categorias, de acordo com os critérios de categorização descritos na subseção 3.1.3 deste trabalho.

3.2.11 Resultados obtidos

Nesta seção são apresentados os resultados obtidos com o processo da revisão sistemática. O Quadro 5 apresenta o resultado quantitativo da busca. A coluna “Resultados” da tabela mostra os resultados retornados pelos mecanismos de busca das bases de dados selecionadas. A coluna “Primeira seleção”, apresenta os resultados após a leitura do título e a leitura do resumo, obedecendo os critérios de inclusão e exclusão. A coluna “Segunda seleção” apresenta os resultados finais após a leitura completa dos trabalhos, que foram os artigos considerados para a revisão sistemática.

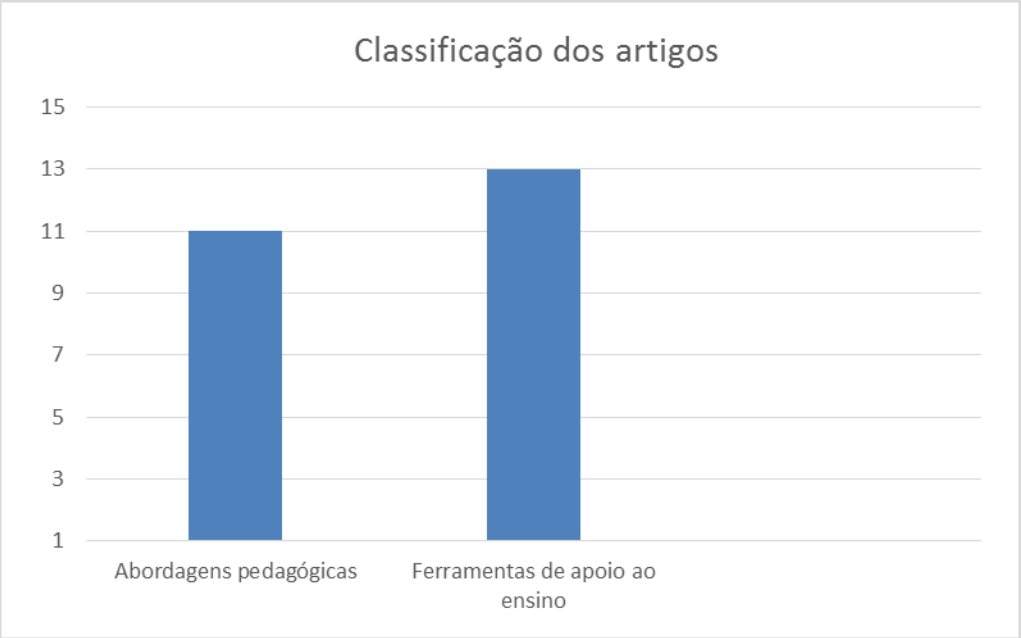
QUADRO 5 –RESULTADO QUANTITATIVO DA BUSCA

Base de Dados	Resultados	Primeira seleção	Segunda seleção
IEEE Xplore	210	37	16
ACM Digital Library	292	12	05
Springer Link	54	06	03
ScienceDirect	63	03	00
	614	58	24

FONTE: O autor (2017).

Os artigos foram classificados em Abordagens pedagógicas e Ferramentas de apoio ao ensino, de acordo com os critérios de categorização descritos na subseção 3.1.3 deste trabalho. A Figura 3 apresenta o gráfico com as classificações dos artigos, após a seleção final.

FIGURA 3 – CLASSIFICAÇÃO DOS ARTIGOS ENCONTRADOS COM A REVISÃO SISTEMÁTICA



FONTE: O autor (2017).

O Quadro 6 apresenta a relação dos todos os artigos selecionados e que foram considerados para a revisão sistemática.

QUADRO 6 –ARTIGOS CONSIDERADOS PARA A REVISÃO SISTEMÁTICA

Título	Base de Dados	Ano	Classificação
An environment for teaching object-oriented programming: objectKarel	Springer Link	2003	Ferramentas de apoio ao ensino
Using Jeroo to introduce object-oriented programming	ACM Digital Library	2003	Ferramentas de apoio ao ensino
Teaching OO concepts-a new approach	IEEE Xplore	2004	Abordagens pedagógicas
Orienting the Teaching of an Introductory Object-Oriented Programming to Meet the Learning Objective	IEEE Xplore	2004	Ferramentas de apoio ao ensino

Título	Base de Dados	Ano	Classificação
ActiveTutor	IEEE Xplore	2005	Ferramentas de apoio ao ensino
Discovery learning of object-oriented modelling with exploration modules in secondary Informatics education	Springer Link	2006	Abordagens pedagógicas
Re-designing an OOP course based on BlueJ	IEEE Xplore	2007	Ferramentas de apoio ao ensino
Teaching of programming languages: An introduction to dynamic learning objects	IEEE Xplore	2009	Abordagens pedagógicas
LCM Exploration and Practicell, In OOP Teaching. Scalable Computing and Communications	IEEE Xplore	2009	Abordagens pedagógicas
The Jigsaw Technique: Experiences Teaching Analysis Class Diagrams	ACM Digital Library	2009	Abordagens pedagógicas
Teaching Object-Oriented Programming with Games	IEEE Xplore	2009	Ferramentas de apoio ao ensino
A blended E-learning experience in a course of object oriented programming fundamentals	ACM Digital Library	2009	Ferramentas de apoio ao ensino
Active learning in Telecommunication Engineering: A case study	IEEE Xplore	2010	Abordagens pedagógicas
Teaching object oriented programming course using cooperative learning method based on game design and visual object oriented environment	IEEE Xplore	2010	Ferramentas de apoio ao ensino
Teaching OOP With Financial Literacy	IEEE Xplore	2011	Abordagens pedagógicas
Game-based Approach and its Feasibility to Support the Learning of	IEEE Xplore	2011	Ferramentas de apoio ao

Object-Oriented Concepts and Programming			ensino
Título	**Base de Dados**	**Ano**	**Classificação**
Object Oriented Design through game development in XNA	IEEE Xplore	2013	Abordagens pedagógicas
Active learning methods for teaching OOAD course	IEEE Xplore	2013	Ferramentas de apoio ao ensino
Teaching design first; interleaved with object-oriented programming in a software engineering course	IEEE Xplore	2014	Abordagens pedagógicas
Learning object-oriented paradigm by playing computer games: concepts first approach	Springer Link	2014	Ferramentas de apoio ao ensino
Using Flipped Classroom, Peer Discussion, and Just-in-Time Teaching to Increase Learning in a Programming Course	IEEE Xplore	2015	Abordagens pedagógicas
Does Outside-In Teaching Improve the Learning of Object-Oriented Programming?	ACM Digital Library	2015	Abordagens pedagógicas
An Enhanced Software Tool to Aid Novices in Learning Object Oriented Programming (OOP)	IEEE Xplore	2015	Ferramentas de apoio ao ensino
Use of the Alice Visual Environment in Teaching and Learning Object-Oriented Programming	ACM Digital Library	2015	Ferramentas de apoio ao ensino

FONTE: O autor (2017).

3.3 FERRAMENTA PARA O APOIO ENSINO-APRENDIZAGEM DO MODELO ORIENTADO A OBJETOS DURANTE A CONSTRUÇÃO DO DIAGRAMA DE CLASSES

Com a revisão sistemática da literatura foi possível identificar vários trabalhos que apoiam o processo de ensino-aprendizagem do modelo orientado a objetos. Porém, não foram identificados trabalhos que utilizam ferramentas computacionais que permitem analisar o processo de ensino-aprendizagem do modelo orientado a objetos, durante o uso da ferramenta. Assim, como já mencionado na seção 2.1, o objetivo geral desta pesquisa é conceber e avaliar um método que permita avaliar os resultados do processo de ensino-aprendizagem do modelo orientado a objetos, durante a criação de Diagramas de Classe, com o uso de uma ferramenta computacional.

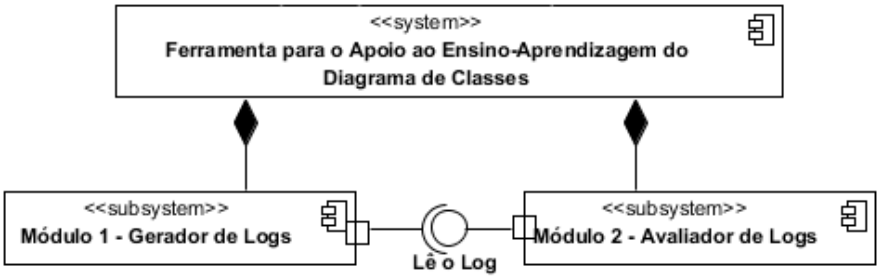
Essa seção descreve o desenvolvimento, a implementação e testes de uma ferramenta para apoiar o processo de ensino-aprendizagem do modelo orientado a objetos, durante a construção de Diagramas de Classe. O principal objetivo da ferramenta é coletar e analisar os dados gerados pelos alunos durante a construção de Diagramas de Classe. Com os dados, o professor consegue identificar e abordar, de maneira individual, os alunos que apresentam dificuldades nos conceitos de orientação a objetos que são necessários para a criação destes diagramas. Com isso, é concebido um método que apoia o processo de ensino e aprendizagem do modelo orientado a objetos.

O objetivo é apresentar a metodologia utilizada para o desenvolvimento da Ferramenta para o Apoio ao Ensino-Aprendizagem do Modelo Orientado a Objetos Durante a Construção do Diagrama de Classes, por meio de experimentos com alunos do curso de graduação em Análise e Desenvolvimento de Sistemas, de uma Universidade. Também é apresentada a ferramenta criada para testar o método desenvolvido.

Primeiramente, foi decidido que seria criada uma ferramenta para coletar informações de todas as ações executadas pelos alunos, durante a construção de Diagramas de Classe. Com isso, também foi decidido como deveria ser feita a coleta dos dados dos exercícios executado pelos alunos, e como seria feita a análise dos mesmos. Optou-se pela divisão da arquitetura da ferramenta em dois módulos, conforme apresentado na Figura 4. O Módulo 1, “Gerador de Logs”, é responsável por

coletar todas as ações que cada aluno executa durante a construção de um Diagrama de Classes. O Módulo 1 é um *plugin* desenvolvido para o Microsoft Visio (versão 2013), que é uma ferramenta que auxilia na criação de diversos diagramas (Okada e Sass 2015). A justificativa para utilizar o Microsoft Visio é que os alunos da universidade, sujeitos do experimento, utilizam essa ferramenta para a criação de Diagramas de Classe. O Microsoft Visio fornece um ambiente de desenvolvimento integrado usando VBA para controlar formas e desenhos, ler ou escrever dados a partir de fontes externas, como um banco de dados. O Módulo 2, “Avaliador de Logs”, foi desenvolvimento no ambiente Eclipse (versão Neon Release 4.6.0), sendo responsável por ler, consolidar e apresentar os dados gerados (*log*) pelo Módulo 1.

FIGURA 4 – ARQUITETURA DA FERRAMENTA PARA O APOIO AO ENSINO-APRENDIZAGEM DO MODELO ORIENTADO A OBJETOS DURANTE A CONSTRUÇÃO DO DIAGRAMA DE CLASSES

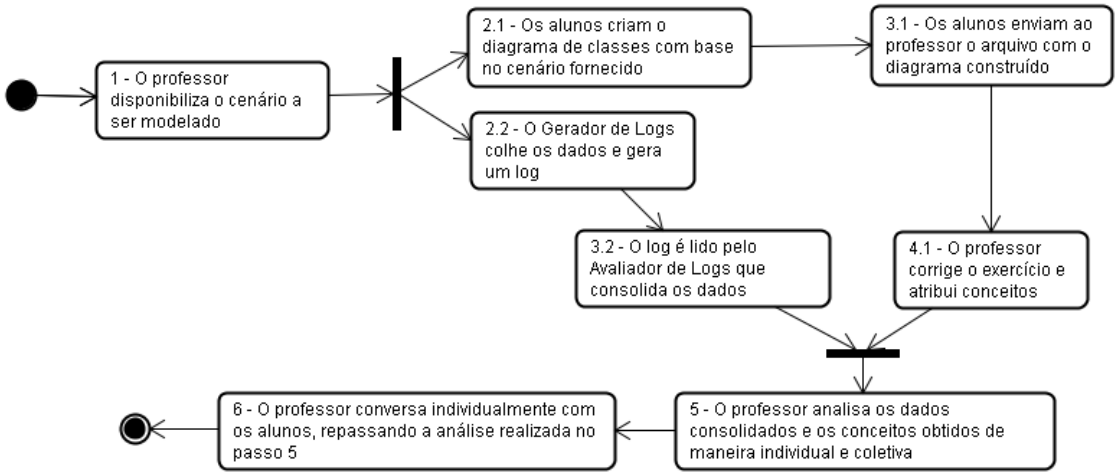


FONTE: O autor (2017).

Foi escolhida a divisão da Ferramenta para o Apoio ao Ensino-Aprendizagem do Modelo Orientado a Objetos Durante a Construção do Diagrama de Classes em dois módulos pois, desta forma, é possível que o Módulo 1 seja substituído por *plugins* para outras ferramentas de desenvolvimento gráfico, como o Umbrello, o Visual Paradigm ou o Astah*, ampliando o campo de aplicação para o Módulo 2.

Após a decisão sobre a arquitetura da ferramenta, foi desenvolvido o Fluxo de Execução dos Experimentos, conforme descrito no diagrama da Figura 5 e apresentado nos parágrafos a seguir.

FIGURA 5 – FLUXO DE EXECUÇÃO DOS EXPERIMENTOS



FONTE: O autor (2017).

No passo 1, apresentado na Figura 5, o professor disponibiliza o cenário a ser modelado pelos alunos, descrito em um texto. Esse cenário deve ser interpretado pelos alunos, que devem criar o Diagrama de Classes no Microsoft Visio, conforme indicado no passo 2.1, da Figura 5.

Enquanto os alunos criam o Diagrama de Classes, o Gerador de Logs desenvolvido para o Visio colhe todas as ações dos alunos em *background*, gerando um *log*, conforme indicado pelo passo 2.2, na Figura 5. Um exemplo parcial de *log* gerado é apresentado no Quadro 7, a partir de alguns fragmentos gerados. No Quadro 7 é possível observar os fragmentos de todas as ações possíveis realizadas pelos alunos no Diagrama de Classes.

QUADRO 7 – INFORMAÇÕES GERADAS NO ARQUIVO DE LOG

Fragmento	Log
1	CREATE CLASS;Pessoa;12/06/2017;22:53:21
2	RENAME CLASS;PessoaFisica;Cliente;12/06/2017;22:58:42
3	REMOVE CLASS;Fornecedor;12/06/2017;23:01:38
4	CREATE ATTRIBUTE;-id: int;12/06/2017;22:53:36;Pessoa
5	RENAME ATTRIBUTE;-cpf: String;-cpfCliente: String;12/06/2017;22:58:50;Cliente

Fragmento	Log
6	REMOVE ATTRIBUTE;-nome: String;12/06/2017;23:03:42;Cliente
7	CREATE OPERATION;+validarCpf(): boolean;12/06/2017;22:57:20;PessoaFisica
8	RENAME OPERATION;+validarCpf(): boolean;+validarCpf(cliente: Cliente): boolean;12/06/2017;23:06:19;Cliente
9	REMOVE OPERATION;+validarCpf(cliente: Cliente): boolean;12/06/2017;23:13:27;Cliente
10	CREATE INHERITANCE;Pessoa->Cliente;12/06/2017;23:14:54;123
11	REMOVE INHERITANCE;12/06/2017;23:16:01;123
12	CREATE ASSOCIATION;Pedido->Cliente;12/06/2017;23:17:04;128
13	REMOVE ASSOCIATION;12/06/2017;23:17:51;128
14	CREATE AGGREGATION;Cliente->Dependente;12/06/2017;23:19:25;141
15	REMOVE AGGREGATION;12/06/2017;23:19:57;141
16	CREATE COMPOSITION;Pedido->ItemPedido;12/06/2017;23:20:59;154
17	REMOVE COMPOSITION;12/06/2017;23:21:35;154

FONTE: O autor (2017).

Ao todo, o Gerador de Logs tem a capacidade de gerar 17 instruções diferentes no arquivo de *log* e cada instrução é associada a uma ação que o aluno executa quando está modelando o Diagrama de Classes. O Quadro 8 apresenta cada uma das 17 instruções e em que momento elas acontecem, durante a construção do Diagrama de Classes.

QUADRO 8 – INSTRUÇÕES GERADAS PELO GERADOR DE LOGS

Instrução	Ação
CREATE CLASS	Instrução gerada sempre que um aluno cria uma nova classe no diagrama de classes.
RENAME CLASS	Instrução gerada sempre que um aluno altera o nome de uma classe no diagrama de classes.
REMOVE CLASS	Instrução gerada sempre que um aluno exclui uma classe do diagrama de classes.
CREATE ATTRIBUTE	Instrução gerada sempre que um aluno cria um novo atributo no diagrama de classes.
RENAME ATTRIBUTE	Instrução gerada sempre que um aluno altera um atributo no diagrama de classes.

Instrução	Ação
REMOVE ATTRIBUTE	Instrução gerada sempre que um aluno exclui um atributo do diagrama de classes.
CREATE OPERATION	Instrução gerada sempre que um aluno cria um novo método no diagrama de classes.
RENAME OPERATION	Instrução gerada sempre que um aluno altera um método no diagrama de classes.
REMOVE OPERATION	Instrução gerada sempre que um aluno exclui um método do diagrama de classes.
CREATE INHERITANCE	Instrução gerada sempre que um aluno cria um relacionamento de herança entre duas classes no diagrama de classes.
REMOVE INHERITANCE	Instrução gerada sempre que um aluno exclui um relacionamento de herança entre duas classes no diagrama de classes.
CREATE ASSOCIATION	Instrução gerada sempre que um aluno cria um relacionamento de associação simples entre duas classes no diagrama de classes.
REMOVE ASSOCIATION	Instrução gerada sempre que um aluno exclui um relacionamento de associação simples entre duas classes no diagrama de classes.
CREATE AGGREGATION	Instrução gerada sempre que um aluno cria um relacionamento de agregação entre duas classes no diagrama de classes.
REMOVE AGGREGATION	Instrução gerada sempre que um aluno exclui um relacionamento de agregação entre duas classes no diagrama de classes.
CREATE COMPOSITION	Instrução gerada sempre que um aluno cria um relacionamento de composição entre duas classes no diagrama de classes.
REMOVE COMPOSITION	Instrução gerada sempre que um aluno exclui um relacionamento de composição entre duas classes no diagrama de classes.

FONTE: O autor (2017).

Como mencionado anteriormente, o algoritmo que gera os *logs,* enquanto os alunos estão modelando o Diagrama de Classes, foi escrito em VBA, que é a linguagem de programação que o Microsoft Visio usa para controlar formas e desenhos. Quando se programa em VBA, utiliza-se os eventos disponíveis pelo ambiente. Os eventos em VBA monitoram as ações realizadas pelos usuários, onde cada ação disparada pelo usuário normalmente está associada a um evento. Para a solução, foram programados quatro eventos:

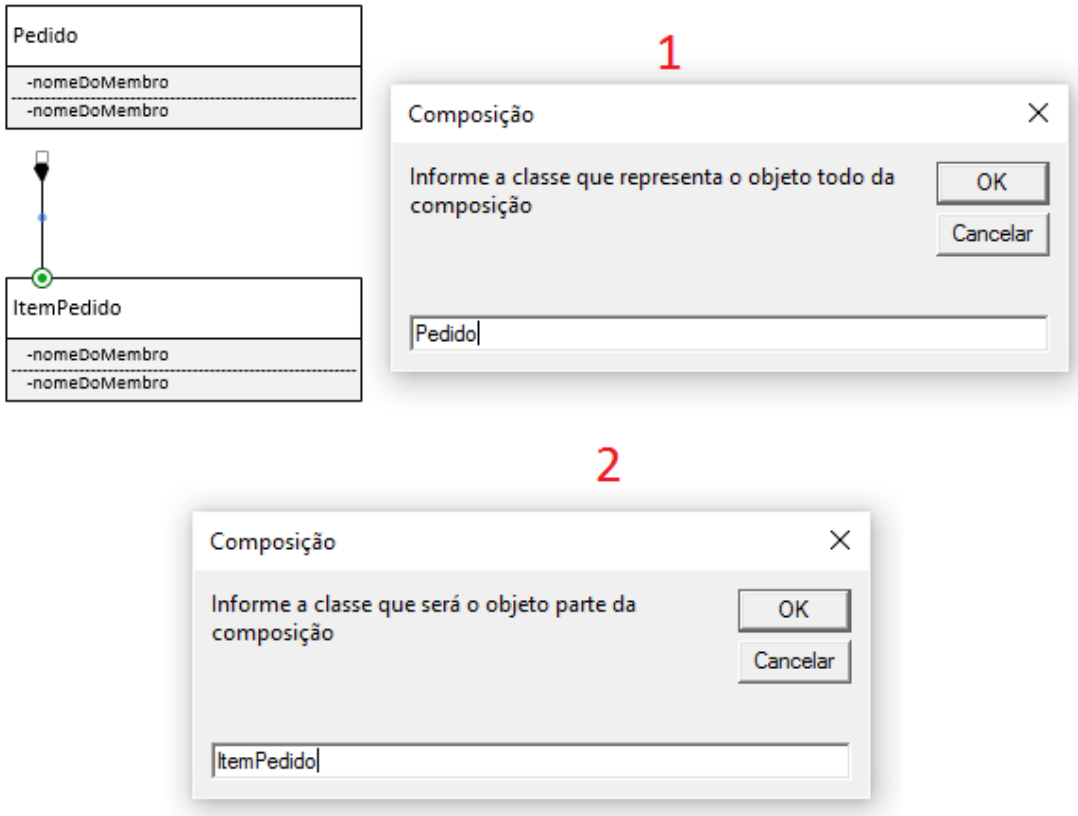
- *Document_ShapeAdded:* Este evento ocorre após uma ou mais formas serem adicionadas a um documento. Neste evento o algoritmo reconhece a forma que foi adicionada ao Diagrama de Classes. As formas reconhecidas e tratadas pelo algoritmo são: classe, membro (que pode ser um atributo ou método), herança, associação simples, agregação e composição.
- *Document_ShapeExitedTextEdit:* Este evento ocorre depois que uma forma não está mais aberta para a edição de texto interativa. Neste evento o algoritmo trata as trocas de nomes sugeridas pelo Visio pelos nomes definidos pelo aluno. Por exemplo, quando o aluno cria uma nova classe no Diagrama de Classes, o Visio sugere que essa classe se chame *Nome da Classe*. Neste momento o aluno edita o nome da classe, fazendo com que o algoritmo entenda que uma nova classe está sendo incorporada ao diagrama. Este evento também irá tratar as edições no Diagrama de Classes, que são quando o aluno altera nomes de atributos, nomes de métodos e atributos para métodos.
- *Document_QueryCancelSelectionDelete:* Este evento ocorre quando uma forma é excluída do Diagrama de Classes. Neste evento o algoritmo trata os elementos que foram excluídos do diagrama.
- *Document_BeforeShapeTextEdit:* Este evento ocorre quando uma forma é editada no Diagrama de Classes. Neste evento o algoritmo preserva o nome anterior do elemento que está sendo alterado no Diagrama de Classes. Por exemplo, se um aluno resolve alterar o nome da classe de Pedidos para Pedido, o evento preserva o nome anterior, ou seja Pedidos, para indicar no *log* o nome antigo e o novo nome da classe alterada.

O Visio reconhece os elementos do Diagrama de Classes como *shapes*. Assim, a classe é um *shape*, os membros (atributos ou métodos) são *shapes* e os relacionamentos também são *shapes*. Desta maneira, o algoritmo deve interpretar quais *shapes* estão sendo incorporados, editados, excluídos e associados no Diagrama de Classes. Inicialmente, quando o aluno cria um membro dentro de uma classe, o Visio

não entende que o membro criado, que pode ser um atributo ou método, pertence a aquela classe. Neste momento o algoritmo deve reconhecer a qual classe pertence um determinado atributo ou método.

Um dos grandes desafios encontrados no desenvolvimento do algoritmo foi o reconhecimento de quais classes são envolvidas em um relacionamento (associação simples, herança, agregação ou composição). Quando o aluno cria um relacionamento no Diagrama de Classes, o Visio não reconhece quais classes estão associadas. O que ele faz é fornecer qual a posição das classes e do relacionamento no diagrama. Neste caso, o algoritmo teria muito trabalho para calcular e descobrir quais classes estão relacionadas. Para facilitar a geração do *log*, quando um relacionamento é criado no Diagrama de Classes, o algoritmo interage com o aluno perguntando quais as classes do relacionamento. A Figura 6 apresenta essa interação, quando um relacionamento de composição é criado.

FIGURA 6 – INTERAÇÃO DO MICROSOFT VISIO COM A AÇÃO EXECUTADA PELO ALUNO



FONTE: O autor (2017).

O mesmo acontece para as criações dos relacionamentos de associação simples, herança e agregação.

Outro problema é interpretar qual o relacionamento que está sendo excluído do Diagrama de Classes. A maneira que o algoritmo trata as exclusões de relacionamentos do Diagrama de Classes é realizado pelo identificador do *shape*, ou simplesmente *id*. Ao criar um relacionamento de associação simples, herança, agregação ou composição no Diagrama de Classes, o algoritmo guarda o *id* do *shape*. Isso pode ser percebido analisando as linhas 10, 11, 12, 13, 14, 15, 16 e 17 do Quadro 7. Pode-se perceber que a última instrução destas linhas é um valor numérico, que é o *id* do *shape*. As linhas 16 e 17, por exemplo, criam e excluem, respectivamente, um relacionamento de composição. Na linha 16 do Quadro 7, o *id* é 154. Este valor foi guardado na inclusão

do relacionamento. Se este relacionamento for excluído do Diagrama de Classes, que é o que acontece na linha 17 do Quadro 7, sabe-se que a exclusão gerada no *log* é para o relacionamento de composição criado na linha 16 do Quadro 7, pois o *id* é o mesmo, ou seja, 154. Os quadros a seguir apresentam, de forma textual, os fluxos e comportamentos para cada uma das instruções que podem ser geradas no *log* pelo Gerador de Logs, de acordo com a ação do aluno. O Quadro 9 apresenta o fluxo da instrução CREATE CLASS.

QUADRO 9 – FLUXO DA INSTRUÇÃO CREATE CLASS

CREATE CLASS			
Ação do aluno	Ação do Visio	Evento disparado pelo Visio	Ação do algoritmo
Insere uma classe no diagrama de classes	Cria um shape de classe, chamado Nome da Classe, no diagrama de classes	Document_ShapeAdded	
Seleciona o nome da classe para renomear	Posiciona o cursor em Nome da Classe para editar	Document_BeforeShapeTextEdit	Recupera o nome atual do shape
Digita o nome da classe e confirma	Muda o nome da classe	Document_ShapeExitedTextEdit	Verifica se o nome anterior do shape é Nome da Classe. Se sim, gera o log com informações da classe criada. Caso contrário realiza outras verificações

FONTE: O autor (2017).

A seguir, no Quadro 10, é apresentado o fluxo da instrução RENAME CLASS.

QUADRO 10 – FLUXO DA INSTRUÇÃO RENAME CLASS

RENAME CLASS			
Ação do aluno	Ação do Visio	Evento disparado pelo Visio	Ação do algoritmo
Seleciona o nome da classe para renomear	Posiciona o cursor no nome da classe a ser renomeada	Document_BeforeShapeTextEdit	Recupera o nome atual do shape
Digita o novo nome para a classe e confirma	Muda o nome da classe	Document_ShapeExitedTextEdit	Verifica se o shape que está sendo alterado é uma classe. Se sim gera o log com informações da classe renomeada. Caso contrário realiza outras verificações

FONTE: O autor (2017).

A seguir, no Quadro 11, é apresentado o fluxo da instrução REMOVE CLASS.

QUADRO 11 – FLUXO DA INSTRUÇÃO REMOVE CLASS

REMOVE CLASS			
Ação do aluno	Ação do Visio	Evento disparado pelo Visio	Ação do algoritmo
Seleciona a classe que será excluída	Deixa o shape de classe selecionado		
Pressiona a tecla delete	Exclui o shape de classe do diagrama de classes	Document_QueryCancelSelectionDelete	Verifica se o shape é uma classe. Se sim, gera o log com informações da classe excluída

FONTE: O autor (2017).

A seguir, no Quadro 12, é apresentado o fluxo da instrução CREATE ATTRIBUTE.

QUADRO 12 – FLUXO DA INSTRUÇÃO CREATE ATTRIBUTE

CREATE ATTRIBUTE			
Ação do aluno	Ação do Visio	Evento disparado pelo Visio	Ação do algoritmo
Insere um membro no diagrama de classes	Cria um shape de membro, chamado -nomeDoMembro, no diagrama de classes	Document_ShapeAdded	
Seleciona o nome do membro para renomear	Posiciona o cursor em -nomeDoMembro para editar	Document_BeforeShapeTextEdit	Recupera o nome atual do shape
Digita o nome do atributo e confirma	Muda o nome do atributo	Document_ShapeExitedTextEdit	Verifica se o nome anterior do shape é -nomeDoMembro. Se sim, gera o log com informações do atributo criado. Caso contrário realiza outras verificações

FONTE: O autor (2017).

A seguir, no Quadro 13, é apresentado o fluxo da instrução RENAME ATTRIBUTE.

QUADRO 13 – FLUXO DA INSTRUÇÃO RENAME ATTRIBUTE

RENAME ATTRIBUTE			
Ação do aluno	Ação do Visio	Evento disparado pelo Visio	Ação do algoritmo
Seleciona o nome do atributo para renomear	Posiciona o cursor no nome do atributo a ser renomeado	Document_BeforeShapeTextEdit	Recupera o nome atual do shape
Digita o novo nome para o atributo e confirma	Muda o nome do atributo	Document_ShapeExitedTextEdit	Verifica se o shape que está sendo alterado é um atributo. Se sim gera o log com informações do atributo renomeado. Caso contrário realiza outras verificações

FONTE: O autor (2017).

A seguir, no Quadro 14, é apresentado o fluxo da instrução REMOVE ATTRIBUTE.

QUADRO 14 – FLUXO DA INSTRUÇÃO REMOVE ATTRIBUTE

REMOVE ATTRIBUTE			
Ação do aluno	Ação do Visio	Evento disparado pelo Visio	Ação do algoritmo
Seleciona o atributo que será excluído	Deixa o shape de membro selecionado		
Pressiona a tecla delete	Exclui o shape de membro do diagrama de classes	Document_QueryCancelSelectionDelete	Verifica se o shape é um membro e se é um atributo. Se sim, gera o log com informações do atributo excluído

FONTE: O autor (2017).

A seguir, no Quadro 15, é apresentado o fluxo da instrução CREATE OPERATION.

QUADRO 15 – FLUXO DA INSTRUÇÃO CREATE OPERATION

CREATE OPERATION			
Ação do aluno	Ação do Visio	Evento disparado pelo Visio	Ação do algoritmo
Insere um membro no diagrama de classes	Cria um shape de membro, chamado -nomeDoMembro, no diagrama de classes	Document_ShapeAdded	
Seleciona o nome do membro para renomear	Posiciona o cursor em -nomeDoMembro para editar	Document_BeforeShapeTextEdit	Recupera o nome atual do shape
Digita o nome do método, seus parâmetros de entrada e seu tipo, e confirma	Muda o nome do método	Document_ShapeExitedTextEdit	Verifica se o nome anterior do shape é -nomeDoMembro. Se sim, verifica se é um método. Se sim, gera o log com informações do método criado. Caso contrário realiza outras verificações

FONTE: O autor (2017).

A seguir, no Quadro 16, é apresentado o fluxo da instrução RENAME OPERATION.

QUADRO 16 – FLUXO DA INSTRUÇÃO RENAME OPERATION

RENAME OPERATION			
Ação do aluno	Ação do Visio	Evento disparado pelo Visio	Ação do algoritmo
Seleciona o nome do método para renomear	Posiciona o cursor no nome do método a ser renomeado	Document_BeforeShapeTextEdit	Recupera o nome atual do shape
Digita as novas características para o método e confirma	Muda o nome do método	Document_ShapeExitedTextEdit	Verifica se o shape que está sendo alterado é um método. Se sim gera o log com informações do método renomeado. Caso contrário realiza outras verificações

FONTE: O autor (2017).

A seguir, no Quadro 17, é apresentado o fluxo da instrução REMOVE OPERATION.

QUADRO 17 – FLUXO DA INSTRUÇÃO REMOVE OPERATION

REMOVE OPERATION			
Ação do aluno	Ação do Visio	Evento disparado pelo Visio	Ação do algoritmo
Seleciona o método que será excluído	Deixa o shape de membro selecionado		
Pressiona a tecla delete	Exclui o shape de membro do diagrama de classes	Document_QueryCancelSelectionDelete	Verifica se o shape é um membro e se é um método. Se sim, gera o log com informações do método excluído

FONTE: O autor (2017).

A seguir, no Quadro 18, é apresentado o fluxo da instrução CREATE INHERITANCE.

QUADRO 18 – FLUXO DA INSTRUÇÃO CREATE INHERITANCE

CREATE INHERITANCE			
Ação do aluno	Ação do Visio	Evento disparado pelo Visio	Ação do algoritmo
Insere um relacionamento de herança no diagrama de classes	Cria um shape de relacionamento de herança no diagrama de classes	Document_ShapeAdded	Verifica qual o shape de relacionamento está sendo incorporado ao documento. Se é um shape de herança, pergunta ao aluno a classe que representa a classe pai da herança
Informa o nome da classe pai no relacionamento de herança			Armazena o valor informado pelo aluno
			Pergunta ao aluno a classe que será a classe filha da herança
Informa o nome da classe filha no relacionamento de herança			Armazena o valor informado pelo aluno
			Gera o log com informações do relacionamento de herança criado

FONTE: O autor (2017).

A seguir, no Quadro 19, é apresentado o fluxo da instrução REMOVE INHERITANCE.

QUADRO 19 – FLUXO DA INSTRUÇÃO REMOVE INHERITANCE

REMOVE INHERITANCE			
Ação do aluno	Ação do Visio	Evento disparado pelo Visio	Ação do algoritmo
Seleciona o relacionamento de herança a ser excluído	Deixa o shape de herança selecionado		
Pressiona a tecla delete	Exclui o shape de herança do diagrama de classes	Document_QueryCancelSelectionDelete	Verificar se o shape que está sendo removido está dentro ou fora de uma classe. Se estiver fora, verifica se é um shape de herança. Se sim, gera o log com informações sobre a herança excluída

FONTE: O autor (2017).

A seguir, no Quadro 20, é apresentado o fluxo da instrução CREATE ASSOCIOATION.

QUADRO 20 – FLUXO DA INSTRUÇÃO CREATE ASSOCIOATION

CREATE ASSOCIOATION			
Ação do aluno	Ação do Visio	Evento disparado pelo Visio	Ação do algoritmo
Insere um relacionamento de associação simples no diagrama de classes	Cria um shape de relacionamento de associação simples no diagrama de classes	Document_ShapeAdded	Verifica qual o shape de relacionamento está sendo incorporado ao documento. Se é um shape de associação simples, pergunta ao aluno a classe que será referenciada pela associação
Informa o nome da classe que será referenciada no relacionamento de associação			Armazena o valor informado pelo aluno
			Pergunta ao aluno a classe que conterá o objeto da associação
Informa o nome da classe que conterá o objeto da associação			Armazena o valor informado pelo aluno
			Gera o log com informações do relacionamento de associação simples criado

FONTE: O autor (2017).

A seguir, no Quadro 21, é apresentado o fluxo da instrução REMOVE ASSOCIOATION.

QUADRO 21 – FLUXO DA INSTRUÇÃO REMOVE ASSOCIOATION

REMOVE ASSOCIOATION			
Ação do aluno	Ação do Visio	Evento disparado pelo Visio	Ação do algoritmo
Seleciona o relacionamento de associação simples a ser excluído	Deixa o shape de associação simples selecionado		
Pressiona a tecla delete	Exclui o shape de associação simples do diagrama de classes	Document_QueryCancelSelectionDelete	Verificar se o shape que está sendo removido está dentro ou fora de uma classe. Se estiver fora, verifica se é um shape de associação simples. Se sim, gera o log com informações sobre a associação simples excluída

FONTE: O autor (2017).

A seguir, no Quadro 22, é apresentado o fluxo da instrução CREATE AGGREGATION.

QUADRO 22 – FLUXO DA INSTRUÇÃO CREATE AGGREGATION

CREATE AGGREGATION			
Ação do aluno	Ação do Visio	Evento disparado pelo Visio	Ação do algoritmo
Insere um relacionamento de agregação no diagrama de classes	Cria um shape de relacionamento de agregação no diagrama de classes	Document_ShapeAdded	Verifica qual o shape de relacionamento está sendo incorporado ao documento. Se é um shape de agregação, pergunta ao aluno a classe que representa o objeto todo da agregação
Informa o nome da a classe que representa o objeto todo da agregação			Armazena o valor informado pelo aluno
			Pergunta ao aluno a classe que será o objeto parte da agregação
Informa o nome da a classe que será o objeto parte da agregação			Armazena o valor informado pelo aluno
			Gera o log com informações do relacionamento de agregação criado

FONTE: O autor (2017).

A seguir, no Quadro 23, é apresentado o fluxo da instrução REMOVE AGGREGATION.

QUADRO 23 – FLUXO DA INSTRUÇÃO REMOVE AGGREGATION

REMOVE AGGREGATION			
Ação do aluno	Ação do Visio	Evento disparado pelo Visio	Ação do algoritmo
Seleciona o relacionamento de agregação a ser excluído	Deixa o shape de agregação selecionado		
Pressiona a tecla delete	Exclui o shape de agregação do diagrama de classes	Document_QueryCancelSelectionDelete	Verificar se o shape que está sendo removido está dentro ou fora de uma classe. Se estiver fora, verifica se é um shape de agregação. Se sim, gera o log com informações sobre a agregação excluída

FONTE: O autor (2017).

A seguir, no Quadro 24, é apresentado o fluxo da instrução CREATE COMPOSITION.

QUADRO 24 – FLUXO DA INSTRUÇÃO CREATE COMPOSITION

CREATE COMPOSITION			
Ação do aluno	Ação do Visio	Evento disparado pelo Visio	Ação do algoritmo
Insere um relacionamento de composição no diagrama de classes	Cria um shape de relacionamento de composição no diagrama de classes	Document_ShapeAdded	Verifica qual o shape de relacionamento está sendo incorporado ao documento. Se é um shape de composição, pergunta ao aluno a classe que representa o objeto todo da composição
Informa o nome da a classe que representa o objeto todo da composição			Armazena o valor informado pelo aluno
			Pergunta ao aluno a classe que será o objeto parte da composição
Informa o nome da a classe que será o objeto parte da composição			Armazena o valor informado pelo aluno
			Gera o log com informações do relacionamento de composição criado

FONTE: O autor (2017).

A seguir, no quadro 25, é apresentado o fluxo da instrução REMOVE COMPOSITION.

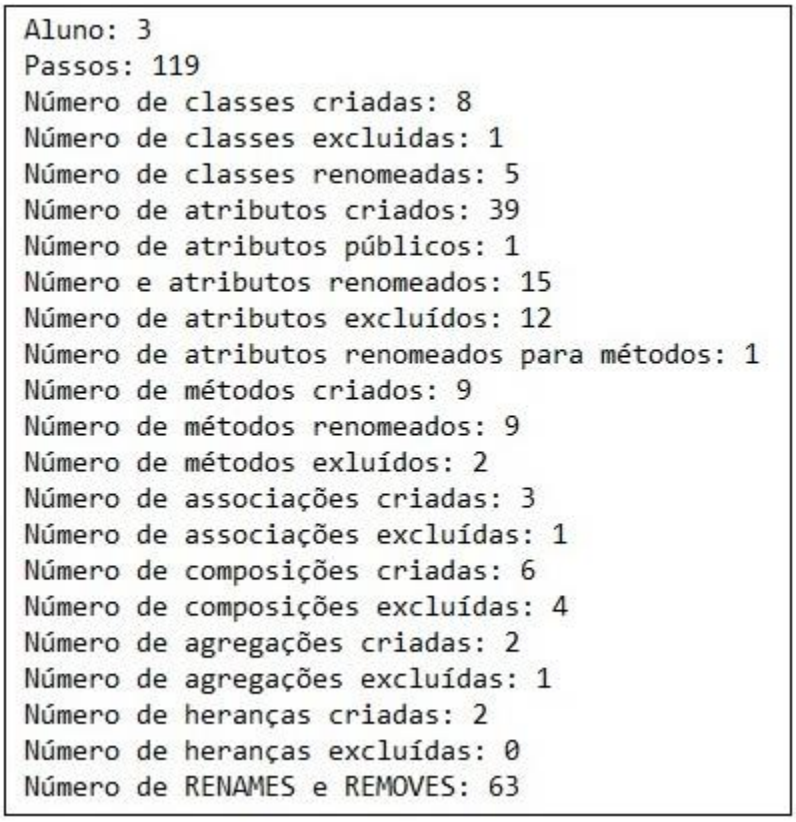
QUADRO 25 – FLUXO DA INSTRUÇÃO REMOVE COMPOSITION

REMOVE COMPOSITION			
Ação do aluno	Ação do Visio	Evento disparado pelo Visio	Ação do algoritmo
Seleciona o relacionamento de composição a ser excluído	Deixa o shape de composição selecionado		
Pressiona a tecla delete	Exclui o shape de composição do diagrama de classes	Document_QueryCancelSelectionDelete	Verificar se o shape que está sendo removido está dentro ou fora de uma classe. Se estiver fora, verifica se é um shape de composição. Se sim, gera o log com informações sobre a composição excluída

FONTE: O autor (2017).

Terminado o exercício, os alunos enviam ao professor o arquivo com o diagrama construído, conforme o passo 3.1 da Figura 5, e o aplicativo Avaliador de Logs (Módulo 2) recebe o *log* gerado para cada aluno pelo Gerador de Logs no Visio. O objetivo do aplicativo escrito em Java é sintetizar o *log* gerado pelo Gerador de Logs implementado no Visio. O *log* está em formato texto, como mostra o quadro 7. Portanto, o Avaliador de Logs deve ler cada instrução gerada no *log* e produzir o resultado da Figura 7.

FIGURA 7 – RESULTADO GERADO PELO AVALIADOR DE LOGS



FONTE: O autor (2017).

Analisando o resultado apresentado pela Figura 7, pode-se perceber que o Avaliador de Logs é capaz de produzir 21 resultados, que são detalhados no Quadro 26.

QUADRO 26 – INSTRUÇÕES GERADAS PELO AVALIADOR DE LOGS

Resultado	Descrição
Passos	Determina o número de instruções que foram gerados no *log*.
Número de classes criadas	Determina o número de classes que o aluno criou durante a construção do diagrama de classes
Número de classes excluídas	Determina o número de classes que o aluno excluiu durante a criação do diagrama de classes
Número de classes renomeadas	Determina o número de classes que os alunos mudaram o nome durante a construção do diagrama de classes
Número de atributos criados	Determina o número de atributos criados pelos alunos durante a construção do diagrama de classes, sem considerar a quais classes estes atributos pertencem
Número de atributos públicos	Determina o número de atributos públicos criados pelos alunos durante a construção do diagrama de classes, sem considerar a quais classes estes atributos pertencem
Número de atributos renomeados	Determina a quantidade de vezes que os alunos mudaram as características dos atributos durante a construção do diagrama de classes. Neste caso pode-se considerar como mudança a troca do nome do atributo, seu tipo ou o seu modificador de acesso
Número de atributos excluídos	Determina o número de atributos que os alunos excluíram durante a construção do diagrama de classes, sem considerar a quais classes estes atributos pertencem
Número de atributos renomeados para métodos	Determina a quantidade de vezes que os alunos mudaram o atributo para um método durante a construção do diagrama de classes, sem considerar a quais classes estes atributos pertencem
Número de métodos criados	Determina o número de métodos criados pelos alunos durante a construção do diagrama de

	classes, sem considerar a quais classes estes métodos pertencem
Resultado	Descrição
Número de métodos renomeados	Determina a quantidade de vezes que os alunos mudaram as características dos métodos durante a construção do diagrama de classes. Neste caso pode-se considerar como mudança a troca do nome do método, seus parâmetros, seu tipo de retorno ou o seu modificador de acesso
Número de métodos excluídos	Determina o número de métodos que os alunos excluíram durante a construção do diagrama de classes, sem considerar a quais classes estes métodos pertencem
Número de associações criadas	Determina o número de relacionamentos de associações simples criadas pelos alunos durante a construção do diagrama de classes
Número de associações excluídas	Determina o número de relacionamentos de associações simples excluídas pelos alunos durante a construção do diagrama de classes
Número de composições criadas	Determina o número de relacionamentos de composição criados pelos alunos durante a construção do diagrama de classes
Número de composições excluídas	Determina o número de relacionamentos de composição excluídos pelos alunos durante a construção do diagrama de classes
Número de agregações criadas	Determina o número de relacionamentos de agregação criados pelos alunos durante a construção do diagrama de classes
Número de agregações excluídas	Determina o número de relacionamentos de agregação excluídos pelos alunos durante a construção do diagrama de classes
Número de heranças criadas	Determina o número de relacionamentos de herança criados pelos alunos durante a construção do diagrama de classes
Número de herança excluídas	Determina o número de relacionamentos de herança excluídos pelos alunos durante a construção do diagrama de classes

Resultado	Descrição
Número de RENAMES e REMOVES	Determina a quantidade de renomeações e exclusões que aconteceram com classes, atributos, métodos e relacionamentos, durante a construção do diagrama de classes

FONTE: O autor (2017).

No passo 4.1, observado na Figura 5, o professor corrige o exercício feito por cada aluno e atribui conceitos para o mesmo. A ideia é que o professor faça a correção dos Diagramas de Classe construídos pelos alunos, para depois cruzar os resultados desta correção com os resultados gerados pelo Avaliador de Logs da ferramenta. No passo 5 da Figura 5, o professor analisa os dados consolidados pelo Avaliador de Logs (Módulo 2) e os conceitos obtidos com a correção dos Diagramas de Classe. Os resultados destas análises serão apresentados na seção 4 deste trabalho.

3.4 EXPERIMENTOS REALIZADOS

Essa seção tem como objetivo apresentar os experimentos realizados com a Ferramenta para o Apoio Ensino-Aprendizagem do Diagrama de Classes. Foram realizados 4 experimentos, com alunos do curso de graduação em Análise e Desenvolvimento de Sistemas em uma Universidade de Curitiba. O Quadro 27 indica o nome do curso de graduação, a disciplina cursada pelos alunos participantes, o período da disciplina, as datas de realização de todos os experimentos e o número de alunos por experimento.

QUADRO 27 – RESUMO DOS EXPERIMENTOS REALIZADOS

Experimento	1	2	3	4
Curso	Análise e Desenvolvimento de Sistemas	Análise e Desenvolvimento de Sistemas	Análise e Desenvolvimento de Sistemas	Análise e Desenvolvimento de Sistemas
Disciplina cursada pelos alunos participantes	Desenvolvimento de Sistemas com Java	Desenvolvimento de Sistemas com Java	Desenvolvimento de Sistemas com C#	Desenvolvimento de Sistemas com C#
Período da disciplina	2º	2º	3º	4º
Data da execução do experimento	23/08/2016	03/10/2016	14/03/2017	30/05/2017
N° de alunos no experimento	20	16	22	20

FONTE: O autor (2017).

Para os experimentos, os alunos construíram o Diagrama de Classes no Microsoft Visio, durante uma avaliação. Na Universidade, local dos experimentos, os alunos são avaliados por competência e não por nota. Assim, cada disciplina do curso é composta por um conjunto de competências, classificadas em necessárias e complementares. Para serem aprovados em uma determinada disciplina, os alunos devem obter 100% das competências necessárias e no mínimo 60% das competências complementares. Se os alunos obtiverem no mínimo 50% das competências necessárias e no mínimo 50% das competências complementares, eles ficam com o status pendente. Por fim, se os alunos não obtiverem no mínimo 50% das competências necessárias ou no mínimo 50% das competências complementares, eles estão reprovados, precisando cursar a disciplina novamente. Para cada competência, a nota é binária: SIM ou NÃO. Os critérios estabelecidos pelo professor para que os alunos possam adquirir as competências são baseados em indicadores de competências. Por exemplo, para um aluno adquirir a competência relacionada a classes, são considerados os seguintes indicadores:

- Os nomes de classes refletem o domínio do problema?
- A quantidade de classes que modeladas são adequadas ao domínio do problema?

- As classes modeladas resolvem o problema apresentado?

Se a resposta for sim, para no mínimo 60% dos indicadores, o aluno adquire a competência. Vale destacar que a correção das avaliações, pelo professor, é subjetiva e pode variar de professor para professor, pois a verificação de um modelo que envolve abstração fica sujeito a esse tipo de condição.

Como possíveis ameaças à validade deste trabalho, estão: a definição de conceitos de avaliação para os alunos por um único professor, sendo o próprio pesquisador; a aplicação dos testes em um único curso de graduação; a aplicação dos testes em uma única universidade.

3.4.1 Primeiro experimento

No primeiro experimento, realizado no dia 23/08/2016, participaram 20 alunos do 2º período do curso de Análise e Desenvolvimento de Sistemas. A disciplina que serviu como apoio para o experimento foi Desenvolvimento de Sistemas com Java e nela os alunos estudam os conceitos iniciais de orientação a objetos. Durante as aulas, os alunos aprendem a modelar o Diagrama de Classes, bem como a implementar os modelos criados na linguagem de programação Java. São estudados os conceitos de classes e objetos, relacionamentos entre classes e polimorfismo na análise e implementação de projetos. São trabalhados também os conceitos de interfaces e exceções. A disciplina de Desenvolvimento de Sistemas com Java é composta por 10 competências, sendo 5 classificadas como necessárias e 5 classificadas como complementares. Anteriormente, os alunos já haviam cursado as seguintes disciplinas: Algoritmos de Programação I, Fundamentos de Sistemas de Informação, Algoritmos de Programação II e Sistemas Operacionais. Nas disciplinas de Algoritmos de Programação I e Algoritmos de Programação II, os alunos tiveram contato com os conceitos iniciais em programação de computadores usando a pseudo linguagem Portugol e Linguagem C, ambas no paradigma procedural/estruturado.

No experimento foram avaliadas 2 competências: Identificar classes, atributos e seus modificadores de acesso para a construção de diagramas de classe UML; Identificar métodos para a construção de diagramas de classe UML. O cenário disponibilizado, a ser modelado pelos alunos, está apresentado no Quadro 28. Esse

cenário foi interpretado pelos alunos, que criaram o diagrama de classes correspondente no Microsoft Visio.

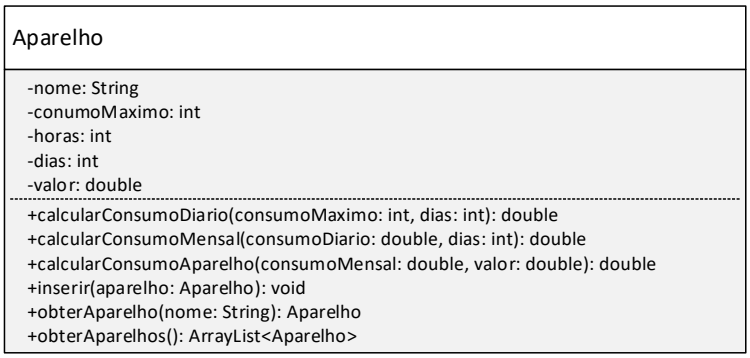
QUADRO 28 – CENÁRIO PARA O EXPERIMENTO 1

Parte	Cenário
1	Construir as classes necessárias para o cálculo de consumo de energia elétrica de aparelhos eletroeletrônicos.
2	Para isso, é necessário saber: a) Qual o aparelho; b) Qual o consumo máximo do aparelho em watts; c) Quantas horas o aparelho é ligado por dia; Quantos dias o aparelho fica ligado durante o mês; Qual o valor do kilowatt-hora.
3	Com base nos dados do aparelho, o sistema deverá apresentar: a) Consumo diário do aparelho (consumo máximo em watts* número de horas que o aparelho é ligado por dia = x; x/1000); b) Consumo mensal do aparelho (consumo diário do aparelho * número de dias que o aparelho ficará ligado no mês); c) Consumo do aparelho em R$ (consumo mensal do aparelho * valor do kilowatt-hora).
4	São requisitos do sistema: a) O sistema deverá permitir o cadastro de vários aparelhos; b) O sistema não deverá permitir aparelhos cadastrados com o mesmo nome; c) O sistema deverá permitir a consulta dos aparelhos cadastrados, pelo nome do aparelho; d) O sistema deverá listar os nomes de todos os aparelhos cadastrados.

FONTE: O autor (2017).

O professor também criou o Diagrama de Classes para o cenário definido no Quadro 28, para fins de comparação entre o modelo criado pelo professor e o modelo criado pelos alunos. O Diagrama de Classes construído pelo professor é apresentado na Figura 8.

FIGURA 8 – DIAGRAMA DE CLASSES DO EXPERIMENTO 1



FONTE: O autor (2017).

Para este experimento, que é o primeiro, tanto o Gerador de Logs quanto o Leitor de Logs estavam em processo de desenvolvimento, portanto, à época, ambos não estavam aptos a gerar informações sobre os relacionamentos entre classes. Por este motivo, o cenário disponibilizado aos alunos foi bem simples, gerando como resultado final apenas uma classe no Diagrama de Classes.

O Quadro 29 representa os dados obtidos do experimento 1. Nele é possível verificar que o aluno 1 executou 31 passos para a construção do Diagrama de Classes, que 13 atributos foram criados, que 7 métodos foram renomeados, entre outros. Na seção 4, estes dados serão analisados e discutidos mais detalhadamente.

QUADRO 29 – RESULTADOS OBTIDOS DO EXPERIMENTO 1

Aluno	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	Professor
N° de passos total	31	42	17	29	22	19	21	23	39	18	31	20	31	16	15	19	32	21	41	13	12
Atributos criados	13	13	7	7	5	6	5	5	9	5	8	12	5	8	5	6	3	7	5	5	5
Atributos renomeados	10	13	3	4	6	4	1	1	2	1	2	0	5	2	1	4	8	2	5	3	0
Atributos excluídos	0	3	0	1	1	1	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0
Total de intervenções sobre atributos	23	29	10	12	12	11	7	6	11	6	11	12	10	11	6	10	11	9	10	8	5
Métodos criados	0	3	3	6	5	3	6	6	1	7	6	0	6	3	4	5	8	4	10	2	6
Métodos renomeados	7	6	1	7	3	5	6	7	4	4	4	0	13	1	4	2	11	5	19	2	0
Métodos excluídos	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Total de intervenções sobre métodos	7	9	4	13	8	8	12	13	5	11	10	0	19	4	8	7	19	9	29	4	6

FONTE: O autor (2017).

3.4.2 Segundo experimento

No segundo experimento, realizado no dia 03/10/2016, participaram 16 alunos do 2º período do curso de Análise e Desenvolvimento de Sistemas. A disciplina que serviu como apoio para o experimento foi Desenvolvimento de Sistemas com Java e nela os alunos estudam os conceitos iniciais de orientação a objetos. Durante as aulas, os alunos aprendem a modelar o Diagrama de Classes, bem como implementar os modelos criados na linguagem de programação Java. São estudados os conceitos de classes e objetos, relacionamentos entre classes e polimorfismo na análise e implementação de projetos. São trabalhados também os conceitos de interfaces e exceções. A disciplina de Desenvolvimento de Sistema com Java é composta por 10 competências, sendo 5 classificadas como necessárias e 5 classificadas como complementares. Anteriormente, os alunos já haviam cursado as seguintes disciplinas: Algoritmos de Programação I, Fundamentos de Sistemas de Informação, Algoritmos de Programação II e Sistemas Operacionais. Nas disciplinas de Algoritmos de Programação I e Algoritmos de Programação II, os alunos tiveram contato com os conceitos iniciais em programação de computadores usando a pseudo linguagem Portugol e Linguagem C, ambos no paradigma procedural/estruturado.

No experimento foram avaliadas 6 competências: Identificar classes, atributos e seus modificadores de acesso para a construção de diagramas de classe UML; Identificar métodos para a construção de diagramas de classe UML; Utilizar o conceito de herança em diagramas de classe UML; Utilizar o conceito de associação simples em diagramas de classe UML; Utilizar o conceito de composição em diagramas de classe UML; Utilizar o conceito de agregação em diagramas de classe UML. O cenário disponibilizado, a ser modelado pelos alunos, está apresentado no Quadro 30. Esse cenário foi interpretado pelos alunos, que criaram o diagrama de classes correspondente no Microsoft Visio. Para os experimentos 3 e 4 também foram utilizados o cenário do Quadro 30.

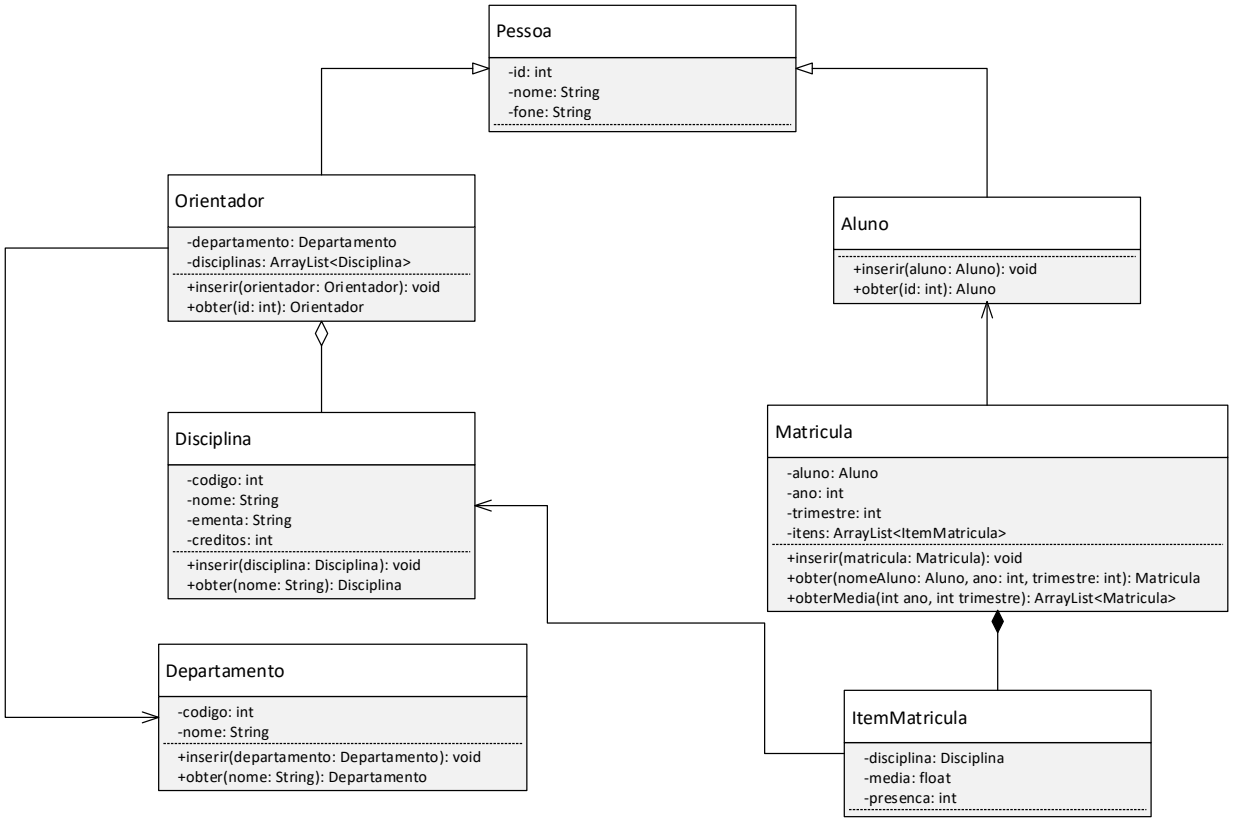
QUADRO 30 – CENÁRIO PARA OS EXPERIMENTOS 2, 3 E 4

Parte	Cenário
1	Uma universidade precisa desenvolver um sistema para gerenciar algumas atividades. A universidade é dividida em departamentos. Cada departamento tem um código e nome. Ao cadastrar um departamento no sistema, o seu nome não pode ser repetido.
2	Para cada disciplina, deve-se armazenar o código, o nome, a ementa e o número de créditos. Ao cadastrar uma disciplina, o seu nome não pode ser repetido.
3	Para o orientador é necessário armazenar o número, o nome, um telefone para contato, o departamento ao qual pertence e as disciplinas ministradas por ele. Considere que o número de disciplinas ministradas pelo orientador vai de 1 a n. Ao cadastrar o orientador, o seu número não pode ser repetido.
4	Cada aluno da universidade possui um número, nome e o telefone para contato. Ao cadastrar um aluno, o seu número não pode ser repetido.
5	Um aluno pode se matricular em várias disciplinas. Para cada disciplina que o aluno se matricular, pretende-se saber a média final obtida e a frequência (presença). Na matrícula também deve constar o ano da matrícula e o trimestre (1º, 2º, 3º ou 4º). Um aluno não pode ser matricular mais de uma vez no mesmo ano e mesmo trimestre.
6	Como consulta, o sistema deve retornar a média dos alunos em um determinado ano e trimestre. Por exemplo, o usuário informa o ano e o trimestre e o sistema apresenta todos os alunos matriculados no ano e trimestre informados, apresentando a média de cada um, bem como a média geral.

FONTE: O autor (2017).

O professor também criou o Diagrama de Classes para o cenário definido no Quadro 30, para fins de comparação entre o modelo criado pelo professor e o modelo criado pelos alunos. O Diagrama de Classes construído pelo professor é apresentado na Figura 9.

FIGURA 9 – DIAGRAMA DE CLASSES DOS EXPERIMENTOS 2, 3 E 4



FONTE: O autor (2017).

Para este experimento, que é o segundo, já foram considerados os conceitos de relacionamentos entre classes. Assim, tanto o Gerador de Logs quanto o Leitor de Logs já estavam aptos para gerar e analisar informações sobre herança, associação simples, agregação e composição.

O Quadro 31 representa os dados obtidos do experimento 2. Nele é possível verificar que o aluno 1 executou 62 passos para a construção do Diagrama de Classes, que 22 atributos foram criados, que 6 métodos foram renomeados, que 8 associações simples foram criadas, entre outros. Na seção 4, estes dados serão analisados e discutidos mais detalhadamente.

QUADRO 31 – RESULTADOS OBTIDOS DO EXPERIMENTO 2

Aluno	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	Professor
N° de passos total	62	120	84	118	83	96	78	134	78	88	115	81	66	76	97	68	43
Classes criadas	6	8	7	5	7	7	7	8	7	6	7	6	5	7	5	7	7
Classes renomeadas	0	5	2	2	2	5	1	1	3	0	17	0	2	6	3	1	0
Classes excluídas	0	1	0	1	0	0	1	1	1	0	0	0	0	0	1	1	0
Total de intervenções sobre classes	6	14	9	8	9	12	9	10	11	6	24	6	7	13	9	9	7
Atributos criados	22	39	27	26	31	27	22	35	28	30	24	22	22	28	39	23	18
Atributos renomeados	2	15	5	16	10	10	10	13	3	12	19	28	1	4	12	3	0
Atributos excluídos	4	12	6	2	15	4	2	13	8	9	3	4	3	2	5	1	0
Atributos renomeados para métodos	1	1	0	0	0	3	0	4	0	1	0	0	0	0	1	1	0
Total de intervenções sobre atributos	29	67	38	44	56	44	34	65	39	52	46	54	26	34	57	28	18
Métodos criados	5	9	15	15	6	10	14	15	11	11	14	10	13	12	14	10	11
Métodos renomeados	6	9	6	9	0	9	9	18	2	9	4	5	12	4	11	5	0
Métodos excluídos	0	2	0	1	1	0	1	2	1	0	1	1	2	0	1	0	0
Total de intervenções sobre métodos	11	20	21	25	7	19	24	35	14	20	19	16	27	16	26	15	11
Associações simples criadas	8	3	3	0	1	7	0	5	6	2	8	1	1	1	2	5	3
Associações simples excluídas	4	1	1	0	0	5	0	4	3	1	6	0	0	1	1	3	0
Total de intervenções sobre associações simples	12	4	4	0	1	12	0	9	9	3	14	1	1	2	3	8	3
Heranças criadas	2	2	2	0	2	2	3	3	2	2	2	2	0	1	0	2	2
Heranças excluídas	0	0	0	0	0	0	1	1	0	0	0	0	0	1	0	0	0
Total de intervenções sobre heranças	2	2	2	0	2	2	4	4	2	2	2	2	0	2	0	2	2
Agregações criadas	0	2	1	9	5	2	2	2	2	1	2	1	3	2	2	0	1
Agregações excluídas	0	1	2	7	2	1	0	0	1	0	1	0	1	1	0	0	0
Total de intervenções sobre agregações	0	3	3	16	7	3	2	2	3	1	3	1	4	3	2	0	1
Composições criadas	1	6	5	14	1	3	5	5	0	3	5	1	1	5	0	4	1
Composições excluídas	1	4	2	11	0	1	0	4	0	1	2	0	0	1	0	2	0
Total de intervenções sobre composições	2	10	7	25	1	4	5	9	0	4	7	1	1	6	0	6	1

FONTE: O autor (2017).

3.4.3 Terceiro experimento

No terceiro experimento, realizado no dia 14/03/2017, participaram 22 alunos do 3º período do curso de Análise e Desenvolvimento de Sistemas. A disciplina que serviu como apoio para o experimento foi Desenvolvimento de Sistemas com C#, que tem como objetivo capacitar o aluno no desenvolvimento de sistemas desktop para a plataforma Windows. Durante as aulas, os alunos projetam sistemas orientados a objetos. São estudados os conceitos de classes e objetos, herança e polimorfismo para análise e implementação de sistemas. São trabalhados também os conceitos de interfaces, exceções, criação de Diagramas de Classes e persistência de dados. A

disciplina de Desenvolvimento de Sistema com C# é composta por 08 competências, sendo 4 classificadas como necessárias e 4 classificadas como complementares. Anteriormente, os alunos já haviam cursado as seguintes disciplinas: Algoritmos de Programação I, Fundamentos de Sistemas de Informação, Algoritmos de Programação II, Sistemas Operacionais, Desenvolvimento de Sistemas com Java e Banco de Dados I. Nas disciplinas de Algoritmos de Programação I e Algoritmos de Programação II os alunos tiveram contato com os conceitos iniciais em programação de computadores usando a pseudo linguagem Portugol e Linguagem C, ambos no paradigma procedural/estruturado. Na disciplina de Desenvolvimento de Sistemas com Java os alunos conheceram os conceitos básicos de orientação a objetos. Assim, para o experimento, considera-se que os alunos já possuem conhecimento básico de orientação a objetos.

No experimento foram avaliadas 6 competências: Identificar classes, atributos e seus modificadores de acesso para a construção de diagramas de classe UML e implementação de sistemas orientados a objetos; Identificar métodos para a construção de diagramas de classe UML e implementação de sistemas orientados a objetos; Utilizar o conceito de herança em diagramas de classe UML e implementação de sistemas orientados a objetos; Utilizar o conceito de associação simples em diagramas de classe UML e implementação de sistemas orientados a objetos; Utilizar o conceito de composição em diagramas de classe UML e implementação de sistemas orientados a objetos; Utilizar o conceito de agregação em diagramas de classe UML e implementação de sistemas orientados a objetos. O cenário disponibilizado, a ser modelado pelos alunos, está apresentado no Quadro 30. Esse cenário foi interpretado pelos alunos, que criaram o diagrama de classes correspondente no Microsoft Visio.

O professor também criou o Diagrama de Classes para o cenário definido no Quadro 30, para fins de comparação entre o modelo criado pelo professor e o modelo criado pelos alunos. O Diagrama de Classes construído pelo professor é apresentado na Figura 9.

Para este experimento, que é o terceiro, já foram considerados os conceitos de relacionamentos entre classes. Assim, tanto o Gerador de Logs quanto o Leitor de Logs já estavam aptos para gerar e analisar informações sobre herança, associação simples, agregação e composição.

O Quadro 32 representa os dados obtidos do experimento 3. Nele é possível verificar que o aluno 1 executou 87 passos para a construção do Diagrama de Classes, que 30 atributos foram criados, que 4 métodos foram renomeados, que 6 associações simples foram criadas, entre outros. Na seção 4, estes dados serão analisados e discutidos mais detalhadamente.

QUADRO 32 – RESULTADOS OBTIDOS DO EXPERIMENTO 3

Aluno	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	Prof
N° de passos total	87	115	75	81	135	54	78	109	70	84	75	101	83	128	60	122	58	93	94	66	84	124	43
Classes criadas	7	7	6	7	6	6	8	7	7	8	6	6	6	9	6	7	6	9	8	7	7	8	7
Classes excluídas	1	0	0	2	0	0	1	1	0	0	0	0	3	2	0	2	0	3	2	0	0	0	0
Classes renomeadas	0	3	1	0	2	1	1	0	0	3	1	0	1	4	0	1	3	2	0	2	1	2	0
Total de intervenções sobre classes	8	10	7	9	8	7	10	8	7	11	7	6	10	15	6	10	9	14	10	9	8	10	7
Atributos criados	30	37	26	27	37	20	28	29	23	32	30	26	26	36	19	45	19	13	14	27	31	33	18
Atributos renomeados	3	21	3	5	40	0	6	25	3	13	10	11	12	9	1	12	2	24	19	6	9	13	0
Atributos excluídos	8	12	2	12	5	0	8	6	4	7	0	7	1	10	0	13	0	0	2	9	6	11	0
Atributos renomeados para métodos	1	4	0	0	0	0	0	0	0	0	0	1	0	3	0	1	0	2	1	0	1	0	0
Total de intervenções sobre atributos	42	74	31	44	82	20	42	60	30	52	40	45	39	58	20	71	21	39	36	42	47	57	18
Métodos criados	12	10	12	6	13	13	12	12	10	2	13	15	11	11	10	21	15	4	15	6	10	20	11
Métodos renomeados	4	9	13	0	20	0	5	3	4	2	8	15	11	19	1	4	3	20	11	0	5	14	0
Métodos excluídos	1	0	0	0	0	2	0	1	0	0	0	2	0	0	1	6	0	0	0	0	0	4	0
Total de intervenções sobre métodos	17	19	25	6	33	15	17	16	14	4	21	32	22	30	12	31	18	24	26	6	15	38	11
Associações simples criadas	6	1	6	8	5	0	4	8	8	3	4	5	1	11	8	1	3	7	2	1	6	7	3
Associações simples excluídas	3	0	3	6	2	0	0	4	5	1	1	1	0	7	5	0	1	4	0	0	2	3	0

Aluno	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	Prof
Total de intervenções sobre associações simples	9	1	9	14	7	0	4	12	13	4	5	6	1	18	13	1	4	11	2	1	8	10	3
Heranças criadas	4	3	2	3	2	2	2	0	2	6	0	1	0	3	2	2	2	2	2	2	2	2	2
Heranças excluídas	2	1	0	1	0	0	0	0	0	4	0	1	0	1	0	0	0	0	0	0	0	0	0
Total de intervenções sobre heranças	6	4	2	4	2	2	2	0	2	10	0	2	0	4	2	2	2	2	2	2	2	2	2
Agregações criadas	0	3	0	1	2	5	0	6	1	2	0	1	3	2	4	1	3	0	7	3	2	0	1
Agregações excluídas	0	1	0	0	1	3	0	6	1	0	0	1	2	1	3	1	1	0	3	1	1	0	0
Total de intervenções sobre agregações	0	4	0	1	3	8	0	12	2	2	0	2	5	3	7	2	4	0	10	4	3	0	1

FONTE: O autor (2017).

3.4.4 Quarto experimento

No quarto experimento, realizado no dia 30/05/2017, participaram 20 alunos do 4º período do curso de Análise e Desenvolvimento de Sistemas. A disciplina que serviu como apoio para o experimento foi Desenvolvimento de Sistemas com C#, que tem como objetivo capacitar o aluno no desenvolvimento de sistemas desktop para a plataforma Windows. Durante as aulas, os alunos projetam sistemas orientados a objetos. São estudados os conceitos de classes e objetos, herança e polimorfismo para análise e implementação de sistemas. São trabalhados também os conceitos de interfaces, exceções, criação de Diagramas de Classes e persistência de dados. A disciplina de Desenvolvimento de Sistema com C# é composta por 08 competências, sendo 4 classificadas como necessárias e 4 classificadas como complementares. Anteriormente, os alunos já haviam cursado as seguintes disciplinas: Algoritmos de Programação I, Fundamentos de Sistemas de Informação, Algoritmos de Programação II, Sistemas Operacionais, Desenvolvimento de Sistemas com Java, Banco de Dados I, Programação em PHP, Engenharia de Software, Redes de Computadores e Sistemas Operacionais. Nas disciplinas de Algoritmos de Programação I e Algoritmos de Programação II os alunos tiveram contato com os conceitos iniciais em programação de computadores usando a pseudo linguagem Portugol e Linguagem C, ambos no paradigma procedural/estruturado. Na disciplina de Desenvolvimento de Sistemas com Java os alunos conheceram os conceitos básicos de orientação a objetos. Assim, para

o experimento, considera-se que os alunos já possuem conhecimento básico de orientação a objetos.

No experimento foram avaliadas 6 competências: Identificar classes, atributos e seus modificadores de acesso para a construção de diagramas de classe UML e implementação de sistemas orientados a objetos; Identificar métodos para a construção de diagramas de classe UML e implementação de sistemas orientados a objetos; Utilizar o conceito de herança em diagramas de classe UML e implementação de sistemas orientados a objetos; Utilizar o conceito de associação simples em diagramas de classe UML e implementação de sistemas orientados a objetos; Utilizar o conceito de composição em diagramas de classe UML e implementação de sistemas orientados a objetos; Utilizar o conceito de agregação em diagramas de classe UML e implementação de sistemas orientados a objetos. O cenário disponibilizado, a ser modelado pelos alunos, está apresentado no Quadro 30. Esse cenário foi interpretado pelos alunos, que criaram o diagrama de classes correspondente no Microsoft Visio.

O professor também criou o Diagrama de Classes para o cenário definido no Quadro 30, para fins de comparação entre o modelo criado pelo professor e o modelo criado pelos alunos. O Diagrama de Classes construído pelo professor é apresentado na Figura 9.

Para este experimento, que é o quarto, já foram considerados os conceitos de relacionamentos entre classes. Assim, tanto o Gerador de Logs quanto o Leitor de Logs já estavam aptos para gerar e analisar informações sobre herança, associação simples, agregação e composição.

O Quadro 33 representa os dados obtidos do experimento 4. Nele é possível verificar que o aluno 1 executou 97 passos para a construção do Diagrama de Classes, que 32 atributos foram criados, que 0 métodos foram renomeados, que 0 associações simples foram criadas, entre outros. Na seção 4, estes dados serão analisados e discutidos mais detalhadamente.

QUADRO 33 – RESULTADOS DO EXPERIMENTO 4

Aluno	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	Professor
N° de passos total	97	89	79	65	186	82	71	61	183	84	40	79	75	83	139	69	48	307	73	88	43
Classes criadas	6	7	6	5	6	6	4	4	6	7	4	6	5	7	7	7	5	10	5	5	7
Classes excluídas	1	1	1	0	1	0	0	0	2	0	0	0	0	1	1	0	0	2	0	0	0
Classes renomeadas	1	4	4	0	3	0	1	0	0	1	0	0	0	0	1	0	1	3	0	1	0
Total de intervenções sobre classes	8	12	11	5	10	6	5	4	8	8	4	6	5	8	9	7	6	15	5	6	7
Atributos criados	32	10	11	20	29	31	31	24	53	28	23	26	25	33	37	23	11	50	26	29	18
Atributos renomeados	42	22	23	11	34	23	29	10	35	6	2	12	13	3	18	9	16	129	7	4	0
Atributos excluídos	1	1	3	0	27	1	1	3	22	9	5	0	7	6	14	0	0	23	8	4	0
Atributos renomeados para métodos	5	1	1	1	1	3	1	0	6	0	0	0	0	2	1	0	4	11	0	1	0
Total de intervenções sobre atributos	80	34	38	32	91	58	62	37	116	43	30	38	45	44	70	32	31	213	41	38	18
Métodos criados	2	5	4	11	18	7	1	8	8	17	0	6	9	12	19	11	0	3	17	19	11
Métodos renomeados	0	29	15	3	31	3	3	4	22	4	0	18	4	2	18	1	5	40	4	11	0
Métodos excluídos	0	1	0	0	3	0	0	0	1	2	0	0	1	0	2	0	0	0	0	4	0
Total de intervenções sobre métodos	2	35	19	14	52	10	4	12	31	23	0	24	14	14	39	12	5	43	21	34	11
Associações simples criadas	0	5	1	0	8	1	0	1	15	6	0	0	0	3	0	3	0	9	2	5	3
Associações simples excluídas	0	0	1	0	3	1	0	0	9	0	0	0	0	1	0	1	0	5	0	2	0
Total de intervenções sobre associações simples	0	5	2	0	11	2	0	1	24	6	0	0	0	4	0	4	0	14	2	7	3
Heranças criadas	0	3	6	0	4	0	0	0	2	3	6	1	0	2	0	9	0	7	0	0	2
Heranças excluídas	0	0	3	0	2	0	0	0	2	1	0	1	0	0	0	5	0	3	0	0	0
Total de intervenções sobre heranças	0	3	9	0	6	0	0	0	4	4	6	2	0	2	0	14	0	10	0	0	2
Agregações criadas	2	0	0	7	5	3	0	3	0	0	0	2	6	1	10	0	3	1	1	1	1
Agregações excluídas	1	0	0	4	5	0	0	0	0	0	0	1	3	0	7	0	1	0	1	1	0
Total de intervenções sobre agregações	3	0	0	11	10	3	0	3	0	0	0	3	9	1	17	0	4	1	2	2	1
Composições criadas	4	0	0	2	3	2	0	2	0	0	0	5	2	7	3	0	2	7	2	1	1
Composições excluídas	0	0	0	1	3	1	0	2	0	0	0	1	0	3	1	0	0	4	0	0	0
Total de intervenções sobre composições	4	0	0	3	6	3	0	4	0	0	0	6	2	10	4	0	2	11	2	1	1

FONTE: O autor (2017).

3.5. RESULTADOS, ANÁLISE E DISCUSSÃO

Essa seção apresentará os resultados mais relevantes dos experimentos. Estes resultados serão analisados e discutidos em detalhes, permitindo o entendimento de como as informações geradas pela Ferramenta de Apoio ao Ensino-Aprendizagem do Diagrama de Classes podem ajudar alunos e professores no processo de ensino-aprendizagem do modelo orientado a objetos.

No total, 78 alunos participaram dos experimentos, divididos em 4 turmas. As turmas dos experimentos 2, 3 e 4 utilizaram o mesmo exemplo de exercício (ver seções 3.4.2, 3.4.3 e 3.4.4) em cada um dos experimentos.

Em todos os experimentos foram investigadas possíveis correlações entre os dados gerados pelos experimentos e os conceitos obtidos pelos alunos. O cálculo do coeficiente de correlação indica o quanto duas variáveis estão vinculadas, relacionadas, o quanto uma afeta a outra, proporcional ou inversamente proporcional. O coeficiente de correlação deve estar entre os valores -1 e 1 e quanto mais próximo de 1 for o resultado, mais fortemente correlacionadas estão as duas variáveis. Por outro lado, quanto mais próximo de 1 negativo, mais inversamente proporcional estão correlacionadas estas duas variáveis. A fórmula do coeficiente de correlação do momento do produto Pearson, r, é:

$$r = \frac{\sum (x - \bar{x})(y - \bar{y})}{\sqrt{\sum (x - \bar{x})^2 \sum (y - \bar{y})^2}}$$

onde x e y são as médias de amostra MÉDIA(matriz1) e MÉDIA(matriz2).

A interpretação dos resultados do cálculo do coeficiente de correlação de Pearson é demonstrada no Quadro 34.

QUADRO 34 - INTERPRETAÇÃO DOS RESULTADOS DO CÁLCULO DO COEFICIENTE DE CORRELAÇÃO DE PEARSON

Coeficiente de correlação	Interpretação
0,00 a 0,19	Correlação muito fraca
0,20 a 0,39	Correlação fraca
0,40 a 0,69	Correlação moderada
0,70 a 0,89	Correlação forte
0,90 a 1,00	Correlação muito forte

FONTE: O autor (2017).

Percebeu-se, nos experimentos realizados, que em alguns momentos os conceitos obtidos pelos alunos têm uma correlação expressiva com as intervenções realizadas durante a construção do Diagrama de Classes.

No experimento 1, por exemplo, foi verificada a correlação entre as intervenções sobre os atributos da classe com a competência relacionada aos atributos, se foi atingida ou não. São consideradas intervenções sobre atributos, a soma das seguintes ações: Atributos criados; Atributos renomeados; Atributos excluídos. O Quadro 35 apresenta as criações, alterações e exclusões de atributos que os alunos executaram para a construção do Diagrama de Classes, bem como a soma destas ações. Também é apresentada a informação se a competência relacionada a atributos foi atingida ou não. Para a competência atingida, o valor é 1 e para a competência não atingida, o valor é 0.

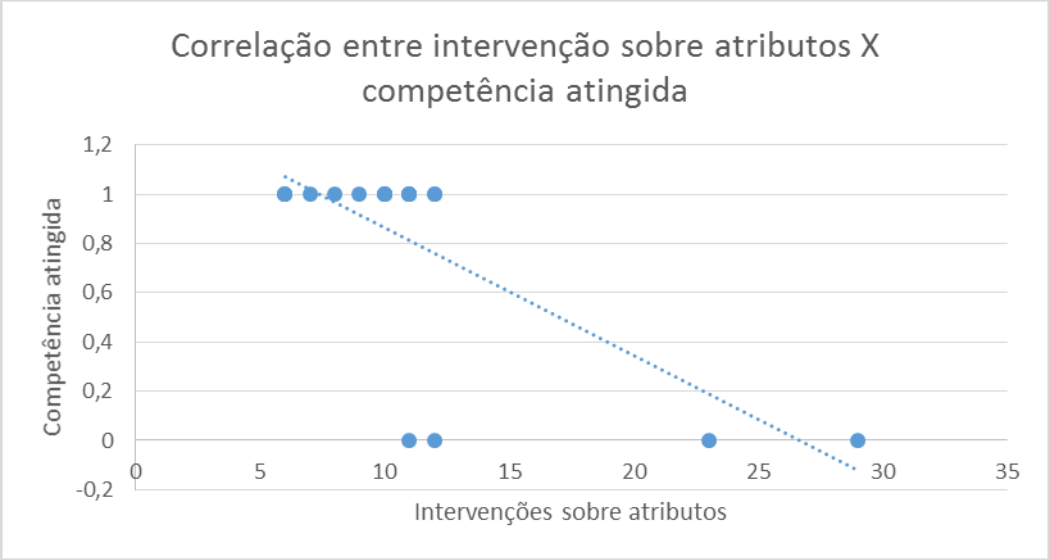
QUADRO 35 – INTERVENÇÕES SOBRE OS ATRIBUTOS DO DIAGRAMA DE CLASSES NO EXPERIMENTO 1

Aluno	Atributos criados	Atributos Renomeados	Atributos Excluídos	Intervenções	Competência Atingida
1	13	10	0	23	0
2	13	13	3	29	0
3	7	3	0	10	1
4	7	4	1	12	1
5	5	6	1	12	1
6	6	4	1	11	1
7	5	1	1	7	1
8	5	1	0	6	1
9	9	2	0	11	1
10	5	1	0	6	1
11	8	2	1	11	0
12	12	0	0	12	0
13	5	5	0	10	1
14	8	2	1	11	1
15	5	1	0	6	1
16	6	4	0	10	1
17	3	8	0	11	1
18	7	2	0	9	1
19	5	5	0	10	1
20	5	3	0	8	1

FONTE: O autor (2017).

A correlação entre o número de intervenções sobre atributos e o conceito obtido foi de r = -0,70, significando que há uma forte correlação, indicando que quanto menor a intervenção sobre os atributos, durante a criação do Diagrama de Classes, maior será o conceito atribuído, ou seja, o aluno atinge a competência. A Figura 10 apresenta graficamente o resultado desta correlação.

FIGURA 10 – CORRELAÇÃO ENTRE INTERVENÇÃO SOBRE ATRIBUTOS X COMPETÊNCIA ATINGIDA NO EXPERIMENTO 1



FONTE: O autor (2017).

No experimento 2, a referida correlação foi de r = 0,10. Já no experimento 3, a mesma correlação foi de r = -0,15 e no experimento 4, a mesma correlação foi de r = 0,16. Ou seja, cada uma das quatro turmas mostrou um comportamento diferente. Neste caso, para a turma do experimento 1, o professor conseguiu identificar quem são os alunos que tem mais intervenções sobre os atributos, durante a construção do Diagrama de Classes, e auxiliá-los com essa possível dificuldade.

Ainda no experimento 1, foi verificada a correlação entre as intervenções sobre os métodos da classe com a competência relacionada aos métodos, se foi atingida ou não. São consideradas intervenções sobre métodos, a soma das seguintes ações: Métodos criados; Métodos renomeados; Métodos excluídos. O Quadro 36 apresenta as criações, alterações e exclusões de métodos que os alunos executaram para a construção do Diagrama de Classes, bem como a soma destas ações. Também é apresentada a informação se a competência relacionada a métodos foi atingida ou não.

QUADRO 36 - INTERVENÇÕES SOBRE OS MÉTODOS DO DIAGRAMA DE CLASSES NO EXPERIMENTO 1

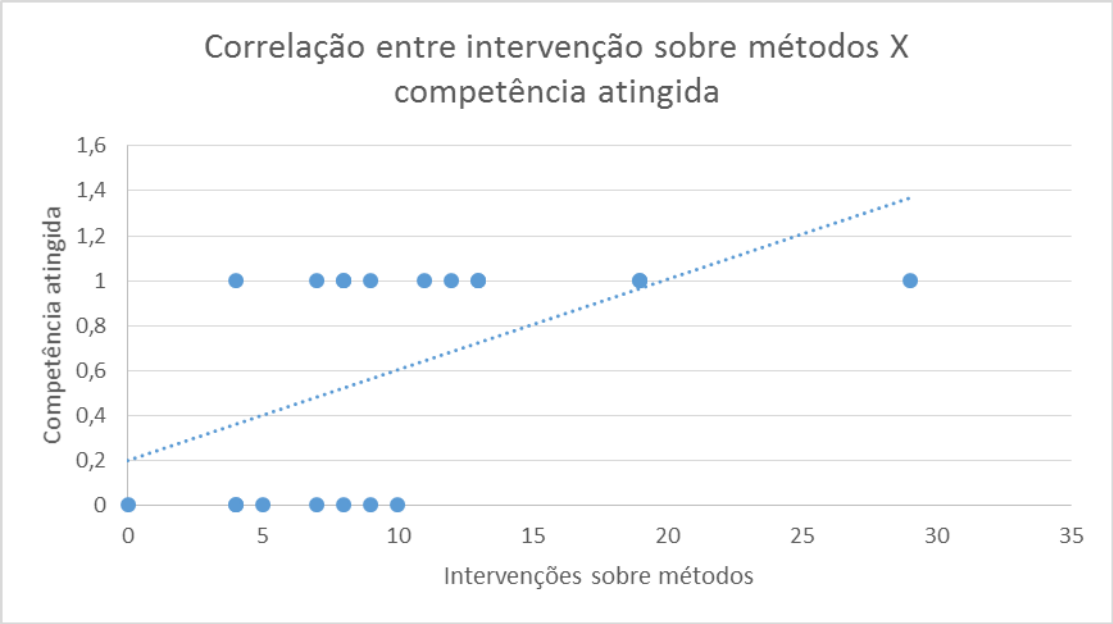
Aluno	Métodos criados	Métodos Renomeados	Métodos Excluídos	Intervenções	Competência Atingida
1	0	7	0	7	0
2	3	6	0	9	0
3	3	1	0	4	1
4	6	7	0	13	1
5	5	3	0	8	0
6	3	5	0	8	1
7	6	6	0	12	1
8	6	7	0	13	1
9	1	4	0	5	0
10	7	4	0	11	1
11	6	4	0	10	0
12	0	0	0	0	0
13	6	13	0	19	1
14	3	1	0	4	0
15	4	4	0	8	1
16	5	2	0	7	1
17	8	11	0	19	1
18	4	5	0	9	1
19	10	19	0	29	1
20	2	2	0	4	0

FONTE: O autor (2017).

A correlação entre o número de intervenções sobre atributos e o conceito obtido foi de r = 0,52, significando que há uma correlação moderada, indicando que quanto maior a intervenção sobre os métodos, durante a criação do Diagrama de Classes, maior será o conceito atribuído, ou seja, o aluno atinge a competência. A Figura 11 apresenta graficamente o resultado desta correlação. Este resultado pode demonstrar que, apesar dos alunos terem atingido a competência, o número elevado de

intervenções pode significar possíveis dificuldades dos alunos, já que os mesmos podem chegar ao modelo correto por meio de tentativa erro.

FIGURA 11 - CORRELAÇÃO ENTRE INTERVENÇÃO SOBRE MÉTODOS X COMPETÊNCIA ATINGIDA NO EXPERIMENTO 1



FONTE: O autor (2017).

No experimento 2, foi verificada a correlação entre as intervenções sobre o relacionamento de associação simples com a competência relacionada ao relacionamento de associação simples, se foi atingida ou não. São consideradas intervenções sobre os relacionamentos de associações simples, a soma das seguintes ações: Associações criadas; Associações excluídas. O Quadro 37 apresenta as criações e exclusões de relacionamentos de associação simples que os alunos executaram para a construção do Diagrama de Classes, bem como a soma destas ações. Também é apresentada a informação se a competência relacionada ao relacionamento de associação simples foi atingida ou não.

QUADRO 37 - INTERVENÇÕES SOBRE AS ASSOCIAÇÕES SIMPLES DO DIAGRAMA DE CLASSES NO EXPERIMENTO 2

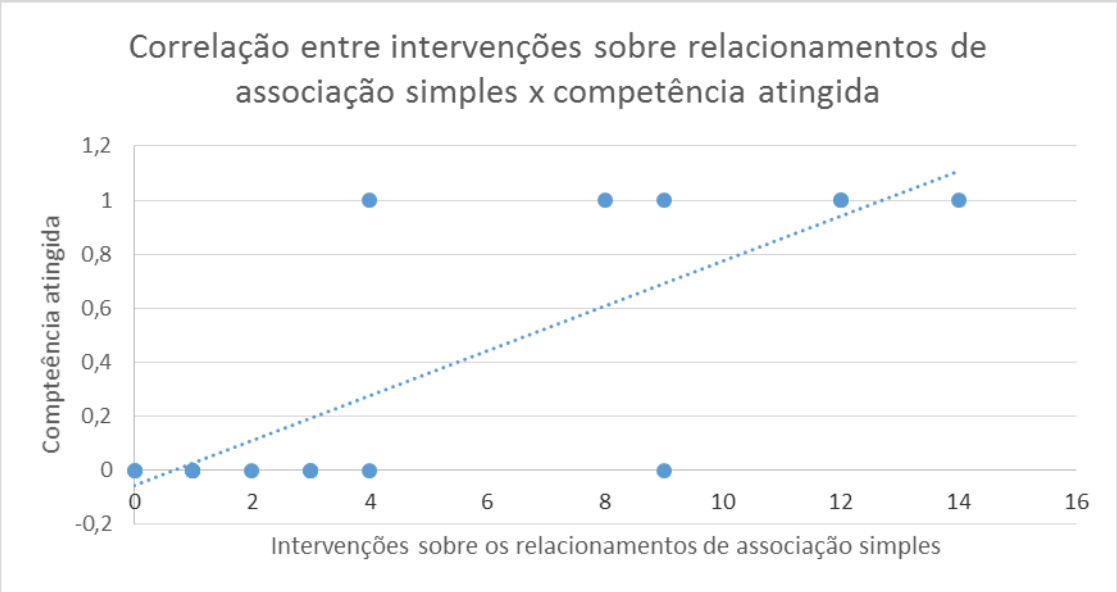
Aluno	Associações criadas	Associações excluídas	Intervenções	Competência Atingida
1	8	4	12	1
2	3	1	4	1
3	3	1	4	0
4	0	0	0	0
5	1	0	1	0
6	7	5	12	1
7	0	0	0	0
8	5	4	9	0
9	6	3	9	1
10	2	1	3	0
11	8	6	14	1
12	1	0	1	0
13	1	0	1	0
14	1	1	2	0
15	2	1	3	0
16	5	3	8	1

FONTE: O autor (2017).

A correlação entre o número de intervenções sobre os relacionamentos de associação simples e o conceito obtido foi de r = 0,78, significando que há uma correlação forte, indicando que quanto maior a intervenção sobre os relacionamentos de associação simples, durante a criação do Diagrama de Classes, maior será o conceito atribuído, ou seja, o aluno atinge a competência. Sabe-se que o aprendizado do relacionamento de associação é um dos mais difíceis, para os alunos. O resultado apresentado demonstra claramente que os alunos realizam muitas intervenções, ou seja, criam e excluem diversas vezes os relacionamentos de associação simples entre as classes, até encontrarem a solução para o cenário apresentado. Por outro lado, os

alunos que realmente não dominam o assunto, sequer tentam intervir no modelo para chegar ao resultado. A Figura 12 apresenta graficamente o resultado desta correlação.

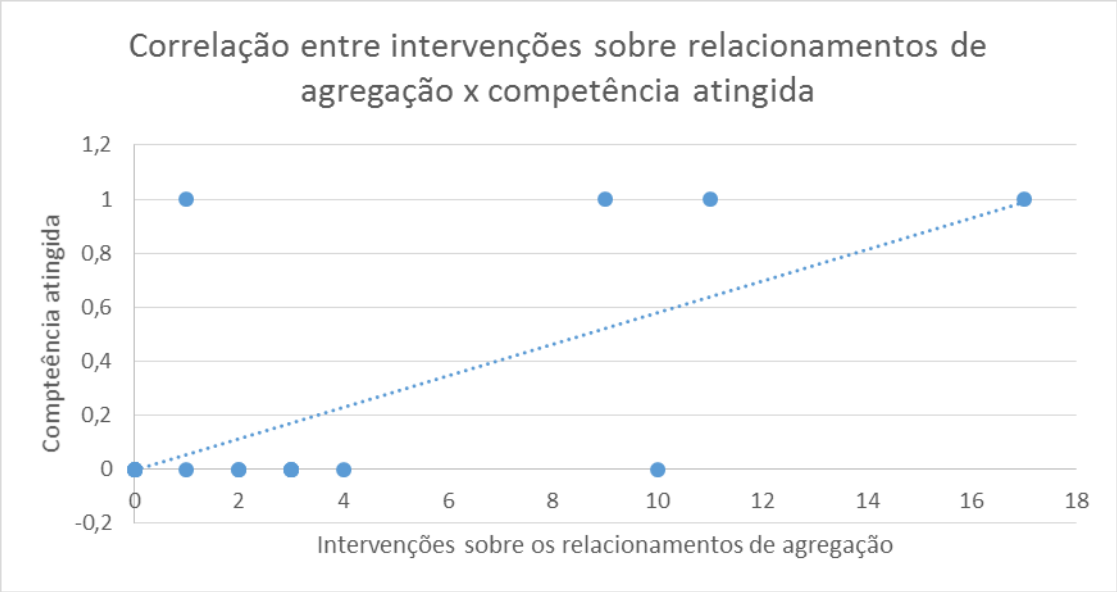
FIGURA 12 - CORRELAÇÃO ENTRE INTERVENÇÃO SOBRE RELACIONAMENTOS DE ASSOCIAÇÃO SIMPLES X COMPETÊNCIA ATINGIDA NO EXPERIMENTO 2



FONTE: O autor (2017).

No experimento 4, a correlação entre o número de intervenções sobre o relacionamento de agregação e o conceito obtido foi de r = 0,66, significando que há uma correlação moderada, indicando que quanto maior a intervenção sobre os relacionamentos de agregação, durante a criação do Diagrama de Classes, maior será o conceito atribuído, ou seja, o aluno atinge a competência. A Figura 13 apresenta graficamente o resultado desta correlação.

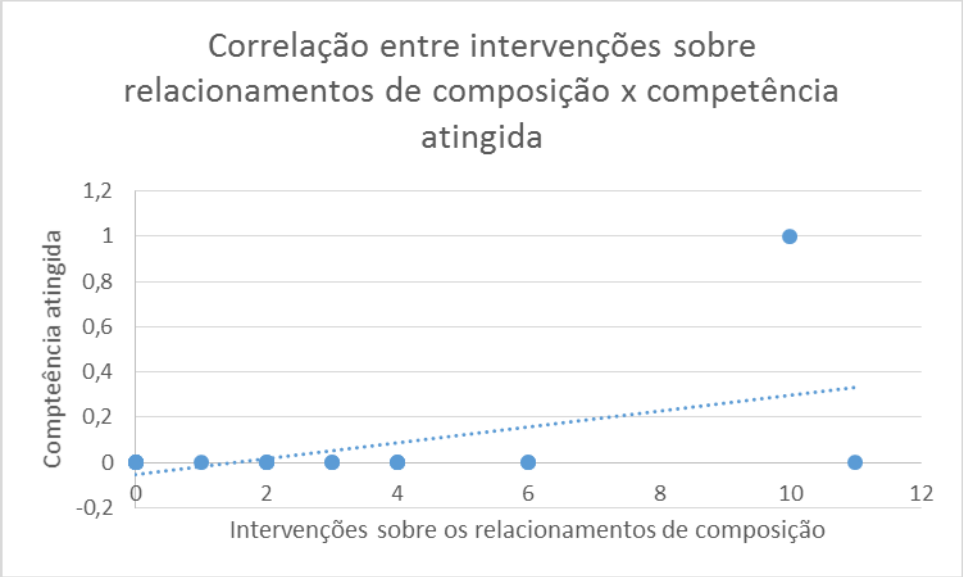
FIGURA 13 - CORRELAÇÃO ENTRE INTERVENÇÃO SOBRE RELACIONAMENTOS DE AGREGAÇÃO X COMPETÊNCIA ATINGIDA NO EXPERIMENTO 4



FONTE: O autor (2017).

Ainda no experimento 4, a correlação entre o número de intervenções sobre o relacionamento de composição e o conceito obtido foi de r = 0,51, significando que há uma correlação moderada, indicando que quanto maior a intervenção sobre os relacionamentos de composição, durante a criação do Diagrama de Classes, maior será o conceito atribuído, ou seja, o aluno atinge a competência. A Figura 14 apresenta graficamente o resultado desta correlação.

FIGURA 14 - CORRELAÇÃO ENTRE INTERVENÇÃO SOBRE RELACIONAMENTOS DE COMPOSIÇÃO X COMPETÊNCIA ATINGIDA NO EXPERIMENTO 4



FONTE: O AUTOR (2017).

Com relação a atributos renomeados para métodos, no experimento 2, 44% dos alunos fizeram esse procedimento, no experimento 3 foram 36% e, no experimento 4 foram 65%. Indagando individualmente os alunos que fizeram esse procedimento, verificou-se uma dificuldade maior desses alunos, em relação aos que não renomearam atributos para métodos, em distinguir atributos de métodos a partir do cenário proposto.

Com relação à correlação do conceito atribuído em relação ao tempo para gerar o diagrama em minutos, no experimento 2 tem-se r = 0,40, no experimento 3, tem-se r = 0,36 e, no experimento 4, tem-se r = 0,41, indicando uma correlação moderada nos três casos, ou seja, quanto mais tempo o aluno demorou para fazer o diagrama, melhor foi o seu conceito.

Para a correlação do conceito atribuído em relação ao número de passos por minuto, no experimento 2 tem-se r = -0,46, no experimento 3, tem-se r = -0,36 e, no experimento 4, tem-se r = -0,15, indicando uma correlação inversamente moderada nos dois primeiros casos, ou seja, quanto menos passos o aluno fez por minuto, melhor foi o seu conceito. Neste caso, provavelmente, o aluno com melhor domínio do tema leva mais tempo analisando o cenário antes de fazer uma intervenção, enquanto que, o

aluno com pior domínio do tema faz mais intervenções por unidade de tempo a fim de tentar mais soluções, analisando menos o cenário.

O Avaliador de Logs gera um alerta toda a vez que o número de passos total por minuto for maior que a média desse valor na turma, a fim de que o professor possa aconselhar o aluno com esse comportamento a analisar melhor cada intervenção que for fazer.

Para fins de comparação com os modelos criados pelos alunos, o professor também criou o Diagrama de Classes com base no cenário entregue aos alunos, sendo que o modelo criado pelo professor é considerado como o “Melhor cenário”. Pode-se observar na última coluna do Quadro 38, que em relação aos atributos criados, renomeados, excluídos e atributos renomeados para métodos, a única intervenção que o professor realiza durante a construção do Diagrama de Classes é a criação dos atributos. Para os alunos, espera-se que isso não aconteça, pois os mesmos estão em processo de aprendizagem. As informações do Quadro 38 são referentes ao experimento 2.

Com o intuito de auxiliar o professor a identificar os problemas que os alunos tiveram durante a criação do diagrama, o Avaliador de Logs gera um alerta para cada aluno, sempre que o total de intervenções sobre um elemento do diagrama de classes é menor que o total de intervenções que o professor fez, ou ainda maior que o dobro do total de intervenções que o professor fez. O Quadro 38 apresenta estes alertas nas linhas “Abaixo de MIN” e “Acima de MAX”, respectivamente. Entende-se que o valor abaixo de *min* deve ser gerado, pois não tem sentido os alunos terem menos intervenções que o professor, que realizou intervenções apenas na criação do elemento. Já para o valor acima de *max*, considera-se que um valor aceitável para o máximo de intervenções é o dobro do que o professor realizou. Também é possível perceber, no Quadro 38, a linha “Competência de atributos atingida”, que informa quais alunos atingiram e quais alunos não atingiram o conceito, ou competência, no momento que o professor corrigiu o Diagrama de Classes. O valor 1 indica que a competência foi atingida e o valor 0 indica que a competência não foi atingida. Assim, sugere-se que os alunos que tem o total de intervenções abaixo do mínimo, que é min = 7, e acima do

máximo, que é max = 36, devem ser recomendados a estudar atributos. Também serão recomendados a estudar atributos, aqueles alunos que não atingiram a competência.

QUADRO 38 – ANÁLISE DO ITEM “ATRIBUTOS” OBTIDOS COM O EXPERIMENTO 2

Aluno	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	Prof
Atributos criados	30	37	26	27	37	20	28	29	23	32	30	26	26	36	19	45	19	13	14	27	31	33	18
Atributos renomeados	3	21	3	5	40	0	6	25	3	13	10	11	12	9	1	12	2	24	19	6	9	13	0
Atributos excluídos	8	12	2	12	5	0	8	6	4	7	0	7	1	10	0	13	0	0	2	9	6	11	0
Atributos renomeados para métodos	1	4	0	0	0	0	0	0	0	0	0	1	0	3	0	1	0	2	1	0	1	0	0
Total de intervenções sobre atributos	42	74	31	44	82	20	42	60	30	52	40	45	39	58	20	71	21	39	36	42	47	57	18
Competência de atributos atingida	1	1	0	0	0	1	1	1	1	0	1	1	0	1	1	1	1	1	1	1	1	1	-
Incentivados a estudar atributos																							
Acima de MAX	X	X		X	X		X	X		X	X	X	X	X		X		X		X	X	X	-
Abaixo de MIN																							-
Competência não atingida			X	X	X					X			X										-

FONTE: O autor (2017).

No Quadro 38, a linha “Acima de MAX”, indica quais alunos serão recomendados a estudar atributos. Pode-se observar nessa linha, que vários alunos foram selecionados mesmo tendo atingido a competência. Estes alunos tem um total de intervenções acima do número de intervenções aceitáveis, sendo assim eles podem ter atingido a competência por meio de tentativa e erro ou qualquer outro fator que não dispensa eventuais dificuldades para alcançar os objetivos propostos.

A linha “Abaixo de MIN”, do Quadro 38, não possui nenhum indicador. Todos os alunos realizaram o mínimo de intervenções aceitáveis.

Já a linha “Competência não atingida”, do Quadro 38, seleciona os alunos que de fato não atingiram a competência. Ao analisar estes apontamentos, percebe-se que os alunos 5, 10 e 13 tem duas marcações. Além de não terem atingido a competência, eles têm um elevado número de intervenções, acima do número de intervenções

aceitáveis. Para estes alunos, o incentivo para estudar atributos deve ser prioritária e altamente recomendável.

QUADRO 39 - ANÁLISE DO ITEM “MÉTODOS” OBTIDOS COM O EXPERIMENTO 2

Aluno	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	Prof
Métodos criados	12	10	12	6	13	13	12	12	10	2	13	15	11	11	10	21	15	4	15	6	10	20	11
Métodos renomeados	4	9	13	0	20	0	5	3	4	2	8	15	11	19	1	4	3	20	11	0	5	14	0
Métodos excluídos	1	0	0	0	0	2	0	1	0	0	0	2	0	0	1	6	0	0	0	0	0	4	0
Total de intervenções sobre métodos	17	19	25	6	33	15	17	16	14	4	21	32	22	30	12	31	18	24	26	6	15	38	11
Competência de métodos atingida	1	1	1	0	0	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	-
Incentivados a estudar métodos																							
Acima de MAX			X		X							X		X		X		X	X			X	-
Abaixo de MIN				X						X										X			-
Competência não atingida				X	X					X													-

FONTE: O autor (2017).

Essas interpretações e análises devem ser estendidas ao demais elementos do Diagrama de Classes. O Quadro 39 apresenta as análises ao elemento “métodos” do Diagrama de Classes, referentes ao segundo experimento. Já o Quadro 40, apresenta as análises dos elementos “associações simples ou comuns”, “herança”, “composição” e “agregação”, também referentes ao segundo experimento.

QUADRO 40 - ANÁLISE DOS ITENS “ASSOCIAÇÕES SIMPLES OU COMUNS”, “HERANÇA”, “COMPOSIÇÃO” E “AGREGAÇÃO” OBTIDOS COM O EXPERIMENTO 2

Aluno	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	Prof
Associações comuns criadas	6	1	6	8	5	0	4	8	8	3	4	5	1	11	8	1	3	7	2	1	6	7	3
Associações comuns excluídas	3	0	3	6	2	0	0	4	5	1	1	1	0	7	5	0	1	4	0	0	2	3	0
Total de intervenções sobre associações	9	1	9	14	7	0	4	12	13	4	5	6	1	18	13	1	4	11	2	1	8	10	3
Competência de associações atingida	0	0	0	0	0	0	1	1	1	0	1	1	0	1	1	0	1	1	1	1	0	1	-
Incentivados a estudar associações simples																							
Acima de MAX	X		X	X	X			X	X					X	X			X			X	X	-
Abaixo de MIN		X				X							X			X			X	X			-
Competência não atingida	X	X	X	X	X	X				X			X			X					X		-
Heranças criadas	4	3	2	3	2	2	2	0	2	6	0	1	0	3	2	2	2	2	2	2	2	2	2
Heranças excluídas	2	1	0	1	0	0	0	0	0	4	0	1	0	1	0	0	0	0	0	0	0	0	0
Total de intervenções sobre heranças	6	4	2	4	2	2	2	0	2	10	0	2	0	4	2	2	2	2	2	2	2	2	2
Competência de herança atingida	1	1	1	1	0	1	1	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0	-
Incentivados a estudar herança																							
Acima de MAX	X	X		X						X				X									-
Abaixo de MIN								X			X		X										-
Competência não atingida					X			X		X	X	X	X			X	X	X	X	X	X	X	-
Agregações criadas	0	3	0	1	2	5	0	6	1	2	0	1	3	2	4	1	3	0	7	3	2	0	1
Agregações excluídas	0	1	0	0	1	3	0	6	1	0	0	1	2	1	3	1	1	0	3	1	1	0	0
Total de intervenções sobre agregações	0	4	0	1	3	8	0	12	2	2	0	2	5	3	7	2	4	0	10	4	3	0	1
Competência de agregação atingida	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	0	1	0	0	-
Incentivados a estudar agregação																							
Acima de MAX		X			X	X		X					X	X	X		X		X	X	X		-
Abaixo de MIN	X		X				X				X							X				X	-
Competência não atingida	X	X	X	X	X	X	X	X	X	X	X	X	X			X		X	X		X	X	-
Composições criadas	3	2	1	2	0	2	2	1	2	1	2	6	4	0	0	4	0	2	5	2	1	5	1
Composições excluídas	2	1	0	1	0	0	1	0	0	0	0	2	2	0	0	1	0	1	3	0	0	2	0
Total de intervenções sobre composições	5	3	1	3	0	2	3	1	2	1	2	8	6	0	0	5	0	3	8	2	1	7	1
Competência de agregação atingida	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	1	-
Incentivados a estudar composição																							
Acima de MAX	X	X		X		X	X		X		X	X	X			X		X	X	X		X	-
Abaixo de MIN					X									X	X		X						-
Competência não atingida	X	X	X	X	X	X			X	X	X		X	X	X	X	X	X	X	X	X		-

FONTE: O autor (2017).

Nesta seção foram apresentados e discutidos os resultados mais relevantes dos experimentos. É importante observar que os resultados gerados pela Ferramenta de Apoio ao Ensino-Aprendizagem do Modelo Orientado a Objetos Durante a Construção do Diagrama de Classes não resolvem os problemas do processo de ensino-aprendizagem do modelo orientado a objetos, mas podem ser um importante método para aprimorar o processo de ensino-aprendizagem. A ferramenta desenvolvida oferece informações específicas sobre cada aluno, que podem ser utilizadas pelo professor para incentivar cada um a estudar determinados temas, a fim de melhorar o desempenho dos mesmos no entendimento do modelo orientado a objetos e na criação de Diagramas de Classes.

4 CONCLUSÃO

Este trabalho apresenta a Ferramenta para o Apoio ao Ensino-Aprendizagem do Modelo Orientado a Objetos Durante a Construção do Diagrama de Classes. Tal ferramenta foi testada, por meio de experimentos, com alunos do curso de graduação em Análise e Desenvolvimento de Sistemas em uma Universidade de Curitiba. A Ferramenta para o Apoio ao Ensino-Aprendizagem do Modelo Orientado a Objetos Durante a Construção do Diagrama de Classes foi construída com a implementação de dois módulos, sendo eles: O “Gerador de Logs”, que é responsável por coletar todas as ações que cada aluno executa durante a construção de um Diagrama de Classes e o “Avaliador de Logs”, sendo responsável por ler, consolidar e apresentar os dados gerados pelo “Gerador de Logs”. O “Avaliador de Logs” permitiu revelar comportamentos dos alunos durante a construção dos Diagramas de Classes.

A Ferramenta para o Apoio ao Ensino-Aprendizagem do Modelo Orientado a Objetos Durante a Construção do Diagrama de Classes fornece dados negligenciados pelas ferramentas de construção de diagramas, podendo ser um importante método para aprimorar o processo de ensino-aprendizagem do modelo Orientado a Objetos. A ferramenta desenvolvida oferece informações específicas sobre cada aluno, que podem ser utilizadas pelo professor para incentivar cada um a estudar determinados temas, a fim de melhorar o desempenho dos mesmos no entendimento dos conceitos do modelo Orientado a Objetos e na criação de Diagramas de Classes. Não foram encontrados trabalhos relacionados ao tema na literatura, que avaliassem as características da construção e as dificuldades dos alunos durante o processo de construção do Diagrama de Classes, em *background*, como feito neste trabalho, desta forma sendo inovador neste aspecto.

Alguns resultados mostram que os conceitos obtidos pelos alunos têm uma correlação expressiva com as intervenções realizadas durante a construção do Diagrama de Classes. No experimento 1, a correlação entre o número de intervenções sobre atributos e o conceito obtido foi de r = -0,70, significando que há uma forte correlação, indicando que quanto menor a intervenção sobre os atributos, durante a criação do Diagrama de Classes, maior será o conceito atribuído, ou seja, o aluno

atinge a competência. A correlação entre o número de intervenções sobre atributos e o conceito obtido foi de r = 0,52, significando que há uma correlação moderada, indicando que quanto maior a intervenção sobre os métodos, durante a criação do Diagrama de Classes, maior será o conceito atribuído, ou seja, o aluno atinge a competência. No experimento 2, a correlação entre o número de intervenções sobre os relacionamentos de associação simples e o conceito obtido foi de r = 0,78, significando que há uma correlação forte, indicando que quanto maior a intervenção sobre os relacionamentos de associação simples, durante a criação do Diagrama de Classes, maior será o conceito atribuído, ou seja, o aluno atinge a competência. No experimento 4, a correlação entre o número de intervenções sobre o relacionamento de agregação e o conceito obtido foi de r = 0,66, significando que há uma correlação moderada, indicando que quanto maior a intervenção sobre os relacionamentos de agregação, durante a criação do Diagrama de Classes, maior será o conceito atribuído, ou seja, o aluno atinge a competência. O “Avaliador de Logs” também gera um alerta para cada aluno, sempre que o total de intervenções sobre um elemento do diagrama de classes é menor que o total de intervenções que o professor fez, ou ainda maior que o dobro do total de intervenções que o professor fez. Assim, consegue-se identificar os alunos que tem mais dificuldades em um determinado item do Diagrama de Classes.

A princípio, a comunidade a ser beneficiada pela Ferramenta para o Apoio ao Ensino-Aprendizagem do Modelo Orientado a Objetos Durante a Construção do Diagrama de Classes são os professores e alunos que utilizam o Visio para a criação do diagrama de classes.

Como trabalhos futuros, o “Gerador de Logs” da presente ferramenta pode ser desenvolvido e adaptado para ser utilizado como *plugin* para o Umbrello, o Visual Paradigm, o Astah, entre outros, aumentando o campo de atuação da ferramenta. Além disso, pode-se pensar na criação de uma aplicação que dispense o uso de ferramentas auxiliares, como planilhas eletrônicas, para o cálculo e armazenamento de informações ao longo do tempo. Como isso é possível explorar grandes quantidades de dados à procura de padrões consistentes, como regras de associação ou sequências temporais, para detectar relacionamentos sistemáticos entre variáveis, detectando assim novos subconjuntos de dados usando o processo de Data Mining.

REFERÊNCIAS

Anquan, J., Yuqing, L., Bailiang, C., Jihua, Y. & Jie, Z. The education reform and innovation of object-oriented programming course in Normal University. in Computer Science and Education (ICCSE), 2010 5th International Conference on 700–703 (IEEE, 2010).

Bardin, L. Análise de conteúdo (3a ed.). Lisboa: Edições 70, (2007).

Boticki, I., Katic, M. & Martin, S. Exploring the Educational Benefits of Introducing Aspect-Oriented Programming Into a Programming Course. IEEE Transactions on Education 56, 217–226 (2013).

Brinda, T. Discovery learning of object-oriented modelling with exploration modules in secondary Informatics education. Education and Information Technologies 11, 105–119 (2006).

Coad, P. & Yourdon, E. Análise Baseada em Objetos. Rio de Janeiro: Campus, (1992).

Dorn, B. & Sanders, D. Using Jeroo to introduce object-oriented programming. in Frontiers in Education, 2003. FIE 2003 33rd Annual 1, T4C–22–7 Vol.1 (2003).

Dwarika, J. & de Villiers, M. R. R. Use of the Alice Visual Environment in Teaching and Learning Object-oriented Programming. in Proceedings of the 2015 Annual Research Conference on South African Institute of Computer Scientists and Information Technologists 14:1–14:10 (ACM, 2015).

EE International Conference on Advanced Learning Technologies, 2007. ICALT 2007 660–664 (2007).

Esteves, M., Fonseca, B., Morgado, L. & Martins, P. Improving Teaching and Learning of Computer Programming through the Use of the Second Life Virtual World. British Journal of Educational Technology 42, 624–637 (2011).

Fournier, J.-P. ActiveTutor. in Fifth IEEE International Conference on Advanced Learning Technologies, 2005. ICALT 2005 113–115 (2005).

Franco, M.L.P.B. Análise de conteúdo. Brasília: Líber livro Editora, (2007).

Gálvez, J., Guzmán, E. & Conejo, R. A blended E-learning experience in a course of object oriented programming fundamentals. Knowledge-Based Systems 22, 279–286 (2009).

Giordano, D. & Maiorana, F. Object Oriented Design through game development in XNA. in Interdisciplinary Engineering Design Education Conference (IEDEC), 2013.

Giordano, D. & Maiorana, F. Teaching design first; interleaved with object-oriented programming in a software engineering course. in 2014 IEEE Global Engineering Education Conference (EDUCON) 1085–1088 (2014).

Guedes, G.T. A. UML 2. “Uma Abordagem Prática”, São Paulo, Novatec, (2009).

HADDAD, M. C. L. et al. Enfermagem médico-cirúrgica: uma nova abordagem de ensino e sua avaliação pelo aluno. Revista Latino-Am. Enfermagem, Ribeirão Preto, v. 2, (1993).

Henrique, M. S. & Rebouças, A. D. D. S. Objetos de Aprendizagem para auxiliar o ensino de conceitos do Paradigma de Programação Orientada a Objetos. RENOTE 13, (2016).

Hinterholz, O. Tepequém: uma nova Ferramenta para o Ensino de Algoritmos nos Cursos Superiores em Computação. in XVII-Anais do Workshop sobre Educação em Informática (2009).

Hosanee, Y. & Panchoo, S. An enhanced software tool to aid novices in learning Object Oriented Programming (OOP). in 2015 International Conference on Computing, Communication and Security (ICCCS) 1–7 (2015).

Janke, E., Brune, P. & Wagner, S. Does Outside-In Teaching Improve the Learning of Object-Oriented Programming? in 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering (ICSE) 2, 408–417 (2015).

Jonsson, H. Using flipped classroom, peer discussion, and just-in-time teaching to increase learning in a programming course. in Frontiers in Education Conference (FIE), 2015. 32614 2015. IEEE 1–9 (IEEE, 2015).

Kanakaraddi, S. G., Naragund, J. G. & Chikaraddi, A. K. Active learning methods for teaching OOAD course. in MOOC Innovation and Technology in Education (MITE), 2013 IEEE International Conference in 47–52 (2013).

Kayama, M., Ogata, S., Masymoto, K., Hashimoto, M. & Otani, M. A Practical Conceptual Modeling Teaching Method Based on Quantitative Error Analyses for Novices Learning to Create Error-Free Simple Class Diagrams. in 2014 IIAI 3rd International Conference on Advanced Applied Informatics (IIAIAAI) 616–622 (2014).

Kitchenham, B. Procedures for Performing Systematic Reviews. Joint Technical Report, TR/SE-0401 and NICTA 0400011T.1, Keele University, (2004).

Krause, P. An Achievement Degree Analysis Approach to Identifying Learning Problems in Object-Oriented Programming. Trans. Comput. Educ. 14, 20:1–20:15 (2014).

Kuljis, J. Orienting the teaching of an introductory object-oriented programming to meet the learning objective. in 26th International Conference on Information Technology Interfaces, 2004 401–406 Vol.1 (2004).

Larman, C. Utilizando UML e Padrões. Editora Bookman, (2007).

Li, L. & Xu, Y. The teaching research on a case of object-oriented programming. in 2010 5th International Conference on Computer Science&Education 619–621 (2010).

Liu, L., Liu, J., Zhuang, H. & Wang, Z. LCM Exploration and Practice in OOP Teaching. in Scalable Computing and Communications; Eighth International Conference on Embedded Computing, 2009.

Livovský, J. & Porubän, J. Learning object-oriented paradigm by playing computer games: concepts first approach. centr.eur.j.comp.sci. 4, 171–182 (2014).

Martín, E., Lázaro, C. & Hernán-Losada, I. Active learning in Telecommunication Engineering: A case study. in 2010 IEEE Education Engineering (EDUCON) 1555–1562 (2010).

Moreira, M. A., & Masini, E. F. S. Aprendizagem significativa: A teoria de David Ausubel. São Paulo: Editora Moraes (1982).

Moura, I. C. & van Hattum-Janssen, N. Teaching a CS introductory course: An active approach. Computers & Education 56, 475–483 (2011).

Narasimhamurthy, U. & Al Shawkani, K. Teaching of programming languages: An introduction to dynamic learning objects. in International Workshop on Technology for Education, 2009.

Oliveira, C. A., Conte, M. F. & Riso, B. G. Aspects on Teaching/Learning with Object Oriented Programming for Entry Level Courses of Engineering. (1998).

Pears, A. et al. A Survey of Literature on the Teaching of Introductory Programming. in Working Group Reports on ITiCSE on Innovation and Technology in Computer Science Education 204–223 (ACM, 2007).

Portillo, J. A. P.-S. & Campos, P. G. The Jigsaw Technique: Experiences Teaching Analysis Class Diagrams. in 2009 Mexican International Conference on Computer Science (ENC) 289–293 (2009).

Rais, A. E., Sulaiman, S. & Syed-Mohamad, S. M. Game-based approach and its feasibility to support the learning of object-oriented concepts and programming. in Software Engineering (MySEC), 2011 5th Malaysian Conference in 307–312 (2011).

Satratzemi, M., Xinogalos, S. & Dagdilelis, V. An environment for teaching object-oriented programming: objectKarel. in The 3rd IEEE International Conference on Advanced Learning Technologies, 2003. Proceedings 342–343 (2003).

Stein Júnio, A.V., Malucelli, A. & Bastos, L.C. Especificação de sistema de informação de microáreas de risco utilizando a abordagem orientada a objetos. Revista Eletrônica de Enfermagem, [S.l.], v. 11, n. 4, p. 866-76, dez. 2009. ISSN 1518-1944, (2009)

Thomasson, B., Ratcliffe, M. & Thomas, L. Identifying Novice Difficulties in Object Oriented Design. in Proceedings of the 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education 28–32 (ACM, 2006).

Westin, L. K. & Nordstrom, M. Teaching OO concepts-a new approach. in Frontiers in Education, 2004. FIE 2004. 34th Annual F3C–6–11 Vol. 2 (2004).

Winbland, E. Software orientado a objeto. São Paulo : Makron Books, (1993).

Wirfs-Brock, R. J. Looking for powerful abstractions [object oriented technology]. IEEE Software 23, 13–15 (2006).

Xinogalos, S. Object-Oriented Design and Programming: An Investigation of Novices' Conceptions on Objects and Classes. Trans. Comput. Educ. 15, 13:1–13:21 (2015).

Xinogalos, S., Sartatzemi, M., Dagdilelis, V. & Evangelidis, G. Teaching OOP with BlueJ: A case study. in null 944–946 (IEEE, 2006).

Xinogalos, S., Satratzemi, M. & Dagdilelis, V. Re-designing an OOP course based on BlueJ. in Seventh IE

Yan, L. Teaching Object-Oriented Programming with Games. in Sixth International Conference on Information Technology: New Generations, 2009. ITNG '09 969–974 (2009).

Yulia & Adipranata, R. Teaching object oriented programming course using cooperative learning method based on game design and visual object oriented environment. in 2010 2nd International Conference on Education Technology and Computer (ICETC) 2, V2–355–V2–359 (2010).

Zhu, H. Teaching OOP With Financial Literacy. IEEE Transactions on Education 54, 328–331 (2011).