UNIVERSIDADE DE SÃO PAULO

Instituto de Ciências Matemáticas e de Computação

Uma infraestrutura de apoio ao desenvolvimento de aplicações educacionais móveis para o ensino e aprendizagem de fundamentos de programação

Anderson da Silva Marcolino

Tese de Doutorado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-CCMC)

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura: _____

Anderson da Silva Marcolino

Uma infraestrutura de apoio ao desenvolvimento de
aplicações educacionais móveis para o ensino e
aprendizagem de fundamentos de programação

Tese apresentada ao Instituto de Ciências
Matemáticas e de Computação – ICMC-USP,
como parte dos requisitos para obtenção do título
de Doutor em Ciências – Ciências de Computação e
Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e
Matemática Computacional

Orientadora: Profa. Dra. Ellen Francine Barbosa

USP – São Carlos
Julho de 2019

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

d545i da Silva Marcolino, Anderson
Uma infraestrutura de apoio ao desenvolvimento
de aplicações educacionais móveis para o ensino e
aprendizagem de fundamentos de programação /
Anderson da Silva Marcolino; orientador Ellen
Francine Barbosa. -- São Carlos, 2019.
211 p.

Tese (Doutorado - Programa de Pós-Graduação em
Ciências de Computação e Matemática Computacional) --
Instituto de Ciências Matemáticas e de Computação,
Universidade de São Paulo, 2019.

1. Fundamentos de Programação. 2. Aprendizagem
Móvel. 3. Linhas de Produto de Software. 4.
Engenharia de Software baseada em Reúso. I.
Francine Barbosa, Ellen, orient. II. Título.

Anderson da Silva Marcolino

**An infrastructure to support the development of mobile
learning applications for the teaching and learning of
programming fundamentals**

Doctoral dissertation submitted to the Institute of
Mathematics and Computer Sciences – ICMC-USP, in
partial fulfillment of the requirements for the degree of
the Doctorate Program in Computer Science and
Computational Mathematics. *FINAL VERSION*

Concentration Area: Computer Science and
Computational Mathematics

Advisor: Profa. Dra. Ellen Francine Barbosa

USP – São Carlos
July 2019

AGRADECIMENTOS

Primeiramente a Deus, pela oportunidade de alcançar o que, a princípio, era inalcançável. À minha Orientadora, Profa. Dra. Ellen Francine Barbosa, pela paciência, amizade, compreensão, sabedoria, e por ter me orientado muito além da pesquisa. Aos meus pais, Adauto Aparecido Marcolino e Ivanilde Júlia da Silva Marcolino, minha Família, amigos e colegas, meus professores, orientadores e àqueles que se foram, mas me ensinaram muito.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

*“São as nossas escolhas que revelam o que realmente somos,
muito mais do que as nossas qualidades.”*
(J. K. Rowling)

RESUMO

MARCOLINO, A. S. **Uma infraestrutura de apoio ao desenvolvimento de aplicações educacionais móveis para o ensino e aprendizagem de fundamentos de programação.** 2019. 211 p. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2019.

A importância das disciplinas de programação nas grades curriculares de cursos de computação, tanto no contexto nacional como internacional e as limitações e problemas apresentados em tais disciplinas, tem levado a um esforço significativo em pesquisas na área da Informática na Educação para propor soluções de software no domínio de ensino e aprendizagem de programação, com destaque para a adoção de soluções para a modalidade de aprendizagem eletrônica (*e-learning*). No entanto, é possível notar limitações nas pesquisas já conduzidas, principalmente em relação à quantidade ainda reduzida de estudos que adotam a aprendizagem móvel (*m-learning*) como modalidade para ensinar a programar. Observa-se, ainda, a incipiência de pesquisas que adotam metodologias provenientes da Engenharia de Software baseada em Reúso para apoiar o processo de criação de produtos de software educacionais. A adoção de tais metodologias pode permitir uma variação maior de produtos educacionais, permitindo o desenvolvimento de modo a atender um maior número de professores, instituições e domínios. Neste contexto, esta pesquisa de doutorado tem como objetivo propor e desenvolver uma infraestrutura para o desenvolvimento de aplicações educacionais móveis para o ensino de fundamentos de programação a serem utilizadas em nível superior. Para isso, diferentes soluções já existentes foram avaliadas permitindo a criação de um catálogo de requisitos educacionais móveis que serviu de artefato para o estabelecimento de uma arquitetura de linha de produtos de software (LPS) e de uma ferramenta de apoio, intitulada TANGRAM (*an infrasTructure to support the development of mobile leArning applicatioNs for the teachinG and leaRning of progrAMming fundamentals*), que permite a criação facilitada dos produtos dessa linha. Adicionalmente, um conjunto inicial de funcionalidades para a plataforma móvel foi desenvolvido. Para verificar a viabilidade do uso de tal infraestrutura, diversas avaliações foram conduzidas nas diferentes etapas de concepção da mesma. Ao final, foram observadas evidências preliminares de que a LPS e a ferramenta TANGRAM possibilitam a criação de aplicações educacionais móveis para serem utilizadas na mitigação de problemas no ensino e aprendizagem de fundamentos de programação. Além de contribuições complementares referentes ao catálogo de requisitos para aplicações educacionais móveis, as escolhas metodológicas e tecnológicas na concepção do projeto arquitetural, e um conjunto de gestos de interação para telas sensíveis ao toque a integrar as aplicações educacionais, que possibilitam a condução de trabalhos futuros na exploração dos mesmos no que se refere à mitigação de problemas no ensino e aprendizagem de fundamentos de programação.

Palavras-chave: Fundamentos de Programação, Aprendizagem Móvel, Linhas de Produto de Software, Engenharia de Software baseada em Reúso.

ABSTRACT

MARCOLINO, A. S. **An infrastructure to support the development of mobile learning applications for the teaching and learning of programming fundamentals.** 2019. 211 p. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2019.

The importance of programming disciplines in computer curricula courses worldwide has increasing the efforts of researches in informatics in education area to propose software for the mitigation of problems in programming domain, highlighting the adoption of electronic learning solutions (e-learning). Meanwhile, it is possible to identify limitations in researches that adopting other learning modalities, as mobile learning modality (m-learning) for teaching of programming. Furthermore, it is also noticed a lack of adoption of reused-based software engineering methodologies, for supporting the process of development of educational software for programming domain, in special, software product lines (SPL). The adoption of such methodologies may allow a higher variation of educational products and a higher support for a greater number of professors and institutions. In this perspective, this doctoral research aimed at the establishment of an infrastructure to support the development of m-learning applications for the teaching of programming fundamentals in undergraduates courses. For that, several software solutions in programming domain were analysed allowing the creation of a m-learning requirement catalog. Such catalog was used as the main artifact for the establishment of an SPL and a support tool entitled TANGRAM (*an infrasTructure to support the development of mobile leArning applicatioNs for the teachinG and leaRning of progrAMming fundamentals*). The tool was proposed to facilitate the creation of applications considering the SPL's artifacts and their management. Additionally, an initial set of features for the mobile learning applications were developed, e.g., a catalog of gestures. To investigate the viability in the use of such infrastructure, several evaluations were conducted in each phase of conception of the SPL. At the end, preliminar positive evidence indicates that both LPS and TANGRAM tool may allow the development of m-learning applications for the mitigation of problems in the teaching and learning of programming fundamentals. Besides, contributions related with the m-learning requirement catalog, the methods and technologies selected for the conduction of the architectural project, and a set of gesture for touchscreen interfaces were also identified, resulting in an infrastructure which may be better investigate and adopted to improve the processes of teaching and learning in the programming fundamentals domain.

Keywords: Programming Fundamentals, Mobile Learning, Software Product Lines, Reuse-Based Software Engineering.

LISTA DE ILUSTRAÇÕES

Figura 1 – Resumo das Atividades da Pesquisa de Doutorado.	27
Figura 2 – Informações sobre o Questionário: a) Regiões do Professores Participantes, b). Perfil dos Professores, c). Atividades do Questionário e d). Perfil dos alunos.	34
Figura 3 – Composição e Características do <i>m-learning</i>	43
Figura 4 – Classificação Geral de Sistemas de <i>m-Learning</i>	44
Figura 5 – Aplicações: a) <i>Moodle Mobile 2</i> e b) <i>Blackboard Mobile</i>	46
Figura 6 – Atividades Essenciais de LPS.	48
Figura 7 – Ciclo de Vida do Arcabouço SPLE.	50
Figura 8 – Estratégias (a) Extrativa, (b) Reativa e (c) Proativa.	52
Figura 9 – Custo Acumulado por Número de Sistemas.	54
Figura 10 – Atividades do Processo de Busca e Seleção para LPS Educacionais.	59
Figura 11 – Atividades do Processo de Busca e Seleção de Estudos Primários.	67
Figura 12 – Estratégias de Software e Ano de Desenvolvimento.	71
Figura 13 – Fases de Concepção do Catálogo de Requisitos.	77
Figura 14 – Composição para o Estudo de Ivanovi, Xinogalos e Komlenov (2011).	80
Figura 15 – Composição para o Estudo de Watson, Li e Lau (2011).	80
Figura 16 – Excertos de dois Mockups da Aplicação Móvel Avaliada.	89
Figura 17 – Página Principal do Catálogo de Requisitos.	95
Figura 18 – Exemplo de Apresentação de Requisito.	96
Figura 19 – Visão Macro da Arquitetura da LPS e da Ferramenta de Apoio.	103
Figura 20 – Resultados Gerais da Avaliação.	109
Figura 21 – Resultados da Avaliação por Conjuntos de Aspectos.	109
Figura 22 – Resultados da Avaliação - Professores de Programação.	110
Figura 23 – Resultados da Avaliação - Engenheiros de Software.	110
Figura 24 – Resultados da Avaliação - Desenvolvedor Móvel.	110
Figura 25 – Resultados da Avaliação - Especialistas M-learning.	111
Figura 26 – Atividades para a Evolução do Modelo Arquitetural.	113
Figura 27 – Modelo Conceitual Atualizado da Arquitetura das Aplicações Móveis.	114
Figura 28 – Modelo de Componentes da LPS.	117
Figura 29 – Modelo de Classes do microserviço de Conteúdo Educacional.	119
Figura 30 – Classificação de Assertividade do Modelo da Arquitetura.	120
Figura 31 – Modelo de Componentes Final da Arquitetura LPS.	124

Figura 32 – Diagrama de Atividades da Integração do SPLE com XP e TDD.	126
Figura 33 – Página Inicial da TANGRAM desenvolvida com Angular.	128
Figura 34 – Página de Criação da Aplicação com Angular.	129
Figura 35 – Página Inicial da TANGRAM desenvolvida com React.	129
Figura 36 – Página do Painel de Controle de Criação da Aplicação com React.	130
Figura 37 – Itens Candidatos a Variações no Contexto dos Produtos da LPS.	133
Figura 38 – Tela Seleção das <i>Features</i> para Criação de Configuração.	137
Figura 39 – Modelo de Características dos Módulos Educacionais para o Ensino Formal.	138
Figura 40 – Modelo de Características dos Módulos Educacionais para o Ensino Informal.	138
Figura 41 – Página de Criação das Questões e Seleção das <i>Tags</i>	140
Figura 42 – Página de Criação das Atividades.	143
Figura 43 – Tela Principal da TANGRAM.	144
Figura 44 – Níveis de Conhecimento dos Participantes.	145
Figura 45 – Total Participantes que Já Utilizaram Ferramentas Educacionais.	146
Figura 46 – <i>Box Plot</i> das Respostas para as Questões do SUS.	148
Figura 47 – Excerto da Ajuda da Ferramenta TANGRAM.	149
Figura 48 – Catálogo de Gestos <i>M-learning</i> e Trechos de Código que Representam.	152
Figura 49 – <i>Gesture Tool</i> e Usuário Inserindo Gesto durante a Segunda Avaliação.	153
Figura 50 – Editor <i>CodeMirror</i> contendo código <i>HTML</i> e <i>JavaScript</i>	154
Figura 51 – Utilização da TANGRAM para Seleção da <i>feature</i> de Gestos.	155
Figura 52 – Resumo das Atividades e Resultados da Pesquisa.	159
Figura 53 – Resumo das Atividades Futuras.	163
Figura 54 – Informações sobre o Questionário: a) Regiões do Professores Participantes, b). Perfil dos Professores, c). Atividades do Questionário e d). Perfil dos alunos.	187
Figura 55 – Ranque de Resultados por Categoria de Problemas.	189
Figura 56 – Diagrama do Paradigma Procedural.	203
Figura 57 – Diagrama do Paradigma Orientado a Objetos.	204
Figura 58 – Gestos Selecionados na Primeira Avaliação.	207
Figura 59 – <i>Gesture Tool</i> e Usuário Inserindo Gesto durante a Segunda Avaliação.	209
Figura 60 – Número de Tentativas por Estrutura de Código.	209
Figura 61 – Satisfação dos Participantes na Utilização dos Gestos.	210

LISTA DE QUADROS

Quadro 1 – Categorias de Problemas de Programação.	33
Quadro 2 – Respostas dos Professores para Segunda (QP2) e Terceira (QP3) Questões.	36
Quadro 3 – Respostas dos Alunos para o Pré-teste (QEA) e Pós-teste (QEB).	38
Quadro 4 – Vantagens e Desvantagens de <i>M-learning</i>	45
Quadro 5 – Tecnologias Adotadas.	63
Quadro 6 – Problemas Mapeados.	64
Quadro 7 – Modalidades de Ensino e Especificações Técnicas.	70
Quadro 8 – Classificação das Referências por Categorias de Problemas em Programação.	79
Quadro 9 – Comparação das Funcionalidades das Aplicações <i>m-learning</i>	82
Quadro 10 – Requisitos de Conteúdo de Aprendizagem.	83
Quadro 11 – Requisitos de Atividades de Aprendizagem.	84
Quadro 12 – Requisitos de <i>Feedback</i> e Resultados.	84
Quadro 13 – Requisitos de Monitoramento e Desempenho de Aprendizagem.	85
Quadro 14 – Requisitos de Mecanismos para o Ensino e Aprendizagem de Programação.	85
Quadro 15 – Requisitos de Avaliação de Aprendizagem.	86
Quadro 16 – Requisitos de Apoio ao Ensino e Aprendizagem.	86
Quadro 17 – Requisitos de Usuários.	87
Quadro 18 – Requisitos Não Funcionais.	87
Quadro 19 – Requisitos da Prova de Conceito.	90
Quadro 20 – Justificativas para Classificação dos Participantes.	121
Quadro 21 – Vantagens e Desvantagens da Arquitetura segundo Participantes.	122
Quadro 22 – Sugestões e Críticas por Participante.	149
Quadro 23 – Estudos Primários Recuperados.	196
Quadro 24 – Estudos Primários Recuperados.	198
Quadro 25 – Estrutura de Códigos em C e Python.	205
Quadro 26 – Resultados da Primeira Avaliação do Catálogo de Gestos.	207

LISTA DE TABELAS

Tabela 1 – Categorias de Problemas de Programação e Níveis de Incidência.	35
Tabela 2 – Estudos Primários Recuperados por Base de Busca.	60
Tabela 3 – Estudos Primários Seleccionados Ordenados por Ano.	60
Tabela 4 – Estudos Primários Recuperados por Base de Busca.	68
Tabela 5 – Catálogo de software e Aplicações Educacionais para o Ensino de Programação	69
Tabela 6 – Síntese das Respostas da Avaliação Qualitativa.	92
Tabela 7 – Resultados Coletados e Estatística Descritiva do SUS.	147
Tabela 8 – Categorias de Problemas de Programação e Níveis de Incidência	188
Tabela 9 – Estudos Primários Recuperados por Base de Busca.	195
Tabela 10 – Estudos Primários Recuperados por Base de Busca.	197

SUMÁRIO

1	INTRODUÇÃO	23
1.1	Contexto e Motivação	23
1.2	Objetivos e Questões da Pesquisa	26
1.3	Organização	28
2	FUNDAMENTAÇÃO TEÓRICA	31
2.1	Considerações Iniciais	31
2.2	Ensino e Aprendizagem de Programação	31
2.2.1	<i>Problemas no Ensino e Aprendizagem de Programação</i>	32
2.2.2	<i>Questionários sobre Problemas no Ensino e Aprendizagem de Pro- gramação</i>	33
2.3	Aprendizagem Móvel	39
2.4	Linhas de Produto de Software	46
2.4.1	<i>Atividades Essenciais</i>	48
2.4.2	<i>O Arcabouço SPLE</i>	49
2.4.3	<i>A Abordagem SMarty</i>	51
2.4.4	<i>Estratégias de Concepção do Núcleo de Artefatos</i>	52
2.4.5	<i>Benefícios e Limitações</i>	53
2.5	Considerações Finais	55
3	LPS EDUCACIONAIS E SOFTWARES PARA O ENSINO E APREN- DIZAGEM DE PROGRAMAÇÃO	57
3.1	Considerações Iniciais	57
3.2	Linhas de Produto de Software no Domínio Educacional e de Ensino e Aprendizagem de Programação	57
3.2.1	<i>Metodologia, Planejamento e Execução</i>	58
3.2.2	<i>Análise dos Resultados</i>	60
3.2.3	<i>Discussões Gerais e Lacunas Encontradas</i>	64
3.3	Software Educacionais para o Ensino e Aprendizagem de Programação	66
3.3.1	<i>Metodologia, Planejamento e Execução</i>	66
3.3.2	<i>Análise dos Resultados</i>	67
3.3.3	<i>Discussões Gerais e Lacunas Encontradas</i>	72
3.4	Considerações Finais	73

4	CATÁLOGO DE REQUISITOS <i>M-LEARNING</i> NO DOMÍNIO DE PROGRAMAÇÃO	75
4.1	Considerações Iniciais	75
4.2	Proposta do Catálogo	76
4.3	Metodologia e Concepção	77
4.4	O Catálogo de Requisitos <i>M-learning</i> no Domínio de Programação	83
4.5	Avaliações do Catálogo de Requisitos	88
4.5.1	<i>Estudo de Caso</i>	88
4.5.2	<i>Questionário com Professores</i>	90
4.6	Ameaças à Validade das Avaliações	94
4.7	O Catálogo Final	95
4.8	Considerações Finais	96
5	UMA LPS PARA APLICAÇÕES <i>M-LEARNING</i> NO DOMÍNIO DE PROGRAMAÇÃO	99
5.1	Considerações Iniciais	99
5.2	Gerenciamento de Produtos	100
5.3	Engenharia de Requisitos de Domínio	100
5.4	Projeto de Domínio	101
5.5	Avaliação do Modelo Conceitual da Arquitetura	107
5.5.1	<i>Projeto e Execução</i>	108
5.5.2	<i>Resultados</i>	108
5.6	Evolução da Arquitetura	112
5.6.1	<i>Metodologia</i>	112
5.6.2	<i>Condução da Evolução da Arquitetura</i>	113
5.6.3	<i>Avaliação dos Novos Modelos Arquiteturais</i>	118
5.6.4	<i>Realização de Domínio e Testes de Domínio</i>	123
5.6.5	<i>Contribuições da Arquitetura no Domínio de Ensino e Aprendizagem de Programação</i>	131
5.7	Considerações Finais	134
6	A FERRAMENTA TANGRAM	135
6.1	Considerações Iniciais	135
6.2	Visão Geral da Ferramenta	136
6.3	Avaliação de Usabilidade com Professores	144
6.3.1	<i>Planejamento, Metodologia e Execução</i>	145
6.3.2	<i>Descrição e Análise dos Resultados</i>	145
6.3.3	<i>Ameaças à Validade da Avaliação</i>	148
6.4	Aplicações para o Ensino e Aprendizagem de Programação	150

6.4.1	<i>Desenvolvimento da Aplicação Móvel e Integração dos Gestos como Serviços</i>	153
6.5	Considerações Finais	156
7	CONCLUSÕES	157
7.1	Contribuições da Pesquisa	160
7.2	Limitações da Pesquisa	162
7.3	Trabalhos Futuros	162
7.4	Publicações Resultantes	164
REFERÊNCIAS		167
APÊNDICE A QUESTIONÁRIO SOBRE PROBLEMAS NO ENSINO E APRENDIZAGEM DE PROGRAMAÇÃO		
		185
A.1	Questionário sobre Problemas no Ensino e Aprendizagem de Programação	185
A.1.1	<i>Análise das Respostas do Questionário com Professores</i>	186
A.1.2	<i>Análise das Respostas do Questionário com alunos</i>	191
A.1.3	<i>Desafios no Ensino e Aprendizagem de Programação</i>	194
APÊNDICE B LPS EDUCACIONAIS E SOFTWARES PARA O ENSINO E APRENDIZAGEM DE PROGRAMAÇÃO: UMA NOVA ITERAÇÃO		
		195
B.1	LPSs no Domínio Educacional e no Domínio de Programação	195
B.2	Softwares para o Ensino e Aprendizagem de Programação	197
B.3	Conclusões	199
APÊNDICE C CATÁLOGO DE GESTOS PARA AS APLICAÇÕES EDUCACIONAIS MÓVEIS		
		201
C.1	Catálogo de Gestos para as Aplicações Educacionais Móveis	201
C.1.1	<i>Planejamento e Concepção</i>	201
C.1.2	<i>Primeira Avaliação do Catálogo de Gestos</i>	205
C.1.2.1	<i>Análise dos Resultados</i>	206
C.1.3	<i>Segunda Avaliação do Catálogo de Gestos</i>	208
C.1.3.1	<i>Análise dos Resultados</i>	208
C.1.4	<i>Ameaças à Validade das Avaliações</i>	210

INTRODUÇÃO

Esta tese de doutoramento visa o estabelecimento de uma linha de produtos de software (LPS) e uma ferramenta, intitulada TANGRAM, para apoiar o desenvolvimento de aplicações móveis educacionais para o ensino e aprendizagem de fundamentos de programação. À união desta LPS, de seus diferentes artefatos, e da ferramenta de apoio é referida nesta pesquisa como uma infraestrutura. O objetivo em se estabelecer tal infraestrutura é de permitir a condução de investigações quanto à combinação de diferentes características, a fim de mitigar problemas no domínio de programação considerando-se a plataforma móvel.

Este capítulo apresenta o contexto e motivação para a condução desta tese de doutorado (Seção 1.1); os objetivos, questões de pesquisa e aspectos metodológicos relacionados (Seção 1.2); e, ao final, a organização textual dos demais capítulos que compõem esta tese (Seção 1.3).

1.1 Contexto e Motivação

Diversas iniciativas têm sido conduzidas para garantir a melhora do processo de ensino e aprendizagem em diferentes áreas, cursos e disciplinas (JOHNSON; ADAMS; CUMMINS, 2016; HAWKRIDGE; VINCENT; HALES, 2018). Na área de computação, por exemplo, tem-se observado a intensificação de esforços em disciplinas que envolvem aspectos teóricos e práticos de fundamentos de programação (SOUZA; BATISTA; BARBOSA, 2015; DUNCAN; BELL; TANIMOTO, 2014; MARCOLINO; BARBOSA, 2017a). Esses esforços são reflexo, sobretudo, da importância das disciplinas de programação nas grades curriculares de cursos como Ciência da Computação, Sistemas de Informação, Engenharia de Computação e afins, tanto no contexto nacional (SBC, 2003) como no cenário internacional (PRESS; PRESS, 2005).

Aprender a programar consiste, de fato, em um processo difícil e exigente para a maioria dos alunos. Mesmo a aprendizagem de conceitos básicos e sua aplicação na resolução de problemas concretos pode levar a questões complexas para muitos alunos (SALES; DANTAS,

2010). Neste contexto, a adoção de software educacionais em tal domínio tem sido considerada e incentivada (SOUZA; BATISTA; BARBOSA, 2015; DUNCAN; BELL; TANIMOTO, 2014; MARCOLINO; BARBOSA, 2017a; PAPADAKIS *et al.*, 2019; POPAT; STARKEY, 2019). Porém, mesmo com diferentes ferramentas e software que apoiam as disciplinas de programação, este domínio ainda apresenta diversos problemas e limitações que carecem de melhores soluções. No caso dos alunos, há dificuldades na aprendizagem dos conceitos de programação, a aplicação de tais conceitos, o entendimento do funcionamento de programas e falta de motivação (SOUZA; BATISTA; BARBOSA, 2015; DUNCAN; BELL; TANIMOTO, 2014; MARCOLINO; BARBOSA, 2017a; GOMES; MENDES, 2007; ESPER; FOSTER; GRISWOLD, 2013). No caso dos professores, há dificuldades relacionadas à criação de conteúdos e atividades educacionais e em realizar acompanhamento individual em salas com um grande contingente de alunos (SOUZA; BATISTA; BARBOSA, 2015; DUNCAN; BELL; TANIMOTO, 2014; MARCOLINO; BARBOSA, 2017a; GOMES; MENDES, 2007; ESPER; FOSTER; GRISWOLD, 2013).

Nesta perspectiva, a investigação e o desenvolvimento de software direcionados a mitigar e sanar os problemas de ensino e aprendizagem de programação fazem-se cada vez mais necessários. Por outro lado, é sabido que o processo de desenvolvimento de software é custoso e complexo (PRESSMAN, 2015; SOMMERVILLE, 2010), sendo tais problemas ainda mais significativos quando considerados em domínios mais complexos. O domínio educacional, por exemplo, integra questões socioculturais, econômicas, legais, tecnológicas e didático-pedagógicas (AROYO; DICHEVA, 2004; BACICH; NETO; TREVISANI, 2015; MURRAY; BLESSING; AINSWORTH, 2003; MARCOLINO; BARBOSA, 2016; CRAIG *et al.*, 2012).

Os software educacionais devem atender com sucesso o objetivo educacional para o qual foram concebidos e, ainda, contemplar conceitos e práticas procedentes de uma série de teorias educacionais (ILLERIS, 2018; ABACHI; MUHAMMAD, 2014; MARCOLINO; BARBOSA, 2016; RITTER; MURRAY; HAUSMANN, 2018; BELLANCA, 2010; AROYO; DICHEVA, 2004). Adicionalmente, diferentes soluções tecnológicas devem proporcionar aos alunos um ambiente motivante à aprendizagem, incluindo, muitas vezes, funcionalidades que se assemelham às encontradas em aplicações de redes sociais e de lazer (MARCOLINO; BARBOSA, 2017a; ANDERSON; DRON, 2018). Logo, a simples adoção de software educacionais não corrobora na melhoria dos processos de ensino e aprendizagem se estes não estiverem alinhados com as expectativas dos alunos e professores e às práticas pedagógicas (FOFONCA *et al.*, 2018a).

Surge, portanto, a necessidade de se repensar os meios pelos quais será garantido que o processo educacional resulte em níveis satisfatórios de aprendizagem, tais como os obtidos em sala de aula ou, até mesmo, àqueles que ultrapassem tal nível, visto que surgem recursos que, até então, não estavam disponíveis nos ambientes físicos (e.g., acompanhamento, monitoramento e *feedback* personalizado) (SOUZA; BATISTA; BARBOSA, 2015; FOFONCA *et al.*, 2018a; FOFONCA *et al.*, 2018b; SINCLAIR; AHO, 2018). Esses recursos têm sido fornecidos, em sua maioria, pela adoção de diferentes tecnologias da informação e comunicação (TIC) e das

modalidades de aprendizagem que delas são derivadas (FOFONCA *et al.*, 2018a; JAMIESON-PROCTOR, 2018; IEA, 2018; BARTHOLO; AMARAL; CAGNIN, 2009; ALRASHEEDI; CAPRETZ, 2018; FU; HWANG, 2018; HEFLIN; SHEWMAKER; NGUYEN, 2017; PIMMER; MATEESCU; GRÖHBIEL, 2016).

De fato, TIC e suas modalidades de ensino, com destaque para os ambientes de aprendizagem móvel (*m-learning*, do inglês *mobile learning*), têm levado o processo de ensino e aprendizagem para além dos ambientes físicos das escolas. Por meio do *m-learning*, alunos têm a possibilidade de aprender em qualquer lugar e a qualquer momento. Já os professores possuem canal direto com seus alunos por meio de funcionalidades presentes em tais soluções de software (CROMPTON, 2013; BARAN, 2014; SOAD *et al.*, 2017a; IEA, 2018).

As diferentes especificidades dos dispositivos móveis, por sua vez, ampliam ainda mais a complexidade no desenvolvimento de aplicações educacionais, uma vez que são incluídos novos meios de interação, sensores, telas reduzidas, consumo de energia, uso de redes móveis, etc. (FALVO JUNIOR *et al.*, 2014; DUARTE FILHO; BARBOSA, 2013; SOAD *et al.*, 2017a; MARCOLINO; BARBOSA, 2016). Como reflexo dessa complexidade, há o aumento do custo no processo de desenvolvimento destas soluções (FALVO JUNIOR *et al.*, 2014; SOMMERVILLE, 2010; CHIMALAKONDA; NORI, 2018), que ocorrem, muitas vezes, sem a integração de estratégias de reúso.

A adoção de estratégias de reúso provenientes da Engenharia de Software estabelecem meios de criar aplicações que, considerando a possibilidade de seleção e variação de diferentes funcionalidades, possibilitam um melhor apoio aos professores e alunos, visto que se torna possível desenvolver com menor custo e em menor prazo, soluções diversas (FALVO JUNIOR *et al.*, 2014; CHIMALAKONDA; NORI, 2018), com redução da complexidade e integração de meios para superar limitações no domínio educacional que forem aplicadas. Isso não é facilmente obtido quando essas aplicações são desenvolvidas utilizando metodologia tradicional ou singular, a qual é projetada para atender um conjunto seletivo de requisitos e, em consequência, abrangem soluções mais pontuais (MARCOLINO; BARBOSA, 2015b; METZGER; POHL, 2014; FALVO JUNIOR *et al.*, 2014; JUNIOR; GIMENES; MALDONADO, 2010).

No contexto de abordagem de reúso, destacam-se as LPS, que têm suas decisões de projeto postergadas para fases posteriores de desenvolvimento, permitindo, assim, a criação de um número mais amplo de produtos finais. No contexto educacional, LPS podem permitir a criação de diferentes configurações de produtos por meio da combinação de diferentes *features*, facilitando também a integração de diferentes elementos educacionais a serem disponibilizadas nas soluções, i.e., conteúdos educacionais, atividades educacionais (POHL; BOCKLE; LINDEN, 2005; GOMAA, 2004; GUANA; CORREAL, 2013; FIORI *et al.*, 2012; DHUNGANA *et al.*, 2010).

Assim, a adoção de LPS para conceber software para o ensino de programação pode, além de permitir a redução de custos já comprovada em estudos de caso da indústria (SEI,

2013; CAPILLA *et al.*, 2014), auxilia na criação de um laboratório para exploração e criação de soluções que, por meio da integração de diferentes *features*, possam vir a mitigar os problemas neste domínio. Contudo, tanto LPS quanto *m-learning* são ainda pouco explorados no contexto de ensino e aprendizagem de programação, estabelecendo um campo interessante a ser investigado (MARCOLINO; BARBOSA, 2015b; AURELIANO; TEDESCO, 2012; MARCOLINO; BARBOSA, 2015a).

Considerando (i) as lacunas da literatura em relação às aplicações e software educacionais para o domínio de ensino de programação; (ii) os problemas no domínio de ensino de programação; e (iii) a incipiência na exploração de LPS e da modalidade *m-learning*, o presente trabalho de doutorado tem como principal objetivo investigar, propor e desenvolver uma infraestrutura de apoio ao desenvolvimento de aplicações educacionais móveis no ensino e aprendizagem de fundamentos de programação.

1.2 Objetivos e Questões da Pesquisa

Considerando o contexto e motivação apresentados, este projeto de doutorado tem como objetivo principal a proposição de uma infraestrutura de apoio ao desenvolvimento de aplicações educacionais móveis para o ensino e aprendizagem de fundamentos de programação em nível superior. Tal infraestrutura consiste no estabelecimento de uma LPS e de uma ferramenta, denominada TANGRAM, para apoiar a criação das aplicações educacionais móveis. De modo geral, a LPS é direcionada diretamente para professores e desenvolvedores de aplicações educacionais e, indiretamente, para os alunos, que utilizarão os produtos da linha em contextos educacionais.

Aos professores é disponibilizada uma ferramenta de apoio ao desenvolvimento de aplicações educacionais móveis, com o objetivo de facilitar e de reduzir a necessidade de conhecimentos técnicos para a criação de aplicações no contexto de uma LPS. Tais aplicações, por sua vez, podem ser investigadas no sentido de apoiar o ensino e aprendizagem de programação e a mitigação de problemas neste domínio. Para desenvolvedores, além da disponibilização de um catálogo de requisitos para aplicações móveis no domínio de programação, disponibiliza-se uma arquitetura da linha de produtos que, baseada no conceito de serviços e microsserviços, possibilita a reutilização de artefatos e a integração facilitada de novas *features*, promovendo a evolução da LPS de modo mais simplificado. Finalmente, os alunos também se beneficiam, já que podem utilizar as aplicações móveis para aprender ou complementar seus estudos em fundamentos de programação, por meio de seus dispositivos móveis já adotados em outras atividades diárias.

Nesta perspectiva, um conjunto de objetivos específicos foram considerados para atingir-se o objetivo principal:

1. Identificar os problemas educacionais no domínio de ensino de programação e os software

e aplicações existentes para apoiar na mitigação dos mesmos. Para isso, foram realizadas pesquisas com alunos e professores, além da condução de um mapeamento sistemático (MS);

2. Identificar LPS educacionais gerais e no domínio de ensino de programação por meio da condução de um MS;
3. Propor e avaliar um catálogo de requisitos para o desenvolvimento dos diferentes artefatos da LPS e de suas aplicações educacionais móveis, considerando os problemas de programação e as soluções identificadas. Por meio do catálogo e suas avaliações, também foi possível estabelecer um conjunto de funcionalidades candidatas a serem desenvolvidas;
4. Propor e avaliar a arquitetura da LPS e das aplicações associadas com intuito de verificar sua viabilidade de concepção;
5. Desenvolver uma ferramenta de apoio ao desenvolvimento das aplicações educacionais móveis e a avaliar, em relação a sua usabilidade, na perspectiva de professores e desenvolvedores de software e;
6. Desenvolver componentes e demais artefatos relacionados para comporem um grupo inicial de aplicações móveis da LPS, principalmente no que tange ao processo de interação com os dispositivos móveis.

A Figura 1 apresenta um diagrama com as atividades que resumem a proposta deste trabalho de doutorado, necessárias para se alcançar o objetivo principal.

Figura 1 – Resumo das Atividades da Pesquisa de Doutorado.

Fonte: Elaborada pelo autor.

Com respeito à Base Teórica (Figura 1, Item (a)), buscou-se identificar a fundamentação teórica sobre: (i) TIC e *M-learning*; (ii) os processos de ensino e aprendizagem, teorias educacionais, e o ensino e aprendizagem de programação; (iii) problemas no ensino e aprendizagem de programação no cenário mundial e brasileiro; e (iv) os software para ensinar programação.

Com respeito às Decisões de Projeto (Figura 1, Item (b)), um catálogo de requisitos para aplicações móveis educacionais no domínio de programação, integrando teorias educacionais e problemas em tal domínio foi estabelecido. Este catálogo serviu de entrada para a concepção de modelos da arquitetura da LPS e o consequente desenvolvimento de seus artefatos.

Quanto aos aspectos de desenvolvimento, (Figura 1, Item (c)), a ferramenta TANGRAM foi desenvolvida para auxiliar no processo de definição de aplicações a serem renderizadas. Para a condução dos itens (b) e (c) o arcabouço de engenharia de linhas de produto de software (SPLE) (POHL; BOCKLE; LINDEN, 2005) foi adotado, dividindo e permitindo a condução sistêmica das diferentes etapas de concepção da LPS, e consequente infraestrutura.

Adicionalmente, no decorrer da realização das diferentes etapas que constituem este trabalho, avaliações experimentais, principalmente em nível qualitativo, foram conduzidas. Estas tiveram como objetivo avaliar diferentes aspectos dos artefatos propostos e, também, coletar evidências iniciais quanto a questões teóricas, técnicas e educacionais no decorrer do estabelecimento do catálogo, da LPS, da ferramenta e de sua utilização no desenvolvimento de aplicações móveis.

Destarte, a principal questão de pesquisa que guia este estudo é:

- *Uma infraestrutura, composta por uma LPS para criação de aplicações educacionais móveis, e uma ferramenta de apoio para a criação destas aplicações, estabelece meios para mitigar os diferentes problemas no ensino e aprendizagem de fundamentos de programação?*

Finalmente, é importante destacar que este trabalho está em consonância com diferentes trabalhos desenvolvidos no grupo de pesquisa de Computação Aplicada à Educação e Tecnologia Social Avançada (CAEd) e no Laboratório de Engenharia de Software (LabES) do ICMC-USP. Em destaque para a investigação de problemas no ensino e aprendizagem de programação, com a proposição do uso de testes de software para mitigar alguns desses problemas (SOUZA; BATISTA; BARBOSA, 2015); a proposição de uma LPS genérica no domínio móvel (FALVO JUNIOR *et al.*, 2014); a definição de requisitos móveis educacionais de domínio geral (SOAD *et al.*, 2017a); a definição de uma linguagem de padrões móveis educacionais (FIORAVANTI; BARBOSA, 2018b; FIORAVANTI; BARBOSA, 2018a); e o estabelecimento de uma família de experimentos no domínio de ensino e aprendizagem de programação e avaliação de qualidade de aplicações móveis educacionais (SOAD; FILHO; BARBOSA, 2016).

1.3 Organização

Esta tese está organizada da seguinte forma: neste capítulo foram discutidos o contexto, a motivação e os objetivos deste trabalho de doutorado. O Capítulo 2 discute aspectos educacionais

relacionados ao ensino de programação e à modalidade de aprendizagem móvel, e aspectos técnicos e metodológicos sobre abordagens provenientes da engenharia de software baseada em reuso, com destaque para as LPS. O Capítulo 3 apresenta dois mapeamentos sistemáticos conduzidos para identificar LPS educacionais e software educacionais para o ensino de programação. No Capítulo 4 é apresentado o estabelecimento de um catálogo de requisitos móveis no domínio de programação. O Capítulo 5 detalha a proposta, concepção, evolução e desenvolvimento da arquitetura da linha de produto de software para a criação das aplicações móveis educacionais. O Capítulo 6 descreve a ferramenta desenvolvida para apoiar a criação das aplicações educacionais e questões referentes aos produtos da LPS. O Capítulo 7 traz as conclusões desta pesquisa, suas contribuições e trabalhos futuros.

FUNDAMENTAÇÃO TEÓRICA

2.1 Considerações Iniciais

Os principais temas desta pesquisa estão relacionados ao ensino de Computação, no que se refere ao ensino e aprendizagem de fundamentos de programação; à Informática na Educação, no que refere-se às tecnologias da informação e comunicação, em especial *m-learning* e seus aspectos educacionais; e à Engenharia de Software, no que tange a metodologia de linhas de produto de software e o processo de desenvolvimento de software.

Nesse sentido, este capítulo apresenta uma síntese dos fundamentos necessários para a compreensão deste trabalho de doutorado. Primeiramente, são apresentados conceitos relacionados ao ensino e aprendizagem de programação, com destaque à identificação de problemas neste domínio (Seção 2.2). Em seguida são apresentados os conceitos básicos sobre TIC e, principalmente, os relacionados à aprendizagem móvel, suas vantagens, desvantagens e exemplos de ambientes de aprendizagem móvel (Seção 2.3). Por fim, discutem-se abordagens provenientes da Engenharia de Software baseada em reúso, em especial, linhas de produto de software, suas atividades essenciais e algumas das abordagens e estratégias mais significativas, adotadas para apoiar o desenvolvimento da LPS desta pesquisa (Seção 2.4).

2.2 Ensino e Aprendizagem de Programação

Fundamentos de programação é um dos principais tópicos ensinados nas disciplinas introdutórias dos cursos de computação (PAPADAKIS *et al.*, 2019; POPAT; STARKEY, 2019; NG; CHIU, 2019; WATSON; LI, 2014; AURELIANO; TEDESCO, 2012; DUNCAN; BELL; TANIMOTO, 2014; SBC, 2003; PRESS; PRESS, 2005). Mais do que aprender a sintaxe de uma linguagem de programação, em tais disciplinas os alunos devem desenvolver um pensamento computacional, que os permita resolver problemas por meio de habilidades como abstração,

decomposição, generalização, avaliação e projeto de algoritmos (LÓPEZ; GARCÍA-PEÑALVO, 2016). Logo, o desenvolvimento de tais habilidades, vão além do domínio de programação, visto que, os alunos que as desenvolvem se tornam melhor preparados para o mercado de trabalho e também para a vida (WANGENHEIM; NUNES; SANTOS, 2014; SEEHORN *et al.*, 2011; MARCOLINO; BARBOSA, 2017a; AURELIANO; TEDESCO, 2012; DUNCAN; BELL; TANIMOTO, 2014).

Dentre os principais benefícios em aprender a programar, destacam-se: (i) capacidade de desenvolver software (WANGENHEIM; NUNES; SANTOS, 2014; SEEHORN *et al.*, 2011), (ii) estímulo da aprendizagem do pensamento computacional, que se trata de uma abordagem para resolver problemas de um modo que possa ser implementada em um computador envolvendo conceitos como abstração, recursão, iteração, entre outros (WANGENHEIM; NUNES; SANTOS, 2014; WING, 2008); (iii) estímulo da aprendizagem quanto ao pensamento matemático, visto que são aplicados formalismos matemáticos ao se programar, além de estimular a aprendizagem da matemática por meio de atividades exploratórias e solução de problemas (WANGENHEIM; NUNES; SANTOS, 2014; PEA; KURLAND, 1984; LIAO; BRIGHT, 1991; COCKBURN; BRYANT, 1996); (iv) estímulo ao desenvolvimento de habilidade de trabalho em grupo por meio de diferentes abordagens, como a programação em pares, na qual se trabalha colaborativamente para resolver problemas (COCKBURN; WILLIAMS, 2001); e (v) capacidade de criar e aplicar soluções mais abrangentes, de modo mais rápido, para aqueles indivíduos já especializados em programar (ROBINS; ROUNTREE J. AN ROUNTREE, 2003).

Não obstante, apesar dessas vantagens e também da presença obrigatória na maioria dos cursos superiores em nível nacional e internacional (SBC, 2015; PRESS; PRESS, 2005), aprender a programar é um processo complexo para grande parte dos alunos, o que resulta em desafios para os professores que lecionam tais disciplinas (PITEIRA; COSTA, 2013; PITEIRA; COSTA, 2012).

2.2.1 Problemas no Ensino e Aprendizagem de Programação

As dificuldades enfrentadas no processo de ensino na modalidade presencial, como o grande contingente de especificidades de aprendizagem dos discentes, limitação de recursos e, muitas vezes, a insistência, em utilizar métodos que enfatizam mais a teoria, ao invés de aplicações práticas, acabam tornando os processos de ensinar e aprender a programar mais complexos e difíceis. Soma-se, ainda, a escolha de linguagens de programação que, muitas vezes, dificulta a aplicação e a aprendizagem dos conceitos fundamentais de programação (MARCOLINO; BARBOSA, 2017a; HARTANTO, 2014; COSTA; APARICIO, 2014; PITEIRA; COSTA, 2012; LAHTINEN; ALA-MUTKA; JÄRVINEN, 2005).

Os relatos de diversos problemas e limitações encontrados no ensino e na aprendizagem de programação, ainda que complementares, carecem de uma classificação geral. Nesta perspectiva, Souza, Batista e Barbosa (2015) conduziram um mapeamento sistemático (MS) para

identificar os problemas e também as principais estratégias para mitigá-los, não se restringindo ao nível superior de ensino. O mapeamento buscou estudos primários entre os períodos de 2010 e 2014 na base eletrônica SCOPUS¹, visto que esta indexa as principais bases eletrônicas em Ciência da Computação. No total, 70 estudos foram selecionados para extração dos resultados. O Quadro 1 apresenta as seis categorias identificadas por meio de informações dos 70 estudos. A coluna coluna “%”, corresponde à quantidade de estudos que relatam tais problemas em suas respectivas categorias.

Quadro 1 – Categorias de Problemas de Programação.

ID	Categoria de Problemas	%	Descrição
P1	Aprendizagem de conceitos de programação.	38%	Dificuldades relacionadas à aprendizagem de conceitos de programação, tais como recursão, ponteiros, estruturas de repetição, classes e objetos.
P2	Aplicação de conceitos de programação.	24%	Dificuldades em utilizar os conceitos de programação aprendidos durante a construção de programas.
P3	Compreensão de programas.	11%	Dificuldades em ler e entender programas.
P4	Fatoração e refatoração de programas.	3%	Dificuldades em dividir o programa em módulos, funções, classes, etc.
P5	Motivação.	22%	Dificuldades relativas à falta de interesse e/ou desânimo dos alunos em realizar a atividade de programação.
P6	Dificuldades relacionadas aos professores.	2%	Dificuldades dos professores em ensinar os conceitos de programação, desenvolver materiais e exercícios de apoio, bem como avaliar os trabalhos de programação.
Total		100%	

Fonte: Adaptada de Souza, Batista e Barbosa (2015).

Com o objetivo de identificar se os problemas especificados no MS de Souza, Batista e Barbosa (2015) também correspondem aos problemas enfrentados no Brasil, um questionário *online* foi realizado com 59 professores das redes pública e privada, de diferentes localidades do Brasil. Além de solicitar que os participantes indicassem quais das categorias de problemas corresponderam aos mais evidentes em suas instituições (por meio de uma escala), os mesmos foram questionados quanto a estratégias e software utilizados em sala, e se havia outros problemas relevantes, não mencionados no Quadro 1. Adicionalmente, para buscar compreender as dificuldades por parte dos alunos, um questionário específico com 18 alunos foi aplicado presencialmente. A seguir é apresentado um resumo da pesquisa e os resultados obtidos. O planejamento, aplicação e análises detalhada das respostas coletadas estão disponíveis no Apêndice A.

2.2.2 Questionários sobre Problemas no Ensino e Aprendizagem de Programação

O esquema da Figura 2 apresenta um diagrama com o panorama geral dos questionários aplicados e os perfis dos participantes. A Figura 2 a) apresentam de quais regiões os 59 professores participantes são; a Figura 2 b) apresenta o grau de escolaridade dos professores; a Figura 2

¹ <<https://www.scopus.com>>

c) apresenta as atividades realizadas na execução da pesquisa e; a Figura 2 d) apresenta o perfil 18 dos alunos participantes em relação ao ano que cursam no ensino médio. Ainda que o perfil dos usuários considerados inicialmente como usuários das aplicações sejam alunos do ensino superior, o questionário aplicado com alunos do ensino médio justificado porque tais alunos serão os ingressantes em cursos superiores de tecnologia, logo, os problemas relatados por esses alunos são considerados importantes e pertinentes no contexto à pesquisa.

Figura 2 – Informações sobre o Questionário: a) Regiões do Professores Participantes, b). Perfil dos Professores, c). Atividades do Questionário e d). Perfil dos alunos.

Fonte: Elaborada pelo autor.

O questionário para os professores (QP), foi composto por três questões, a saber:

- QP1.** O quão precisas são as categorias de problemas relacionados ao ensino de programação (Quadro 1), quando analisadas na perspectiva dos problemas que você tem encontrado durante suas aulas? Classifique-as, considerando a escala de 5 níveis (1-“Mais Incidente” a 5-“Menos Incidente”).
- QP2.** Você tem adotado estratégias ou software para mitigar os problemas enfrentados no processo de ensino de programação? Descreva-os.
- QP3.** Há outros problemas relacionados ao ensino de programação que não tenham sido considerados nas seis categorias de problemas analisadas?

As respostas para a primeira questão (QP1), referente à classificação das categorias de problemas são apresentadas na Tabela 1. Nela estão descritas a soma dos percentuais para cada um dos cinco níveis de incidência, sendo que as colunas de 1 até 5 correspondem à ordenação fornecida pelos respondentes, de qual problema é mais frequente (1), o segundo mais frequente (2) e assim por diante.

Tabela 1 – Categorias de Problemas de Programação e Níveis de Incidência.

ID	Categoria de Problemas	% por Nível de Incidência					Total
		1	2	3	4	5	
P1	Aprendizagem de conceitos de programação.	32,20%	32,20%	15,25%	15,25%	5,10%	100%
P2	Aplicação de conceitos de programação.	35,60%	28,80%	16,95%	13,55%	5,10%	100%
P3	Compreensão de programas.	15,25%	35,60%	42,38%	6,77%	0,00%	100%
P4	Fatoração e refatoração de programas.	15,25%	23,72%	37,31%	16,95%	6,77%	100%
P5	Motivação.	23,72%	25,42%	22,04%	20,33%	8,49%	100%
P6	Dificuldades relacionadas aos professores.	15,25%	8,49%	33,90%	23,72%	18,64%	100%

Fonte: Dados da pesquisa.

A classificação realizada pelos participantes indica a categoria **Aplicação de conceitos de programação** (P2) como a mais incidente, correspondendo a 35,6%. A categoria classificada com incidência 2 e 3, 35,6% e 42,38% respectivamente, referem-se à **Compreensão de programas** (P3). Finalmente, a categoria **Dificuldades relacionadas aos professores** (P6) corresponde às incidências 4 e 5, 23,72% e 18,64% respectivamente. O empate entre os níveis de incidência deve-se à indicação dos participantes, que podiam classificar uma categoria com um mesmo nível de incidência. Ao comparar os resultados do MS com os da pesquisa, nota-se uma alteração na ordem dos problemas considerados mais incidentes. Os resultados da pesquisa consideraram a categoria **Aplicação de conceitos de programação** (P2) mais incidente, enquanto no MS tinha-se como mais incidente a categoria **Aprendizagem de conceitos de programação** (P1). Como segunda categoria mais incidente tem-se a categoria **Compreensão de programas** (P3), que no MS era a categoria de quarta posição. Finalmente, como categoria menos incidente, tanto no MS quanto na classificação, tem-se a referente às **Dificuldades relacionadas aos professores** (P6), com 23,72% e 18,64%.

O Quadro 2 resume as respostas dos professores para a segunda (QP2) e terceira (QP3) questões.

Considerando as diferentes estratégias de aprendizagem indicadas (QP2), a que apresentou maiores indicações corresponde à aplicação de diversas atividades de aprendizagem, principalmente, listas de exercícios para motivar os alunos a aplicar conceitos aprendidos de modo prático para que, assim, seja possível identificar mais rapidamente dúvidas e dificuldades. Finalmente, os software indicados são adotadas como complemento, permitindo e facilitando a aplicação dos conhecimentos aprendidos em sala.

Em relação às categorias e demais problemas de programação enfrentados (QP3), todos

Quadro 2 – Respostas dos Professores para Segunda (QP2) e Terceira (QP3) Questões.

Questão	Resumo das Respostas
QP2	As principais estratégias e softwares adotados para mitigar problemas no ensino de programação, foram: (i) uso de linguagens simplificadas de programação, i.e., linguagens com poucas estruturas complexas, variáveis fracamente tipadas e um conjunto menor de palavras reservadas, como a linguagem Python; (ii) uso de cursos de programação <i>online</i> , i.e., Udacity (https://www.udacity.com/) e Coursera (https://www.coursera.org/); (iii) uso de abordagens para motivar a aprendizagem, i.e., aprendizagem baseada em problemas, condução de várias atividades de aprendizagem, como questões e desafios e competição de programação, como as maratonas de programação; (iv) uso de softwares para complementar às atividades de aprendizagem desenvolvidas em sala de aula, i.e., Scratch (https://scratch.mit.edu/), Greenfoot (www.greenfoot.org/), VisualLog (www.apoioinformatica.inf.br/produtos/visualg), Ambientes de Desenvolvimento Integrado (IDEs) (<i>Integrated Development Environments</i>), Coding DOJO (http://dojopuzzles.com/), BlueJ (www.bluej.org/) e URI Online Judge (https://www.urionlinejudge.com.br/) e; (v) uso de conteúdos de aprendizagem abordando diferentes meios de representação, como vídeos, apresentações de slides e ilustrações divertidas e atraentes na concepção dos alunos.
QP3	Os principais problemas destacados pelos professores estão relacionados a: (i) conceitos básicos necessários para que os alunos possam entender os conceitos básicos de programação envolvidos; (ii) desmotivação; e (iii) distração com outras tecnologias. No que se refere aos softwares de apoio, estes não foram considerados completos o suficiente para atender as necessidades dos professores. Finalmente, um último comentário de destaque menciona que os programas de disciplinas são frequentemente fixos e, de algum modo, desatualizados.

Fonte: Dados da pesquisa.

os problemas relatados, segundo os professores, são cobertos pelas seis categorias de problemas classificados. Contudo, tais resultados evidentemente não podem ser generalizados para todas as instituições brasileiras. Logo, as respostas apresentam evidências preliminares de que os problemas de programação encontrados em nível internacional também são refletidos nas instituições de ensino no Brasil. Por outro lado, mesmo com a limitação da amostra, esta pode ser incrementada com futuras replicações do questionário.

Já o questionário com alunos foi aplicado presencialmente em um curso de introdução à programação para Internet, ofertado na Universidade de São Paulo à comunidade em geral. A aplicação ocorreu em dois momentos: (1) pré-teste (QEA), com cinco questões, aplicado no primeiro dia do curso; e pós-teste (QEB), quatro questões, aplicado no último dia do curso.

As questões aplicadas no pré-teste (QEA) foram:

QEA1. Por que você quer aprender a programar?

QEA2. Por que aprender a programar é importante para o seu futuro?

QEA3. Se pudesse, você adotaria tecnologias móveis para aprender a programar?

QEA4. Quais estratégias ou abordagens você gostaria usar em software que objetivam ensinar programação?

QEA5. Que outros meios você gostaria que fossem utilizados para apoiar o processo de aprendizagem de programação?

As questões aplicadas no pós-teste (QEB) foram:

QEB1. Se pudesse, você adotaria tecnologias móveis para aprender a programar?

QEB2. Quais tópicos você têm ou teve mais dificuldade em aprender?

QEB3. Por que você acredita ter tais dificuldades nesses tópicos especificamente?

QEB4. Você possui alguma sugestão para melhorar o curso de programação que realizou?

Destaca-se que a repetição da questão QEA3 no pós-teste, por meio da questão QEB1, procura identificar se os alunos mantêm a mesma resposta após a realização do curso.

Um total de 18 alunos responderam aos questionários. Destes, 10 responderam o pré-teste (QEA) e o pós-teste (QEB), 4 responderam apenas o pré-teste e 4 responderam apenas o pós-teste. O Quadro 3 resume as respostas dos alunos.

A resposta unânime à QEA1 destaca a preocupação dos alunos e a importância em saber programar. A resposta à questão QEA4 pode auxiliar na identificação de estilos de aprendizagem (FELDER; SILVERMAN *et al.*, 1988; LIYANAGE; GUNAWARDENA; HIRAKAWA, 2014) e os consequentes meios para melhor engajar os alunos a aprender a programar (KELLEHER, 2009). Já a resposta à QEA5 apresenta sugestões que podem ser melhor exploradas em software para apoiar disciplinas e cursos de programação.

A inclusão da terceira questão do pré-teste e primeira questão do pós-teste (QEA3 e QEB) foi realizada devido à incipiência de soluções para apoiar o ensino de programação por meio de dispositivos móveis. Essa carência será melhor discutida no Capítulo 3, no qual um mapeamento sistemático que buscou identificar tais soluções é apresentado (MARCOLINO; BARBOSA, 2015b).

Com base nas respostas das questões QEB3 e QEB4 foi possível identificar que tais considerações são abordadas nas seis categorias de problemas propostas por Souza, Batista e Barbosa (2015). Adicionalmente, ainda sobre a QEB3 e as dificuldades enfrentadas no curso, os alunos recebiam algumas tarefas a serem resolvidas em casa, sem qualquer ferramenta ou material de apoio, além do fornecido durante as aulas presenciais. As dúvidas foram todas sanadas em sala de aula pelo instrutor. Outra consideração refere-se a dificuldades do instrutor em identificar as limitações dos alunos, uma vez que era necessário e esperado que cada aluno reportasse tais dificuldades ao instrutor no decorrer das aulas, o que não ocorreu com frequência.

Finalmente, as sugestões feitas na QEB4 indicam que os alunos consideraram os tópicos um tanto quanto avançados para a base de conhecimento que possuíam. Nesta perspectiva, eles

Quadro 3 – Respostas dos Alunos para o Pré-teste (QEA) e Pós-teste (QEB).

Questão	Resumo das Respostas
Pré-Teste (QEA)	
QEA1	A maioria dos alunos indicaram que querem aprender a programar pois pretendem seguir carreira na área e que tecnologia está presente em todos os setores da sociedade.
QEA2	Respostas similares à questão QEA1 foram obtidas. Os alunos indicaram que é importante aprender a programar, porque eles podem entender as tecnologias que os rodeiam, além de ser crucial para suas carreiras.
QEA4	Dentre as indicações de estratégias e/ou abordagens que gostariam de utilizar em softwares para ensinar programar, 4 alunos indicaram vídeos; 2 jogos; 8 a mesclagem de vídeos, animações, textos, figuras e jogos; e 1 respondeu estratégias que utilizem poucos textos e textos curtos, justificando que seriam mais engajantes que textos longos.
QEA5	Em relação a outros meios para aprender a programar, 6 alunos demonstraram interesse em aplicações móveis, 4 em aprendizagem de programação com robôs, 2 por meio de cursos online, 2 por jogos, e 2 por meio de software para computadores.
Pós-Teste (QEB)	
QEB2	Em relação aos tópicos com maiores dificuldades, 1 aluno indicou estruturas de repetição, 1 estruturas condicionais, 7 vetores e matrizes, 7 funções, e 1 indicou não possuir dificuldades nos tópicos especificamente ministrados no curso.
QEB3	A principal dificuldade relatada pelos alunos correspondeu em seguir a teoria ensinada, 2 deles indicaram o nível de dificuldade das atividades, e 3 não indicaram as razões.
QEB4	Duas sugestões foram indicadas para propor melhoras no curso que os alunos participaram: (1) necessidade de ensinar mais detalhadamente fundamentos de programação; e (2) adoção de outras linguagens de programação, além de JavaScript.
Pré-Teste (QEA) e Pós-Teste (QEB)	
QEA3 e QEB1	13 alunos indicaram que adotariam dispositivos móveis para aprender a programar no pré-teste, por permitire estudar em qualquer lugar e a qualquer hora. Apenas um participante disse que não utilizaria, justificando que computadores fornecem ambientes de desenvolvimento mais completos. Em relação as respostas do pós-teste, novamente 13 aprendizes indicaram que adotariam dispositivos móveis e um disse que não utilizaria.

Fonte: Dados da pesquisa.

expressaram a necessidade de aprender melhor os fundamentos de programação antes de uma linguagem específica, que no contexto do curso foi a linguagem JavaScript.

Nesta perspectiva, com base nos problemas categorizados em [Souza, Batista e Barbosa \(2015\)](#) e nos questionários conduzidos com professores e alunos elenca-se os principais desafios a serem investigados e superados nesta área:

- Direcionar esforços para a proficiência em matemática e raciocínio lógico, de modo a impulsionar a qualidade no processo de ensino e aprendizagem de programação.
- Conceber soluções de software mais flexíveis, que agreguem meios de melhorar o processo de *feedback*; identificação de limitações e resolução de dúvidas; conteúdos e atividades de aprendizagem com duração e descrições curtas; estratégias e simulações visuais, e.g., ilustrações, objetos 3D; e interações colaborativas e cooperativas.

- Conceber ferramentas para apoiar os professores durante e depois nas aulas no que se refere ao monitoramento do desempenho de seus alunos;
- Adotar soluções mais motivantes e engajadoras para os alunos;
- Integrar funcionalidades e facilidades que os alunos encontram em aplicações do cotidiano; e.g. aplicações de redes sociais, jogos.
- Criar conteúdos por meio de diferentes estratégias de apresentação, considerando a separação de conceitos de lógica e algoritmos de conceitos específicos de linguagens de programação e;
- Adotar de dispositivos móveis para apoiar o ensino e aprendizagem.

Mais detalhes a respeito do estudo da aplicação dos questionários sobre problemas no ensino e aprendizagem de programação estão disponíveis em (MARCOLINO; BARBOSA, 2017a).

É importante observar que estes desafios estão relacionados ao domínio de programação e não estão restritos ao Brasil (MARCOLINO; BARBOSA, 2015b). Porém, em comparação com outros países e suas iniciativas para impulsionar o setor tecnológico (i.e., inclusão de disciplinas de programação no currículo de cursos elementares), muitas mudanças ainda são necessárias. Entre elas, destaca-se a necessidade de melhor preparar os professores para que criem o hábito de utilizar tecnologias inovadoras e que possam enriquecer suas aulas. Tais professores, principalmente das escolas públicas, necessitam de mais motivação e treinamento (AURELIANO; TEDESCO, 2012; WANGENHEIM; NUNES; SANTOS, 2014; BORDINI *et al.*, 2016), além de necessitarem do respaldo dos órgãos educacionais federais, estaduais e municipais no que tange à atualização de currículos e planos de ensino que integrem o pesamento computacional e programação em suas aulas.

Com base nos problemas categorizados e desafios elencados, a adoção de modalidades de ensino a distância, como a aprendizagem móvel, torna-se atrativa para permitir a investigação de possíveis soluções para mitigar ou sanar problemas no domínio de programação. Na próxima seção são apresentados aspectos gerais sobre aprendizagem móvel.

2.3 Aprendizagem Móvel

TIC têm modificado cada vez mais as dimensões pessoais de inter-relacionamento com o mundo, da percepção da realidade e da interação com o tempo e o espaço. A miniaturização das tecnologias de comunicação vem permitindo uma grande flexibilidade, mobilidade e personalização, facilitando a individualização dos processos de comunicação. Tecnologias portáteis, juntamente com as redes computacionais, tornam-se cada vez mais presentes no cotidiano, promovendo acesso a informações de maneira fácil e rápida (GALAFASSI; GLUZ; GALAFASSI,

2013; FREITAS *et al.*, 2012; ALMEIDA, 2005; SCHLÜNZEN, 2005; HADDAD, 2002; BALA, 2018). Tal constatação é claramente visível em pesquisas, como as conduzidas pelo Instituto Brasileiro de Geografia Estatística (IBGE) e o Comitê de Gestor de Internet no Brasil (CGI).

A pesquisa conduzida pelo CGI, de 2013 (CGI, 2014), apresenta uma evolução significativa na adoção de TIC. Por exemplo, o total de residências brasileiras com, ao menos, um celular aumentou cerca de 30,7%, de 2013 em relação a 2005, totalizando 90%. Outro aumento expressivo corresponde ao número de computadores, que teve sua quantidade ampliada em torno de 40,4%, totalizando 59%. O acesso a Internet também foi ampliado, de 13,70% para 43%, indicando um aumento de 29,3% em 2013, também em relação a 2005. Já a pesquisa mais recente, realizada pelo IBGE em 2017, identificou que 92,7% dos domicílios possuem algum dispositivo móvel (IBGE, 2018).

Nesta perspectiva, a adoção de TIC é intensificada também na área educacional (HADDAD, 2002; BALA, 2018; ALRASHEEDI; CAPRETZ, 2018; FU; HWANG, 2018; HEFLIN; SHEWMAKER; NGUYEN, 2017; PIMMER; MATEESCU; GRÖHBIEL, 2016). Tal crescimento tem favorecido o surgimento de modalidades de ensino a distância, i.e., aprendizagem eletrônica ou *e-learning* (*electronic learning*), aprendizagem móvel ou *m-learning* (do inglês *mobile learning*), aprendizagem por jogos ou *g-learning* (do inglês *game learning*), aprendizagem ubíqua ou *u-learning* (do inglês *ubiquitous learning*) e aprendizagem vestível ou *w-learning* (do inglês *wearable learning*) (MOREIRA; ROCHA, 2018; MICCIOLO *et al.*, 2018). Tais modalidades de ensino, constituídas de características e vantagens provenientes das tecnologias que adotam, apresentam-se como um campo de pesquisa em potencial para, não somente promover o ensino e aprendizagem, como também dinamizá-lo.

A necessidade de se estar fisicamente em um ambiente para aprender, tal como ocorre no ensino presencial, acaba limitando as novas possibilidades trazidas pelo uso das TIC e suas modalidades de aprendizagem. Além disso, tal modo de ensino não é muito favorável à exploração de diferentes estilos de aprendizagem, tais como exames e discussões *online*, colaboração, materiais interativos, animações, vídeos, possibilidade de mobilidade e a busca de conhecimento por meio da Internet (SARRAB, 2015). Essas limitações têm levado à busca por novos métodos educacionais de aprendizagem; a introdução do *e-learning*, modalidade bastante difundida e utilizada, é exemplo desta tentativa.

Tem-se como *e-learning* a combinação de tecnologia e pedagogia, apresentando três critérios fundamentais (FERNANDES; FERREIRA, 2011; SHESHASAYEE; BEE, 2018): (i) é transmitido em rede, o que torna possível a atualização, armazenamento, recuperação, distribuição e compartilhamento instantâneos da instrução ou informação; (ii) é fornecido ao usuário final por meio do computador, utilizando a tecnologia padrão da Internet; e (iii) concentra-se em uma visão ampla de aprendizagem, contemplando soluções que utilizem paradigmas distintos dos tradicionais. Os diferentes elementos elencados possibilitam então a representação do *e-learning* como a convergência de tecnologias, como os computadores *desktop*, a conexão

com a Internet representada pela pilha de protocolos TCP/IP (*Transfer Control Protocol e Internet Protocol*, respectivamente) e elementos de aprendizagem.

Contudo, ainda que o *e-learning* possibilite a interação entre usuários e diferentes elementos educacionais esta modalidade de ensino também limita o aprendiz, no sentido de não permitir que este, em constante movimento, possa decidir quando e em que momento aprender. Com o surgimento e difusão dos dispositivos móveis, passa-se para um novo estágio da aprendizagem e reformulações de como ambientes virtuais podem ser utilizados de modo a fornecer, sobretudo, liberdade aos seus usuários.

Os dispositivos móveis permitem acesso a informações fora dos lares, empresas e instituições educacionais, tornando-se pervasivos ao cotidiano (SARRAB, 2015). Soma-se a esta mudança a ampliação e difusão do uso da Internet nas últimas décadas, em nível global, bem como a evolução de sua dinâmica com base no surgimento de novas tecnologias (BALAJI, 2018). Na Web 2.0, novas tecnologias de desenvolvimento e interação tornaram a rede mundial de computadores mais interativa e acessível, ampliando e facilitando ainda mais a obtenção de informação, aprendizagem e interações sociais e colaborativas (COCHRANE, 2011). Já na Web 3.0, há uma sinergia de informações, na qual estrutura-se todo o conteúdo disponível na rede mundial de computadores dentro dos conceitos de aprendizagem de máquina (do inglês, *machine learning*) e web semântica, fazendo uso dos dados provenientes dos dispositivos móveis (CABADA *et al.*, 2018; BALAJI, 2018). Surgem assim novas facilidades, como apresentar conteúdo com base nas preferências de navegação dos usuários por meio, principalmente, de serviços mantidos na nuvem (derivado do termo em inglês *cloud computing*) (ABDEL-BASSET; MOHAMED; CHANG, 2018; VARGHESE; BUYYA, 2018). Adicionalmente, já se fala na Web 4.0, que possivelmente será caracterizada por autonomia de atividades e pela sensação de estar usando um sistema único, formado pelo conglomerado, ou ecossistemas, de diversas soluções (KURGUN; KURGUN; AKTAŞ, 2018).

Assim, o uso de dispositivos móveis (i.e., telefones celulares, computadores portáteis e *tablets*) propicia a difusão e crescimento da aprendizagem móvel ou *m-learning*. Em suas definições iniciais, *m-learning* é tido como um *e-learning* que utiliza dispositivos móveis. Contudo, estudos como o de Sharma e Kitchens (2004) apontam diferenças significativas entre as duas modalidades de ensino. Em 2001, Uhlig *et al.* (2005) definiu esse novo formato de aprendizagem como sendo uma nova possibilidade utilizando tecnologias móveis, devido à crescente demanda por processos de aprendizagem flexíveis e que permitissem mobilidade espacial geográfica.

Em 2002, Farooq *et al.* (2002) atribuíram uma nova definição a *m-learning*, como sendo um processo educacional e de aprendizagem usando dispositivos específicos, em qualquer lugar e a qualquer momento. Essa definição é similar à proposta em 2003, que indica *m-learning* como sendo um novo mecanismo de aprendizagem que ocorre em qualquer lugar e momento, por meio de dispositivos computacionais móveis (SARRAB, 2015).

Contudo, o termo aprendizagem móvel, segundo [El-Hussein e Cronje \(2010\)](#), vai além do uso de dispositivos móveis para aprender. O termo mobilidade indica também o novo advento sociocultural em que os alunos também não se encontram em lugares fixos, movem-se e, o contexto social no qual se inserem acaba por influenciar o modo como aprendem por meio de *m-learning*. [Traxler \(2007\)](#) indica que aprendizagem móvel não trata apenas de mobilidade ou aprendizagem em si, como era entendido, mas sim parte de uma nova concepção de sociedade móvel, na qual os usuários estão em constante deslocamento, mas sempre conectados.

[Baran \(2014\)](#) aponta que as definições de *m-learning* envolvem diversas características além de mobilidade, tais como acesso, relação imediata, cognição situada – teoria que postula que o conhecimento é inseparável de fazer, argumentando que todo o conhecimento está situado na atividade vinculado a contextos sociais, culturais e físicas ([BROWN; COLLINS; DUGUID, 1989](#)); ubiquidade, conveniência, comodidade e contextualidade. *M-learning* possibilita aos professores e alunos o acesso ubíquo e contínuo (transparente) de informações, comodidade, conveniência e imediatismo. Tais características são importantes e valiosas para os professores, assim como para garantir a aprendizagem dos seus alunos, tornando-se oportunidades para aprendizagem individual, localizada, colaborativa e informativa, sem limitações impostas pelo aprender em salas de aula físicas ([BARAN, 2014](#)).

Apesar das diferentes definições ao longo dos anos, nota-se uma convergência em termos das mesmas. Deste modo, *m-learning* é definida nesta tese como segue ([MOREIRA; ROCHA, 2018](#); [BARAN, 2014](#); [TRAXLER, 2007](#); [AARRENIEMI-JOKIPELTO, 2005](#)): modalidade de aprendizagem caracterizada por permitir interações efetivas entre usuários (alunos, professores e tutores), permitindo que contribuam, participem e acessem ambientes educacionais por meio de dispositivos móveis (celulares, assistentes digitais pessoais (PDAs), *smartphones*, *tablets*, *laptops*, e outros) a qualquer lugar e a qualquer hora.

A Figura 3 (a) sintetiza a ideia de *m-learning*, representando a convergência de TIC, advindas da pilha de protocolos TCP/IP, da tecnologia móvel e da tecnologia apresentada nos sistemas de *e-learning* ([AARRENIEMI-JOKIPELTO, 2005](#)). Já a Figura 3 (b) apresenta uma visão detalhada de *m-learning*, que formada pela convergência de aspectos dos dispositivos, dos alunos e aspectos sociais, define características específicas da modalidade ([KOOLE, 2009](#)): usabilidade dos dispositivos, interações de aprendizagem e tecnologias sociais.

Ademais, além de características específicas do processo de ensino e aprendizagem, as aplicações que promovem a aprendizagem móvel possuem um conjunto de características técnicas comuns. De acordo com [Huang, Huang e Hsieh \(2008\)](#), algumas delas são mencionadas em diversos estudos, sendo elas: (1) garantir a disponibilidade e acesso a informações via rede; (2) atrair e motivar os alunos nas atividades de aprendizagem independentemente da localização física; (3) apoiar a realização de estudos em grupo; (4) melhorar a comunicação e aprendizagem colaborativa, quando utilizados em sala de aula; e (5) permitir a entrega rápida de conteúdos.

Tal como o estudo de [Huang, Huang e Hsieh \(2008\)](#), a literatura apresenta diversos

Figura 3 – Composição e Características do *m-learning*.

Fonte: Adaptada de [Aarreniemi-Jokipielto \(2005\)](#), [Koole \(2009\)](#).

trabalhos que visam especificar mais precisamente os *m-learning* em relação à diversidade de funcionalidades e características que apresentam que, como visto, acabam gerando dificuldades em os definir.

[Georgieva, Smrikarov e Georgiev \(2005\)](#) estabelecem uma classificação, tendo como base indicadores como o tipo de dispositivos móveis e a tecnologia de comunicação utilizada para disponibilização de materiais de aprendizagem e informações administrativas. São considerados, ainda, aspectos como disponibilidade de conexão com Internet, apoio a normas e padrões de *e-learning*, localização dos usuários e mecanismos de comunicação entre professores e alunos. A Figura 4 apresenta a classificação proposta.

De modo similar, [Duarte Filho e Barbosa \(2013\)](#) desenvolveram um catálogo de requisitos para ambientes de aprendizagem móvel. Mais especificamente, com base em diferentes estudos, foi estabelecido um catálogo com diferentes requisitos esperados em aplicações de aprendizagem móveis. Este catálogo foi aperfeiçoado na perspectiva de requisitos de qualidade para aplicações móveis, resultando em um conjunto mais preciso de requisitos, intitulado, *ReqML-Catalog* ([SOAD et al., 2017b](#)). A classificação de [Georgieva, Smrikarov e Georgiev \(2005\)](#) e os catálogos de [Duarte Filho e Barbosa \(2013\)](#) e [Soad et al. \(2017b\)](#) foram utilizados para avaliar e complementar um catálogo de requisitos *m-learning* discutido no Capítulo 4.

A diversidade de funcionalidades e características apresentadas nos trabalhos supracitados e em outros estudos levam à identificação de diversos benefícios na adoção modalidade de aprendizagem móvel ([SARRAB, 2015](#); [ROSHAN](#); [DARYUSH](#); [BAHAREH, 2013](#); [BHULLAR, 2014](#); [TOTEJA](#); [KUMAR, 2012](#); [KHADDAGE et al., 2015](#); [BARAN, 2014](#); [MOREIRA](#);

Figura 4 – Classificação Geral de Sistemas de *m-Learning*.

Fonte: Adaptada de Georgieva, Smrikarov e Georgiev (2005).

ROCHA, 2018; BANO *et al.*, 2018). Contudo, apesar dos significativos benefícios e avanços proporcionados no contexto educacional, por ser um conceito mais recente, necessitando de mais investigações, inclusive quanto aos seus benefícios reais (BANO *et al.*, 2018), o *m-learning* também apresenta limitações e desafios para sua efetiva adoção (ROSHELLE; PEA, 2002; LIU; HAN; LI, 2010; BHULLAR, 2014; SARRAB, 2015; EL-HUSSEIN; CRONJE, 2010). O Quadro 4 resume as principais vantagens e desvantagens de *m-learning*.

Nota-se, portanto, que os desafios a serem considerados para se promover o desenvolvimento de sistemas de aprendizagem móvel e usufruir de seus benefícios são significativos. Exemplos de tais esforços são refletidos em diversas pesquisas, tais como as que investigam e propõem diferentes catálogos e classificações de requisitos (PARSONS; RYU; CRANSHAW, 2006; KOOLE, 2009; GEORGIEVA; SMRIKAROV; GEORGIEV, 2005; DUARTE FILHO; BARBOSA, 2013; KHADDAGE *et al.*, 2015; SOAD *et al.*, 2017b), propostas de arquiteturas (SHARMA; KITCHENS, 2004), modelos de *m-learning* (BARKER; KRULL; MALLINSON, 2005), padrões (FIORAVANTI; BARBOSA, 2018c), tendências atuais (TING, 2005; KHADDAGE *et al.*, 2015; BANO *et al.*, 2018) e ambientes e aplicações de aprendizagem móveis para

Quadro 4 – Vantagens e Desvantagens de *M-learning*.

Aprendizagem Móvel
Vantagens
Centralização da aprendizagem no aluno, sendo o professor ou tutor apenas um coadjuvante na construção do saber;
Apoio às necessidades específicas de aprendizagem do aluno, que tem o controle sobre as atividades de aprendizagem;
Flexibilização com acesso a conteúdos em qualquer lugar e em qualquer momento;
Dinamização e aproveitamento de tempo ocioso, possibilitando o aprender durante deslocamento em viagens, filas de espera, etc.;
Coleta de resultados e informações por diferentes meios, como áudio, escrita, vídeo, sensores, etc.;
Redução do custo em relação aos dispositivos para e-learning;
Redução de barreiras culturais pelos diferentes canais de comunicação que possui;
Adaptação e sensibilidade ao contexto;
Personalização da aprendizagem, visto que o estudante o faz em um dispositivo próprio e o "ajusta" às suas necessidades, além de ter acesso a conteúdo personalizado;
Eliminação de barreiras tecnológicas e provê aprendizagem social;
Capacidade de motivar experiências de aprendizagem independente e colaborativa;
Auxílio no combate à resistência no uso de TICs;
Remoção de barreiras formais impostas pelas experiências tradicionais de ensino, engajando alunos relutantes;
Elevação da autoestima e autoconfiança dos alunos, visto que se tornam responsáveis e construtores do seu próprio conhecimento, por meio do m-learning; e
Capacidade de fornecimento de feedback imediato.
Desvantagens
Restrições quanto ao tamanho das telas;
Acesso a conteúdo e interações ofensivas por meio de dispositivos móveis;
Capacidade de armazenamento reduzida;
Vida útil da bateria limitada, ou seja, dependência de baterias;
Carência de padrões arquiteturais de desenvolvimento;
Diferentes plataformas móveis como iOS, Android, Windows, etc.;
Diferentes produtores de hardware e competitividade no mercado que, apesar de reduzir custos, criam especificidades no processo de desenvolvimento de aplicações, devido falta de padronização;
Comunicação com taxas de transmissão geralmente menores do que as das redes fixas;
Aspectos de usabilidade, como tamanho de tela reduzida, e de segurança, como aplicações maliciosas;
Prevenção de divulgação de informações dos alunos na rede;
Pode criar um sentimento de isolamento;
Pode necessitar de uma curva de aprendizagem maior para alunos não técnicos;
As aplicações existentes não são facilmente integradas com as tecnologias móveis do ambiente;
Resistência dos professores para usar conteúdos digitais e TICs.

Fonte: Sarrab (2015), Roshan, Daryush e Bahareh (2013), Bhullar (2014), Toteja e Kumar (2012), Khaddage *et al.* (2015), Baran (2014), Moreira e Rocha (2018), Bano *et al.* (2018), Roschelle e Pea (2002), Liu, Han e Li (2010), Bhullar (2014), El-Hussein e Cronje (2010).

domínio geral, tanto comerciais quanto gratuitos, como o *Moodle Mobile* ² (Figura 5 a)) e o *Blackboard Mobile* ³ (Figura 5 b)).

² <https://docs.moodle.org/dev/Moodle_Mobile>

³ <<http://www.blackboard.com/platforms/mobile/overview.aspx>>

Figura 5 – Aplicações: a) Moodle Mobile 2 e b) Blackboard Mobile.

Fonte: Imagens extraídas de <<https://moodle.org/>> e <<http://www.blackboard.com/platforms/mobile/overview.aspx>>.

Assim, com os benefícios da modalidade de aprendizagem móvel e seus desafios, a incipiência de estudos que exploram reuso de funcionalidades no contexto de linhas de produto de software (MARCOLINO; BARBOSA, 2015a) (melhor discutido no Capítulo 3), e as evidências iniciais sobre a possível adoção desses dispositivos para se aprender a programar, conforme visto nos resultados do questionário conduzidos com alunos (Subseção 2.2.1).

Assim, com base nas especificidades dos diferentes benefícios no uso de dispositivos móveis para se aprender, e na incipiência de sua adoção com metodologias de reuso, a ser melhor discutida no Capítulo 3, a modalidade de aprendizagem móvel foi escolhida para ser investigada no contexto de mitigar e sanar problemas relacionados ao domínio de programação, por meio da criação de aplicações móveis utilizando-se da metodologia de linhas de produto de software, fundamentada a seguir.

2.4 Linhas de Produto de Software

O reuso, em Engenharia de Software, é definido como qualquer procedimento que produza ou ajude a produzir um sistema, permitindo utilizar algo desenvolvido previamente (PRESSMAN, 2015; SOMMERVILLE, 2010). Incorpora-se em um novo produto qualquer artefato previamente definido, seja este código, especificações de requisitos e projeto, plano de teste, etc., proporcionando diversos benefícios. Busca-se, portanto, com a Engenharia de Software baseada em Reuso, adotar abordagens que procuram maximizar a reutilização em software existentes, com destaque para as linhas de produto de software.

Uma LPS, segundo Northrop e Clements (2009), corresponde a um conjunto de soluções de software que compartilham *features* comuns e gerenciáveis, que satisfazem às necessidades

de um segmento particular do mercado ou de uma missão específica, sendo desenvolvida a partir de ativos centrais, de modo sistemático (CAPILLA; BOSCH; KANG, 2013; NORTHROP; CLEMENTS, 2009). Com base nessa definição, alguns conceitos chave são destacados (NORTHROP; CLEMENTS, 2009; GOMAA, 2004; CAPILLA; BOSCH; KANG, 2013):

- **Features⁴**: indicam requisitos ou características reusáveis. Um requisito representa uma necessidade; logo, ao menos um membro da LPS deve ser capaz de satisfazê-lo; esse membro da LPS, por sua vez, é representado por uma *feature*;
- **Ativos centrais (core assets)**: representam um conjunto de artefatos prontos para serem reusados no desenvolvimento de novos produtos. Dentre eles, destacam-se os componentes de software, arquitetura, documentação e outros artefatos. A arquitetura carrega consigo as possibilidades de variabilidade da LPS. Logo, o ativo arquitetura é ponto chave de uma LPS, inviabilizando o projeto, caso seja mal projetado;
- **Modo Sistemático**: cada ativo tem um processo associado que define como ele pode ser usado no desenvolvimento de novos produtos. O conjunto desses processos combinados com processos que descrevem o acoplamento dos ativos centrais definem um plano de produção para cada produto ou uma configuração. O plano de produção, por sua vez, define um procedimento de derivação para um novo produto;
- **Segmento particular de mercado**: ou também especificado como missão, deve ser delimitado e bem compreendido em uma LPS, refletindo na definição do escopo e direcionamento das estratégias de negócio. A definição assertiva auxilia na redução da complexidade do gerenciamento dos ativos e de suas variabilidades;
- **Variabilidade**: é a forma como os membros de uma família de produtos podem se diferenciar entre si, ou seja, é o que permite distinguir os diversos produtos de uma LPS (WEISS; LAI, 1999). De acordo com Booch, Rumbaugh e Jacobson (1999), variabilidade corresponde à habilidade de um software ou artefato ser modificado, personalizado ou configurado para uso em um contexto particular. A variabilidade é descrita por pontos de variação, variantes e restrições entre variantes (LINDEN; SCHMID; ROMMES, 2007):
 - Um **ponto de variação** permite a resolução de variabilidades em artefatos genéricos de uma LPS por meio de suas variantes associadas. De acordo com Jacobson, Griss e Jonsson (1997), um ponto de variação identifica um ou mais locais nos quais a variação irá ocorrer. Assim, um ponto de variação pode ocorrer em artefatos em diferentes níveis de abstração.

⁴ O termo *feature* não foi traduzido neste trabalho pois a expressão correspondente em português não expressa tecnicamente o que o conceito significa.

- Uma **variante** representa os possíveis elementos pelos quais um ponto de variação pode ser resolvido. Também pode representar uma maneira de resolver uma variabilidade diretamente.
- Para a resolução de um ponto de variação, mediante a seleção de uma ou mais variantes, devem ser obedecidas as **restrições entre variantes**. Tais restrições definem os relacionamentos entre uma ou mais variantes.

Desse modo, para estabelecimento de uma LPS é necessária a condução sistêmica de diferentes atividades. Nesta perspectiva, três principais atividades foram definidas pelo *Software Engineering Institute* (SEI) por meio da iniciativa *Product Line Practice* (PLP) (NORTHROP; CLEMENTS, 2009) para auxiliar na concepção de LPS.

2.4.1 Atividades Essenciais

As três atividades essenciais de LPS (NORTHROP; CLEMENTS, 2009) são: (i) Desenvolvimento do Núcleo de Artefatos, correspondente à Engenharia de Domínio; (ii) Desenvolvimento do Produto, conhecida também como Engenharia de Aplicação; e (iii) Gerenciamento de LPS. A Figura 6 representa a interação entre tais atividades.

Figura 6 – Atividades Essenciais de LPS.

Fonte: Northrop e Clements (2009).

No desenvolvimento do núcleo de artefatos são definidos e criados os aspectos comuns e variáveis da LPS. Como principais artefatos concebidos nessa etapa estão o modelo de *features* (KANG *et al.*, 1990), um conjunto de artefatos de domínio, e o mapa de *features*, que relaciona cada *feature* com seus artefatos de domínio respectivos (NORTHROP, 2008).

No desenvolvimento do produto é feita a derivação de produtos da LPS. Como processos que compõem tal atividade tem-se a criação de uma configuração do produto, responsável por

listar as *features* selecionados para o produto que será derivado, e o processo de derivação do produto, que por meio do mapa de *features* e a configuração do produto, permite gerar o produto final, com os artefatos de domínio (NORTHROP, 2008).

A atividade de gerenciamento, por sua vez, gerencia a interação entre as atividades de desenvolvimento do núcleo e do desenvolvimento do produto, para que ocorram de acordo com um planejamento coordenado (NORTHROP, 2008).

Em síntese, a interação entre as três atividades, por meio dos círculos da Figura 6, indica que estas são interligadas e iterativas. A atividade de desenvolvimento do núcleo de artefatos interage com a atividade de desenvolvimento dos produtos, gerando-os de forma sistemática, por meio da atividade de gerenciamento, que contempla a gerência técnica e organizacional.

Entre as diversas abordagens que apoiam as diferentes atividades de concepção e gerenciamento de uma LPS, estão as que apoiam a fase de análise de domínio, como a *Feature-Oriented Domain Analysis* (FODA) (KANG *et al.*, 1990), *Feature-Oriented Reuse Method* (FORM) (KANG *et al.*, 1998) e a *Feature-Object Oriented Modeling* (FOOM) (AJILA; TIERNEY, 2002). Algumas que apoiam a concepção dos ativos centrais da linha, com destaque para às estratégias ativa, proativa e reativa, definidas por Krueger (2002). Algumas que apoiam a definição de processos e metodologias como a *Product Line Software Engineering* (PuLSE) (BAYER *et al.*, 1999), arcabouço *Software Product Line Engineering* (SPLE) (Pohl, Bockle e Linden (2005), *Family-Oriented Abstraction, Specification, and Translation* (FAST) (WEISS; LAI, 1999) e *Product Line UML-based Software Engineering* (PLUS) (GOMAA, 2004). E, finalmente, as que apoiam o gerenciamento de variabilidades, como a *Stereotype-based Management of Variability* (SMarty) (MARCOLINO *et al.*, 2013; JUNIOR; GIMENES; MALDONADO, 2010), modelo ortogonal de variabilidade (POHL; BOCKLE; LINDEN, 2005), e a *Component-Based Application Development* (KobrA) (ATKINSON; BAYER; MUTHIG, 2000).

No contexto desta pesquisa foram adotados o arcabouço SPLE para auxiliar na condução das diferentes atividades para concepção da linha; a SMarty, para modelagem dos diagramas arquiteturais; e a estratégia proativa de estabelecimento dos artefatos de software.

2.4.2 O Arcabouço SPLE

O arcabouço intitulado Engenharia de Linhas de Produto de Software (SPLE) (do inglês, *Software Product Line Engineering*), proposto por Pohl, Bockle e Linden (2005), torna-se um forte aliado no processo de definição e estabelecimento de uma LPS. A Figura 7 apresenta o SPLE e seus subprocessos.

O SPLE possui nove sub-processos que subdividem-se em conjuntos menores de atividades. Tais subprocessos estão estabelecidos em dois ciclos de vida: o de (1) Engenharia de Domínio, que resulta nos ativos comuns, que juntos formam a plataforma da LPS, sendo também responsável por definir o escopo da linha e garantir que tal plataforma tenha as variabilidades

Figura 7 – Ciclo de Vida do Arcabouço SPLE.

Fonte: Adaptada de Pohl, Bockle e Linden (2005).

necessárias para apoiar o escopo dos produtos almejados; e o (2) de Engenharia de Aplicação que, ao considerar os ativos comuns e as variabilidades, integra-os para criar instâncias de produtos (NORTHROP; CLEMENTS, 2009). Em algumas abordagens, faz-se presente a atividade de Engenharia de Família, correspondendo ao gerenciamento organizacional e técnico da LPS (GOMAA, 2004). No SPLE, esta atividade é integrada à Engenharia de Domínio.

Uma última consideração a respeito do SPLE é que este foi desenvolvido considerando diferentes iniciativas conhecidas, podendo integrar diferentes métodos de desenvolvimento, e.g., RUP (*Rational Unified Process*) (POHL; BOCKLE; LINDEN, 2005), o tornando mais dinâmico e adaptável.

A seleção deste arcabouço para apoiar esta pesquisa se deve à sua completude no que tange às diferentes facetas para a concepção e manutenção da LPS. Finalmente, a capacidade do SPLE em integrar outros métodos e estratégias é também determinante no contexto deste projeto, já que a integração da ferramenta de apoio TANGRAM levou à modificações sucintas no processo de concepção da linha, no que se refere ao gerenciamento dos artefatos na criação de aplicações e na definição de diferentes módulos educacional, sendo melhor discutidas no Capítulo 5. Na perspectiva de modelagem dos diferentes diagramas da LPS, optou-se pela adoção da abordagem SMarty.

2.4.3 A Abordagem SMarty

O arcabouço SPLE indica a adoção do modelo ortogonal de variabilidade (POHL; BOCKLE; LINDEN, 2005) para representar as variabilidades por meio de um modelo. Contudo, uma desvantagem com a adoção de uma notação gráfica específica como essa, acaba requerendo o estudo para o entendimento da mesma, estudo esse que deve ser realizado pelos projetistas e também pela equipe de desenvolvedores que deverão interpretar tais modelos. Adicionalmente, a criação de tais modelos leva ao uso de ferramentas que as apoiam, e que em muitos casos são pagas. Para minimizar a necessidade de aprendizado de uma nova notação e também possíveis problemas com ferramentas e entendimento de tais modelos, a notação SMarty foi adotada (OLIVEIRAJR; GIMENES; MALDONADO, 2013; MARCOLINO *et al.*, 2013; MARCOLINO; OLIVEIRAJR, ; MARCOLINO *et al.*, 2017).

SMarty apoia a representação e gerenciamento de variabilidades em modelos UML (i.e., casos de uso, sequência, classes, atividades e componentes). Esses modelos são estendidos por meio de um Perfil UML, intitulado *SMartyProfile*. O perfil UML permite a integração facilitada dos atributos e elementos de SMarty em qualquer ferramenta UML que permite a importação de perfis UML. Além disso, SMarty possui um conjunto de diretrizes que guiam os usuários na aplicação e identificação de variabilidades em modelos UML, tal conjunto é chamado de *SMartyProfile*.

O *SMartyProfile* compreende os seguintes estereótipos: «variability», «mandatory», «optional», «alternative_OR», «alternative_XO»; «mutex» e «requires». Ele possui ainda um conjunto de atributos para apresentar informações sobre as variabilidades, como a rastreabilidade entre elas, o número máximo e mínimo de número de variantes, todos eles incluídos em um elemento gráfico de comentário da UML.

Outra razão para a seleção de SMarty foram os bons resultados do conjunto de avaliações experimentais, nas quais o processo de identificação de variabilidades foi comparado com o método de engenharia de software baseada em UML (PLUS) (do inglês, *UML-based Software Engineering Method*) e a abordagem de Ziadi *et al.* (MARCOLINO *et al.*, 2013; MARCOLINO; OLIVEIRAJR; GIMENES, 2014). Mesmo com uma melhor representação do método PLUS para o modelo de classes, principalmente pelo número reduzido de estereótipos, SMarty apresenta-se como uma melhor opção pelo conjunto de suas diretrizes. Ademais, como a abordagem permite o gerenciamento de variabilidades de seus estereótipos em um perfil, a adição de novos elementos sintáticos é facilitado (OLIVEIRAJR; GIMENES; MALDONADO, 2013), permitindo a constante evolução do núcleo de artefatos e atualização dos modelos, quando incluídas novas características. Finalmente, para a concepção dos artefatos de software, optou-se pela estratégia proativa.

2.4.4 Estratégias de Concepção do Núcleo de Artefatos

Para a criação dos artefatos de software da LPS, é necessário utilizar alguma técnicas que auxiliem no estabelecimento desses artefatos. Krueger (2002) apresenta três estratégias (Figura 8) – proativa, reativa e extrativa – para povoar o núcleo de uma LPS.

Figura 8 – Estratégias (a) Extrativa, (b) Reativa e (c) Proativa.

Fonte: Adaptada de Krueger (2002).

A estratégia extrativa, apresentada na Figura 8 (a), utiliza software existentes para formar a base inicial da LPS. A adoção desta estratégia deve refletir no uso restrito de reengenharia e tecnologias complexas para promover a extração dos diferentes artefatos a serem reutilizados.

A adoção da estratégia reativa, representada na Figura 8 (b), refere-se quando a LPS evolui incrementalmente, ou seja, quando novos produtos ou novos requisitos são necessários e não estão disponíveis no núcleo sendo, portanto, extraídos. É adequada quando, inicialmente, não é possível prever todos os requisitos para a derivação dos produtos.

A última estratégia é a proativa, apresentada na Figura 8 (c), correspondendo a atividades de desenvolvimento realizadas sequencialmente para que os requisitos sejam projetados, analisa-

dos e então implementados. Ocorre em um cenário previsível, no qual o conjunto de produtos a serem derivados são estáveis e podem ser definidos e planejados antecipadamente.

Nesta pesquisa, adotou-se a estratégia proativa, pela: (a) indisponibilidade de software de ensino e aprendizagem de programação para a condução da extração sistêmica (discutido no Capítulo 3); (b) disponibilização de um catálogo de requisitos *m-learning* no domínio de programação (discutido no Capítulo 4), a permitir a identificação prévia das funcionalidades a integrarem às diferentes aplicações; (c) necessidade de se desenvolver artefatos compatíveis com as tecnologias e a ferramenta de apoio TANGRAM; e (d) facilidade no desenvolvimento de artefatos de software do zero, devido ao estabelecimento de um projeto arquitetural desenvolvido com apoio da SMarty (discutido no Capítulo 5).

2.4.5 Benefícios e Limitações

O conceito de linhas de produtos é referência às linhas de produção das indústrias de manufatura, em especial, automobilísticas do final do século XIX (CARVALHO *et al.*, 2013). A adoção da abordagem de LPS como forma efetiva de reutilização sistemática de artefatos de software vem se consolidando no decorrer dos anos, tanto na indústria como na academia (POHL; BOCKLE; LINDEN, 2005; LINDEN; SCHMID; ROMMES, 2007; REINHARTZ-BERGER; KEMELMAN, 2019; KRUEGER; CLEMENTS, 2018; RABISER *et al.*, 2018). Essa adoção é motivada por uma série de fatores, como os mencionados por Linden, Schmid e Rommes (2007) e Pohl, Bockle e Linden (2005):

- **Redução de custos de desenvolvimento:** O reúso de artefatos disponibilizados em um repositório central diminui os custos no desenvolvimento de cada produto. Entretanto, é necessário o investimento inicial na criação deste núcleo de artefatos, antes da redução efetiva dos custos;
- **Melhoria da qualidade:** Os artefatos criados são revisados e testados em muitos produtos. Logo, a garantia de qualidade aumenta, tanto na detecção como na correção de falhas, o que, consequentemente, melhora a qualidade de todos os produtos;
- **Redução do *time to market*:** O tempo de produção inicial é mais alto, visto que o núcleo de artefatos deve ser produzido primordialmente. No entanto, o tempo posterior de produção de cada novo produto pode ser reduzido, uma vez que estes serão instanciados a partir de artefatos já existentes;
- **Redução do esforço de manutenção:** ao modificar um artefato do núcleo, as modificações são propagadas para todos os produtos que dele se utilizam, o que pode ser explorado visando a diminuição do tempo global de manutenção. Por exemplo, a realização de testes isolados em produtos pode ser explorada como forma de testar uma LPS;

- **Contribuição para a evolução:** Para atender à criação de produtos com novas funcionalidades, novos artefatos devem ser inseridos no núcleo da LPS. Isso permite a evolução da LPS, visto que é possível melhorar o desenvolvimento, garantindo a evolução dos produtos e a redução de esforços de desenvolvimento; e
- **Contribuição para redução da complexidade:** O aumento de solicitações de funcionalidades, por parte dos cliente, acarreta o aumento da complexidade dos produtos. No entanto, com a possibilidade de reusar partes comuns em uma LPS, a complexidade acaba sendo reduzida de forma significativa;

Já a principal limitação, que dificulta a obtenção dos benefícios mencionados, corresponde à dificuldade em planejar cuidadosamente a base de reuso a ser utilizada. O problema resume-se em modificar toda a estrutura organizacional, de processos de negócio e tecnológica. Tal alteração, em organizações de desenvolvimento de software, não é trivial, e a adoção inicial incorreta de LPS pode se tornar cara ou fatal (CLEMENTS; NORTHROP, 2002; NORTHROP, 2008).

Com base em estudos, Clements e Northrop (2002), Northrop (2008) apresentam um gráfico relacionando o custo acumulado com o número de sistemas desenvolvidos (Figura 9). Nota-se que linhas de produto iniciam com custos estimados maiores que o desenvolvimento de sistemas únicos. Contudo, após três sistemas desenvolvidos aproximadamente, LPS apresentam uma diminuição nos custos, se comparado ao desenvolvimento tradicional. A determinação do sucesso da adoção de LPS deve-se, portanto, à capacidade de permitir que os sistemas desenvolvidos possibilitem um maior portfólio de produtos no domínio definido, o que reflete na necessidade das mudanças mencionadas, nos diferentes contextos das empresas de software.

Figura 9 – Custo Acumulado por Número de Sistemas.

Fonte: Adaptada de Clements e Northrop (2002).

2.5 Considerações Finais

Neste capítulo foi sintetizada a fundamentação teórica sobre o domínio de ensino e aprendizagem de programação, em especial, uma pesquisa que permitiu identificar problemas e soluções, (i.e., software, estratégias didáticas), em tal domínio. Foram apresentadas, ainda, as principais características que tornam os dispositivos móveis atrativos como meios para se ensinar e a aprender a programar. Finalmente, discutiu-se a abordagem LPS, suas três atividades essenciais e diferentes arcabouços, abordagens e estratégias de concepção, bem como as justificativas para a seleção de algumas dessas para o estabelecimento da LPS desta pesquisa.

A classificação e pesquisa sobre problemas no domínio de programação fornece meios para se investigar soluções e estratégias para a mitigação de tais problemas. Para isso, adota-se a modalidade *m-learning* no desenvolvimento de aplicações móveis por meio de uma LPS e ferramenta de apoio. A adoção da abordagem de reúso, por sua vez, permite que diferentes problemas sejam integrados às *features* das aplicações, possibilitando a exploração e identificação de possíveis produtos de software que possam atender melhor os professores e alunos nos processos de ensino e aprendizagem de fundamentos programação.

Com o objetivo de identificar as LPS educacionais, em especial, as de domínio em programação; e identificar os software que apoiam o ensino e aprendizagem de programação, dois mapeamentos sistemáticos (MS) foram conduzidos e são apresentados no próximo capítulo.

LPS EDUCACIONAIS E SOFTWARES PARA O ENSINO E APRENDIZAGEM DE PROGRAMAÇÃO

3.1 Considerações Iniciais

Este capítulo discute dois mapeamentos sistemáticos (MS), nos quais o primeiro busca identificar linhas de produto educacionais e no domínio de programação (Seção 3.2) e o segundo apresenta software e aplicações educacionais para o ensino e aprendizagem de programação (Seção 3.3).

A condução de um MS no domínio de LPS educacionais e de programação, além de identificar possíveis lacunas na utilização de tal metodologia de reúso, busca apresentar propostas que possam ser utilizadas na condução do desenvolvimento da LPS desta pesquisa. Já o MS que busca identificar software e aplicações educacionais no domínio de ensino e aprendizagem de programação, discute questões tecnológicas e educacionais adotadas nestas soluções, bem como os problemas que elas buscam solucionar e suas lacunas. Adicionalmente, objetiva-se com a identificação de tais soluções propor um catálogo de requisitos para *m-learning*, fundamental para a definição do escopo e da arquitetura da LPS proposta e geração das *features* que serão integradas a formar as aplicações educacionais móveis.

3.2 Linhas de Produto de Software no Domínio Educacional e de Ensino e Aprendizagem de Programação

A fim de identificar as LPS existentes no domínio educacional, evidenciando a existência de possíveis linhas para o ensino de programação e distinguindo estratégias de concepção das mesmas para o estabelecimento da LPS proposta nesta pesquisa, um MS foi conduzido.

Em linhas gerais, mapeamento sistemático (MS) de literatura é um método de estudo secundário, que tem como objetivo identificar os estudos primários existentes, permitindo a visualização do estado da arte sobre o tema investigado (PETERSEN *et al.*, 2008). As diretrizes propostas por Kitchenham e Charters (KITCHENHAM; CHARTERS, 2007) e indicações de Biolchini et al. (BIOLCHINI; MIAN; NATALI, 2005) foram seguidas para a condução deste MS.

3.2.1 Metodologia, Planejamento e Execução

A concepção das questões de pesquisa seguiu o critério PICOC (*population, intervention, comparison, outcome, context*) (PETTICREW; ROBERTS, 2006), sendo:

- **População:** linhas de produto de software.
- **Intervenção:** propostas de LPS na literatura que atendam o desenvolvimento de software educacional para apoiar o ensino e aprendizagem em geral e, em especial, no domínio de ensino e aprendizagem de programação.
- **Comparação:** não se aplica.
- **Resultados:** visão geral dos estudos que propõem o uso de reuso por meio de LPS para apoiar o processo de ensino e aprendizagem. Espera-se obter contribuições significativas para a definição da LPS proposta nesta pesquisa, bem como as lacunas relacionadas a LPS de domínio educacional.
- **Contexto:** acadêmico e industrial, nos últimos dez anos;

Quatro questões foram derivadas após a definição do PICOC, a saber:

Q1. Quais as LPS existentes na literatura que atendem o desenvolvimento de software educacional para apoiar o ensino e aprendizagem em diversos domínios?

Q1.2 Alguma delas atende ao domínio de ensino e aprendizagem de programação?

Q2. Quais as especificações técnicas das LPS identificadas?

Q3. Quais dificuldades e/ou problemas têm sido enfrentados em LPS sob a perspectiva de ensino e aprendizagem?

Q4. Quais as avaliações e/ou validações experimentais foram aplicadas a estas propostas de LPS?

A Figura 10 resume as tarefas conduzidas para a recuperação dos estudos primários.

A primeira atividade corresponde ao planejamento, em que foram definidas as questões de pesquisa. Em seguida foram especificados os termos de busca para serem aplicados nas bibliotecas digitais e bases indexadas (ACM, IEEE, Science Direct e Scopus).

Figura 10 – Atividades do Processo de Busca e Seleção para LPS Educacionais.

Fonte: Elaborada pelo autor.

A definição ocorreu com base em indicações de professores, estudos sobre LPS e execução piloto. Os principais termos utilizados, em inglês, foram: “*software product line*”, “*education*”, “*educational*”, “*eletronic learning*”, “*mobile learning*”, “*television learning*”, “*distance learning*” e “*blended learning*”, com sinônimos e derivações. A primeira e segunda iterações foram realizadas com o objetivo de refinar e aprimorar os termos de busca. Optou-se por não incluir termos relacionados à programação, visto que na busca por LPS de domínio educacional geral, as de domínio de ensino e aprendizagem de programação seriam incluídas nos resultados. Após refinamento, a aplicação dos termos de busca direcionados aos títulos e resumos ocorreu no período de dezembro/2014 e janeiro/2015.

Para auxiliar nas fases de seleção preliminar e final, foram definidos critérios de inclusão e exclusão, eliminando estudos primários incongruentes aos objetivos da revisão. Os critérios de inclusão estão diretamente relacionados aos estudos que respondem às questões da revisão; já os de exclusão contemplam: (i) estudos que não apresentam o conceito de LPS aplicados ao ensino e aprendizagem em diferentes domínios; (ii) estudos resumidos; (iii) estudos duplicados e (iv) estudos que não puderam ser recuperados.

Com a aplicação dos termos de busca nas respectivas bases, 116 estudos foram recuperados. Destes, seis estavam duplicados. Os 110 estudos considerados inéditos passaram então pelo processo de seleção preliminar – leitura do título, palavras-chave, resumo e aplicação dos critérios de inclusão e exclusão. Um total de 104 estudos foram descartados na etapa e 12 foram considerados para a seleção final. A Tabela 4 apresenta a quantidade de estudos, no decorrer das fases de seleção, por base de busca¹.

Na etapa de seleção final os estudos foram lidos na íntegra. Dos 12 estudos, três foram descartados e nove foram considerados para a extração de dados e resposta às questões de pesquisa, correspondendo a um percentual de 8,18%, dos 110 estudos recuperados.

A existência de um estudo primário que não estava indexado nas bases de busca, por ter sido publicado em 2014, levou à inclusão manual deste, somando-se aos nove estudos reportados.

¹ O protocolo completo e as referências podem ser consultados em <<https://goo.gl/trRMBI>>

Tabela 2 – Estudos Primários Recuperados por Base de Busca.

#	Base	Qtd.	Descartados (Sel. Preliminar)	Sel. Preliminar	Descartados (Sel. Final)	Sel. Final
1	ACM	21	20	1	1	0
2	IEEE	37	30	7	2	5
3	Science Direct	36	35	1	0	1
4	Scopus	22	19	3	0	3
5	Outra	1	–	–	–	1
Total		116	104	12	3	10

Fonte: Dados da pesquisa.

A Tabela 3 apresenta os nove artigos selecionados pela busca nas bases e um adicionado manualmente, totalizando dez estudos para a extração de dados. Adicionalmente, apresenta-se a distribuição temporal dos estudos.

Tabela 3 – Estudos Primários Selecionados Ordenados por Ano.

Id	Título	Referência	Ano	Publicado em
E01	Towards the Establishment of a Software Product Line for Mobile Learning Applications	(FALVO JUNIOR <i>et al.</i> , 2014)	2014	Int. Conf. on Software Engineering and Knowledge Engineering.
E02	Building Families of Software Products for E-Learning Platforms: A Case Study.	(BARREIRO; GARCIA-SAIZ; PANTALEON, 2014)	2014	Iberoamericana Tech. Learning Journal.
E03	ISE-SPL: Uma Abordagem Baseada em Linha de Produtos de Software aplicada a geração Automática de Sistemas para Educação Médica na Plataforma E-learning.	(CARVALHO <i>et al.</i> , 2013)	2013	Journal of Biomedical Engineering.
E04	What Makes it Hard to Apply Software Product Lines to Educational Technologies?	(CHIMALAKONDA; NORI, 2013)	2013	Int. Workshop on PL. Apch. in Software Engineering.
E05	A Software Engineering Perspective for Accelerating Educational Technologies.	(CHIMALAKONDA; NORI, 2012b)	2012	Int. Conf. Advanced Learning Tech.
E06	Accelerating Educational Technologies Using Software Product Lines.	(CHIMALAKONDA; NORI, 2012a)	2012	Int. Conf. Tech. Enhanced Education.
E07	Software Product Line Engineering for E-learning Applications: A Case Study.	(SANCHEZ; GARCIA-SAIZ; ZORRILLA, 2012)	2012	Int. Symp. Comp. in Education.
E08	The Design of Codeless E-learning Animation Materials Developing Platform.	(ZHOU <i>et al.</i> , 2009)	2009	Int. Conf. Machine Learning and CybermeTIC.
E09	Product Lines for Digital Information Products.	(OBERWEIS; PANKRATIUS; STUCKY, 2007)	2007	Journal of Information Systems.
E10	Aspect-oriented re-engineering of e-learning courseware.	(PANKRATIUS; STUCKY; VOSSEN, 2005)	2005	Journal of Learning Organization.

Fonte: Dados da pesquisa.

3.2.2 Análise dos Resultados

A seguir são apresentadas as respostas e discussões para cada uma das questões de pesquisa. Para facilitar a referência aos estudos, o código da coluna “Id” da Tabela 3 foi utilizado na identificação de cada um dos estudos primários recuperados.

Q1. *Quais as propostas existentes de LPS na literatura que atendem o desenvolvimento de software educacional para apoiar o ensino e aprendizagem em diversos domínios?*

Os 7 estudos que propõem LPS educacionais que apoiam o ensino e aprendizagem são apresentados na Tabela 3.

Em *Towards the Establishment of a Software Product Line for Mobile Learning Applications* [E01] é apresentado a proposta de uma LPS para aplicações móveis educacionais

(*m-learning*), denominada M-SPLearning, de domínio educacional geral. A M-SPLearning foi desenvolvida seguindo o paradigma proativo de desenvolvimento, com arquitetura orientada a serviços (SOA), em trabalho de mestrado no grupo no qual a pesquisa deste doutorado está sendo desenvolvida. Neste sentido, por ter sido a única linha que aborda *m-learning* e por se mostrar bastante abrangente, suas contribuições foram significativas no estabelecimento da LPS desta pesquisa.

Em *E-learning Web Miner* (EIWM) [E02, E07] uma LPS é desenvolvida por meio de refatoração de artefatos extraídos de uma aplicação de mineração de dados chamado *E-learning Web Miner* (EIWM). O EIWM fornece informações úteis aos instrutores na educação virtual para que o processo de ensino e aprendizagem possa ser aprimorado. A linha gera, portanto, produtos que realizam a mineração de informações com diferentes configurações e contextos, para ambientes *e-learning*.

No estudo *Interactive Spaced-Education SPL* (ISESPL) [E03] é apresentada a concepção de uma abordagem baseada em LPS aplicada à geração automática de sistemas para educação médica para plataforma *e-learning*, com base na ferramenta *Interactive Spaced-Education* (ISE) para educação médica.

Em *Towards Automating the Development of a Family of eLearning Systems* (TALES) [E04, E05, E06] propõe-se a criação de uma LPS voltada à alfabetização por meio de mineração de artefatos de nove sistemas *e-learning* existentes. Todos os sistemas foram desenvolvidos por equipes geograficamente distribuídas e com processos de desenvolvimento específicos. Essas propostas atendem 284 milhões de pessoas na Índia, ensinando 22 dialetos provenientes da língua indiana.

Em [E05] os autores apresentam a pretensão futura de aprimorar o uso de LPS no domínio educacional. Já em [E06] os autores apresentam o perfil da pesquisa, em um nível mais resumido. Em última instância, detalhes técnicos são pouco discutidos, como o próprio autor relata, limitando a utilização de conceitos utilizados nos trabalhos em outros nichos.

Em *Codeless e-learning Animation Materials Developing Platform* [E08] é proposta uma LPS para a criação de materiais de animação para ambientes *e-learning*, na qual os usuários podem construir animações de modo colaborativo por meio de interações com componentes, sem a necessidade de escrever código.

A existência de cinco tipos de representações para os conteúdos educacionais (texto, figuras, vídeo, áudio e animação), conforme definido pelo comitê da IEEE de padrões tecnológicos de aprendizagem (IEEE, 2002), requerem atenção e consequentemente recursos para o seu desenvolvimento. Entre eles destacam-se os de efeitos visuais e interações, que podem estimular o interesse dos estudantes, aprimorando a compreensão em experiências de aprendizagem.

Em [E09], uma proposta denominada *Product Lines for digitAl iNformation producTs* (PLANT) aplica conceitos de LPS para produtos de informação digital (DIPs). A abordagem

gerencia as similaridades de DIPs, por meio da definição de requisitos comuns, limita as variabilidades antecipadamente, assim como planeja e coordena o reuso. Resumidamente, trata-se de uma abordagem para a concepção de DIPs em diferentes domínios.

No estudo intitulado *Aspect-oriented Re-engineering of E-learning Courseware* [E10] é proposto o processo de reengenharia orientado a aspectos de materiais educacionais *e-learning*. Utilizando programação orientada a aspectos, os interesses transversais são extraídos de materiais educacionais existentes, tornando-se componentes que incluídos em uma LPS, fornecem um arcabouço global para coordenar a reengenharia e reuso desses.

Q1.2. *Alguma delas atende ao domínio específico para o ensino e aprendizagem de programação?*

Como observado nos estudos primários, nenhuma das LPS propostas foi desenvolvida especificamente para promover o ensino e aprendizagem de programação.

Nota-se que do total de sete propostas, três mantêm domínio na criação de conteúdo educacional – um de animação, e os demais conteúdos para diferentes áreas e disciplinas, uma proposta têm domínio em ferramenta de auxílio no processo de ensino e aprendizagem, uma em *e-learning* para alfabetização e uma para *m-learning*.

Q2. *Quais as especificações técnicas das LPS que apoiam o desenvolvimento de produtos de software, no processo de ensino e aprendizagem?*

Para responder esta pergunta, o Quadro 5 resume, por proposta, as tecnologias utilizadas. Para melhor identificar quais atividades da LPS elas apoiam, estas foram classificadas em uma das três atividades:

1. **Engenharia ou Gerenciamento de Família:** atividade que designa o processo de gerenciamento organizacional e técnico da LPS e que, em alguns casos, é parte integrante da Engenharia de Domínio;
2. **Engenharia de Domínio:** responsável pela criação e gerência do núcleo de artefatos (similaridades e variabilidades); e
3. **Engenharia de Aplicação:** atividade na qual os produtos são instanciados por meio dos artefatos provenientes da atividade anterior (Engenharia de Domínio).

A coluna “Abordagem” indica qual(is) o(s) tipo(s) de abordagem(ns) (KRUEGER, 2002) são empregadas nas propostas. Em exceção aos demais, o estudo [E08] não especifica (U) o tipo de abordagem que utiliza. A coluna “Outras Tecnologias” resume as tecnologias utilizadas de modo geral, não sendo possível identificar precisamente em quais das três atividades foram utilizadas. Quanto à coluna “Eng. de Família”, apenas uma abordagem relata um sistema utilizado para o gerenciamento de configurações, no caso, o CCMS, o qual a referência não foi

indicada. Para as demais atividades, os estudos pontuam a utilização da abordagem *Feature-oriented domain analysis* (FODA), na qual o modelo de características é definido e tecnologias e ferramentas específicas adotadas.

Quadro 5 – Tecnologias Adotadas.

Id	Referência	Abordagem	Eng. de Família	Eng. Domínio	Eng. Aplicação	Outras Tecnologias
1	E01	P	?	Catálogo de requisitos no domínio móvel (DUARTE FILHO; BARBOSA, 2013), SMarty (JUNIOR; GIMENES; MALDONADO, 2010), FODA (KANG <i>et al.</i> , 1990).	<i>Representational State Transfer</i> (REST), Java, Android.	Service-Oriented Architecture (SOA).
2	E02, E07	E	?	FODA (KANG <i>et al.</i> , 1990), Hydra tool, <i>Epsilon generation Language</i> (ROSE <i>et al.</i> , 2008).	?	HTML, AJAX, dynamic web pages.
3	E03	E, P	?	FODA (KANG <i>et al.</i> , 1990).	<i>Software product lines on-line Tools</i> (S.P.L.O.T.).	Django, Python, PostgreSQL.
4	E04, E05, E06	E	?	?	eScript tool (reference not found).	<i>Java Script Flash Language</i> (JSFL), visual basic e C++.
5	E08	U	?	?	?	<i>Codeless development</i> .
6	E09	P	?	?	<i>desktop workflow engine</i> (DWE).	Regras de derivação, gramática, e o <i>Query and program EXecution Nets</i> (QX Nets) (AALST; HEE, 2004).
7	E10	E	Sistema de gerenciamento de configuração e comparação (CCMS).	?	?	Aspect Oriented Programming (AOP), Reengenharia, Engenharia Reversa.

P-Proativa, E-Extrativa, R-Reativa, U-Não especificada.

Fonte: Dados da pesquisa.

Q3. *Quais dificuldades e/ou problemas têm sido enfrentados em LPS sob a perspectiva de ensino e aprendizagem?*

O Quadro 6 apresenta o mapeamento das abordagens em relação aos principais problemas relatados e que motivaram a adoção de LPS nos trabalhos primários recuperados. Para facilitar a identificação dos problemas, seis categorias foram criadas: (1) custos, que refere-se ao valor elevado para se desenvolver software educacionais; (2) esforços, que se refere à quantidade de recursos necessários para a concepção de software educacionais; (3) ausência de padrões, indicando a necessidade de se estabelecer padrões para os diferentes artefatos educacionais reutilizados; (4) incipiência no domínio explorado, que se refere a necessidade de se investigar a adoção de LPS no domínio educacional; (5) dificuldades no desenvolvimento, que se refere a dificuldade no desenvolvimento de software educacionais, principalmente em relação ao entendimento do domínio e suas regras de negócio; e (6) outros problemas, não incluídos nas categorias anteriores, que se refere especificamente, a transversalidade de conteúdos desenvolvidos e integrados a diferentes módulos.

Quadro 6 – Problemas Mapeados.

Id	Estudo	Custo	Esforços	Padronização	Domínio	Desenvolvimento	Outros Problemas
1	E01	X	X	X	X	X	
2	E02, E07		X	X	X		
3	E03		X			X	
4	E04, E05, E06	X	X	X		X	
5	E08				X	X	
6	E09	X		X		X	
7	E10		X	X		X	X

O “X” relaciona o problema (coluna) com o respectivo trabalho (linha).

Fonte: Dados da pesquisa.

Q4. Quais as avaliações e/ou validações experimentais conduzidas a estas propostas de LPS?

Apenas dois estudos primários apresentaram alguma iniciativa de validação/avaliação das propostas nos estudos recuperados. Em [E03] é conduzida uma validação da LPS, utilizando a *Software Product Lines on-line Tools* (S.P.L.O.T.)², que avalia o modelo de características da linha, e uma avaliação do processo de criação dos produtos, por 30 docentes da área de saúde, identificando tempo e facilidade na concepção de três produtos.

Em [E01], também com o intuito de validar as características levantadas na análise de domínio, um questionário com 20 especialistas da área de LPS e *m-learning* foi conduzido, permitindo ajustes no catálogo de requisitos inspecionado. Os demais trabalhos não apresentam validação, porém alguns indicam como pretensão futura a realização de avaliações experimentais.

3.2.3 Discussões Gerais e Lacunas Encontradas

Como pode ser observado nos estudos primários recuperados no MS, esses atendem a diferentes domínios no contexto educacional. *E-learning* e os conteúdos associados são os que apresentam esforços predominantes para permitir o desenvolvimento de software educacionais. Em contrapartida, nota-se a incipiência de propostas para *m-learning*. Finalmente, quando transferido o foco para LPS no domínio de ensino de programação, observa-se a inexistência de iniciativas na área.

Identificou-se também a ausência de especificações metodológicas e técnicas utilizadas na concepção e manutenção das propostas de LPS. A difusão de tais informações podem viabilizar a realização das diferentes atividades que envolvem o estabelecimento de uma LPS, de modo a reduzir esforços e custos, uma vez que tais atividades não são triviais e podem limitar a adoção dessa abordagem.

Em relação as estratégias para concepção de LPS, o estudo [E07] ganha destaque, já que os autores utilizam um software existente para conduzir a extração de artefatos para popular o núcleo de uma LPS. Esse software foi desenvolvido por um dos autores (ZORRILLA, 2011), o

² <<http://www.splot-research.org/>>

que facilita o acesso ao código fonte e a consequente extração de seus artefatos. É justamente a disponibilização de software que tornam a abordagem extrativa pouco utilizada, visto que a extração só é possível quando há disponibilização de software para a condução mesma. Ademais, em determinados domínios, faz-se necessário esforços significativos para adaptar tais artefatos à LPS que deverá integrar (KRUEGER, 2002).

Em relação aos diferentes problemas no domínio educacional que tentam ser mitigados pela adoção de LPS estão, principalmente, a busca por redução de esforços, padronização e desenvolvimento sistêmico. Logo, a soma de tais vantagens com a necessidade de se desenvolver *features* para mitigar problemas no domínio de programação, criam um cenário propício à exploração de LPS, uma vez que essa abordagem permite a geração de uma gama maior de produtos que podem, deste modo, atender um maior número de usuários.

Já a incipiência de LPS que geram produtos *m-learning* no domínio de ensino e aprendizagem de programação e os esforços nítidos no desenvolvimento de *features* para *e-learning* criam um cenário desafiador, no sentido de investigar se essas *features* podem ser adaptadas para *m-learning*.

Finalmente, em relação a adoção de estratégias para popular a LPS, faz-se necessária a adoção da estratégia proativa, uma vez que, como identificado por um MS conduzido para recuperar software para o ensino e aprendizagem de programação, discutido na próxima seção, poucos software estão disponibilizados de modo a permitirem a extração de seus artefatos. Adicionalmente, o processo de extração dos software existentes acabam encontrando barreiras no projeto arquitetural e escolhas tecnológicas, o que dificulta ainda mais o processo de adaptação de tais artefatos e o atendimento dos mesmos à LPS projetada. A dificuldade na obtenção dos fontes de projetos para possível extração de artefatos é uma limitação presente. No entanto, a adoção de LPS e sua concepção, mesmo que proativamente, traz benefícios que, em curto prazo, favorecem a evolução do portfólio de produtos instanciados de uma linha (KRUEGER; CLEMENTS, 2018; RABISER *et al.*, 2018).

Este mapeamento teve seus resultados publicados em (MARCOLINO; BARBOSA, 2015a), sendo que uma nova iteração a identificar possíveis novas LPS no domínio educacional e de ensino de programação foi conduzido recentemente. Essa atualização é apresentada brevemente no Apêndice B.

Nesta atualização cinco estudos foram selecionados no período de 2014 até 2018, com exceções para dois estudos primários que não foram recuperados na primeira execução do mapeamento, referente ao ano de 2012. Assim, constatou-se poucos estudos que adotam a modalidade *m-learning* no contexto de LPS. Contudo, os estudos identificados se mostram interessantes nos contextos tecnológico e educacional que estão inseridos. Sendo, portanto, considerado para complementar os estudos primários identificados na primeira iteração do MS.

3.3 Software Educacionais para o Ensino e Aprendizagem de Programação

Um segundo MS foi conduzido a fim de identificar de software para ensino e aprendizagem de programação para permitir: (i) a criação de um catálogo de requisitos para os produtos da LPS desta pesquisa; (ii) possível extração de artefatos dos mesmos também para a LPS aqui discutida; e (iii) a disponibilização da lista de tais software, para contribuir com a comunidade escolar que procuram soluções em tal domínio. Este MS buscou trabalhos em nível nacional e internacional, não se restringindo aos software e aplicações utilizados apenas no ensino de programação a iniciantes, como visto em [Aureliano e Tedesco \(2012\)](#).

3.3.1 Metodologia, Planejamento e Execução

Este MS também seguiu as diretrizes propostas por ([KITCHENHAM; CHARTERS, 2007](#)) e por ([BIOLCHINI; MIAN; NATALI, 2005](#)). Assim, o critério PICOC foi definido para o estabelecimento das questões de pesquisa, sendo:

- **População:** software educacionais.
- **Intervenção:** software educacionais no domínio de ensino e aprendizagem de programação.
- **Comparação:** não se aplica.
- **Resultados:** identificar os tipos de software que apoiam o ensino e aprendizagem de programação, gerando uma base referencial teórica sobre esses software, incluindo especificações técnicas, metodológicas e educacionais (e.g., modalidades de ensino, estratégias didáticas e pedagógicas).
- **Contexto:** acadêmico, nos últimos dez anos;

Ao todo, cinco questões de pesquisa (QP) foram definidas para auxiliar neste MS, sendo elas:

QP1. Quais software educacionais têm apoiado o ensino de programação?

QP2. Quais as modalidades de ensino apoiadas?

QP3. Qual a linguagem de programação, sistema operacional (SO) e técnicas de desenvolvimento são adotadas?

QP4. Quais as estratégias de ensino adotadas?

QP5. De que forma os software e aplicações educacionais foram avaliados?

As diversas atividades conduzidas para a busca, seleção dos estudos e para a realização da extração dos dados³ são apresentadas na Figura 11.

Figura 11 – Atividades do Processo de Busca e Seleção de Estudos Primários.

Fonte: Elaborada pelo autor.

A primeira atividade (Figura 11), corresponde ao planejamento, no qual foram criadas as questões, com apoio do critério PICOC (PETTICREW; ROBERTS, 2006). Em seguida, juntamente com duas iterações piloto, os termos de busca foram definidos: “software educacional” e “ensino de programação” e “*educational software*” e “*programming teaching*”, com seus sinônimos e variações.

Os termos de busca definidos foram aplicados nas bibliotecas digitais e bases indexadas da ACM, IEEE, ScienceDirect, Compendex e Scopus (em inglês). Também foram consideradas as bases da Revista Brasileira de Informática na Educação (RBIE) e de periódicos da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) (em português). A busca foi realizada em 2014, abrangendo o período de 2004 a 2014.

Na atividade de seleção preliminar foram aplicados critérios de inclusão e exclusão. O critério de inclusão restringe a seleção aos estudos que abordam software ou aplicações educacionais para o ensino de programação e conteúdos relacionados. Já os critérios de exclusão foram: estudos primários resumidos, estudos duplicados, estudos indisponíveis e estudos em outros idiomas que não sejam o inglês e o português. Ao final da seleção preliminar foram obtidos 122 estudos.

Os 122 estudos primários selecionados foram submetidos à leitura na íntegra, sendo que 81 foram selecionados para responder as questões de busca definidas. A Tabela 4 sumariza os resultados obtidos nas etapas de seleção.

3.3.2 Análise dos Resultados

Os estudos recuperados apresentaram maior concentração de publicação nos anos de 2012 a 2013, como pode ser visto na Tabela 5. A seguir são apresentadas as respostas para as questões de pesquisa do MS.

³ O protocolo completo e as referências podem ser consultados em <<https://goo.gl/4hu1gZ>>

Tabela 4 – Estudos Primários Recuperados por Base de Busca.

#	Base	Qtd.	Descartados (Sel. Preliminar)	Sel. Preliminar	Descartados (Sel. Final)	Sel. Final
1	ACM	38	35	3	2	1
2	Compendex	397	325	72	20	52
3	IEEE	40	29	11	3	8
4	Periódicos CAPES	63	62	1	1	0
5	RBIE	10	9	1	0	1
6	ScienceDirect	17	11	6	2	4
7	Scopus	621	593	28	13	15
Total		1186	1064	122	41	81

Fonte: Dados da pesquisa.

Q1. *Quais software educacionais têm apoiado o ensino de programação?*

A Tabela 5 apresenta o catálogo dos 81 estudos e o nome dos respectivos software/iniciativas identificados para promover o ensino de programação. Mais detalhes são apresentados nas respostas às demais questões de pesquisa.

Q2. *Quais as modalidades de ensino apoiadas?*

O Quadro 7 (a) apresenta as modalidades de ensino consideradas nos estudos primários recuperados. Os estudos que não especificaram explicitamente a modalidade foram classificados de acordo com suas características.

Dos 81 estudos, 35 são utilizados isoladamente, em *desktops* para a modalidade de ensino presencial, 30 referem-se a ambientes de aprendizagem eletrônicos (*e-learning*) (GEORGIEVA; SMRIKAROV; GEORGIEV, 2005), dois estudos abordam tecnologias móveis (*m-learning*) e apenas um estudo aborda tanto *m-learning* quanto *e-learning*, sendo classificado como *blended learning*. Constata-se ainda a inexistência de estudos que utilizam a modalidade de ensino *t-learning*.

Q3. *Qual a linguagem de programação, sistema operacional (SO) e técnicas de desenvolvimento são adotadas?*

Os Quadros 7 (b), (c) e (d) apresentam as linguagens de programação, os sistemas operacionais (SOs) e as tecnologias de desenvolvimento adotadas nos software e aplicações, respectivamente.

A linguagem de programação mais utilizada no desenvolvimento dos software e aplicações foi a linguagem Java (24 estudos). O Quadro 7 (c) reflete esta constatação, visto que 35 são multiplataforma. Finalmente, os poucos estudos que apresentam indicações de reuso estão relacionadas à tecnologia Java (7), à utilização do padrão *Model-view-controller* (2), arquiteturas orientadas a serviços (3), e desenvolvimento baseado em componentes (1).

Apenas o estudo [E60] adota o padrão *Shareable Content Object Reference Model*

Tabela 5 – Catálogo de software e Aplicações Educacionais para o Ensino de Programação

Id	Tipo de Publicação	Software/Iniciativa	Ano	Id	Tipo de Publicação	Software	Ano
E1	Conference	Jype	2010	E42	Conference	Generic Code-Memory Diagram Animation Creator	2004
E2	Conference	JFLAP	2013	E43	Conference	PGT	2007
E3	Conference	Cloud C++ Workbook	2013	E44	Conference	Greenfoot	2004
E4	Journal	MoCAS	2014	E45	Conference	TouchDevelop	2012
E5	Workshop	Eclipse com Eli	2005	E46	Conference	BlueFix	2012
E6	Journal	COALA	2005	E47	Conference	Multi-faceted Support for MOOC	2012
E7	Conference	Dolittle	2007	E48	Conference	ViLLE	2010
E8	Conference	Graphical Tutoring of Message Passing	2008	E49	Conference	FLOP	2012
E9	Conference	PlayLOGO 3D	2011	E50	Conference	Web-Based Environment	2005
E10	Conference	A Framework for Programming Process Measurement	2011	E51	Conference	NXT-G	2013
E11	Conference	Java Programming Learning System	2012	E52	Conference	PROGTEST	2011
E12	Conference	TSUNA-TASTE	2011	E53	Conference	SimplifIDE	2008
E13	Conference	Eclipse with Robot Simulator	2012	E54	Conference	BeReT	2006
E14	Conference	HIT Typing, Serious Cube and Two 3D games	2012	E55	Conference	PROgram Visualization Tool	2009
E15	Conference	CABLE	2006	E56	Transactions	Greenfoot	2010
E16	Transactions	Alice, Greenfoot and Scratch	2010	E57	Conference	MUPPETS	2004
E17	Conference	AgentWeb	2012	E58	Conference	CodeSpells	2013
E18	Conference	Tool for web-based animations	2005	E59	Conference	Storytelling Alice	2007
E19	Conference	DrRacket	2011	E60	Conference	KU-UZEM	2010
E20	Conference	CAL-CT Tutors	2011	E61	Conference	Learning Environment for Algorithm and Programming	2009
E21	Conference	VIP Visualization Tool	2007	E62	Transactions	ALVIS Live	2006
E22	Journal	WebHat	2013	E63	Conference	mJeliot	2011
E23	Conference	LDS	2010	E64	Journal	AutoLEP	2004
E24	Conference	ANNA	2008	E65	Conference	Learning Environment for Understanding of Program Division Patterns	2007
E25	Journal	UOM-TMD	2011	E66	Transactions	State-Based Squeak	2012
E26	Conference	C-Sheep	2007	E67	Conference	PROGTEST	2008
E27	Conference	Modkit and LilyPad ProtoSnap board	2013	E68	Conference	RPA System for MOODLE	2011
E28	Conference	Automatically Generated Programming Tutorials	2013	E69	Transactions	Peer-Learning Agent Based on Pair Programming	2010
E29	Journal	JPLAS	2013	E70	Conference	Tool Programming Process Visualizer	2013
E30	Conference	Pick Cricket	2011	E71	Conference	Real-time Monitoring System for Programming Education	2006
E31	Conference	SICAS	2007	E72	Conference	MDAT	2013
E32	Journal	Web-CAT	2008	E73	Transactions	Jeliot ConAn	2010
E33	Conference	Tutoring System	2008	E74	Journal	COALA	2012
E34	Conference	Active-learning Environment	2011	E75	Conference	ASIDE	2013
E35	Conference	MOOC	2013	E76	Conference	Game-Based Concept Visualization for Learning Programming	2011
E36	Workshop	Prolog Visualization System	2009	E77	Conference	JPL	2013
E37	Conference	Corrective Feedback and Concept Visualisation Tool	2011	E78	Journal	QA cycle execution system	2007
E38	Conference	Parson's Programming Puzzles	2006	E79	Conference	3DAPS	2011
E39	Workshop	Jason (AgentSpeak(L))	2012	E80	Conference	JBuilder	2012
E40	Conference	MSWLogo	2007	E81	Conference	LEGO Mindstorms Simulator (LMS)	2008
E41	Conference	Pen-Based Flowchart Recognition System	2008				

Fonte: Dados da pesquisa.

Quadro 7 – Modalidades de Ensino e Especificações Técnicas.

a) Modalidades de Ensino

#	Modalidade	Qtd.
1	Ensino presencial mediado por tecnologias	35
2	<i>Eletronic learning (e-learning)</i>	30
3	Não especificado	5
4	Ensino presencial mediado com Jogos /Gameificação ou Design de Jogos	4
5	Ensino presencial mediado por Robôs/ProtoSnap Board	3
6	<i>Blended learning (B-learning)</i>	2
7	<i>Mobile Learning (M-learning)</i>	2
Total		81

b) Linguagens de Programação

#	Linguagem	Qtd.
1	Não especificado	47
2	Java	24
3	Outras Linguagens	6
4	C#/ASP.Net	4
Total		81

d) Sistema Operacional/Plataforma

#	SO/Plataforma	Qtd.
1	Multiplataforma	35
2	Não especificado	34
3	Windows	5
4	Linux e Windows	2
5	Windows Phone	1
6	Mac OS X	1
7	Windows e iOS	1
8	Android	1
9	UNIX	1
Total		81

c) Tecnologias de Desenvolvimento

#	Tecnologias de Desenvolvimento	Qtd.
1	Não especificado	60
2	Tecnologia Java	7
3	Arquitetura <i>Model-View-Controller</i> (MVC)	2
4	Arquitetura orientada a serviços	3
5	Conjunto de instruções RISC (linguagem de máquina)	1
6	Google App Engine	1
7	Abordagem baseada em componentes	1
8	Módulo de física <i>Open Dynamics Engine</i>	1
9	Integração com camadas de middleware multi agente	1
10	Cliente móvel, servidor e protocolo de comunicação baseado em XML	1
11	Desenvolvimento com Hot-Potato	1
12	XML para troca de mensagens (servidor e módulo)	1
13	Arquitetura de terceira geração, com armazenador de logs integrado.	1
Total		81

Fonte: Dados da pesquisa.

(SCORM)⁴. A adoção de tal padrão permite a criação e reutilização de componentes, principalmente em sistemas de gerenciamento de conteúdos como o Moodle⁵. Esse padrão auxilia ainda na compatibilidade de recursos educacionais sendo uma estratégia interessante para permitir a integração entre componentes e até software educacionais.

Q4. Quais as estratégias de ensino adotadas?

Para classificar os software e as estratégias que utilizam, a classificação de ferramentas de software de Pears *et al.* (2007) foi adotada, sendo ela: (i) ferramentas de visualização; (ii) de avaliação automática; (iii) de apoio à programação; (iv) micromundos; e (v) outras ferramentas, que usam estratégias específicas, não incluídas nas categorias anteriores.

A Figura 12 apresenta o *bubble plot* dos software categorizados e a quantidade destes em relação ao ano em que foram publicadas. Nota-se uma diversificação nas estratégias adotadas. Algumas delas integram conceitos pedagógicos interessantes para motivar e instruir os alunos, como o CABLE (*Cognitive Apprenticeship-Based Learning Environment*) [E15], que fornece

⁴ <<http://www.scormsoft.com/scorm>>

⁵ <<https://moodle.org>>

retorno ao instrutor por meio de avaliação cognitiva da aprendizagem do aluno. Já o SimplifIDE [E53] propõe um IDE simplificado para facilitar o entendimento de conceitos relacionados à programação.

Figura 12 – Estratégias de Software e Ano de Desenvolvimento.

Fonte: Dados da pesquisa.

Alguns estudos abordam o ensino de programação para níveis básicos, como em [E16], que apresenta três ferramentas que apoiam o ensino em diferentes fases: *Alice* para estudantes de nível primário; *Scratch*, para os de idade de 8 a 16 anos; e o *Greenfoot*, para os com idade superior a 14 anos. As três integram conceitos visuais e lúdicos para promover e motivar os estudantes.

Entre estratégias de visualização de código, destacam-se as que apresentam o código por meio de blocos de montar, permitindo interação do aluno com um avatar 3D (*3D Avatar Programming System*) [E79], e as que adotam conceitos lúdicos. A utilização de estratégias de visualização apoia a aprendizagem, facilitando-a, principalmente, para os conceitos básicos de programação para alunos de séries elementares. Promove-se também a motivação, uma vez que a diversão é a intermediadora entre aluno e conteúdos ensinados.

Alguns estudos também adotam a interatividade, como o uso de *tablets* e controles de vídeo game para motivar discentes de diferentes níveis escolares, como quatro jogos apresentados no estudo [E14]. Há, ainda, iniciativas que promovem a criação de animações 3D, *Storytelling Alice* [E59], adaptação da ferramenta *Alice* [E16] e as que utilizam metáforas, representando comandos e funções em “feitiços” – CodeSpells [E58]. O uso de metáforas é classificado como micromundos, termo que designa ambientes de aprendizagem nos quais é possível explorar, descobrir e simular acontecimentos da vida real ou de fantasia (JONASSEN, 2007).

A criação de *plugins* com ambientes de desenvolvimento integrado (IDE) como em [E5, E6, E13, E21, E75] também é apresentada como estratégia, com destaque para o COALA (*Computer Assisted Environment for Learning Algorithms*) [E6]. Ele integra *plugins* para a IDE Eclipse, criando um ambiente web integrado. Existem também as ferramentas de apoio à

programação, e as que apoiam a avaliação automatizada, as que utilizam a web, acesso remoto, armazenamento na nuvem, e outras tecnologias de comunicação, como dispositivos móveis [E11, E12, E20, E22, E28, E71, E77]. Algumas propostas englobam agentes inteligentes para fornecer retorno, tanto ao corpo discente como docente, auxiliam as atividades de desenvolvimento e apontam erros cometidos durante as atividades [E67, E68].

Há, por fim, estratégias diferenciadas, classificadas como “outras ferramentas”, como as que utilizam robôs, tanto reais quanto virtuais, para permitir a visualização da interação do código com os mesmos [E27, E30, E34, E51, E81]. Merecem destaque as que utilizam *hardware* especial, como o *Modkit* e *LilyPad ProtoSnap board* [E27], que permitem a programação de *proto boards* costuráveis em roupas para motivar, sobretudo, estudantes do sexo feminino. Outra estratégia que se destaca no monitoramento do desempenho dos alunos e os reporta ao professor, visando que esse possa modificar as estratégias de aula adotadas, é a chamada *Real-time Monitoring System for Programming Education* [E71].

Q5. De que forma os software e aplicações educacionais foram avaliados?

Dos 81 estudos, 24 aplicaram questionários aos usuários para avaliar o uso do software/aplicação. Por meio destes foram identificados problemas e o quão adequadas tais iniciativas eram para ensinar o conteúdo de programação abordado nas perspectivas de alunos e professores.

Dos demais estudos, 24 avaliaram resultados coletados nos software, como notas e pontuações atribuídas na realização de atividade. Desse total, onze estudos adotaram teste de hipóteses, por meio de comparações com outros recursos instrucionais para a aprendizagem de programação. Quatro estudos aplicaram comparação dos resultados entre software, técnicas e grupos de controle. Seis estudos utilizaram técnicas de avaliação específicas, como comparações qualitativas, estudos de caso, hipóteses testadas com questionários, análise do conhecimento pré e pós uso do software. Finalmente, 12 estudos não apresentam avaliações, mas indicaram a possível realização destas em estudos futuros.

3.3.3 Discussões Gerais e Lacunas Encontradas

A partir do MS conduzido foi possível observar que as propostas de software educacionais para o ensino de programação apontam diversas questões e também possibilidades a serem exploradas, conforme elencado a seguir:

Tópicos abordados: a predominância de software e aplicações que abordam o ensino de programação com linguagens específicas, como a linguagem Java e C, deve ser avaliada. De fato, se os fundamentos de programação não estão claramente entendidos pelos estudantes, a utilização de uma linguagem específica pode não resultar no atendimento completo do objetivo educacional almejado, ou seja, aprender a programar. Esse fato ficou evidente nos resultados da pesquisa com alunos (Seção 2.2.2).

Modalidades de Ensino: ainda incipiente exploração de tecnologias, como dispositivos móveis e a restrição no uso de tecnologias em ambientes escolares, visto que estes apoiam principalmente o ensino informal, motivam uma adoção mais efetiva da modalidade *m-learning*.

Especificações Técnicas: as linguagens de programação, sistemas operacionais (SO) ou plataformas e as técnicas adotadas para o desenvolvimento das propostas são aplicadas, essencialmente, no método singular de desenvolvimento. Não foram observados esforços significativos no intuito de facilitar o processo de reúso. Dentre as que primam algum nível de reutilização, nenhuma adota LPS.

A concepção isolada da maioria dos software recuperados, sem a integração de diferentes grupos de pesquisa, e a pouca abrangência em relação aos problemas de programação que enfrentam, restringe a sua adoção para um número maior de usuários. A execução de mais avaliações experimentais e replicações em outros grupos de pesquisa poderia ajudar tanto a identificar se a adoção destes software é válida para outras instituições quanto verificar o quão adaptáveis e amplas são as soluções em relação aos problemas de ensino e aprendizagem de programação que atacam.

Técnicas e Estratégias para Promover o Ensino de Programação: as principais estratégias adotadas aplicam conceitos visuais, elementos lúdicos e outras utilizam a Internet para garantir que as atividades possam ter continuidade fora dos limites escolares. Porém, em conjunto com as modalidades de ensino e também especificações técnicas, a utilização de outros elementos tecnológicos, tais como as tecnologias móveis, ainda é pouco explorada. Estas tecnologias, amplamente difundidas, tornam-se meios em potencial de atingir um maior número de alunos e, consequentemente, um meio de motivá-los a aprender a programar. Portanto, identificar se tais estratégias são válidas e se podem expandir as fronteiras da aprendizagem de programação ao serem integradas a modalidades de ensino a distância é uma lacuna interessante a ser estudada.

Este mapeamento teve seus resultados publicados em (MARCOLINO; BARBOSA, 2015b) e, assim como o MS anterior, foi reexecutado considerando o período de 2014 até 2018. Essa atualização é apresentada brevemente no Apêndice B.

Em síntese, um total de 34 soluções foram recuperadas nesta nova iteração da execução do mapeamento, sendo que apenas três adotaram dispositivos móveis. Este cenário motiva a condução de maiores investigações no domínio de aplicações móveis, tal como foi identificado na primeira iteração do MS, para explorar melhor identificar seu potencial como meio de promover o ensino e aprendizagem.

3.4 Considerações Finais

A condução do MS sobre LPS no domínio educacional permitiu identificar a incipiência de LPS no domínio de ensino e aprendizagem de programação; algumas das técnicas e estratégias

que estão sendo adotadas para a concepção de LPS, bem como para o auxílio na criação de soluções por meio destas linhas. Apenas uma iniciativa no domínio de aplicações móveis educacionais foi identificada.

Já o MS para identificar software para o ensino e aprendizagem de programação permitiu a identificação de poucas iniciativas que adotam *m-learning*. Destaca-se, ainda, as informações quanto as tecnologias adotadas no desenvolvimento desses software e suas avaliações. A principal contribuição foi a possibilidade de identificar diversas estratégias de ensino que podem ser refletidas em funcionalidades a serem adaptadas para integrar aplicações móveis educacionais para o ensino e aprendizagem de programação.

Os software e suas funcionalidades, bem como as funcionalidades e estratégias identificadas nas LPS de domínio educacional, foram utilizadas para a concepção de um catálogo de requisitos *m-learning* no domínio de ensino e aprendizagem de programação, apresentado no próximo capítulo. O catálogo é necessário para o entendimento do domínio de programação e também das questões educacionais envolvidas. Além disso, é ele que fornecerá as especificações para o desenvolvimento das *features*, fundamentais para o estabelecimento da LPS proposta nesta pesquisa.

CATÁLOGO DE REQUISITOS *M-LEARNING* NO DOMÍNIO DE PROGRAMAÇÃO

4.1 Considerações Iniciais

Os requisitos de um software são as descrições do que este deve fazer, os serviços que serão oferecidos e as restrições em seu funcionamento. Tais requisitos refletem as necessidades dos clientes para uma solução que serve a uma finalidade determinada. Geralmente são classificados como requisitos funcionais, que concentram os serviços que o sistema deve fornecer; e não funcionais, que indicam as restrições aos serviços do sistema. Dá-se o nome de Engenharia de Requisitos, ao processo de identificar, analisar, documentar e verificar tais requisitos (SOMMERVILLE, 2010).

Nesta perspectiva, um catálogo de requisitos foi estabelecido seguindo as principais etapas da Engenharia de Requisitos. De modo similar, nota-se na literatura várias propostas de catálogos de requisitos e classificações no contexto de aplicações *m-learning*, como a proposta de Georgieva, Smrikarov e Georgiev (2005), que apresenta uma classificação geral para sistemas *m-learning*; o arcabouço de Parsons, Ryu e Cranshaw (2007), composto por um conjunto de requisitos para aplicações *m-learning* de domínios gerais; o catálogo de Duarte Filho e Barbosa (2013), que descreve requisitos genéricos para ambientes *m-learning*, e que foi atualizado por Soad *et al.* (2017a), em relação aos de requisitos de qualidade; e a proposta de Jalil, Beer e Crowther (2015), que identifica um grupo de requisitos pedagógicos para *m-learning*. No entanto, esses estudos discutem classificações específicas e um conjunto de requisitos não pertinentes ao domínio de ensino e aprendizagem de programação; em alguns casos, os requisitos são pouco detalhados, e poucos são os que discutem aspectos pedagógicos com a inclusão de embasamento em teorias educacionais existentes.

Neste sentido, o catálogo de requisitos proposto neste trabalho, além de servir como

principal artefato na proposição e desenvolvimento da arquitetura da LPS, busca apoiar o desenvolvimento de aplicações educacionais móveis para mitigar problemas no ensino e aprendizagem de programação em geral. Para isso, integra aos seus requisitos, teorias educacionais e indica categorias de problemas no domínio de programação que podem ser mitigados quando tal requisito é desenvolvido em um artefato de software. Este capítulo discute a proposta do catálogo (Seção 4.2), a metodologia e concepção do catálogo (Seção 4.3), o catálogo desenvolvido (Seção 4.4), e as avaliações conduzidas (Seção 4.5).

4.2 Proposta do Catálogo

O principal objetivo no desenvolvimento de um catálogo *m-learning* no domínio de programação é definir as *features* e o consequente escopo da LPS, resultando na proposta arquitetural da LPS investigada. No entanto, o catálogo proposto vai além das especificações dos requisitos, uma vez que busca integrar em cada um dos seus requisitos: (1) teorias educacionais que descrevem estratégias pedagógicas e didáticas que seu respectivo requisito apoiar; e (2) problemas de ensino e aprendizagem de programação que podem ser atacados com base no desenvolvimento de tal requisito. A integração dessas teorias e problemas busca apoiar, portanto, desenvolvedores, no que diz respeito ao entendimento dos conceitos educacionais envolvidos; e professores, no que se refere ao apoio na definição de quais estratégias educacionais e problemas de programação querem usar e mitigar, respectivamente.

Neste sentido, um conjunto de teorias educacionais foram investigadas e alinhadas com as funcionalidades descritas nos requisitos. Adicionalmente, por meio do MS executado para identificar software para o ensino de programação (MARCOLINO; BARBOSA, 2015a) (Capítulo 3), e o questionário conduzido com professores e alunos sobre problemas de programação enfrentados (Capítulo 2), foi possível relacionar tais funcionalidades com a classificação de problemas de programação proposta por Souza, Batista e Barbosa (2015), e complementada com uma sétima categoria, que engloba todas as seis, sendo denominada “Categoria P0 - Geral”, utilizada principalmente nos requisitos não funcionais.

A proposta da integração das teorias educacionais com os requisitos de software busca, ainda, auxiliar os projetistas de software no entendimento das teorias e elementos educacionais envolvidos, visto que o levantamento de requisitos com os interessados, que possuem maior conhecimento na área educacional do que na área de desenvolvimento de software, pode vir dificultar o processo de concepção de software.

Por fim, ao utilizar uma abordagem de reúso, os diferentes componentes e artefatos devem manter um nível satisfatório de coesão e fraco acoplamento, o que pode ser influenciado diretamente pelo conjunto de requisitos que serão desenvolvidos em tais artefatos de software (SOMMERVILLE, 2010). Esse nível de coesão e acoplamento, bem como a necessidade de refletir os requisitos em *features*, incentivam ainda o uso da estratégia proativa de concepção da

LPS, apresentada no Capítulo 2 e discutida no Capítulo 3, visto que o esforço necessário para se reutilizar um componente e adaptá-lo a determinados requisitos é, em muitos casos, mais custoso do que se desenvolver tal *feature* por completo (KRUEGER; CLEMENTS, 2018).

4.3 Metodologia e Concepção

Ao analisar os software para apoiar o processo de ensino e aprendizagem de programação (MARCOLINO; BARBOSA, 2015b; MARCOLINO; BARBOSA, 2017a), é possível notar algumas nuances: (i) há muitas *features* em comum, principalmente as relacionadas com interações *online* e atividades; e (ii) há diversas variabilidades entre as características observadas, especialmente as que apresentam uso de hardware específico nas modalidades que apoiam. Este conjunto de variabilidades e semelhanças, juntos com a necessidade de desenvolver aplicações *m-learning* para apoiar o ensino e aprendizagem de programação, ampliam a necessidade de investigar a possível adoção e adaptação de funcionalidades advindas de diferentes modalidades no contexto de *m-learning*, apoiando, assim, o domínio alvo. Com base em tal constatação, este catálogo também buscou apresentar um entendimento mais detalhado quanto à Engenharia de Domínio, de modo a criar um referencial que sirva de patamar para a investigação da aplicação de abordagens de reúso neste domínio. A metodologia e concepção do catálogo foi dividida em cinco fases, representadas na Figura 13 e descritas em seguida.

Figura 13 – Fases de Concepção do Catálogo de Requisitos.

Fonte: Elaborada pelo autor.

1. Identificação de estudos que descrevem funcionalidades e características de software educacional provenientes de diferentes modalidades, no domínio de ensino e aprendizagem de programação (MARCOLINO; BARBOSA, 2015b; MARCOLINO; BARBOSA, 2017a);
2. Extração, estabelecimento e agrupamento de todos os possíveis requisitos (funcionais e não funcionais);
3. Categorização dos requisitos em conjuntos específicos que expressem características em comum entre eles, bem como atribuição de teorias educacionais e categorias de problemas de programação que sugerem mitigar, para cada um dos requisitos, quando pertinente;

4. Validação e adaptação do catálogo considerando:

4.1 Requisitos baseados em catálogos *m-learning* preexistentes de propósito geral (GEORGIEVA; SMRIKAROV; GEORGIEV, 2005; PARSONS; RYU; CRANSHAW, 2007; DUARTE FILHO; BARBOSA, 2013; JALIL; BEER; CROWTHER, 2015), ISO/IEC 25010, que define o Padrão Internacional para Qualidade de Produtos de Software (ISO/IEC JTC/ 1/SC 7, 2011), e a ISO/IEC 15288, que define o Processo de Ciclo de Vida de Sistemas (ISO/IEC/IEEE, 2008);

4.2 Requisitos identificados por meio da comparação dos mesmos com aplicações *m-learning* existentes;

5. Primeira Versão do Catálogo de Requisitos M-learning;

6. Avaliações:

6.1 Avaliação dos requisitos essenciais no projeto de uma aplicação *m-learning* de propósito geral (estudo de caso e prova de conceitos); e

6.2 Avaliação qualitativa do catálogo por especialistas.

7. Versão final do catálogo de requisitos.

Após a definição das fases para o estabelecimento do catálogo, sua concepção foi conduzida. Na Fase 1, dos 81 software identificados no mapeamento sistemático (MARCOLINO; BARBOSA, 2015a), 33 foram considerados na extração dos requisitos, correspondendo aos estudos mais atuais (a partir de 2011). A lista completa dos estudos primários utilizados para extração está disponível em <https://andersonmarcolino.gitlab.io/app/static/media/Catalog_references.effd2d5e.pdf>.

Na Fase 2, utilizando esses 33 estudos, 498 requisitos foram extraídos e agrupados, resultando em 225 requisitos. Esta fase também incluiu o processo de análise dos requisitos provenientes de outras modalidades, identificando se poderiam ser adaptados para *m-learning*. A análise de outros catálogos *m-learning* e consideração de requisitos dos mesmos, bem como a consideração de especificidades tecnológicas encontradas, também foram consideradas.

De modo similar, na Fase 3, os 225 requisitos foram analisados e comparados em uma nova interação, resultando em 83 requisitos, classificados em 9 categorias: “Conteúdos de Aprendizagem”, “Atividades de Aprendizagem”, “Feedback e Resultados”, “Monitoramento e Desempenho de Aprendizagem”, “Mecanismos para Ensino de Programação”, “Avaliação de Aprendizagem”, “Apoio ao Ensino e Aprendizagem”, “Usuários”, e “Requisitos Não Funcionais”. Além de cada requisito receber um conjunto de palavras-chave e teorias educacionais relacionadas também foram associados os possíveis problemas de programação que possam vir a mitigar/sanar.

Para identificação dos problemas de programação, atribuídos para cada requisito, foram consideradas as indicações feitas nos 33 estudos primários utilizados para a concepção do

catálogo de requisitos. Os problemas relatados foram então dispostos em uma das seis categorias de problemas de programação propostas por Souza, Batista e Barbosa (2015) e apresentadas no Capítulo 1, complementadas por uma categoria geral, que indica a tentativa de mitigação abrangendo todas as categorias. As categorias indicadas em cada requisito sugerem que a funcionalidade ali descrita pode vir a auxiliar na mitigação ou solução daquele problema em específico.

O Quadro 8 apresenta as 33 referências disponíveis *online*¹, classificadas em uma ou mais categorias de problemas de ensino e aprendizagem de programação (SOUZA; BATISTA; BARBOSA, 2015), de acordo com o que cada estudo descreveu. Nota-se que a grande maioria é aplicável a mais de uma categoria de problema. Contudo, tal diversificação ainda deve considerar os problemas específicos dentro de cada categoria; o tipo de modalidade que apoiam (i.e., formal ou informal); suas especificidades educacionais, aderidas a uma das nove categorias utilizadas para concentrar os requisitos; e paradigmas de programação (i.e., orientados a objetos ou procedural).

Quadro 8 – Classificação das Referências por Categorias de Problemas em Programação.

ID	Categoria de Problemas	Estudos Primários (Referências)
P1	Aprendizagem de conceitos de programação.	2, 8, 4, 7, 9, 10, 11, 12, 13, 15, 17, 20, 21, 22, 23, 24, 26, 28, 30, 31, 33.
P2	Aplicação de conceitos de programação.	2, 8, 4, 7, 9, 10, 11, 12, 13, 14, 15, 17, 18, 20, 21, 22, 23, 24, 26, 28, 29, 30, 31, 33.
P3	Compreensão de programas.	2, 8, 4, 7, 9, 10, 11, 12, 13, 15, 17, 18, 20, 21, 22, 23, 24, 26, 28, 29, 30, 31, 33.
P4	Fatoração e refatoração de programas.	2, 8, 4, 7, 9, 10, 11, 12, 13, 15, 16, 17, 20, 21, 22, 23, 24, 26, 28, 29, 30, 31, 33.
P5	Motivação.	4, 10, 12, 14, 13, 17, 18, 21, 22, 25, 30, 31.
P6	Dificuldades relacionadas aos professores.	2, 4, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 19, 20, 21, 22, 25, 28.
Geral	Atende todas as categorias.	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 17, 18, 20, 22, 25, 29, 30.

Fonte: Elaborada pelo autor.

As Figuras 14 e 15 apresentam dois exemplos do quão abrangente são os estudos primários selecionados para definir o catálogo de requisitos.

Na Figura 14, o estudo primário de referência 12 (IVANOVI; XINO GALOS; KOM-LENOV, 2011), indica o apoio à todas as categorias de problemas no ensino e aprendizagem de programação. Esse estudo, em especial, apresenta a proposta de um curso para ensino de orientação a objetos e, deste modo, abrange diversos software que acabam por complementar uns aos outros, o que justifica a capacidade de atender as categorias de problemas. Contudo, ao observar o item modalidades, nota-se que, segundo o estudo, o apoio fornecido concentra-se na modalidade formal. No item paradigma de programação, atende à orientação a objetos. No que se refere às categorias de requisitos, não atende, ou não deixa explícito, o atendimento à categoria de controle de usuários. Finalmente, adota várias teorias educacionais, que poderiam

¹ <https://andersonmarcolino.gitlab.io/app/static/media/Catalog_references.effd2d5e.pdf>

ainda ser divididas, tal como foram durante a concepção do catálogo de requisitos, para apoiar decisões relacionadas às mesmas, quando pertinente ao professor.

Figura 14 – Composição para o Estudo de [Ivanovi, Xinogalos e Komlenov \(2011\)](#).

Problemas no Ensino e Aprendizagem de Programação	Modalidade	Categorias de Requisitos/Features	
P1 - Aprendizagem de conceitos de programação. ✓	Formal ✓	Conteúdos de aprendizagem ✓	Mecanismos para Ensino de Programação. ✓
P2 - Aplicação de conceitos de programação. ✓	Informal ✗	Atividades de aprendizagem ✓	Apoio ao Ensino e Aprendizagem. ✓
P3 - Compreensão de programas. ✓	Paradigma de Programação	Avaliação de aprendizagem ✓	Controle de Usuários. ✗
P4 - Fatoração e refatoração de programas. ✓		Feedback e resultados. ✓	Não funcionais. ✓
P5 - Motivação. ✓		Monitoramento e desempenho de aprendizagem. ✓	
P6 - Dificuldades relacionadas aos professores. ✓	Ambos ✗	Teorias Educacionais	
Geral - Atende todas as categorias. ✓		 ✓	

Fonte: Elaborada pelo autor.

Figura 15 – Composição para o Estudo de [Watson, Li e Lau \(2011\)](#).

Problemas no Ensino e Aprendizagem de Programação	Modalidade	Categorias de Requisitos/Features	
P1 - Aprendizagem de conceitos de programação. ✗	Formal ✓	Conteúdos de aprendizagem ✗	Mecanismos para Ensino de Programação. ✓
P2 - Aplicação de conceitos de programação. ✓	Informal ?	Atividades de aprendizagem ✗	Apoio ao Ensino e Aprendizagem. ✗
P3 - Compreensão de programas. ✗	Paradigma de Programação	Avaliação de aprendizagem ✓	Controle de Usuários. ✗
P4 - Fatoração e refatoração de programas. ✗		Feedback e resultados. ✓	Não funcionais. ✗
P5 - Motivação. ✓		Monitoramento e desempenho de aprendizagem. ✗	
P6 - Dificuldades relacionadas aos professores. ✓	Ambos ✗	Teorias Educacionais	
Geral - Atende todas as categorias. ✗		 ✓	

Fonte: Elaborada pelo autor.

Já na Figura 15, o estudo 14 ([WATSON; LI; LAU, 2011](#)) apoia, principalmente, as categorias de problemas de ensino e aprendizagem de programação relacionadas à aplicação de conceitos de programação, motivação dos alunos, e dificuldades relacionadas aos professores. Neste estudo, propõe-se mecanismos para fornecer *feedback* em tempo real, considerando uma técnica baseada em jogos e conceitos de visualização a compor um software para o ensino de programação. Apesar do estudo não deixar explícito a modalidade apoiada, o que justifica a interrogação na modalidade informal, uma avaliação foi conduzida com estudantes, o que indica apoio ao ensino formal. Adicionalmente, o estudo apoia o paradigma orientado a objetos;

considera as categorias de requisitos/*features* referentes a avaliação e aprendizagem, *feedback* e resultados, e mecanismos para o ensino de programação; e adota diferentes teorias, mas, em especial, as que propõem a aprendizagem baseada em jogos.

O grande número de combinações entre os diferentes itens apresentados nas Figuras 14 e 15 demonstra o potencial em se estabelecer um catálogo de requisitos e o desenvolvimento de um LPS, que possibilitará a investigação da combinação das diferentes *features* na criação de aplicações educacionais móveis no domínio de programação.

Na Fase 4, o catálogo foi validado por meio de comparação do mesmo com outros catálogos de propósito geral (GEORGIEVA; SMRIKAROV; GEORGIEV, 2005; PARSONS; RYU; CRANSHAW, 2007; KEARNEY *et al.*, 2012; DUARTE FILHO; BARBOSA, 2013; KLASSEN; EIBRINK-LUNZENAUER; GLOGGLER, 2013; JALIL; BEER; CROWTHER, 2015), com a ISO/IEC 25010 (ISO/IEC JTC 1/SC 7, 2011) para qualidade de produtos de software e a ISO/IEC 15288, que define o Processo de Ciclo de Vida de Sistemas (ISO/IEC/IEEE, 2008). Esta validação procurou complementar, incluir e alterar os requisitos definidos, a fim de melhor apresentá-los, atendendo os padrões estabelecidos e difundidos na área da Engenharia de Software. Ao final desta avaliação, 14 novos requisitos foram incluídos, principalmente requisitos não funcionais, totalizando 97 requisitos.

Ainda na Fase 4, uma segunda validação considerou sete aplicações *m-learning*, para condução de nova comparação e identificação de funcionalidades ainda não incluídas no catálogo. As sete aplicações consideradas foram: Moodle Mobile², Blackboard Mobile³, BrightSpace⁴, Mobile Sakai⁵, mLean⁶, Mixable⁷ e o Oppia Mobile⁸. Tais aplicações foram escolhidas por serem as com maior número de *downloads* nas principais lojas de aplicativos para dispositivos móveis existentes.

O Quadro 9 resume e compara as aplicações *m-learning* apresentadas com base no catálogo de requisitos proposto por Duarte Filho e Barbosa (2013) no contexto de 10 grupos de requisitos (funcionais e não funcionais), sendo o grupo identificado como “Outros”, um conjunto de indicativos relacionados às tecnologias, avaliações e uso das mesmas. A informações dispostas em tal quadro foram obtidas nos sites das empresas e dos projetos mantenedores.

Após a definição do Quadro 9, as funcionalidades identificadas foram comparadas às do catálogo proposto, no formato de *checklist*. Na inexistência de alguma delas, um requisito relacionado foi adaptado, recebendo a nova funcionalidade. Apenas três requisitos sofreram alteração.

² <https://docs.moodle.org/dev/Moodle_Mobile>

³ <<http://www.blackboard.com/platforms/mobile/overview.aspx>>

⁴ <<http://www.brightspace.com/>>

⁵ <<https://confluence.sakaiproject.org/display/MOBILE/Home>>

⁶ <<http://www.mobilelearning.io/>>

⁷ <<http://www.itap.purdue.edu/studio/mixable/>>

⁸ <<https://digital-campus.org/oppiamobile/overview/>>

Quadro 9 – Comparação das Funcionalidades das Aplicações *m-learning*.

		MOODLE Mobile	Blackboard Mobile	Brightspace	Mobile Sakai	mLearn	Mixable	Oppia Mobile
Comunicação	Resultados e Feedback	●	●	●	●	●	●	●
	Atividades Individuais e em Grupos	●	●	●	●	●	●	?
	Colaboração	●	●	●	●	●	●	?
	Sincronização	●	●	●	●	●	●	●
	Ambiente Colaborativo	●	●	●	●	●	●	?
	Gerenciamento de Avisos e Mensagens	●	●	●	●	●	●	●
Suporte	Customização	●	?	●	?	?	?	●
	Tolerância a Erros	?	x	?	?	●	?	?
	Atualização e Configuração	?	x	?	?	?	?	?
	Instalação	●	●	●	?	●	●	●
Portabilidade	Adaptabilidade	●	●	●	●	●	?	?
	Portável para dispositivos móveis	●	●	●	●	●	●	●
	Coexistência	●	●	●	?	●	●	●
Segurança	Privacidade	●	●	?	●	●	●	●
	Integridade	●	●	●	●	●	●	●
	Disponibilidade	?	x	?	?	?	?	?
	Confiabilidade	●	●	?	?	●	?	?
Nível de Serviço	Custo-benefício	●	?	?	?	?	?	●
	Acordo de Nível de Serviço	?	?	?	?	?	?	?
	Plano de Continuidade	?	?	?	?	?	?	?
Desempenho	Escalabilidade	?	?	●	?	?	?	?
	Utilização de Recursos	?	●	●	?	?	?	?
	Capacidade de Resposta	?	?	?	?	?	?	?
	Armazenamento de Memória	?	●	?	?	?	?	?
	Consumo de Energia	?	?	?	?	?	?	?
	Condições Físicas	?	?	●	?	?	?	?
Usabilidade	Inteligibilidade	●	●	●	●	●	●	●
	Atratividade/Motivação	●	?	●	?	?	●	?
	Apreensibilidade	?	?	?	?	?	?	?
	Layout e Organização	●	●	●	●	●	●	●
	Navegação e Orientação	●	●	●	●	●	●	●
	Acessibilidade	?	?	●	?	?	?	?
Funcionalidade	Ajuda	●	?	?	?	●	?	?
	Adequação	●	●	●	●	●	●	●
	Acurácia	?	?	?	?	?	?	?
	Interoperabilidade	●	●	●	●	●	●	●
	Acesso	●	●	●	●	●	●	●
	Reutilização	●	?	?	?	?	●	?
Educacionais e Pedagógicos	Recursos Multimídia	●	●	●	?	●	?	●
	Conhecimento Just-in-Time	●	●	●	●	●	●	●
	Separação do Ambiente	●	●	●	●	●	●	?
	Gerenciamento de Conteúdo	●	●	●	●	●	●	●
	Atividades Educacionais	●	●	●	●	●	?	●
	Aprendizagem baseada em contexto	?	?	?	?	?	?	?
	Adaptação ao contexto	?	?	?	?	?	?	?
Outros	Interatividade	●	●	●	●	●	●	●
	Open Source	●	x	x	?	x	x	●
	Multiplataforma	●	●	●	?	x	●	x
	Java	x	?	x	●	?	?	?
	Avaliado Ambientes Reais	●	●	●	?	?	?	●
	Avaliado Experimentalmente	?	?	?	?	?	?	?

Legenda: ? - Não identificado ● - Suporta x - Não suporta.

Fonte: Adaptada de Duarte Filho e Barbosa (2013).

Com base na condução das quatro primeiras fases, uma versão inicial do catálogo *m-learning* para aplicações móveis educacionais no domínio de programação, foi estabelecido, sendo então submetido a duas avaliações na Fase 6 para, na Fase 7, obter-se a versão final do

catálogo. Essas fases são discutidas nas próximas seções.

4.4 O Catálogo de Requisitos *M-learning* no Domínio de Programação

Cada um dos requisitos do catálogo é composto por: (1) código de identificação em sua respectiva categoria, (2) palavras-chave, (3) subcategoria, (4) descrição do requisito, (5) explicação do requisito, (6) modalidade educacional de origem, (7) considerações sobre adaptação para *m-learning*, quando necessário, (8) referência cruzada entre requisitos, (9) categorias de problemas de programação relacionados, (10) código das teorias educacionais relacionadas, quando possível, (11) referências bibliográficas utilizadas na concepção dos requisitos, e (12) tipo de variabilidade (LPS).

O total de requisitos para cada uma das 9 categorias são: 6 para “Conteúdos de Aprendizagem” (Quadro 10), 10 para “Atividades de Aprendizagem” (Quadro 11), 7 para “*Feedback* e Resultados” (Quadro 12), 4 para “Monitoramento e Desempenho de Aprendizagem” (Quadro 13), 11 para “Mecanismos para Ensino de Programação” (Quadro 14), 4 de “Avaliação de Aprendizagem” (Quadro 15), 14 para o “Apoio ao Ensino e Aprendizagem” (Quadro 16), 3 para “Usuários” (Quadro 17), e 38 na categoria de “Requisitos Não Funcionais” (Quadro 18).

Quadro 10 – Requisitos de Conteúdo de Aprendizagem.

Categoria	Conteúdos de Aprendizagem	Qtd. Requisitos	6
Resumo	Requisitos relacionados aos Conteúdos de Aprendizagem, cobrem requisitos que provêm a adaptação de conteúdos para plataforma móvel. Especialmente conteúdos de programação e conteúdos que permitam interações mais engajantes por meio dos diferentes sensores dos dispositivos móveis (e.g. acelerômetro, tela sensível ao toque).		
Requisitos			
ID	Descrição	Cat. Problemas	
RFCT001	A aplicação deve permitir o gerenciamento de conteúdos pelos instrutores.	P1, P5, P6.	
RFCT002	A aplicação deve disponibilizar o conteúdo atendendo as especificidades e estratégias educacionais.	P1, P5, P6.	
RFCT003	A aplicação deve disponibilizar o conteúdo por meio de materiais didáticos que explorem diferentes estratégias de apresentação e interação.	P1, P2, P3, P4.	
RFCT004	A aplicação deve permitir a criação de conteúdo para compartilhamento entre os Usuários.	P1, P2, P3, P4.	
RFCT005	A aplicação deve apresentar conteúdo de programação com integração de teste de software.	P1, P2, P3, P4.	
RFCT006	A aplicação deve fornecer mecanismos para a retomada dos conteúdos que estão sendo estudados pelos aprendizes, quando o processo de aprendizagem é pausado/interrompido.	P1, P2, P3, P4.	

Fonte: Elaborada pelo autor.

Em relação aos problemas de programação tratados nos 97 requisitos, 34 estão relacionados com o problema de aprendizagem de programação (Categoria P1) e objetivam fornecer uma melhor explicação de estruturas complexas e conceitos de programação. A categoria que mais abrange requisitos que atacam tais problemas é categoria de “Conteúdos de Aprendizagem”. 35 requisitos tratam de problemas relacionados à aplicação de conceitos de programação (Categoria P2), com ênfase na categoria de “Mecanismos para o Ensino e Aprendizagem de Programação”, a qual concentra diferentes recursos, funcionalidades e ferramentas para permitir a aplicação

Quadro 11 – Requisitos de Atividades de Aprendizagem.

Categoria	Atividades de Aprendizagem	Qtd. Requisitos	10
Resumo	Requisitos de Atividades de Aprendizagem concentram os requisitos relacionados às diferentes atividades realizadas pelos aprendizes, podendo servir como meios de avaliar o desempenho desses e a retenção de conhecimento nos tópicos ensinados. No contexto de programação, atividades com diferentes tipos de interação, principalmente as que usam especificidades dos dispositivos móveis são encorajadas.		
Requisitos			
ID	Descrição	Cat. Problemas	
RFAC001	A aplicação deve fornecer diferentes tipos de atividades educacionais, on-line ou off-line, para avaliar e desenvolver a aprendizagem do aprendiz.	P2, P3, P4.	
RFAC002	A aplicação deve permitir ao instrutor gerenciar diferentes atividades e tarefas.	P6.	
RFAC003	A aplicação deve permitir o envio de respostas para as atividades e tarefas dos alunos.	P1, P2, P3, P4.	
RFAC004	A aplicação deve utilizar diferentes estratégias didático-pedagógicas nas atividades educacionais.	P1, P2, P3, P4.	
RFAC005	A aplicação deve manter um repositório de atividades educacionais.	P6.	
RFAC006	A aplicação deve solicitar usuário e senha, ou algum método de identificação aos seus usuários, no processo de envio de tarefas	P6.	
RFAC007	A aplicação deve solicitar usuário e senha, ou algum método de identificação aos seus usuários, no processo de envio de tarefas	P1, P2, P3, P4.	
RFAC008	A aplicação deve permitir o download das atividades educacionais.	P1, P2, P3, P4.	
RFAC009	A aplicação deve permitir a realização de atividades educacionais com interações multiusuários.	P1, P2, P3, P4.	
RFAC010	A aplicação deve apoiar as atividades de programação com uso de teste de software e suas abordagens.	P1, P2, P3, P4.	

Fonte: Elaborada pelo autor.

Quadro 12 – Requisitos de *Feedback* e Resultados.

Categoria	Feedback e Resultados	Qtd. Requisitos	7
Resumo	Requisitos de Feedback e resultados são utilizadas para fornecer retorno nas diferentes interações com elementos educacionais e com a aplicação, incluindo os resultados das atividades e o desempenho dos aprendizes. No contexto de programação, o feedback é essencial, uma vez que, por meio do retorno de suas atividades, os aprendizes podem corrigir seus erros e aprender por meio deles.		
Requisitos			
ID	Descrição	Cat. Problemas	
RFFR001	A aplicação deve fornecer resposta aos estímulos/interações dos Usuários com sua interface Gráfica ou com suas diferentes funcionalidades.	P0.	
RFFR002	A aplicação deve fornecer feedback aos estímulos de modo completo, conciso e de fácil entendimento.	P0.	
RFFR003	A aplicação deve fornecer feedback a interação dos aprendizes com conteúdos, atividades educacionais,correções e avaliações educacionais.	P0.	
RFFR004	A aplicação deve fornecer feedback no processo de envio e recebimento de respostas de atividades educacionais.	P0.	
RFFR005	A aplicação deve permitir o gerenciamento de feedback aos estímulos dos aprendizes.	P1, P5, P6.	
RFFR006	A aplicação deve fornecer feedback nos diferentes mecanismos de apoio a programação.	P1, P2, P3, P4.	
RFFR007	A aplicação deve fornecer feedback por meio da execução de casos de teste.	P1, P2, P5.	

Fonte: Elaborada pelo autor.

dos conceitos aprendidos, e 35 requisitos estão relacionados com problemas que se referem ao entendimento de conceitos de programação (Categoria P3). Estes são responsáveis por facilitar o processo de entendimento dos tópicos de programação por meio de atividades, logo estão concentrados no grupo de requisitos de “Atividades de Aprendizagem”. 28 requisitos lidam com problemas relacionados a fatoraçoão e refatoraçoão de programas (Categoria P4). Como tais problemas são originários, principalmente, da aplicação de conceitos de programação, duas categorias de requisitos abrangem possíveis soluções, sendo elas “Mecanismos para o Ensino e Aprendizagem de Programação” e “Conteúdos de Aprendizagem”.

Quadro 13 – Requisitos de Monitoramento e Desempenho de Aprendizagem.

Categoria	Monitoramento e Desempenho de Aprendizagem	Qtd. Requisitos	4
Resumo	Os requisitos de Monitoramento e Desempenho de Aprendizagem estão relacionados com o monitoramento das interações dos aprendizes na aplicação, incluindo suas atividades de aprendizagem e o desempenho nessas. No contexto de programação, tempo e número de erros durante a compilação e/ou execução, são exemplos de registros monitorados.		
Requisitos			
ID	Descrição	Cat. Problemas	
RFPM001	A aplicação deve monitorar e registrar o progresso nos conteúdos e desempenho nas atividades educacionais dos aprendizes.	P1, P2, P3, P4.	
RFPM002	A aplicação deve permitir a emissão de relatórios dos registros de progresso e desempenho educacional dos aprendizes.	P6.	
RFPM003	A aplicação deve ser capaz de interromper o registro de tempo dos aprendizes em atividades ou lições quando estas forem interrompidas após um tempo configurável.	P0.	
RFPM004	A aplicação deve permitir que, durante o monitoramento em tempo real das interações educacionais dos aprendizes, os instrutores visualizem e interajam com os aprendizes, também em tempo real.	P1, P2, P3, P4.	

Fonte: Elaborada pelo autor.

Quadro 14 – Requisitos de Mecanismos para o Ensino e Aprendizagem de Programação.

Categoria	Mecanismos para o Ensino e Aprendizagem de Programação	Qtd. Requisitos	11
Resumo	Requisitos de Mecanismos para o Ensino e Aprendizagem de Programação estão relacionados com o processo de ensino e aprendizagem deste domínio, principalmente sobre diferentes elementos educacionais desenvolvidos especificamente para esse fim, como atividades e conteúdos de aprendizagem que os integre, ferramentas para apoio ao desenvolvimento e execução de código, identificação de erros em código, apoio a linguagens de programação. Na maioria dos casos esses requisitos não são independentes. Neste sentido, esta categoria agrega os requisitos que tratam especificamente de questões de tal domínio.		
Requisitos			
ID	Descrição	Cat. Problemas	
RFPG001	A aplicação deve possuir um compilador para compilação de código fonte.	P1, P2, P3, P4.	
RFPG002	A aplicação deve possuir um editor de código.	P2, P3, P4.	
RFPG003	A aplicação deve possuir um depurador de código.	P2, P3, P4.	
RFPG004	A aplicação deve utilizar elementos visuais e gráficos para ensinar programação.	P1, P2, P3, P4.	
RFPG005	A aplicação deve possuir mecanismos para identificar e registrar os códigos de programação dos aprendizes.	P2, P3, P6.	
RFPG006	A aplicação deve possuir interface Gráfica de usuário configurável para os diferentes tipos de usuário.	P5, P6.	
RFPG007	A aplicação deve permitir a importação de bibliotecas ou excertos de códigos desenvolvidos previamente nas atividades de codificação/programação dos aprendizes.	P1, P2, P3.	
RFPG008	A aplicação deve utilizar gamificação em diferentes atividades de programação.	P1, P2, P3, P4.	
RFPG009	A aplicação deve fornecer um ambiente de aprendizagem prático e interativo.	P1, P2, P3, P4.	
RFPG010	A aplicação deve fazer uso dos diferentes sensores móveis e especificidades de hardware dos dispositivos móveis nas atividades de programação.	P1, P5, P6.	
RFPG011	A aplicação deve manter um repositório de casos de teste.	P2, P3, P6.	

Fonte: Elaborada pelo autor.

No que tange os problemas relacionados à motivação (Categoria P5), 32 requisitos visam atacar tal problema, principalmente por meio da exploração de características provenientes das plataformas móveis, como uso de sensores, câmera e outros recursos e hardware. Tais requisitos estão inseridos, principalmente, na categoria de “Atividades de Aprendizagem”. 39 requisitos, tratam problemas relacionados à dificuldade dos professores (Categoria P6), sendo a principal categoria que lida com tais problemas é a que engloba requisitos para o “Apoio ao Ensino e Aprendizagem”.

Finalmente, como indicado anteriormente, uma categoria extra foi incluída no catálogo, P0 - **Geral**, para classificar requisitos de cunho geral, que acabam, indiretamente, auxiliando

Quadro 15 – Requisitos de Avaliação de Aprendizagem.

Categoria	Avaliação de Aprendizagem	Qtd. Requisitos	4
Resumo	Requisitos de Avaliação de Aprendizagem estão relacionados às diferentes avaliações que podem ser conduzidas na aplicação para identificar o desempenho dos alunos, podendo estas serem de diferentes tipos. Elas incluem ainda a avaliação e correção de exercícios, descrevem, ainda, mecanismos para correção automática de exercícios. No contexto de programação, eles provêm meios de analisar e identificar limitações considerando as respostas dos aprendizes, tanto para eles quanto para seus professores.		
Requisitos			
ID	Descrição	Cat. Problemas	
RFAT001	A aplicação deve permitir o gerenciamento de avaliações de aprendizagem.	P1, P2, P3, P6.	
RFAT002	A aplicação deve possuir mecanismos para avaliação/correção automática de atividades de programação.	P1, P2, P3, P4.	
RFAT003	A aplicação deve ser capaz de interromper o registro de tempo dos aprendizes em atividades ou lições quando estas forem interrompidas após um tempo configurável.	P0.	
RFAT004	A aplicação deve fornecer meios para que os instrutores corrijam as avaliações e atividades dos aprendizes manualmente.	P5, P6.	

Fonte: Elaborada pelo autor.

Quadro 16 – Requisitos de Apoio ao Ensino e Aprendizagem.

Categoria	Apoio ao Ensino e Aprendizagem	Qtd. Requisitos	14
Resumo	Requisitos de Apoio ao Ensino e Aprendizagem tratam especificamente de questões relacionadas ao contexto educacional, e.g., administração e formação de turmas, interações entre aprendizes e professores, bem como elementos educacionais que permitam melhor condução dos processos de ensino e aprendizagem, e.g., fóruns. No contexto de programação, encoraja-se o uso de compartilhamento de respostas de exercícios e trechos de código, criação de conteúdos para colegas, interações para sanar dúvidas, entre outros.		
Requisitos			
ID	Descrição	Cat. Problemas	
RFTL001	A aplicação deve permitir o gerenciamento de cursos.	P0.	
RFTL002	A aplicação deve permitir o gerenciamento de turmas.	P6.	
RFTL003	A aplicação deve permitir a criação de grupos de trabalho.	P1, P2, P3, P4.	
RFTL004	A aplicação deve possuir diferentes mecanismos de comunicação/discussão entre os diferentes tipos de Usuários.	P1, P2, P5, P6.	
RFTL005	A aplicação deve possuir mecanismos para apoiar a aprendizagem colaborativa.	P1, P2, P3, P4.	
RFTL006	A aplicação deve possuir mecanismos para apoiar a aprendizagem cooperativa.	P1, P2, P3, P4.	
RFTL007	A aplicação deve utilizar diferentes elementos de interação social baseado em redes sociais.	P1, P5, P6.	
RFTL008	A aplicação deve apoiar a aprendizagem por meio de um tutor inteligente.	P5, P6.	
RFTL009	A aplicação deve manter um organizador/agenda de atividades para os Usuários.	P0.	
RFTL010	A aplicação deve apoiar o uso de materiais didáticos desenvolvidos seguindo padrões de projeto (objetos educacionais) mais conhecidos.	P0.	
RFTL011	A aplicação deve possuir mecanismos de competição (gamificação) para engajar e motivar os aprendizes.	P5.	
RFTL012	A aplicação deve permitir o compartilhamento de projetos e códigos desenvolvidos pelos aprendizes.	P1, P2, P3, P4.	
RFTL013	A aplicação deve permitir o desenvolvimento de habilidades de programação dos aprendizes.	P1, P2, P3, P4.	
RFTL014	A aplicação deve apoiar diferentes modalidades de ensino e suas estratégias didático-pedagógicas.	P0.	

Fonte: Elaborada pelo autor.

em todos os problemas de programação. Por exemplo, o requisito **RFTL001** *A aplicação deve prover gerenciamento de cursos*, extraído da categoria “Apoio ao Ensino e Aprendizagem”, que apesar de não estar relacionado com um problema de programação em si, fornece facilidades para apresentar o desempenho de alunos em determinado curso, permitindo a criação de ranques de competição, quanto adotados elementos lúdicos, entre outros. Em outras palavras, ele apoia a solução de todos os outros problemas, quando integrados com outros requisitos.

É importante mencionar que um requisito pode receber uma ou mais categorias de

Quadro 17 – Requisitos de Usuários.

Categoria	Usuários	Qtd. Requisitos	3
Resumo	Requisitos de Controle de Usuários ou simplesmente Usuários referem-se aos requisitos que concentram o processo de identificação e acesso as aplicações, bem como controle de submissão de itens educacionais por meio da aplicação, apoiando o registro, monitoramento e feedback.		
Requisitos			
ID	Descrição	Cat. Problemas	
RFUS001	A aplicação deve permitir o gerenciamento de usuários.	P0.	
RFUS002	A aplicação deve permitir autenticação dos usuários na aplicação.	P0.	
RFUS003	A aplicação deve permitir o acesso a usuários não autenticados.	P2, P5, P6.	

Fonte: Elaborada pelo autor.

Quadro 18 – Requisitos Não Funcionais.

Categoria	Requisitos Não Funcionais	Qtd. Requisitos	38
Resumo	O conjunto de requisitos não funcionais trata, principalmente, requisitos de qualidade relacionadas ao desenvolvimento das aplicações e seu funcionamento de modo coerente e, como esperado. Consequentemente, eles são transversais a todos as demais categorias de requisitos.		

Fonte: Elaborada pelo autor.

problemas no domínio de ensino de programação. Isso justifica o número total superior a 97, expresso em cada categorias de problemas abrangidos.

Em uma perspectiva educacional, como tal domínio envolve diferentes objetivos educacionais que devem ser alcançados por meio dos processos de ensino e aprendizagem, e parte fundamental destes conceitos se baseiam em teorias educacionais que estão, por sua vez, em constante evolução (HARASIM, 2012), essas teorias se tornam importantes. Contudo, como apresentado nos trabalhos relacionados, não foi possível identificar estudos que relacionam os requisitos educacionais com teorias, fornecendo um apoio mais assertivo e facilitado no que tange ao desenvolvimento de soluções educacionais. Essa lacuna motivou o relacionamento dos requisitos deste catálogo com possíveis teorias educacionais.

As teorias e conceitos abrangem discussões profundas e muitas vezes complexas sobre o processo de ensino e aprendizagem. Consequentemente, considerando a complexidade das questões tecnológicas envolvidas no processo de desenvolvimento, a complexidade acaba aumentando. Outra questão é a dificuldade que educadores e professores têm para expressar suas necessidades (JALIL; BEER; CROWTHER, 2015). Com base nessas constatações, cada requisito recebe uma ou mais teorias educacionais. A intenção é vincular pequenas explicações de teorias educacionais, relacionadas diretamente com o domínio de programação para facilitar a compreensão dos objetivos educacionais que se almejam.

Para demonstrar isso, o requisito **RFTL005** *A aplicação deve fornecer mecanismos para apoiar aprendizagem colaborativa*, por exemplo, está relacionado com o problema de aplicação de conceitos de programação e com as teorias de aprendizagem cognitiva e aprendizagem colaborativa. Essas teorias discutem sobre as necessidades de atividades colaborativas, que estão relacionadas diretamente com o objetivo que enfatiza a necessidade de interação entre os

aprendizes, permitindo um processo de aprendizagem melhor e mais profundo. Mesmo com essas necessidades sendo completamente visíveis nos requisitos, as teorias relacionadas com o requisito podem levar à seleção de mais recursos, que possam vir a criar soluções mais adequadas e, em alguns casos, mais configuráveis, criando maneiras de aumentar a motivação e engajar os aprendizes, além de melhorar o processo de aprendizagem como um todo. A lista completa dos 67 estudos primários utilizados para extração das teorias educacionais está disponível em https://andersonmarcolino.gitlab.io/app/static/media/Theories_references.996a5ee7.pdf.

Além dos problemas de programação e teorias educacionais que sustentam questões pedagógicas e didáticas, o catálogo inclui explicações e comentários adicionais sobre a necessidade na adaptação de recursos provenientes de outras modalidades de ensino, como soluções provenientes do *e-learning*. Isso se deve pelo fato de que os software educacionais mapeados, foram identificados no contexto de diferentes modalidades (Capítulo 3). Entre as principais preocupações sobre a necessidade de adaptação dos requisitos para o nosso catálogo de *m-learning*, pode-se citar: (i) redução dos elementos gráficos nas telas para dispositivos móveis; (ii) implementações de mecanismos de interação com o conteúdo e atividades, mesmo sem conexão com a Internet; (iii) implementações e suporte para diferentes dispositivos de entrada; (iv) implementação de conteúdos e atividades mais pessoais e configuráveis; e (v) preocupações com a limitação de recursos de hardware, como a vida útil da bateria e consumo de memória RAM (do inglês, *Random Access Memory*), no caso de dispositivos mais antigos.

4.5 Avaliações do Catálogo de Requisitos

Avaliações apoiam a coleta de evidências e podem auxiliar na melhoria do objeto de estudo (BASILI; SELBY; HUTCHENS, 1986). Nesse sentido, duas avaliações referentes ao catálogo de requisitos foram conduzidas, uma por meio de um estudo de caso e outra por um questionário aplicado com professores.

4.5.1 Estudo de Caso

A primeira avaliação foi conduzida por meio de uma prova de conceitos. Esta avaliação integrou, além do método de prova de conceitos, elementos do estudo de caso de Jalil, Beer e Crowther (2015) e avaliou os requisitos em um contexto de aplicações educacionais de propósito geral, ou seja, independente de domínio. Para condução da mesma foram realizadas reuniões online via vídeo-conferência, no formato de *kick-off* e *dayling meeting* extraídos da metodologia ágil Scrum (STRAY; MOE; SJOBERG, 2018). Em tais reuniões foi definido um projeto de aplicação para apoiar à aprendizagem na modalidade presencial de um curso de Ciência da Computação, na disciplina de Engenharia de Software.

A aplicação visa permitir aos alunos, depois de terem as lições sobre conteúdos em sala, receberem tarefas para criação e submissão dos tópicos referentes aos conteúdos que aprenderam.

Em seguida o professor da disciplina, ao receber tais tópicos, pode revisar e os enviar novamente aos seus alunos. Tal ciclo deve ser realizado até que o professor aceite totalmente o tópico. Após tal aceite os alunos compartilham entre si os tópicos que criaram, de modo a permitir que todos possam estudar e revisar o que foi visto em sala.

Com base nesta definição, o estudo de caso seguiu para uma análise com desenvolvedores, das solicitações do *stakeholder* e a transcrição das mesmas em funcionalidades. O time era composto por três desenvolvedores de diferentes níveis de experiência com conhecimentos variados em termos de tecnologias, mas todos com domínio em desenvolvimento de aplicações móveis. Para o conjunto de requisitos especificados pelo professor, teorias educacionais relacionadas foram identificadas e lidas para o time. Em seguida, foram identificadas a relação das mesmas com o objetivo da aplicação. Definiu-se que o objetivo final é a criação de conteúdos (tópicos), permitindo que os estudantes se tornem autores de conteúdos, aprendendo por meio da criação dos mesmos, além de compartilhar seus conhecimentos com demais colegas. Neste contexto, as teorias destacadas foram o construtivismo, aprendizagem ativa e teoria *e-learning* (KONG; KWOK; FANG, 2012; HAYTHORNTWHAITE; ANDREWS, 2011). O uso das teorias também foi utilizado para alinhar o objetivo educacional com o objetivo final da aplicação.

Em um terceiro momento, uma lista de componentes e funcionalidades foi criada. Adicionalmente, alguns desenhos da interface ou *mockups* foram criadas para verificar a aceitação do produto pelas partes envolvidas. A Figura 16 apresenta duas telas da aplicação. Na primeira é apresentado o formulário para cadastro do conteúdo e tópicos; na segunda, o formulário apresenta dois campos na coloração vermelha, sendo estes os campos que necessitam de revisão. Ao interagir com tais campos por meio de toque, uma mensagem de *feedback* é apresentada, indicando as revisões que o professor solicitou.

Figura 16 – Excertos de dois Mockups da Aplicação Móvel Avaliada.

The figure shows two mobile application mockups side-by-side. The left mockup is titled 'Content Name' and contains the following elements: a 'Topic Title' text field, an 'Abstract' text field, a 'Text Field' (highlighted with a blue dashed border), a 'Media Field' (highlighted with a red dashed border), an 'Image Field (URL)' text field, an 'Upload' button, and a 'Cancel' button. At the bottom, there are links for 'Add Field', 'Save', 'Submit', 'Help', and 'Cancel'. The right mockup is titled 'Topic Review - Student' and contains the following elements: a 'Content Name' text field, a 'Topic Title' text field (highlighted with a green dashed border), an 'Abstract' text field (highlighted with a green dashed border), a 'Feedback' message box (highlighted with a blue dashed border) containing the text 'Message from Professor indicating what is need to be revised in the touched field.', an 'Image Field (URL)' text field (highlighted with a red dashed border), and buttons for 'Add Field', 'Save', 'Submit', 'Help', and 'Cancel'.

Fonte: Elaborada pelo autor.

Finalmente, após a avaliação das telas pelo time de desenvolvedores e também pelo professor, a lista de componentes e funcionalidades foi complementada com todos os requisitos que refletiam tais funcionalidades no catálogo proposto. O Quadro 19 apresenta, na coluna “Perfil do Usuário” os tipos de usuários que interagirão com os respectivos componente e funcionalidades apresentadas nas colunas subsequentes. A coluna “Requisitos do Catálogo”, por sua vez, apresenta os requisitos do catálogo avaliado que referenciam as funcionalidades desejadas.

Quadro 19 – Requisitos da Prova de Conceito.

Perfil do Usuário	Componente	Funcionalidades	Requisitos do Catálogo
E/P	Login	Estudante ou professor já registrados podem logar na aplicação com suas credenciais.	RFFR001, RNF006
E/P	Registro de Usuário	Usuários sem registro podem se registrar e logar na aplicação.	RFUS001, RFUS002, RFTL002
E/P	Gerenciamento de Usuário	Usuários podem gerenciar seus perfis (criar, consultar/visualizar e atualizar).	RFUS001
E/P	Ajuda	Usuários podem acessar e ler documentos/ajuda da aplicação para apoiá-los no uso da mesma.	RFN003
E/P	Logoff	Usuários podem finalizar suas sessões de acesso na aplicação.	RFUS002
P	Gerenciamento de Conteúdo	Professores podem gerenciar conteúdos na aplicação (criar, consultar/visualizar, atualizar e excluir).	RFCT001
E/P	Gerenciamento de Tópicos	Estudantes e Professores podem gerenciar os tópicos na aplicação (criar, consultar/visualizar, atualizar e excluir).	RFCT003, RFAC002, RFAC004
E	Submeter um Tópico	Estudantes podem criar e submeter um tópico para revisão.	RFFR004, RFAC003, RFAC006
P	Revisão de Tópico	Professores podem revisar os tópicos submetidos pelos seus estudantes.	RFAT004
E/P	Compartilhamento de Tópico	Estudantes, depois de suas revisões e correções podem compartilhar os seus tópicos criados.	RFCT002, RFCT004, RFTL012
E/P	Visualizar Conteúdo	Estudantes podem visualizar conteúdos compartilhados.	RFCT002
E/P	Requisitos Não funcionais.	Requisitos não funcionais especificados com time de desenvolvedores.	RFN001, RFN004, RFN005, RFN007, RFN009, RFN012, RFN019, RFN020, RFN021, RFN024, RFN026, RFN027, RFN032, RFN033, RFN034, RFN035, RFN036, RFN037.

Legenda: E - Estudante, P - Professor.

Fonte: Elaborada pelo autor.

De posse de tal quadro, os desenvolvedores foram consultados para identificar se as funcionalidades descritas referiam-se aos requisitos indicados. Estes confirmaram que os requisitos atendiam precisamente as funcionalidades a serem desenvolvidas e que as teorias deram apoio significativo no entendimento da aplicação, no contexto educacional. Uma consideração final foi dada a respeito da descrição dos requisitos. Os desenvolvedores mencionaram que um requisito abrange um grupo de funcionalidades que poderiam ser transcritos em novos requisitos de menor granularidade. Esse fato já era previsto, uma vez que o catálogo passou, durante seu processo de concepção, de 498 requisitos iniciais para o conjunto final de 97, resultando na integração de tais requisitos e, assim, em uma descrição mais abrangente que apresenta um conjunto mais amplo de funcionalidades a serem desenvolvidas. Logo, não se trata de um problema, mas uma decisão de projeto para estabelecer um grupo menor de requisitos que abranja um grupo maior de funcionalidades relacionadas.

4.5.2 Questionário com Professores

A segunda avaliação foi realizada por meio de um questionário disponibilização de modo online com professores especialistas, tendo este sido validado em seção piloto com aluno de

doutorado da Universidade de São Paulo (ICMC–USP). O objetivo de tal avaliação é analisar as categorias de requisitos e seus requisitos funcionais, em sua totalidade, visto que o estudo de caso buscou avaliar a aplicação prática do catálogo.

Primeiramente, um questionário de caracterização foi realizado para identificar formação, experiência e região em que atuam/residem. Em um segundo momento foi conduzida a avaliação do catálogo. Esta avaliação foi composta por duas questões que avaliaram de modo individual cada requisito e, de modo geral, cada uma das oito categorias de requisitos.

Na primeira questão, os participantes eram convidados a classificar, em uma escala de 1 a 3, sendo 1 “Prioridade baixa”, 2 “Prioridade mediana” e 3 “Prioridade alta”, qual é o grau de prioridade dos requisitos de cada categoria correspondente. Ou seja, ao se considerar um produto inicial, na perspectiva dos participantes, quais dos requisitos e suas respectivas categorias, devem ser priorizados.

A necessidade de identificar requisitos prioritários foi incluída para determinar um conjunto inicial de requisitos a serem desenvolvidos na LPS.

Na segunda questão, os participantes eram convidados a responder se “*O conjunto de requisitos de <categoria> atende as necessidades para a concepção de aplicações móveis educacionais?*”. Para isso, deveriam indicar, em uma escala de 1 a 5: 1 “Discorda totalmente”; 2 “Discorda parcialmente”; 3 “Não concorda nem discorda”; 4 “Concorda parcialmente”; e 5 “Concorda totalmente”, qual o seu nível de satisfação em relação aos requisitos apresentados. Adicionalmente, um espaço para sugestões e comentários foi deixado para cada categoria avaliada. Para essa análise, os participantes tinham acesso ao código de identificação dos mesmos e a descrição resumida, além de *link* de acesso ao catálogo completo.

A avaliação qualitativa contou com 28 participantes. Estes foram convidados por meio de listas de e-mail e redes sociais. O único requisito é que tivessem ministrado aulas de programação no decorrer de suas experiências como docente e/ou estágio.

Dos 28 participantes, 19 são professores em universidades, 2 em faculdades e 7 em institutos técnicos. No total, 4 possuem experiência de até 2 anos; 3, de 2 a 4 anos; 4, de 4 a 6 anos; 13, mais que 6 anos; 1 com 15 anos; 1 com 17 anos e 1 com mais de 30 anos de experiência. Quanto a distribuição geográfica dos mesmos, 12 são do Paraná, 5 de São Paulo, 5 de Minas Gerais, 1 do Mato Grosso do Sul, 1 do Macapá, 1 do Pará, 1 do Distrito Federal, 1 de Santa Catarina e 1 do Rio Grande do Sul.

A Tabela 6 apresenta o percentual de respostas para a questão 1 (coluna “Prioridade (%)”) e para a questão 2 (coluna “Atende as Necessidades? (%)”). A coluna “Resumo dos Comentários”, resume todos os comentários coletados na sua respectiva categoria.

Tabela 6 – Síntese das Respostas da Avaliação Qualitativa.

Categoria	Requisito	Prioridade (%) (1 - Baixa, 2 - Mediana e 3 - Alta)				Atende as Necessidades? (%) (1. Discorda totalmente; 2. Discorda parcialmente; 3. Não concorda nem discorda; 4. Concorda parcialmente; e 5. Concorda totalmente)						Resumo dos Comentários
		1	2	3	Total	1	2	3	4	5	Total	
Atividades de Aprendizagem	RFAC001	3,57	25,00	71,43	100,00	3,57	0,00	3,57	57,14	35,71	100,00	"A aplicação deve permitir uma categorização de atividades". "O requisito RFAC006 é transversal a quase todos os requisitos" e "possibilitar a criação de atividades adaptativas ao perfil e interesse do estudante e não somente do professor".
	RFAC002	3,57	14,29	82,14	100,00							
	RFAC003	0,00	17,86	82,14	100,00							
	RFAC004	0,00	21,43	78,57	100,00							
	RFAC005	0,00	50,00	50,00	100,00							
	RFAC006	7,14	21,43	71,43	100,00							
	RFAC007	17,86	25,00	57,14	100,00							
	RFAC008	21,43	57,14	21,43	100,00							
	RFAC009	17,86	35,71	46,43	100,00							
	RFAC010	17,86	35,71	46,43	100,00							
Conteúdo de Aprendizagem	RFCT001	3,57	14,29	82,14	100,00	3,57	0,00	3,57	53,57	39,29	100,00	"Deve permitir diferentes tipos de conteúdos. O conteúdo deve ser disponibilizado conforme estratégia do professor ou necessidade do aluno. Integrar funcionalidades considerando redes sociais. Teste de software é importante, mas não é a única: agilidade e software, ferramentas, etc."
	RFCT002	3,57	17,86	78,57	100,00							
	RFCT003	3,57	25,00	71,43	100,00							
	RFCT004	17,86	35,71	46,43	100,00							
	RFCT005	21,43	39,29	39,29	100,00							
	RFCT006	0,00	21,43	78,57	100,00							
Avaliação de Aprendizagem	RFAT001	0,00	14,29	85,71	100,00	3,57	3,57	10,71	42,86	39,29	100,00	"Feedback automático e também manual.", "Sugestão de incluir avaliação de pares." e "Acredito que se o aluno sentir que tem o direito de não conseguir completar uma tarefa corretamente e que isso faz parte do seu processo de aprendizagem, a tendência natural é que as "trapaças" (RFAC007) sejam minimizadas."
	RFAT002	3,57	14,29	82,14	100,00							
	RFAT003	21,43	32,14	46,43	100,00							
	RFAT004	14,29	35,71	50,00	100,00							
Feedback e Resultados	RFFR001	3,57	10,71	85,71	100,00	3,57	3,57	7,14	46,43	39,29	100,00	"Apresentar casos de teste ao aluno, mas internamente na aplicação, testar o software com outro conjunto de caso de teste."; "R1 e R3 são parecidos, o segundo parece ser uma explicação do primeiro."; e "Alguns requisitos me pareceram redundantes."
	RFFR002	0,00	7,14	92,86	100,00							
	RFFR003	3,57	7,14	89,29	100,00							
	RFFR004	0,00	14,29	85,71	100,00							
	RFFR005	10,71	32,14	57,14	100,00							
	RFFR006	10,71	14,29	75,00	100,00							
	RFFR007	21,43	39,29	39,29	100,00							
Monitoramento e Desempenho de Aprendizagem	RFPM001	0,00	17,86	82,14	100,00	3,57	3,57	7,14	46,43	39,29	100,00	"O monitoramento deve respeitar o grau de interesse = explorar gamificação" e "RFPM003 não está claro para mim"
	RFPM002	7,14	25,00	67,86	100,00							
	RFPM003	10,71	50,00	39,29	100,00							
	RFPM004	28,57	50,00	21,43	100,00							
Controle de Usuários	RFUS001	3,57	10,71	85,71	100,00	7,14	0,00	3,57	50,00	39,29	100,00	"A aplicação deveria aceitar apenas usuários autenticados"; "Anonimato possui suas vantagens. A integração com outras ferramentas para autenticação" e "No RFUS003, pode ser interessante permitir o acesso a usuários não autenticados com o intuito de despertar o interesse de "curiosos".
	RFUS002	10,71	14,29	75,00	100,00							
	RFUS003	57,14	21,43	21,43	100,00							
Mecanismos para Ensino de Programação	RFPG001	14,29	10,71	75,00	100,00	3,57	3,57	7,14	42,86	42,86	100,00	"No caso a aplicação irá ensinar português/português de programação? Dai não vejo a obrigatoriedade de ter o compilador/depurador."; "Apesar de serem importantes, acredito que boa parte dos requisitos não sejam fundamentais para o ensino de programação"; "Correção "compilador para "compilação"" e "RFPG005 não está clara. RFPG008: porque somente gamificação (talvez o termo ludificação seria melhor) está sendo considerado? Por que não scaffolding, PBL e outros métodos e técnicas pedagógicas?".
	RFPG002	7,14	10,71	82,14	100,00							
	RFPG003	14,29	32,14	53,57	100,00							
	RFPG004	3,57	32,14	64,29	100,00							
	RFPG005	3,57	21,43	75,00	100,00							
	RFPG006	35,71	21,43	42,86	100,00							
	RFPG007	25,00	39,29	35,71	100,00							
	RFPG008	14,29	28,57	57,14	100,00							
	RFPG009	0,00	14,29	85,71	100,00							
	RFPG010	32,14	50,00	17,86	100,00							
	RFPG011	21,43	32,14	46,43	100,00							
Apoio ao Ensino de Aprendizagem	RFTL001	10,71	10,71	78,57	100,00	3,57	0,00	3,57	46,43	46,43	100,00	"Alunos que olham o código fonte de um exercício que um colega desenvolveu, acham que entenderam, não tentam fazer, e na hora da avaliação, não conseguem fazer."; "A aplicação pode criar mecanismos para incentivar a self-regulated learning."; "Senti falta de mais requisitos sobre gamificação."; e "As atividades não devem ser atreladas a cursos e disciplinas. Creio que o ambiente deve ser livre para qualquer pessoa utilizar e aprender, independente de estar participando de um curso ou não."
	RFTL002	7,14	17,86	75,00	100,00							
	RFTL003	3,57	32,14	64,29	100,00							
	RFTL004	7,14	46,43	46,43	100,00							
	RFTL005	7,14	35,71	57,14	100,00							
	RFTL006	3,57	32,14	64,29	100,00							
	RFTL007	25,00	46,43	28,57	100,00							
	RFTL008	14,29	42,86	42,86	100,00							
	RFTL009	14,29	21,43	64,29	100,00							
	RFTL010	7,14	50,00	42,86	100,00							
	RFTL011	7,14	32,14	60,71	100,00							
	RFTL012	21,43	28,57	50,00	100,00							
	RFTL013	0,00	21,43	78,57	100,00							
	RFTL014	0,00	21,43	78,57	100,00							

Fonte: Dados da pesquisa.

Na categoria “Atividades de Aprendizagem” (Tabela 6), dois requisitos tiveram o mesmo percentual de prioridade alta, são eles: **RFAC002** *A aplicação deve permitir ao instrutor gerenciar diferentes atividades e tarefas* e **RFAC003** *O aplicativo deve permitir o envio de respostas para as atividades e tarefas dos alunos*. Em relação a atender as necessidades, a maioria (57,14%) indicou a resposta “4. Concorda Parcialmente”. Já na categoria “Conteúdos de de Aprendizagem”, o requisito classificado com prioridade alta foi o **RFCT001** *A aplicação deve permitir o gerenciamento de conteúdos pelos instrutores*. Em relação à indicação sobre atender as necessidades, a maioria (53,57%) também indicou a resposta “4. Concorda Parcialmente”.

Na categoria “Avaliação de Aprendizagem”, o requisito classificado com prioridade alta foi o **RFAT001** *A aplicação deve permitir o gerenciamento de avaliações de aprendizagem*, e o grupo de requisitos foi classificado que atende as necessidades parcialmente (resposta 4) (42,86%). Já na categoria “Feedback e Resultados”, o requisito classificado com prioridade alta foi o **RFFR002** *A aplicação deve fornecer feedback aos estímulos de modo completo, conciso e de fácil entendimento*. Tal categoria, assim como todas as demais, foi classificada como atendendo as necessidades parcialmente (resposta 4), correspondendo a 46,43% das respostas.

Na categoria “Monitoramento e Desempenho de Aprendizagem”, o requisito classificado com prioridade alta foi o **RFPM001** *A aplicação deve monitorar e registrar o progresso nos conteúdos e desempenho nas atividades educacionais dos aprendizes*. Esta categoria também foi indicada que atende parcialmente, com 46,43% das respostas. Em relação à categoria “Controle de Usuários”, o requisito classificado com prioridade alta foi o **RFUS001** *A aplicação deve permitir o gerenciamento de usuários*. Esta categoria foi classificada também como atendendo parcialmente as necessidades (50,00%). Já na categoria “Mecanismos para Ensino e Aprendizagem de Programação”, o requisito classificado com prioridade alta foi o **RFPG009** *A aplicação deve fornecer um ambiente de aprendizagem prático e interativo*. Tendo sido indicada como atendendo parcialmente as necessidades (42,86%).

Finalmente, na categoria “Apoio ao Ensino e Aprendizagem”, três requisitos tiveram o mesmo percentual de classificação na prioridade alta, sendo eles o **RFTL001** *A aplicação deve permitir o gerenciamento de cursos*, **RFTL013** *A aplicação deve permitir o desenvolvimento de habilidades de programação dos aprendizes* e **RFTL014** *A aplicação deve apoiar diferentes modalidades de ensino e suas estratégias didático-pedagógicas*.

Todos os comentários foram analisados. Alguns deles foram aceitos, levando a evolução dos requisitos. Quanto à transversalidade do requisito **RFAC006**, a necessidade de estar identificado no sistema é obrigatória no contexto de ensino formal, no qual o aluno encontra-se devidamente matriculado em um curso, vinculado a uma instituição de ensino e, portanto, possui controle de frequência (JAGUŠT; BOTIČKI; SO, 2018; SARRAB, 2015). Contudo, no contexto de ensino informal, a autenticação não se torna obrigatória, uma vez que o aprendiz estuda a seu tempo, sem estar vinculado a uma instituição de ensino, podendo ou não realizar as atividades (STRAUB, 2009); isso torna tal requisito opcional em determinados contextos.

Em relação a inclusão de outros tipos de conteúdo de aprendizagem, estes serão de responsabilidade do professor/instrutor, sendo plenamente possível a sua inclusão. No que tange a avaliação em pares, esta já está sendo considerada em requisitos relacionados a atividades cooperativas e colaborativas. Já em relação à utilização de verificação, validação e testes, em especial, a definição de casos de teste, os requisitos consideram a possibilidade de adoção dos mesmos, mas também será critério do instrutor/professor adotá-los ou não. Finalmente, a aprendizagem auto-regulada foi incluída em requisitos relacionados a estratégias de ensino e aprendizagem.

As duas avaliações conduzidas permitiram a identificação de evidências iniciais de que o catálogo atende às necessidades que foram concebidas tanto em termos de aplicação prática, por meio da prova de conceitos e estudo de caso, quanto na avaliação qualitativa, na qual as categorias foram indicadas que atendem parcialmente às necessidades. Neste sentido, avaliações conduzidas com a LPS e as aplicações podem vir a apoiar as evidências coletadas e aprofundar discussões sobre os requisitos e as funcionalidades que expressam.

4.6 Ameaças à Validade das Avaliações

Em relação à avaliação de validade, pode-se destacar as seguintes ameaças:

Ameaças à Validade de Conclusão: a principal ameaça em relação à avaliação qualitativa (principal avaliação conduzida) refere-se ao tamanho da amostra, que pode ser incrementado em estudos futuros.

Ameaças à Validade de Construção: como a análise considerou as respostas qualitativas dos participantes, estes foram convidados como voluntários, visto que tornar a participação obrigatória poderia gerar viés nas respostas coletadas.

Ameaças à Validade Interna:

- **Diferença entre participantes:** a experiência dos professores participantes pode interferir nos resultados coletados, visto que professores mais experientes possuem maior vivência em sala e, conseqüentemente, a análise dos requisitos pode ter ocorrido de modo mais crítico. Contudo, a heterogeneidade do perfil dos participantes é desejada para permitir uma maior generalização dos resultados. Nesse sentido, não se entende como uma ameaça aos resultados; e
- **Efeitos de Fadiga:** por meio da condução de piloto, tempo para responder tal avaliação foi de 30 minutos. Assim, efeitos de fadiga não foram considerados relevantes.

Ameaça à Validade Externa:

- **Instrumentação:** para garantir que o questionário não apresentasse possíveis problemas, um estudo piloto foi conduzido. O estudo permitiu que modificações fossem feitas quanto aos requisitos avaliados, tendo sido por meio deste a decisão de apresentar o código de referência e a descrição resumida de cada requisito, deixando disponível *link* de acesso ao conteúdo completo do catálogo, caso necessário.

4.7 O Catálogo Final

O catálogo final está disponível em <<https://andersonmarcolino.gitlab.io/app/>>. A Figura 17 apresenta a página principal do catálogo de requisitos.

Figura 17 – Página Principal do Catálogo de Requisitos.

Fonte: Elaborada pelo autor.

Já a Figura 18 apresenta um exemplo de requisito e os itens que o compõe.

Figura 18 – Exemplo de Apresentação de Requisito.

Fonte: Elaborada pelo autor.

O presente catálogo de requisitos foi publicado como trabalho em andamento, por não ter sido incluída a avaliação com professores discutida previamente (MARCOLINO; BARBOSA, 2016). A inclusão de tal avaliação foi realizada na publicação do periódico da pesquisa.

4.8 Considerações Finais

Neste capítulo foi apresentado um catálogo de requisitos *m-learning* composto por requisitos específicos do domínio de programação e requisitos educacionais gerais. Objetivou-se, com esse catálogo desenvolver o principal artefato para servir de insumo na concepção de uma LPS para aplicação *m-learning*. Adicionalmente, o catálogo tem como objetivo apoiar a criação de aplicações educacionais móveis para mitigar problemas no ensino e aprendizagem de programação, por meio da integração de teorias educacionais e da identificação de possíveis problemas de programação que cada requisito pode vir a sanar.

Na perspectiva de averiguar se o catálogo atende tais objetivos, o mesmo passou por duas avaliações. Uma delas constituída de prova de conceito com estudo de caso e outra, uma avaliação qualitativa com professores. Assim, foi possível identificar evidências iniciais positivas quanto ao uso e completude das oito categorias de requisitos, e também, quais requisitos são prioritários para integrar aplicações móveis educacionais, na perspectiva de professores de ensino

de programação.

Assim, considerando o catálogo de requisitos, os problemas de programação que cada requisito busca mitigar, e a priorização dos requisitos obtidos na avaliação com professores, uma LPS foi proposta e desenvolvida para permitir a criação de aplicações educacionais móveis para investigar a mitigação dos problemas de ensino e aprendizagem de programação, esta LPS é discutida no próximo capítulo.

UMA LPS PARA APLICAÇÕES *M-LEARNING* NO DOMÍNIO DE PROGRAMAÇÃO

5.1 Considerações Iniciais

As vantagens discutidas sobre a utilização da metodologia de LPS no Capítulo 2, deixam claro o potencial em se obter um portfólio mais amplo de produtos que atendam diferentes necessidades em um domínio específico. Nesta perspectiva, este capítulo apresenta o estabelecimento de uma LPS para o desenvolvimento de aplicações *m-learning* para o ensino e aprendizagem de fundamentos programação. Além da possibilidade de reuso dos diferentes artefatos da LPS, objetiva-se investigar a criação de aplicações que permitam mitigar ou resolver um maior número de problemas no ensino e aprendizagem de programação, diferenciando-se dos software e aplicativos discutidos no MS no Capítulo 3, que são desenvolvidos, em sua maioria, para resolver problemas específicos neste domínio.

Para isso, o catálogo de requisitos *m-learning*, discutido no Capítulo 4, buscou integrar categorias de problemas de programação em cada um de seus requisitos, de modo a permitir a identificação do problemas que podem ser mitigados ao se desenvolver tal requisito em um artefato de software. Contudo, no contexto de uma LPS, cabe garantir que cada uma de suas *features* sejam devidamente concebidas refletindo as soluções propostas nos requisitos. Neste sentido, o catálogo proposto foi adotado como artefato para: (i) definir o escopo da LPS, (ii) definir o projeto arquitetural, e (iii) desenvolver as *features* da LPS.

Com base no arcabouço SPLE e seus cinco sub-processos da Engenharia de Domínio (Capítulo 2), este capítulo discute o estabelecimento da LPS, considerando os sub-processos de gerenciamento de produtos (Seção 5.2); de engenharia de requisitos de domínio (Seção 5.3); de projeto de domínio (Seção 5.4); a avaliação do modelo conceitual da arquitetura da LPS (Seção 5.5); a evolução da arquitetura, a partir dos resultados da primeira avaliação, e sua avaliação

(Seção 5.6); os sub-processos de realização e testes de domínio 5.6.4. Apresenta-se, ainda, uma discussão sobre o modelo arquitetural e suas contribuições no apoio ao desenvolvimento das aplicações e, em especial, como o projeto arquitetural pode auxiliar a mitigar problemas no domínio de ensino e aprendizagem de programação (Seção 5.6.5).

5.2 Gerenciamento de Produtos

O primeiro dos cinco sub-processos do arcabouço SPLE no ciclo de Engenharia de Domínio refere-se ao gerenciamento de produtos (*product management*). Nele são determinadas as *features* variáveis e comuns de uma LPS para a definição do escopo da linha. Caso uma nova característica seja requerida, é por meio deste sub-processo que se avalia o impacto e as implicações na atualização de todos os outros sub-processos envolvidos (LINDEN; SCHMID; ROMMES, 2007). Portanto, é um sub-processo realizado não somente na concepção inicial da linha, mas diante de qualquer eventual modificação e evolução da mesma.

A princípio, este sub-processo é tratado no contexto da concepção do escopo da linha. Deste modo, o principal artefato de entrada para o estabelecimento de tal escopo foi o catálogo de requisitos (Capítulo 4). Por meio deste catálogo foram mapeadas as variabilidades e similaridades da linha, representadas por meio de um modelo de *features*. Esse modelo foi concebido por meio da ferramenta FeatureIDE¹ (THÜM *et al.*, 2014), um *plugin* Eclipse² que gera um arquivo de extensão XML (*eXtensible Markup Language*), capaz de ser importado em outros mecanismos, e.g., um *parser* ou outras ferramentas para o gerenciamento de produtos apoiando, também, a rastreabilidade de código fonte.

Devido à quantidade de *features*, tal modelo está disponibilizado online em <<https://andersonmarcolino.gitlab.io/app/featuremodel>>, bem como a sua representação em XML. A ferramenta FeatureIDE também foi utilizada para conduzir uma primeira validação do modelo de *features*. A validação não identificou nenhuma característica morta (*dead features*) e possibilitou a identificação de 874 produtos válidos considerando o desenvolvimento de *features* para cada um dos 97 requisitos.

5.3 Engenharia de Requisitos de Domínio

Este sub-processo é responsável por cinco tarefas básicas da engenharia de requisitos: elicitação, documentação, negociação, validação/verificação e gerenciamento. Inclui-se, ainda, a concepção dos artefatos de requisitos, e.g., objetivos, *features*, cenários, diagramas, dados, modelos comportamentais e funcionais (LINDEN; SCHMID; ROMMES, 2007).

¹ <goo.gl/JfqxEu>

² <<http://www.eclipse.org/>>

As atividades de engenharia de requisitos deste sub-processo foram discutidas no Capítulo 4, no qual o catálogo de requisitos *m-learning* no domínio de ensino e aprendizagem de programação foi planejado, desenvolvido e avaliado.

As *features*, no contexto de artefatos de requisitos, são as *features* de um sistema visíveis ao usuário final. Em uma LPS, elas definem um conjunto de *features* variáveis e comuns, que necessitam satisfazer um conjunto de restrições para serem utilizadas de modo consistente. Para esta LPS, o modelo de *features* (KANG *et al.*, 1990; METZGER; POHL, 2014) discutido na seção anterior foi adotado como principal artefato a refletir os requisitos em funcionalidades, permitindo a condução do sub-processo de projeto de domínio.

5.4 Projeto de Domínio

A arquitetura de software consiste na definição de seus componentes, suas propriedades externas e seus relacionamentos com outros software. Essa arquitetura, expressa por meio de modelos e podendo ser representada em diferentes níveis de abstração, é resultado do processo de projeto arquitetural. Tal processo, por sua vez, é a ligação crítica entre a engenharia de requisitos e o projeto arquitetural, já que é nele que serão identificados os principais componentes estruturais de um sistema e os relacionamentos entre eles. Neste sub-processo do arcabouço SPLE define-se o projeto da arquitetura de referência da LPS considerando o modelo de *features* desenvolvido em sub-processo anterior.

Uma arquitetura de referência resulta em uma ou mais arquiteturas de software e são originárias da integração entre padrões ou estilos arquiteturais com um modelo de referência (BASS; CLEMENTS; KAZMAN, 2003). No contexto de LPS, uma arquitetura de referência deve retratar os possíveis produtos a serem gerados e é composta pelos seguintes artefatos (POHL; BOCKLE; LINDEN, 2005): (i) a estrutura da arquitetura; (ii) a tessitura da arquitetura (*texture*); e (iii) possíveis e diferentes visões arquiteturais.

Basicamente, a estrutura da arquitetura exhibe, em uma visão macro, as camadas ou pacotes e seus sub-pacotes das aplicações. Esses pacotes, por sua vez, seguem a tessitura arquitetural, que nada mais é do que um conjunto de regras que esses pacotes devem obedecer no sentido de estabelecerem relação, a formarem a solução desejada (e.g., similaridades, variabilidades, pontos de variação, variantes e restrições). Finalmente, a visão da arquitetura representa o sistema em diferentes visões, a capturar as especificações das soluções, para o seu desenvolvimento (POHL; BOCKLE; LINDEN, 2005; MARCOLINO; OLIVEIRA JR; GIMENES, 2014; MARCOLINO *et al.*, 2014; LINDEN; SCHMID; ROMMES, 2007).

Tendo como artefatos o catálogo de requisitos (MARCOLINO; BARBOSA, 2016), o modelo de *features*, e os resultados do mapeamento de LPS (MARCOLINO; BARBOSA, 2015a) (Capítulo 3), no que diz respeito às estratégias e decisões tecnológicas adotadas na concepção dessas LPS, a arquitetura de referência da LPS foi definida. Neste primeiro momento, optou-se

pela modelagem conceitual da arquitetura, com nível de abstração alto. Os objetivos principais deste modelo são: averiguar a viabilidade da arquitetura, e identificar possíveis melhorias para se desenvolver os próximos modelos, de mais baixo nível.

O estilo arquitetural escolhido foi o orientado a serviços (*Service Oriented Architecture*(SOA)). SOA foi selecionados por seguir os princípios chave de padrões arquiteturais e permitir uma melhor reutilização, quando integrado a uma LPS (PATTERNS, 2009). Esse estilo arquitetural permite que as funcionalidades sejam fornecidas como um conjunto de serviços. Tais serviços são fracamente acoplados, visto que utilizam interfaces padronizadas. Assim, por serem autônomos, distribuídos e fracamente acoplados, os serviços preconizam o reuso, tornando-se adequados ao contexto de LPS e outras metodologias de reutilização (PATTERNS, 2009; PAHL; JAMSHIDI; ZIMMERMANN, 2018; HASSAN; AZIZ, 2018).

É importante mencionar que, ainda no contexto de metodologias de reuso, o desenvolvimento dirigido a modelos – *Model-Driven Development*, (MDD) – é uma das metodologias que também tem ganho destaque (CERNY; DONAHOO; TRNKA, 2018). Contudo, mesmo que o MDD permita a transformação de conhecimento entre os serviços de uma solução, o alto nível de abstração usado na abordagem aumenta a complexidade de generalização do projeto e de suas regras de transformação, tornando-a raramente utilizada no contexto de SOA (CERNY; DONAHOO; TRNKA, 2018). Ademais, a ferramenta TANGRAM foi desenvolvida considerando serviços e microserviços web para auxiliar na criação dos produtos da LPS, o que a torna incompatível com a estratégia utilizada no MDD. Os detalhes da implementação e da ferramenta são melhores discutidos no Capítulo 6.

Ainda em relação a SOA, com o intuito de averiguar a validade da arquitetura proposta para a LPS, principalmente na adoção desse padrão arquitetural, um MS para identificar arquiteturas para aplicações móveis educacionais foi conduzido. Este MS buscou, ainda, identificar se há arquiteturas de aplicações móveis que consideram LPS, considerando o período de 2006 a 2016 para recuperação dos estudos primários (MARCOLINO; OLIVEIRAJR,). No total, 26 propostas de arquitetura foram identificadas. Tais propostas foram, então, analisadas e utilizadas na averiguação da assertividade do modelo conceitual desenvolvido. Os estudos permitiram também a identificação do quão viável é a adoção de serviços, para conceber as aplicações móveis no contexto da LPS proposta. Este mapeamento é parte integrante de um artigo publicado (MARCOLINO; OLIVEIRAJR,) e está disponível em <<https://drive.google.com/file/d/0B1DD-RxftMa0eDdfSWRzQmE1N00/view?usp=sharing>>.

A Figura 19 apresenta o modelo conceitual da arquitetura da LPS considerando um diagrama de pacotes, sub-pacotes e uma lista de abordagens e tecnologias adotadas. Destaca-se que tais adoções referem-se à primeira parte da arquitetura. Ou seja, às *features* da LPS e à ferramenta TANGRAM, responsável por facilitar a concepção das aplicações *m-learning*, desenvolvido considerando os trabalhos do MS sobre LPS (Capítulo 3), principalmente, a proposta de Falvo Junior *et al.* (2014).

Figura 19 – Visão Macro da Arquitetura da LPS e da Ferramenta de Apoio.

Fonte: Elaborada pelo autor.

No topo da Figura 19 é apresentado o arcabouço SPLE e suas camadas. A interação entre os sub-processos do ciclo de vida de Engenharia de Domínio (seta verde) apoia a criação das aplicações. A ferramenta de apoio TANGRAM (*application mechanism*) interage com o ciclo de Engenharia de Aplicação para auxiliar na criação das aplicações para o ensino de

programação (seta azul), reduzindo a necessidade de apoio técnico para a criação dos produtos da LPS. Adicionalmente, a ferramenta também apoia a criação dos módulos educacionais que serão disponibilizados para os alunos na aplicação.

O usuário seleciona as *features* por meio de uma interface web (*Web UI*) e envia tais requisições para o servidor (*REST services*), que identifica as mesmas, selecionando a configuração da aplicação (*application template*) desejada. Essa configuração é então salva para posterior uso no desenvolvimento de outras aplicações, servindo como um modelo de configuração.

Os pacotes representados na arquitetura de referência conceitual das aplicações contemplam as nove facetas identificadas e definidas no catálogo de requisitos (Figura 19). Os pacotes, sub-pacotes e relacionamentos em linhas tracejadas representam *features* variáveis; já os representados em linhas sólidas indicam as *features* comuns.

A seta azul indica a especialização da arquitetura das aplicações *m-learning*. A aplicação móvel do cliente apresenta uma visão macro das camadas de apresentação (*Presentation Layer*), de serviços (*Service Layer*), de negócios (*Business Layer - Educational and Programming*), de dados ou persistência (*Data Layer - persistence*), a de elementos transversais (*Cross-Cutting/Orthogonal Services Layer*) e, finalmente, a interação das aplicações com elementos externos (*External Infrastructure e External Sources*).

Neste ponto é destacada uma modificação proposta no arcabouço SPLE para atender a integração da ferramenta de apoio e evolução da linha. Para melhor atender ao domínio educacional escolhido e possibilitar a entrega de novas *features* para as aplicações, os usuários (i.e., professores e alunos) poderão fazer a solicitação de novas funcionalidades por meio da ferramenta TANGRAM. Esse processo é apoiado por *feedback* ao sub-processo de gerenciamento de produtos (seta vermelha).

A estrutura das aplicações *m-learning* apresentada na Figura 19 inclui cinco camadas, além das referentes à infraestrutura externa e fontes externas. Suas camadas e tessitura são caracterizadas como se segue. Para cada uma delas foram adicionados desafios no contexto dos próximos sub-processos do arcabouço SPLE, a serem tratados.

- **Camada de apresentação:** é a camada responsável por fornecer a interface gráfica do usuário e seus componentes presentes no sub-pacote de **componentes de interface de usuário** (*UI Components*). Ela também é responsável por traduzir as interações dos usuários em comandos lógicos que serão redirecionados para as outras camadas de aplicação por meio do sub-pacote de **controladores**.

Os desafios para o sub-processo de **realização de domínio** relacionados com tais componentes é a concepção de interfaces adaptáveis ou responsivas, uma vez que a aplicação deve ser executada em diferentes dispositivos móveis, com diferentes resoluções. Além disso, é necessária a adaptação ou redução de elementos gráficos provenientes de outras modalidades para reduzir a carga de memória dos usuários no reconhecimento da interface

e de suas funcionalidades. O principal recurso para reduzir a carga visual e/ou gráfica dos diferentes elementos a serem disponibilizados para os usuários é concentrando-os em elementos gráficos específicos, como menus flutuantes e botões que ocultam e apresentam outros sub-conjuntos de elementos gráficos quando receberem interações dos usuários.

Já no contexto de interações de entrada de dados, o uso dos teclados virtuais disponíveis nos dispositivos pode vir a desmotivar a inserção de trechos de código mais longos nas tarefas de codificação. Nesta perspectiva, meios mais naturais e motivantes para a inserção de textos devem ser investigados e integrados, e.g., interações gestuais nas telas de multitoque, pareamento com outros dispositivos, entre outros.

- **Camada de Serviços:** as interfaces e contratos de dados, ou tipo de mensagens, são definidos e implementados nesta camada, o que evita a exposição de detalhes dos processos internos ou de regras de negócios usadas na aplicação. As interfaces de serviço expõem a lógica de negócios implementada na aplicação para potenciais consumidores.

Transferência de Estado Representacional (*Representational State Transfer* (REST)) foi escolhido como o estilo de implementação dos serviços web. Ele provê que aplicações e dados expressos em serviços web, também conhecidos como *RESTful*, sejam compatíveis com um conjunto de operações HTTP, facilitando o processo de comunicação e interoperabilidade entre sistemas web. (CHAVES *et al.*, 2018; KIM *et al.*, 2018).

O principal desafio nesta camada é garantir a disponibilidade de tais recursos aos clientes. Mais ainda, o modo como tais serviços irão se comportar com a camada de negócios (*features* de domínio educacional e de programação), considerando o baixo acoplamento.

- **Camada de Negócios:** esta camada engloba tanto questões relacionadas aos processos de ensino e aprendizagem em geral, quanto itens específicos do ensino e aprendizagem de programação. A interface de aplicação (*application facade/ interface*) recebe as requisições e as redireciona para os sub-pacotes desejados. A ideia é prover uma interface simplificada para os componentes da camada lógica de negócios, por meio da combinação de múltiplas operações de negócios em uma única operação, facilitando o uso da camada em questão e reduzindo as dependências.

Uma camada lógica é definida como qualquer lógica de aplicação responsável por recuperar, processar, transformar e gerenciar os dados de aplicação, considerando as políticas e regras de negócios (CERNY; DONAHOO; TRNKA, 2018). No contexto educacional, a camada lógica é responsável por cada mecanismo de apoio aos conteúdos educacionais, atividades educacionais, avaliação, *feedback* de aprendizagem e desempenho.

O sub-pacote de **apoio ao ensino e aprendizagem** recebe os dados coletados nos componentes de interface de usuário e os utiliza para conduzir os processos de negócio de acordo com ordem preestabelecida. Ele lida com a disponibilidade de diferentes mecanismos

educacionais, e.g., conteúdos, atividades, avaliações, *feedback* e desempenho para atingir o objetivo educacional almejado.

Os **elementos educacionais**, por sua vez, agrupam as entidades responsáveis por encapsular a camada lógica e de dados, necessárias para representar elementos do mundo real, tais como alunos, professores, turmas, cursos. Eles provêm acesso necessário para os dados de negócio e funcionalidades relacionadas, validando os dados com as entidades e encapsulando a lógica de aplicação para garantir consistência e implementação de regras de negócio e seus comportamentos. Tutores inteligentes e outras entidades de apoio também estão incluídas neste sub-pacote.

Outras unidades estão diretamente relacionadas com o sub-pacote de **ensino de programação**, responsável por incluir as *features* e funcionalidades do domínio de programação, descrevendo como serão ensinados aos alunos, as estratégias, representações, mecanismos e ferramentas necessárias para permitir a concepção de soluções mais assertivas em relação as partes interessadas.

Quando o usuário selecionar as *features* na interface web (*web UI*) para criar sua aplicação, dicas relacionadas com cada característica e problemas de programação podem ser consultadas com o objetivo de auxiliá-los na criação de sua solução. Adicionalmente, considerações relacionadas com teorias educacionais também poderão ser consultadas.

- **Camada de Dados:** esta camada abstrai a lógica necessária para acessar os repositórios de dados, que podem ser tanto um repositório de dados do lado do cliente ou cache; ou diretamente em uma base de dados remota sincronizadas com funcionalidades da camada de serviços ortogonais/transversais (*cross-cutting/orthogonal services layer*). Eles centralizam funcionalidades de acesso de dados para facilitar a configuração e gerenciamento da aplicação. Os diferentes componentes para estas tarefas estão incluídos no sub-pacote de **componentes de acesso aos dados** (*data access components*).

O sub-pacote de **agentes de serviços** (*Service Agents*) implementa os componentes de acesso aos dados, necessários para a comunicação da aplicação com os serviços da infraestrutura externa (*external infrastructure*). Ele isola os requisitos variados dos serviços de chamada do aplicativo e converte os dados do aplicativo para serem utilizados por serviços externos por meio de requisições HTTP (REST).

O desafio principal nesses componentes é garantir a qualidade no tráfego de dados, apoiado pelos elementos do pacote de serviços transversais. Adicionalmente, o processo de monitorar e armazenar os dados é também um desafio, uma vez que uma possível grande quantidade de acessos dos alunos pode gerar instabilidade e requer a estabilidade para permitir acessos simultâneos.

- **Camada de Serviços Ortogonais/Transversais:** esta camada agrupa, principalmente, funcionalidades e *features* dos requisitos não funcionais, e.g., segurança, configuração, co-

municação (protocolos, tecnologias), mobilidade e especificidades de dispositivos móveis, disponibilidade e internacionalização. Inclui, ainda, a comunicação com fontes externas (*external sources*), e.g., aplicações e outros sistemas educacionais, como *e-learning* ou até mesmo *t-learning*; e repositórios de objetos de aprendizagem (OA) e/ou recursos educacionais abertos (REA) (DAGIENE; GUDONIENE; BARTKUTE, 2018).

O desafio relacionado com essa camada, considerando uma aplicação baseada em serviços, refere-se principalmente em como encapsulá-los. Como as funcionalidades que compõem tal pacote são transversais, o encapsulamento desses componentes não é trivial e muitas vezes não é possível, visto que lidam como especificidades de qualidade espalhadas por todos os componentes da arquitetura. Além disso, considerações sobre qualidade dos serviços devem ser incluídas com base na ordem de prioridade definidas pelas partes interessadas, já que o atendimento de conjuntos amplos de requisitos de qualidade é mais complexo e custoso. Outro fator refere-se às tecnologias de desenvolvimento que podem ser adotadas, influenciando no modo como os componentes se comunicam e são mantidos.

Além das considerações apresentadas em cada uma das camadas da primeira versão da arquitetura, faz-se necessário mencionar duas de caráter geral, a saber:

- A arquitetura pode apoiar o desenvolvimento de aplicações *m-learning* para ensino formal e informal. Essa possibilidade justifica alguns pacotes e sub-pacotes serem indicados como opcionais, conforme ilustrado na Figura 19.
- Os diferentes níveis de abstração dos diagramas podem levar a um melhor entendimento de cada um dos pacotes e componentes presentes em cada sub-pacote. Modelos mais detalhados serão concebidos para a condução dos sub-processos do SPLE. Contudo, considerando a importância de verificar a viabilidade da arquitetura proposta, ainda em nível conceitual, uma avaliação com especialistas foi conduzida.

5.5 Avaliação do Modelo Conceitual da Arquitetura

Considerando a necessidade de identificar se a arquitetura proposta atende as partes interessadas, em termos de desenvolvimento de soluções, e sua viabilidade quanto às suas *features* técnicas, no que tange desenvolvedores, engenheiros de software e especialistas em aplicações móveis educacionais, um questionário *online*, no formato de *checklist* foi conduzido. Adicionalmente, a avaliação do modelo preconiza a identificação de possíveis problemas e melhorias para mesma, antes do processo de desenvolvimento dos artefatos que a compõem, para evitar retrabalho.

5.5.1 Projeto e Execução

A avaliação da arquitetura (Figura 19) correspondeu a uma análise qualitativa, por meio de um questionário *online*, sendo respondido por 31 pessoas que participaram voluntariamente. A maioria desses participantes possuía experiência na área de engenharia de software, com média de 5.2 anos de experiência. 20 são estudantes de mestrado, 11 estudantes de doutorado, seis mestres e dois doutores.

Em relação à experiência dos mesmos, 23 são professores de programação e oito são engenheiros de software. Entre os professores, três são especialistas em *m-learning* e quatro são desenvolvedores móveis.

Antes da aplicação do questionário, um engenheiro de software, especialista em arquitetura de referência da Universidade de São Paulo participou de uma execução piloto para validar a instrumentação e, em consequência, o processo de execução.

A instrumentação utilizada na execução é composta por: (a) termo de consentimento; (b) descrição, objetivos e instruções do estudo; (c) representação gráfica do modelo da arquitetura; (d) descrição textual do modelo da arquitetura; (e) questionário de caracterização; e (f) questionário com 25 questões, das quais uma era opcional, para indicação de considerações e/ou opiniões.

O questionário foi composto de 24 questões baseadas no estudo de Filho e Barbosa (2015). Para cada questão, os participantes deveriam indicar, em uma escala de 1 a 4 se: (1) “concordam totalmente”; (2) “concordam parcialmente”; (3) “não concordam”; (4) “não concordam nem discordam”.

O questionário aplicado, bem como detalhes sobre a análise qualitativa, estão disponíveis em <<https://drive.google.com/open?id=0B1DD-RxftMa0bkRRRGg4ZFFZRmM>>. Para a verificação de diferentes aspectos da arquitetura, ele foi dividido em sete grupos: **aspectos gerais** (questões de 1 a 5), **aspectos conceituais** (questões 6 e 7), **aspectos de infraestrutura** (questões de 8, 9, 10 e 22), **aspectos de modelagem** (11 e 14), **aspectos SOA** (questões 12 e 13), **aspectos de domínio educacional** (questões de 15 a 19), e **aspectos de m-learning** (questões 20, 21, 23 e 24).

5.5.2 Resultados

A Figura 20 apresenta o total geral dos resultados. A maioria dos participantes (52%) concordam totalmente que o modelo conceitual da arquitetura e seus conceitos educacionais e de *m-learning* são factíveis. 42% indicaram que o modelo “*atende parcialmente*” o domínio, 2% indicaram que o modelo “*não atende*”, e 4% indicaram serem indiferentes, “*não concordando nem discordando*”. É importante mencionar que a indicação de não concordar nem discordar foi indicada para os participantes quando estes não conseguissem responder a questão, por falta de domínio da mesma.

Figura 20 – Resultados Gerais da Avaliação.

Fonte: Dados da pesquisa.

Com base nas respostas dos participantes foi possível identificar evidências preliminares de que todos os aspectos arquiteturais avaliados atendem totalmente ao objetivo proposto, que refere-se a atender à necessidade de identificar se a arquitetura proposta atende as partes interessadas, em termos de desenvolvimento de soluções, e sua viabilidade quanto às suas *features* técnicas, no que tange desenvolvedores, engenheiros de software e especialistas em aplicações móveis educacionais. Este aspecto é representado na Figura 21.

Figura 21 – Resultados da Avaliação por Conjuntos de Aspectos.

Fonte: Dados da pesquisa.

Ao analisar as respostas por grupo das partes interessadas, nota-se que o grupo restrito a somente professores de programação (16) indicaram que todos os aspectos foram considerados como “*atendendo completamente*” (Figure 22), com exceção dos **aspectos de modelagem**, apresentando um empate em relação ao número deles que indicaram que “*atende totalmente*” e “*parcialmente*” (42,2%).

Figura 22 – Resultados da Avaliação - Professores de Programação.

Fonte: Dados da pesquisa.

Na perspectiva dos engenheiros de software (8), os **aspectos de infraestrutura** foram “*totalmente aceitos*”, já os aspectos **conceituais** e de **modelagem** apresentaram um empate (50,0%) (Figure 23). Os demais aspectos foram considerados como “*atendidos parcialmente*”.

Figura 23 – Resultados da Avaliação - Engenheiros de Software.

Fonte: Dados da pesquisa.

Na perspectiva dos desenvolvedores móveis (4), todos consideraram todos os aspectos como satisfazendo totalmente (Figure 24).

Figura 24 – Resultados da Avaliação - Desenvolvedor Móvel.

Fonte: Dados da pesquisa.

Finalmente, na perspectiva das respostas dos especialistas em *m-learning* (3), ocorreu empate quanto aos **aspectos de infraestrutura** (50,0%), sendo indicado que “*atendem totalmente*” e “*parcialmente*”. Nos demais aspectos, todos foram considerados como “*atendendo totalmente*” (Figure 25).

Figura 25 – Resultados da Avaliação - Especialistas M-learning.

Fonte: Dados da pesquisa.

Além de obter os resultados apresentados, foram coletadas ainda, algumas considerações, sugestões e melhorias. As de destaque são sintetizadas a seguir:

- Sobre a camada de serviços, participantes indicaram dificuldades no entendimento desta e que o alto nível de abstração do modelo não permite dizer que se a camada de serviços e seus sub-pacotes garantem que os requisitos serão atendidos.
- Participantes demonstraram dúvidas quanto ao atendimento dos requisitos de qualidade, como segurança e escalabilidade, visto que tais *features* são muito sensíveis e serão influenciadas pelo modo como serão derivadas. Neste contexto foi indicado o uso de abordagens específicas para avaliar arquiteturas de referência, e.g., a *Architecture Tradeoff Analysis Method* (ATAM).
- Participantes indicaram que o modelo é fácil para entendimento e completo para a representação de todos os aspectos envolvidos. Em relação à descrição textual, foi indicado que a mesma não explica a relação entre os componentes e que a adoção de um modelo comportamental pode auxiliar neste entendimento.

As respostas completas dos participantes estão disponíveis em <<http://goo.gl/Q4dcBF>>.

Ao analisar as respostas coletadas, uma preocupação principal é destacada: o alto grau de abstração do modelo, que pode limitar o uso do mesmo no processo de desenvolvimento. Como já mencionado, o modelo em alto nível foi submetido a uma avaliação para verificar sua factibilidade e, então, permitir a modelagem em modelos de mais baixo nível.

Contudo, é necessário destacar que os resultados referentes ao atendimento da arquitetura ao objetivo proposto, quando analisados os resultados gerais (Figura 21), indicam que os **aspectos**

de infraestrutura, aspectos SOA e *m-learning* não foram aceitos como esperado, visto que apresentaram a maior quantidade de classificações em relação a “*indiferente*” (1) e “*não atende*” (2). Deste modo, levando em consideração os aspectos observados na avaliação, o modelo arquitetural foi evoluído.

Esse primeiro modelo arquitetural e sua avaliação foram publicados em (MARCOLINO; BARBOSA, 2017b).

5.6 Evolução da Arquitetura

Com base na avaliação do modelo conceitual da arquitetura, melhorias para a arquitetura foram projetadas e conduzidas. Os principais objetivos em evoluir o modelo arquitetural consistem em: (i) garantir melhor atendimento dos objetivos da LPS e maior reúso; (ii) discutir e propor melhorias para as *features* mapeadas, integrando novas *features*, quando necessário; (iii) garantir melhor atendimento aos requisitos de qualidade; (iv) propor um modelo arquitetural considerando conhecimento de desenvolvedores que atuam na indústria; e (v) conceber modelos com nível de abstração mais baixo, considerando a UML, para permitir a condução das próximas etapas do SPLE, i.e., realização de domínio e testes de domínio.

5.6.1 Metodologia

Como metodologia para a evolução da arquitetura, buscou-se integrar os conceitos utilizados para a concepção do modelo inicial com estratégias práticas provenientes da experiência de desenvolvedores de aplicações móveis e sistemas para Internet. Neste contexto, dois desenvolvedores se dispuseram a auxiliar o processo de análise e melhorias na arquitetura, durante o período de um mês.

A Figura 26 resume as seis atividades conduzidas para a concepção do modelo final da arquitetura da LPS. A condução de cada uma dessas atividades é detalhada nas próximas seções.

Figura 26 – Atividades para a Evolução do Modelo Arquitetural.

Fonte: Elaborada pelo autor.

5.6.2 Condução da Evolução da Arquitetura

A primeira atividade corresponde à análise dos requisitos e do modelo arquitetural de alto nível, avaliados anteriormente. Tal análise permitiu a melhoria de alguns requisitos, no que se refere às suas descrições textuais. Seguiu-se para a segunda atividade, na qual foi proposto um novo modelo arquitetural que considerou:

1. o padrão multi-inquilinos, no qual uma única solução de software é executada em um servidor, servindo diferentes grupos de usuários que compartilham um acesso comum a tal instância de software (DELIMITROU; KOZYRAKIS, 2018; ADEWOJO; BASS, 2018);
2. o padrão repositório, que evita o acesso direto aos dados pela camada lógica de negócios, serviços ou outros componentes, reduzindo a duplicação de código, dados de negócio fracamente tipados, a dificuldade em centralizar dados do cache e de vários bancos de

dados, e a incapacidade de testar facilmente a camada lógica (MICROSOFT, 2017; DIAS *et al.*, 2018); e

3. a adoção do conceito de microsserviços, visto que permite o desenvolvimento de software como serviços (SaaS) (NEWMAN, 2015; BALALAIÉ *et al.*, 2018), nas quais que os usuários de diferentes plataformas e tecnologias são tratados como consumidores destes.

A evolução da arquitetura teve como premissa, considerar os pontos indicados na avaliação realizada para atender ao objetivo da LPS em desenvolver aplicações *m-learning* para investigar a mitigação de problemas no ensino e aprendizagem de programação. Considerou-se, ainda, a integração da ferramenta TANGRAM, para facilitar a criação de configurações e aplicações por meio da seleção das *features* da linha.

Proposição e Evolução do Estilo Arquitetural

A Figura 27 reflete a adoção dos padrões e o conceito de microsserviços na atualização do modelo conceitual da arquitetura, proposto anteriormente.

Figura 27 – Modelo Conceitual Atualizado da Arquitetura das Aplicações Móveis.

Fonte: Elaborada pelo autor.

Ao utilizar a TANGRAM para criar uma configuração e aplicação, o usuário cria um registro em banco, esse registro permite que os diferentes usuários (i.e., alunos ou professores) acessem e visualizem apenas o que for pertinente àquilo que lhes foi permitido durante a criação da aplicação. Nesse sentido, considerando o padrão arquitetural adotado, cada usuário é visto como um inquilino, que só tem acesso aos dados que correspondem à permissão que lhes foi

dada, considerando, ainda, a configuração que determina qual conjunto de *features* e problemas de programação que aquela aplicação em específico visa mitigar.

O padrão repositório auxilia, por sua vez, o padrão multi-inquilino. Com o conceito de inquilinos, as aplicações *m-learning* tornam-se dependentes de conexão com a Internet, visto que necessitam recuperar os dados da configuração e da aplicação criadas na TANGRAM, para renderizar a interface ao usuário, bem como para enviar os dados das diferentes interações realizadas. Logo, o padrão repositório auxilia no gerenciamento desses dados e também na criação de *cache* para envio dos dados registrados durante a ausência de conexão, por exemplo.

Já a adoção do conceito de microsserviços torna os produtos da LPS aderentes ao conceito de software como serviço. Ao utilizar microsserviços, os serviços passam a ser unidades menores, desenvolvidas com maior coesão e com baixo acoplamento, tornando-as mais independentes (NEWMAN, 2015). Deste modo, possibilita-se um melhor reflexo dos requisitos nas *features* desenvolvidas, aperfeiçoa-se o reúso, reduz-se falhas no software como um todo, e diminui-se a complexidade de concepção.

Logo, um microsserviço é definido como um componente independente implantável de escopo limitado, que apoia a interoperabilidade por meio de comunicação por mensagens (NADAREISHVILI *et al.*, 2016). Tanto serviços como microsserviços podem utilizar protocolos síncronos (i.e., HTTP/REST, protocolo de acesso a objeto simples (SOAP) (*simple object access protocol*)), ou protocolos assíncronos (i.e., fila de mensagens avançadas (AMQP) (*Advanced Message Queuing Protocol*), transporte de fila de mensagens de telemetria (*message queue telemetry transport*)), facilitando a comunicação com consumidores externos (i.e., outros componentes de hardwares, aplicações de terceiros). Como tanto serviços quanto microsserviços possuem seus próprios repositórios de dados, a consistência de dados é melhorada, e adicionar um padrão de gerenciamento, como o padrão repositório, resultará em uma solução ainda mais robusta.

Por outro lado, um grande quantidade de microsserviços requer cuidados adicionais no que se refere à comunicação entre tais microsserviços, já que muitas vezes tal comunicação torna-se complexa (NADAREISHVILI *et al.*, 2016). Assim, é necessário incluir algum mecanismo que gerencie as requisições, sejam estas originárias de microsserviços ou clientes externos. No modelo atual optou-se pelo uso de um *gateway* da interface de programação da aplicação (*application programming interface*) ou, simplesmente, *API Gateway*. Esse *gateway*, além de auxiliar o redirecionamento de requisições realizada em diferentes protocolos, reduz possíveis falhas por meio de um balanceador de cargas (*load balancer*).

Modelo de Componentes da Arquitetura

Com a visão conceitual da arquitetura atualizada, a terceira atividade refere-se à modelagem da arquitetura considerando a abordagem para representação de variabilidades *SMarty* (OLIVEIRAJR; GIMENES; MALDONADO, 2013; MARCOLINO; BARBOSA, 2015a). Se-

gundo as indicações do SPLE e seu sub-processo de projeto de domínio, a visão lógica é representada por meio do modelo de *features* (LINDEN; SCHMID; ROMMES, 2007). Já em uma visão a ser considerada para o desenvolvimento, pode-se adotar o diagrama de pacotes, de componentes e/ou de classes. Assim, considerando os modelos apoiados pelo *SMarty*, optou-se pelo estabelecimento do modelo de componentes (PRESSMAN, 2015), apresentado na Figura 28.

O modelo de componentes da LPS possui quatro camadas: de aplicações clientes (*client apps*), *API Gateway*, microsserviços (*microservices*) e repositório (*repositories*).

- **Camada de Aplicações de Clientes (*Client Apps layer*):** esta camada apresenta as aplicações (apps), plataformas, páginas da Internet, serviços e outros consumidores em potencial de microsserviços. Os principais clientes considerados neste projeto são as aplicações móveis (*mobile apps*) e páginas da Internet (*websites*). Os clientes possuem apenas a visão e o controlador da aplicação. Toda a lógica de negócios está inserida nos microsserviços. Contudo, eles são o principal meio de integrar todas as *features* desejadas pelo inquilino/usuário, previamente definidas na TANGRAM. Adicionalmente, todos os clientes são considerados variabilidades opcionais, sendo estereotipados como «optional» e possuindo um comentário da UML com informações adicionais. Essas variabilidades são resolvidas durante a criação da configuração na TANGRAM.
- ***API Gateway*:** esta camada identifica os microsserviços disponíveis e registrados em um repositório específico por meio do componente de registro de serviço (*service registry*), redirecionando as requisições recebidas por clientes (inquilinos) internos e/ou externos. Adicionalmente, seu balanceador de cargas gerencia o tráfego de dados e garante que cada uma das requisições recebidas sejam respondidas. Esse componente é também responsável por ampliar os recursos dos serviços, provendo a elasticidade de recursos da nuvem (PRESSMAN, 2015; NEWMAN, 2015; BALALAIE *et al.*, 2018).
- **Camada de Microsserviços (*Microservices layer*):** esta camada inclui todos os microsserviços e estes podem, por sua vez, estarem hospedados em diferentes servidores. As especificidades não funcionais são transversais a todos eles, o que pode gerar redundância de trechos de código para manter o fraco acoplamento. O modelo (Figura 28) apresenta quatro exemplos de microsserviços. Devido a sua natureza diversificada, cada microsserviço é estereotipado como opcional, mas, seguindo o modelo de *features*, alguns deles representam variabilidades exclusivas («alternative_XO») e/ou mutuamente exclusivas. Por exemplo, atividades educacionais podem ser disponibilizadas seguindo uma ordem fixa ou serem realizadas por conveniência. Portanto, essas duas *features* seguem restrições, que as tornam mutuamente exclusivas entre si - ao selecionar uma, a outra não poderá estar presente. Por outro lado, há *features* que são inclusivas, como as relacionadas ao

Figura 28 – Modelo de Componentes da LPS.

Fonte: Elaborada pelo autor.

monitoramento de aprendizagem e desempenho, que necessitam das *features* de *feedback* e apoiam tanto professores quanto alunos.

- **Camada de Repositórios (*Repositories Layer*):** nesta camada estão as estruturas referentes ao padrão repositório e os repositórios de dados. Elas podem estar hospedadas em diferentes servidores e, em alguns casos, são obrigatórias. Essa camada é responsável pelo gerenciamento dos processos de persistência, provendo interfaces no modelo de objetos de acesso de dados (DAO) (*data access object*), quando conveniente, para permitir o acesso aos banco de dados registrados. Ademais, é nela que é coordenado o uso do *cache*.

Modelo de Classes de Microsserviço da Arquitetura

O próximo artefato da arquitetura, definido no SPLE, corresponde ao diagrama de classes (Figura 29)(POHL; METZGER, 2018). Para esta representação, como cada microsserviço expressa uma *feature*, ou um conjunto reduzido destas e, portanto, possuem estruturas similares, um dos microsserviços foi modelado.

A Figura 29 apresenta o diagrama de classes nos seus respectivos componentes, correspondendo ao microsserviço de conteúdo educacional. Três variabilidades são apresentadas no diagrama de classes: dois clientes resolvidos em tempo de projeto (*design time*); e o microsserviço de conteúdo, resolvido em tempo de execução (*runtime*), com base na configuração criada na TANGRAM. O microsserviço de conteúdo educacional é considerado uma variabilidade pois algumas aplicações de aprendizagem podem ser desenvolvidas com foco em prover apenas atividades educacionais para os alunos, não sendo, portanto, obrigatória para todas as aplicações.

De posse dos dois modelos atualizados da arquitetura e representando a mesma em mais baixo nível, os modelos passaram por análise e correções junto aos desenvolvedores, correspondendo à quarta atividade. A análise foi similar ao modelo conceitual da arquitetura. Contudo, visto que os modelos apresentaram nível arquitetural mais baixo, verificou-se também questões de segurança e desempenho, considerando os padrões e estratégias adotadas.

Após análise e nenhuma alteração dos modelos, a quinta atividade referente a uma avaliação qualitativa foi conduzida. Esta avaliação teve como objetivo identificar em uma perspectiva técnica, com desenvolvedores especializados em serviços e microsserviços, se os padrões e conceitos adotados estão adequados ao modelo arquitetural atualizado.

5.6.3 Avaliação dos Novos Modelos Arquiteturais

O método utilizado para avaliar os modelos da arquitetura foi uma análise qualitativa por meio de questionário *online*. Este questionário foi divulgado em redes sociais, lista de e-mails corporativos, sendo os participantes voluntários. O único requisito para participação era a necessidade de possuir conhecimento intermediário ou avançado em microsserviços e serviços.

Figura 29 – Modelo de Classes do microserviço de Conteúdo Educacional.

Fonte: Elaborada pelo autor.

A instrumentação foi dividida em duas partes. A primeira parte do questionário possuía um termo de consentimento e o questionário de caracterização. A segunda parte do questionário era composta pelas figuras dos modelos arquiteturas de componentes e classes, descrição textual de tais modelos, seguidos de quatro questões relacionadas aos modelos arquiteturais, que visam avaliar o quão assertivos tais modelos estão em relação à adoção dos padrões arquiteturais e conceitos referentes aos microserviços e serviços.

Uma execução piloto foi realizada com os dois desenvolvedores para verificar possíveis problemas. Após execução, apenas algumas correções textuais foram realizadas em duas questões. Em seguida o questionário foi disponibilizado e compartilhado.

Um total de 11 participantes responderam ao questionário. A média em anos da experiência dos participantes sobre conceitos de serviços foi de 4 anos, enquanto que, para microserviços, foi de 2,85 anos. Ao analisar o participante com maior experiência em serviços e microserviços, nota-se que há uma diferença de 3 anos. Ou seja, 7 anos de experiência em serviços, contra 4 de microserviços.

Em relação à formação dos 11 participantes, sete são mestres, um é mestrando e três possuem ensino superior completo. Já em termos de localização, cinco são do estado de São

Paulo, quatro do Paraná, um de Santa Catarina e um do país de Moçambique, do continente africano, tendo este concluído mestrado em universidade brasileira.

Análise dos Resultados

A primeira questão solicitou aos participantes para classificassem o quão correta está a arquitetura no que se refere à utilização dos conceitos de microsserviços para o desenvolvimento de aplicações *m-learning*. A resposta foi fornecida por meio de assinalamento em uma escala 0 a 5. O gráfico da Figura 30 apresenta a classificação dos participantes.

Figura 30 – Classificação de Assertividade do Modelo da Arquitetura.

Fonte: Dados da pesquisa.

Nota-se que 36,4% consideraram o modelo “totalmente correto”. O mesmo percentual considera o modelo “parcialmente correto”. Nenhum participante considerou o modelo incorreto ou parcialmente incorreto. Neste sentido, para buscar identificar as considerações que resultaram em tais marcações, na segunda pergunta os participantes eram convidados a justificar a resposta fornecida na primeira questão. O Quadro 20 apresenta a justificativa por participante.

O problema que apresentou maior número de indicações foi a questão de existir microsserviços que acessam a base de dados de outros microsserviços sem o intermédio de um terceiro microsserviço, responsável por realizar tal tarefa. Outra questão indicada que refere-se à responsabilidade do *API Gateway* que, se porventura apresentar alguma falha, ocasionará a queda de todas as aplicações que dele dependem. Esta possível limitação do *API Gateway* já havia sido alertada pelos desenvolvedores, durante a remodelagem da arquitetura, porém as vantagens obtidas se sobrepõem à desvantagem relatada. Com o objetivo de identificar outros benefícios e limitações pela adoção de tais estratégias arquiteturais, a quarta pergunta buscou identificar os *trade-offs* para os modelos apresentados.

O Quadro 21 apresenta as vantagens e desvantagens dos modelos da arquitetura, indicados pelos participantes. Em termos gerais, nota-se que as vantagens em se aderir arquiteturas baseadas em microsserviços são significativas, principalmente no contexto de reúso. Contudo, questões como complexidade para manter tal estrutura, desempenho e a centralização de controle dos

Quadro 20 – Justificativas para Classificação dos Participantes.

Participante (ID)	Classificação	Justificativa
#1	3	"Microserviços que compartilham o mesmo banco de dados é considerado uma prática que pode prejudicar a performance da arquitetura. Comunicação de um microserviço com outro é difícil de visualizar. Se ocorrer via API Gateway, essa pode não ser uma boa prática, pois pode-se reduzir a latência entre os serviços."
#2	3	"Não consegui compreender corretamente o conceito de lookup e a dependência entre Service Registry e API Gateway."
#3	4	"Microserviços não devem acessar a base de dados de outros microserviços. Deve ser incluso um microserviço que seja responsável por tal modelo, de modo a prover dados para os demais. "
#4	5	"Está bem de acordo com a ideia de microserviços. A separação em camadas permite enxergar bem as diferentes partes: Consumidores, Gateway (Broker), Provedores (serviços), e Repositórios. A presença do balanceador de carga é essencial para evitar gargalo e proporcionar escalabilidade, mas poderia ter sido melhor detalhado a forma como funciona. Ele replica os microserviços? Como isso é gerenciado? Outro detalhe é a utilização de SOAP em microserviços. Não que não seja possível, mas pode ir um pouco contra a ideia da comunicação simples. Talvez fosse interessante deixar claro a possibilidade dos repositórios usarem tecnologias diferentes (SQL e diferentes tipos de NoSQL) que é a ideia de persistência poliglota que os microserviços atendem bem."
#5	5	"A arquitetura permite um load balance dos serviços que se mostra bem útil na execução dos serviços pela aplicação. A integração entre os microserviços passam sempre pelo gateway que gerencia todas as demandas. "
#6	4	"A arquitetura proposta segue os princípios esperados, contudo, a mesma possui um único gateway é responsável por gerenciar a distribuição de requisições (outros microserviços/ websites/ mobile). Dependendo da demanda de requisições, isso pode gerar um gargalo. Para atestar a total correteza da arquitetura acredito que a mesma deveria atender a standards de atributos de qualidade. Como não está explicito se a mesma cumpre tais requisitos, prefiro adotar a nota 4."
#7	2	"Atribuída nota 2, pois não é boa prática deixar toda responsabilidade de autenticação e load balancer na API que irá comunicar com os serviços geral após autenticar e realizar o CRUD na base de dados."
#8	4	"No geral a figura 1 e 2 demonstram uma arquitetura de microserviços bem estruturada. Um único ponto que gostaria de destacar é que segundo as definições, os microserviços não devem compartilhar banco de dados, com isso, na minha visão, os quadros ilustrados nas figuras como Repository deveriam ser classificados também como um serviço."
#9	5	O participante não justificou sua resposta.
#10	5	"A arquitetura em camadas é responsável por prover uma abstração entre os módulos, expondo apenas uma interface de comunicação entre componentes de diferentes naturezas. As aplicações que se comunicam através destas camadas serão então capazes de se comunicar com esses serviços independentemente, permitindo assim que diferentes tecnologias sejam utilizadas, de forma a obter o melhor desempenho para cada um desses serviços."
#11	4	"Na parte do Client Apps geralmente a arquitetura pode ter mais camadas. Além disso, não fica claro o que é esse componente microservices nessa mesma camada. A comunicação entre microservices não fica muito clara, geralmente existe a utilização de filas e/ou streams para esse propósito. Geralmente a camada de persistência prioriza-se por ter repositórios independentes para cada microservice."

Fonte: Dados da pesquisa.

microserviços no *API Gateway* podem trazer desvantagens que ocasionarão problemas, se não tratados.

Assim, os resultados apresentaram indícios de que os modelos arquiteturais são assertivos. Entretanto, algumas limitações relacionadas ao acesso compartilhado de repositórios de dados

Quadro 21 – Vantagens e Desvantagens da Arquitetura segundo Participantes.

Participante (ID)	Vantagens	Desvantagens
#1	"Como vantagem pela arquitetura percebe-se que ela está bem dividida embora não seja possível identificar os contextos que cada microserviço está cuidando, sabemos apenas do microserviço detalhado no modelo de classes, então apesar da vantagem da divisão, uma desvantagem é identificar quais seriam esses outros microserviços. Outra vantagem é que o serviço de conteúdo está bem definido, sendo responsável por um único contexto."	Uma desvantagem percebida na arquitetura é que ela é focada em clientes do tipo web e mobile, para isso seria interessante incluir na arquitetura os conceitos de BFF (Back end for Front end - https://samnewman.io/patterns/architectural/bff/).
#2	"Favorece a escalabilidade vertical."	"O fato do DAO Genérico estar fora gera acoplamento, neste caso, o ideal seria ter pelo menos uma interface que o estende servindo de ponte."
#3	"Os serviços são separados por domínio, garantindo que - se necessário - possam ser mantidos por times distintos. Pode-se utilizar qualquer linguagem de programação para o serviço bem como qualquer mecanismo de transmissão de dados, não sendo necessário replicar o mesmo mecanismo/linguagem em todo o modelo. O service registry facilita a escalabilidade de aplicação, pois caso novas instâncias da aplicação sejam necessárias para suportar a carga estas podem ser facilmente registradas. É possível escalar apenas o microserviço que está recebendo mais carga e não toda a aplicação, o que oferece um melhor uso dos recursos computacionais da máquina. O load balancer possibilita distribuir a carga entre as várias instâncias do serviço utilizando qualquer algoritmo de distribuição que for necessário e por último a manutenção do serviço torna-se possível sem ter que desligar todo o sistema garantindo sua disponibilidade."	"Os pontos fracos são a complexidade no desenvolvimento e separação de domínios, aumento no tempo e custo de desenvolvimento dos serviços, falhas em algumas requisições devido a indisponibilidade de serviços e a dependência de versão de serviços que precisa ser bem gerenciada."
#4	"Flexibilidade: é possível facilmente criar um novo microserviço, usando a tecnologia que mais for conveniente (ferramentas certas para casos distintos) e inclui-los no sistema. O deployment se torna muito mais fácil (pois não é necessário recompilar tudo) e adequado ao time-to-market atual. Escalabilidade: é possível criar cópias de um microserviço para atender uma demanda maior, desde que haja algum mecanismo que garanta uma consistência eventual suficiente dos dados. A possibilidade de replicação de microserviços permite tempo de resposta maior em caso de alta demanda do que um sistema sem essa capacidade."	"Desempenho: As várias camadas pelas quais os dados devem passar adicionam naturalmente uma latência maior à resposta, mas são inevitáveis para que se tenha sistemas manuteníveis."
#5	"A organização e separação em serviços do sistema e a possibilidade de reuso são as principais vantagens do modelo. A camada de backend está bem separada do frontend, desta maneira pode-se trabalhar separadamente."	"Não tenho conhecimento e informações suficientes para avaliar as desvantagens do modelo."
#6	"Acredito que os tradeoffs da abordagem são genéricos ao modelo arquitetural proposto e em qualquer cenário eles tendem a se repetir. Independência entre os módulos: microserviços reforçam uma estrutura modular, isso é positivo principalmente em cenários compostos por equipes com grande número de desenvolvedores. Diversidade de Tecnologia: Microserviços possibilitam misturar diversas tecnologias diferentes em um mesmo projeto. Possibilitando que sejam extraídas as melhores características de acordo com o objetivo de cada um."	"Em contrapartida, manter uma consistência e uma comunicação entre os microserviços é difícil. (Todos os desafios de sistemas distribuídos são herdados para esse cenário). Diversidade de Tecnologia Naturalmente, manter a diversidade gera custo extra, pois necessita de uma equipe de qualificada nas mais diversas tecnologias que compõem a arquitetura desenvolvida"
#7	"Controle de acesso aos serviços consoante a autenticação, autenticação dedicada ao serviço (acesso), divisão de camadas acrescentando o micro serviço, o micro service possui arquiteturas diferentes de serviços, os repositórios são diferentes e apresentam contextos diferentes com modelos de Interfaces."	"Modelo de componentes não especifica os serviços explicitamente de acesso, não acede micro serviços mas sim o pacote de services geral o controle pode estar sendo feito pelos métodos e não services, load balancer na API Gateway."
#8	"- Decomposição dos serviços em microserviços atribui flexibilidade, escalabilidade e deploy independente para o contexto de serviço; - API Gateway é um excelente facilitador para realizar o balanceamento de carga e roteamento dos serviços."	"- A abordagem de microserviços implica em implementações mais complexas, pois se trata de sistemas distribuídos; - Como o API Gateway é o ponto de entrada dos serviços ele também pode se tornar um ponto único de falha da aplicação."
#9	"Modelo de componentes - facilita o consumo de informações/integração de diferentes banco de dados e arquiteturas."	Não informou desvantagens.
#10	"A utilização de microserviços permitem que sejam utilizadas diferentes tecnologias e padrões para cada um dos serviços, independentemente, garantindo uma maior flexibilidade na escolha destas."	"Uma desvantagem é que a gestão destes fica delegada ao API Gateway, que não apenas se mostra como um controlador inchado, mas também como um potencial ponto de fragilidade para a arquitetura em si."
#11	Não informou vantagens.	"O modelo de classes você pode ter o risco de ter um grande monolítico na camada de backend o que dificulta por exemplo deploys independentes e serviços que podem trabalhar de forma independente de forma resiliente. Por outro lado, criar uma arquitetura como a do modelo de componentes pode fazer com que a complexidade para executar uma jornada simples possa passar por diversos microserviços."

Fonte: Dados da pesquisa.

entre os microserviços e o acúmulo de funcionalidades no modelo de classes foram apontadas, bem como a responsabilidade do *API Gateway*.

Para condução de uma análise na perspectiva de especialistas da indústria, os desenvolvedores que auxiliaram no processo de concepção dos modelos arquiteturais foram consultados.

Eles indicaram que o uso tanto de serviços quanto microserviços em uma mesma arquitetura é realidade no contexto de SaaS (CERNY; DONAHOO; PECHANEC, 2017; CERNY; DONAHOO; TRNKA, 2018). Quanto ao *API Gateway*, como foi mencionado anteriormente, a vantagem em seu uso sobrepõe-se em relação à desvantagem, mas um modo de contornar tal problema é dividir a carga de responsabilidades entre outros componentes similares, que funcionem como redirecionadores de requisições. Essa estratégia pode ser chamada também de BFF (PERERA; PERERA, 2018; HIPPCHEN *et al.*, 2017).

A ideia de um *back end* para *front end* tem como premissa a dificuldade em desenvolver APIs e integrar nelas diferentes e numerosos microserviços, que muitas vezes são desenvolvidos por terceiros, possuindo especificações e comportamentos diversos e não previstos. Nesta perspectiva, um BFF auxilia na organização de arquiteturas de microserviços para coordenar as funcionalidades entre os diversos microserviços e consequentemente, diversos sistemas. Quando um *app* solicita dados, tal solicitação é traduzida por meio do BFF em uma camada geral e seu componente principal denominado *shim*, que por sua vez se comunica com microserviço previamente registrado (PERERA; PERERA, 2018; SANDOVAL, 2017).

A Figura 31 apresenta o modelo arquitetural de componentes considerando um BFF. Nota-se que na camada que antes correspondia ao *API Gateway*, há diferentes componentes que servem a cada tipo de cliente (inquilino). Deste modo, se o *Mobile shim* falhar, por alguma inconsistência, o cliente *Websites* continuará em funcionamento, visto que estão isolados. Logo, o risco em inviabilizar a utilização dos serviços para todos os clientes é reduzido.

Finalmente, quanto à integração de diferentes *features* e consequentemente, o acúmulo de suas funcionalidades em único micro serviço, isso depende da estratégia e necessidade de reúso. Por exemplo, se uma *feature* específica passa a ser requerida também para outros microserviços, esta seria candidata a ser isolada do componente que integra, tornando-se independente e favorecendo seu reúso. Esta situação não é o caso do microserviço apresentado no diagrama de classes, que buscou representar uma estrutura básica que servirá de modelo aos demais microserviços, e que representa a *feature* para apresentação de conteúdos educacionais (e.g., vídeo, texto, texto com figuras, arquivo do tipo pdf).

Destarte, com base no modelo arquitetural considerando a substituição do *API Gateway* por componentes que atendam determinados tipos de clientes, as fases de realização de domínio e testes de domínio foram conduzidas.

5.6.4 Realização de Domínio e Testes de Domínio

Na realização do domínio (*domain realisation*), as *features* da linha são desenvolvidas, tendo como base os artefatos provenientes dos sub-processos anteriores.

Segundo o arcabouço SPLE, cada *feature* é desenvolvida em um ou mais componentes, por meio de quatro modos diferentes: (1) desenvolvidos na equipe (*make*), fornecendo maior

controle sobre o componente, visto que foi planejado e desenvolvido pela equipe mantenedora da linha; (2) comprado (*buy*) como um componente de prateleira (*off-the-shelf*), ou seja, é comprado já pronto e não segue as especificações de projeto, sendo muitas vezes mais barato do que desenvolver; (3) reutilizado (*mine*) de um software já desenvolvido pela equipe mantenedora da LPS, neste caso pode ser simples ou não adaptar tal componente para seu reuso; e (4) desenvolvido por terceiros (*commission*), ou seja, contrata-se uma empresa para se desenvolver o componente desde o início, o que pode gerar uma lacuna entre as especificações projetadas e o modo como foi desenvolvido, já que se trata de uma equipe terceirizada.

As indicações do arcabouço (*make/buy/mine/commission*) assemelham-se com as propostas por Krueger (2002). Nesta perspectiva, como já discutido no Capítulo 3, optou-se pelo desenvolvimento das *features* do zero, considerando um grupo inicial de *features* candidatas, identificadas nos resultados do questionário conduzido com professores na avaliação do catálogo de requisitos (Capítulo 4). Tal decisão ocorre pela indisponibilidade de software para a extração e a necessidade de esforços para adaptação das *features* das poucas soluções que possibilitavam reuso de seus artefatos.

Em relação ao sub-processo de testes de domínio, este foi conduzido sempre após o desenvolvimento de um novo componente. Segundo as diretrizes do arcabouço SPLE, esse sub-processo é responsável por garantir a qualidade dos ativos e dos produtos da linha (POHL; BOCKLE; LINDEN, 2005). No contexto de LPS, o maior problema no sub-processo de testes envolve as variabilidades, uma vez que a resolução de pontos de variação levam a configurações diferentes e possíveis novos defeitos para uma configuração ainda não testada. Isso impede, por exemplo, a execução da abordagem de testes de força, nos quais todas as configurações e componentes são testados em todas as suas perspectivas (POHL; BOCKLE; LINDEN, 2005).

Assim, testes unitários e de integração foram conduzidos sempre que um novo componente foi desenvolvido; contudo, testes exaustivos não foram conduzidos. A condução de testes unitários foi integrada ao processo de desenvolvimento baseando-se no ciclo e conceitos do desenvolvimento dirigido a testes (TDD) (KARLESKY *et al.*, 2007). Logo, é importante destacar que as atividades do TDD foram adotadas para se definir o ciclo de atividades no processo de desenvolvimento da LPS, o que implica-se em dizer que as atividades do TDD foram adaptadas e muitas modificadas em sua completude. Os testes realizados concentraram-se na técnica funcional, na qual o sistema é considerado uma caixa preta, sendo testado por meio de entradas e avaliação das saídas pelos desenvolvedores, na perspectiva de usuários finais. Os testes utilizados foram em relação a funcionalidades específicas, e.g., criar uma configuração, criar um conteúdo para uma aplicação.

Para dinamizar os sub-processos de realização de domínio e testes de domínio, conceitos provenientes também da metodologia intitulada programação extrema (XP), além do TDD, foram adotados. Esses conceitos são refletidos no diagrama de atividades da Figura 32.

As atividades nos sub-processos de realização de domínio e testes de domínio foram

integradas e se iniciam no **planejamento de lançamentos**. Nesta tarefa são planejados os lançamentos de novas *features*, utilizando como entrada os requisitos considerando mais importantes. Na tarefa seguinte, os **planos de interação** mantêm o gerenciamento do que será desenvolvido, considerando prazos reduzidos (e.g., dias). Se tais planos de interação forem concluídos após uma **interação da fase de desenvolvimento**, esses passam à fase de **testes de integração**, que ao não se identificar defeitos, leva ao **lançamento final** do componente em questão. Em caso de não finalização após a interação, o planejamento de interação é revisto e atualizado para que o ciclo se repita até que uma versão aceitável do componente e de sua respectiva **feature** seja concluída.

Quando executada à cada interação, a **fase de desenvolvimento** leva à atualização dos **modelos da LPS**, incluindo documentos e outros registros dos componentes criados. Após essa atualização, conduz-se as atividades adaptadas do TDD, que executa **testes de falha** para identificar defeitos. Se encontrados, ou seja, se os testes passarem, a **refatoração** da funcionalidade desenvolvida é conduzida com o objetivo de corrigir os defeitos encontrados. Se novos defeitos não forem encontrados, obtêm-se uma **nova funcionalidade**, seguindo para a **integração contínua**, automatizada na ferramenta GitLab³, o que dá continuidade ao ciclo de desenvolvimento. Logo, destaca-se que apenas o ciclo de teste e seus conceitos definidos no TDD foram adotados, e não sua metodologia como um todo (KARLESKY *et al.*, 2007; SOMMERVILLE, 2010). Nesta perspectiva, adotou-se testes unitários que são realizados ao final da criação do modelo do respectivo artefato e mediante o término de seu desenvolvimento, permitindo a identificação de defeitos, para a posterior condução de testes de integração e aceitação do componente a integrar as aplicações móveis.

Com a definição das atividades a serem conduzidas nesses sub-processos, seguiu-se para a escolha das tecnologias a serem adotadas. Para o desenvolvimento do *backend*, ou seja, a lógica para aplicações web mantida nos servidores, adotou-se o uso do arcabouço de aplicação web *express* para *Node.js*⁴ (TILKOV; VINOSKI, 2010; WILSON, 2018). *Node.js* agiliza o processo de desenvolvimento de servidores web, visto que se torna responsável, principalmente, por: (i) tratar do roteamento por meio de URLs, (ii) gerenciar sessões por com *cookies*, (iii) analisar solicitações recebidas (como dados de formulário ou *JavaScript Object Notation* (JSON)), e (iv) impedir requisições incorretas. Ademais, por ser desenvolvido para *Node.js*, outras bibliotecas podem ser incluídas. Este é o caso da *mongoose*, uma biblioteca que propicia soluções baseadas em esquemas para modelar os dados da aplicação para o banco de dados não relacional MongoDB; e a biblioteca *JSON Web Token* (JWT) que possibilita meios de realizar requisições HTTP mais seguras, entre aplicação e servidor(es).

Em relação ao *front end*, que se refere às interfaces do cliente, optou-se, em um primeiro momento, pelo desenvolvimento considerando o arcabouço *Angular*⁵ que posteriormente foi

³ <<https://gitlab.com/>>

⁴ <<https://nodejs.org>>

⁵ <<https://angular.io/>>

substituído pelo *React*⁶. *Angular* foi escolhido pela identificação da adoção de tal tecnologia no contexto de indústrias de desenvolvimento de software, por ser um arcabouço JavaScript conhecido e por seu mantenedor ser a Google. Porém, com o surgimento de diferentes problemas de compatibilidade com pacotes de terceiros, optou-se pela substituição pelo *React*, que é mantido pelo Facebook. A substituição pelo *React* ocorreu devido o mesmo também ser um dos arcabouços JavaScript mais utilizados, e por possuir suporte a biblioteca *React Native*, que permite a criação de aplicações móveis com a integração de pacotes com funcionalidades específicas das plataformas mais conhecidas (i.e., Android e iOS) (GACKENHEIMER, 2015).

As Figura 33 e 34 apresentam a interface desenvolvida para a ferramenta de geração das configurações e aplicações da LPS, denominada TANGRAM, considerando o *framework Angular*.

Figura 33 – Página Inicial da TANGRAM desenvolvida com Angular.

Fonte: Elaborada pelo autor.

⁶ <<https://reactjs.org/>>

Figura 34 – Página de Criação da Aplicação com Angular.

Fonte: Elaborada pelo autor.

Já as Figuras 35 e 36 apresentam duas das interfaces desenvolvidas para TANGRAM, considerando *React*. A Figura 35 apresenta a interface principal para a criação da configuração, de aplicações e solicitação de novas funcionalidades. Já a Figura 36 apresenta o painel de controle para a criação da aplicação.

Figura 35 – Página Inicial da TANGRAM desenvolvida com React.

Fonte: Elaborada pelo autor.

Figura 36 – Página do Painel de Controle de Criação da Aplicação com React.

Fonte: Elaborada pelo autor.

A ferramenta para criação das aplicações permite a seleção de *features* que deverão estar presentes na aplicação. Há ainda o cadastro da *features* desenvolvidas, de modo a permitir o gerenciamento das mesmas. Dentre as informações, destacam-se as rotas de acesso a determinados microserviços. Deste modo, buscou-se facilitar o gerenciamento das *features* desenvolvidas e a facilitação da inclusão de novas *features*.

Nessa perspectiva, notou-se que a adoção de *React*, facilita a integração de novas *features* em um nível mais satisfatório do que a versão anterior da ferramenta, desenvolvida em *Angular*. Com *Angular*, cada componente deveria ser desenvolvido e integrado ao conjunto dos demais componentes. O isolamento de cada um dificultava o processo de evolução da linha, já que o código dos artefatos comuns entre os produtos necessitavam de alterações mais complexas.

Na perspectiva de *React*, os componentes podem ser importados como um elemento a compor outros componentes, sendo independentes e mais facilmente integrados. Outra vantagem é a adoção de *Redux*⁷, uma biblioteca JavaScript de código aberto que permite gerenciar o estado do aplicativo. Tal estado corresponde ao conjunto de dados transmitidos aos componentes. Com *Redux*, toda a aplicação tem acesso aos dados gerais da aplicação, ou seja, acesso ao estado atual dos mesmos, o que permite melhor consistência e dinamismo para se alterar, consistir e salvar tais estados em base de dados, ou simplesmente, para garantir a realização das ações com base nos dados compartilhados. Apesar de serem transparentes para toda a aplicação, o estado compartilhado pode ser mantido por *reducers* exclusivos para cada componente, ou de

⁷ <<https://redux.js.org/>>

modo isolado. Um *reducer* é encarregado de lidar com todas as ações no contexto do *Redux*, e.g., algum componente pedindo para alterar algum dado (GACKENHEIMER, 2015).

Uma última consideração se deve ao uso de *Node.js* e JavaScript. Com a existência de bibliotecas e componentes de terceiros para auxiliar nas tarefas de testes das aplicações e também para o gerenciamento e atualização de documentações de tais aplicações, tem-se uma redução de esforços para permitir um melhor gerenciamento dos diversos componentes desenvolvidos e reutilizados, no contexto da LPS. Isso apoia a sua atualização e gerenciamento, realizado no sub-processo de gerenciamento de produtos, discutido inicialmente neste capítulo. A seguir são discutidas as contribuições da arquitetura e tecnologias adotadas, considerando-se as questões educacionais relacionadas ao domínio de ensino e aprendizagem de fundamentos de programação.

5.6.5 Contribuições da Arquitetura no Domínio de Ensino e Aprendizagem de Programação

O primeiro modelo da arquitetura da LPS adotava o uso apenas de serviços. Considerando que estes serviços agrupariam um número maior de *features*, os esforços no processo de gerenciamento e manutenção para as mesmas seria maior. Isso implica ainda em um maior número de variabilidades em nível de métodos e trechos de código. *Features* de *feedback* e resultado, por exemplo, estariam todas integradas a serviços, a nível de código.

Considerando a evolução que integra também microserviços às especificações da LPS, reduz-se a quantidade de *features* integradas a outros componentes, sejam estes serviços ou microserviços.

Esse padrão auxilia ainda na integração de novas *features*, juntamente com a adoção de *React*, que permite menor acoplamento e desenvolvimento de funcionalidades restritas a componentes. No contexto da presente LPS, que considera apenas um grupo seletivo de *features* para o desenvolvimento de seus primeiros produtos, esse fator é determinante para garantir a evolução da LPS.

Outra contribuição relacionada à proposta arquitetural refere-se à possibilidade de melhor dividir e, assim, possibilitar a investigação da combinação de diferentes *features* e os problemas de ensino e aprendizagem de programação, herdados do catálogo de requisitos *m-learning*, utilizados na sua concepção.

Componentes codificados como microserviços podem concentrar facetas específicas de um requisito, que como já discutido anteriormente, em alguns casos, especificam grupos de funcionalidades similares. Adicionalmente, pode-se especificar melhor os problemas de programação que aquele componente e sua respectiva *feature* buscam mitigar.

Considere, por exemplo, que uma *feature* de conteúdo de aprendizagem, responsável

por apresentar um vídeo a um aluno, está integrada a uma *feature* de *feedback*, que por sua vez está configurada para fornecer informações aos professores e alunos. A integração de tais componentes ocorre de modo centralizado, em um componente de importação de classes (considerando *React* e JavaScript). Deste modo, cada um desses componentes podem fornecer dados e meios específicos de registrar ou não suas interações e possíveis informações para o contexto educacional. Neste exemplo, a finalização da visualização do vídeo pelo aluno pode gerar uma notificação imediata para o professor ou ser registrada, para posterior uso. Tais escolhas implicam ainda na redução ou aumento no consumo de serviços de acesso à Internet.

Em termos de questões que envolvem codificação, ainda que não perceptível para os alunos, *features* específicas de monitoramento e desempenho de aprendizagem podem ser integradas aos componentes que refletem as *features* de mecanismos de programação, permitindo meios mais precisos de acompanhamento pelos professores. Além disso, o confronto dos dados pode permitir a identificação de nuances não perceptíveis quando as atividades de codificação ocorrem em sala, podendo ainda serem auxiliadas com mecanismos de *feedback* automatizado, para instruir melhorias nas limitações identificadas no processo de aprendizagem dos alunos.

Uma outra questão importante é a não divisão das aplicações móveis, como será melhor discutido no próximo capítulo. Ao optar por desenvolver e entregar a aplicação já prevendo possíveis novas *features*, o processo de resolução para especificação de uma aplicação, que considera as configurações e aplicações definidas na TANGRAM, torna-se mais fácil e menos custoso do que se desenvolvido manualmente (o que também é possível). Ademais, considerando o processo de atualização de aplicativos, geralmente gerenciados pelos serviços da própria plataforma do dispositivo móvel, este processo é transparente e facilitado. Imagine, por exemplo, a necessidade de se alterar uma *feature* de apoio ao ensino e aprendizagem relativa a um fórum de mensagens apenas para um grupo seletivo de usuários. Ao disponibilizar e enviar tal atualização apenas para os mesmos, quando a mesma funcionalidade for integrada a outro grupo, e outra aplicação, a atualização deverá ser entregue também a estes. Considerando que uma *feature* foi desenvolvida de modo a alterar a estrutura de registros de dados, que até então não considerava essa atualização da *feature* de fórum, isso resultará em erros para os que não possuem a aplicação atualizada.

Quando comparadas as aplicações da LPS aos software identificados no MS (Capítulo 3) e os 33 estudos considerados para a concepção do catálogo de requisitos, nota-se a vantagem em se adotar uma metodologia de reuso como a LPS. De fato, pode-se gerar um número maior de aplicações que adotem diferentes *features*, atendendo a um possível grupo maior de problemas no domínio.

Outro ponto importante é o impulsionamento na utilização de dispositivos móveis no ensino e aprendizagem de programação ainda identificados como incipientes, conforme resultados do MS conduzido. A concepção das aplicações utilizando-se do conceito de variabilidades permite que tanto os recursos específicos de hardware dos dispositivos possam ser explorados,

quanto novos meios possam ser desenvolvidos e adaptados de modalidades diferentes, como *e-learning*. A necessidade da adaptação de *features* de software para *desktop*, por exemplo, faz-se necessária, principalmente pela incipiência identificada no contexto de *m-learning*. Outra questão é que os alunos, já acostumados com a popularização dos dispositivos móveis e seu uso em tarefas rotineiras, podem se sentir mais motivados a utilizar a plataforma móvel como meio de aprender a programar.

Finalmente, o favorecimento da concepção de uma LPS permite a combinação de diferentes *features* dos dispositivos móveis, com os diferentes elementos e *features* relacionados ao domínio educacional adotado, como as apresentadas na Figura 37. Nela é possível observar as categorias de *features* que são provenientes do catálogo de requisitos (Capítulo 4), para melhor permitir a relação de requisito–serviço/microserviços. Cada retângulo provê e concentra diferentes *features* que poderão ser incrementadas na medida que novos artefatos seja adicionados a LPS.

Figura 37 – Itens Candidatos a Variações no Contexto dos Produtos da LPS.

Fonte: Elaborada pelo autor.

Tais *features* podem variar de acordo com as necessidades, sendo integradas às aplicações da linha, por meio da ferramenta TANGRAM, nos processos de criação da configuração e definição da aplicação e seus módulos educacionais.

Sem um auxílio textual e sistemático para a criação das aplicações, mesmo que realizadas por meio de uma ferramenta, o produto final pode não atender às expectativas, principalmente dos professores, visto que a LPS busca auxiliar, neste primeiro momento, professores e alunos de nível superior, na mitigação de problemas no ensino e aprendizagem de fundamentos de programação. O apoio fornecido pela ferramenta TANGRAM é melhor especificado no próximo capítulo.

5.7 Considerações Finais

Este capítulo apresentou o estabelecimento da arquitetura da LPS e o processo de desenvolvimento da mesma. O arcabouço SPLE foi adotado como guia para auxiliar nos diferentes processos de concepção relacionados à Engenharia de Domínio.

Duas versões da arquitetura foram propostas, considerando avaliação com as partes interessadas e desenvolvedores da indústria de software. A proposta final da arquitetura objetivou garantir o estabelecimento da LPS com a integração da ferramenta TANGRAM, além de buscar definir seus componentes com maior coesão e com baixo acoplamento, a fim de permitir a geração de aplicações móveis educacionais que possam integrar diferentes meios de mitigar problemas no ensino e aprendizagem de programação. Além disso, buscou-se promover o uso dos dispositivos móveis como plataforma de ensino e aprendizagem.

Com o estabelecimento da arquitetura da LPS, o próximo capítulo discute a ferramenta TANGRAM, que apoia a geração de aplicações e sua avaliação de usabilidade e aceitação com professores. Discute-se, ainda, questões técnicas quanto às aplicações móveis geradas na LPS, em especial, uma *feature* concebida por meio de um catálogo de gestos, a ser integrada nas aplicações móveis como alternativa aos meios tradicionais de interação desses dispositivos, nos processos de ensino e aprendizagem de programação.

A FERRAMENTA TANGRAM

6.1 Considerações Iniciais

De modo geral, a utilização da abordagem de LPS requer esforços técnicos significativos na criação e evolução de seus produtos (POHL; BOCKLE; LINDEN, 2005). No contexto educacional, em especial, no ensino e aprendizagem de programação, há ainda a necessidade de se criar os diferentes elementos que serão apresentados por cada *feature* da LPS, formando os módulos educacionais, e.g., para a *feature* de conteúdo, responsável por apresentar os conteúdos educacionais, é necessário criar tais conteúdos por meio de diferentes modos de representação, como figuras, textos, vídeos. Nesse sentido, a ferramenta TANGRAM apoia tanto a criação e gerenciamento das *features* da LPS, quanto à criação dos diferentes módulos educacionais disponibilizados nas aplicações geradas. Esse apoio é complementado pela possibilidade de consulta aos requisitos e seus respectivos problemas de programação que visam mitigar. Espera-se, assim, criar uma infraestrutura que, combinada com a LPS e a ferramenta, forneça meios de investigar soluções mais precisas para tratar dos problemas no domínio de programação, discutidos no decorrer deste trabalho.

Neste capítulo, a ferramenta TANGRAM é discutida, apresentando uma visão geral de suas principais funcionalidades (Seção 6.2) e a avaliação de usabilidade e de aceitação a qual a mesma foi submetida (Seção 6.3). Adicionalmente, discutem-se algumas das *features* candidatas, relacionadas às aplicações móveis (Seção C.1), em especial, o projeto, desenvolvimento e avaliação de um conjunto de gestos de interação, denominado catálogo de gestos (Seção 6.4) integrado a tais características da linha, para permitir interações alternativas aos aprendizes na realização de tarefas de programação a serem disponibilizadas nos produtos da LPS. A escolha de um conjunto de gestos de interação visa identificar se o uso das telas sensíveis ao toque podem melhorar a motivação dos alunos.

6.2 Visão Geral da Ferramenta

A ferramenta TANGRAM (*an infrasTructure to support the development of mobile leArning applicatioNs for the teachinG and leaRning of progrAMming fundamentals*) é uma ferramenta que, juntamente com a LPS e seu núcleo de artefatos, compõem uma infraestrutura para permitir a investigação de soluções para problemas no ensino e aprendizagem de programação por meio de aplicações *m-learning*. Dentre os principais objetivos no estabelecimento desta ferramenta, elenca-se:

- Apoio ao gerenciamento das *features* da LPS, permitindo o cadastro das *features* e demais artefatos que formam o núcleo da LPS, o que facilita também a evolução da LPS, visto que são registradas informações técnicas e de suporte para a condução de tal evolução;
- Apoio à criação das configurações das aplicações, que se refere à seleção das *features* que estão presentes na LPS, a integrar as aplicações móveis educacionais;
- Apoio à criação das aplicações que, considerando a configuração previamente criada, permite a definição dos módulos educacionais correspondentes às *features* selecionadas, sendo estas aplicações disponibilizadas e utilizadas pelos alunos; e
- Apoio ao acompanhamento das interações dos alunos com os diferentes módulos educacionais disponíveis nas aplicações.

O apoio fornecido ao gerenciamento da LPS busca facilitar a identificação das diferentes *features* disponíveis, bem como o processo de evolução da LPS e possíveis atualizações das interfaces da ferramenta ao se inserir uma nova *feature*. Dentre as principais informações mantidas para as *features* que remetem aos componentes de software, estão: interfaces e classes que permitem importação (EcmaScript 6), métodos pertinentes às classes e suas assinaturas, e APIs utilizadas e disponibilizadas pelos componentes.

Para que uma nova *feature* possa integrar um dos nove grupos de *features* existentes, (i.e., os mesmos definidos no catálogo de requisitos), permitindo sua utilização nas aplicações, é necessário importar tal *feature* à respectiva interface de seleção pelo usuário. Adicionalmente, quando necessário, deve-se atualizar a interface de criação da aplicação e do módulo educacional da qual esta nova *feature* fará parte. Logo, com a possibilidade de consulta e registro das *features* preexistentes, é possível direcionar os esforços na modificação e adaptação de componentes específicos que irão integrar tais *features*.

Já no que se remete ao apoio à criação das configurações e das aplicações da LPS, busca-se reduzir os esforços técnicos dos professores para permitir a criação dos produtos da linha que, sem a ferramenta, ocorreria de modo *ad hoc* e por desenvolvedores. A Figura 38 apresenta a tela na qual é feita a seleção das *features*. Essa seleção ocorre por meio de resposta a

perguntas pertinentes a cada um dos oito grupos de *features*. Ao indicar “sim” para uma pergunta, o respectivo grupo de *features* será integrado à configuração.

A decisão para a criação de uma configuração prévia, obrigatória para se definir uma aplicação, foi estabelecida no sentido de permitir que uma mesma configuração possa ser compartilhada entre usuários da ferramenta ou reutilizada pelo usuário que a criou. A definição de uma configuração auxilia, ainda, no processo de criação dos diferentes módulos educacionais que podem ser considerados as instâncias dos grupos de *features* selecionados na configuração.

Figura 38 – Tela Seleção das *Features* para Criação de Configuração.

Fonte: Elaborada pelo autor.

Contudo, o principal objetivo no apoio à criação das aplicações é conduzir tal criação considerando a possível mitigação ou solução dos problemas de ensino e aprendizagem de programação. Para isso, o catálogo de requisitos *m-learning* foi integrado à ferramenta, de modo a permitir sua consulta e seu uso no processo de criação das aplicações.

Nesta perspectiva, a versão atual da ferramenta TANGRAM dá ênfase para a criação de configurações e aplicações que apoiam o ensino formal. O ensino informal é considerado no catálogo de requisitos *m-learning*, possuindo diversos requisitos que o apoiam, e.g., realização de atividades sem a obrigatoriedade de acompanhar ou finalizar uma lição, a inexistência de credenciais de acesso. Contudo, as categorias de problemas de programação identificadas e discutidas anteriormente se concentram no processo de ensino e aprendizagem formais.

Os modelos de *features* (Figuras 39 e 40) apresentam os grupos de *features* educacionais

que podem estar presentes em uma aplicação *m-learning* para o ensino formal (Figura 39) e informal (Figura 40), e as restrições entre os diferentes módulos educacionais que as compõe.

Figura 39 – Modelo de Características dos Módulos Educacionais para o Ensino Formal.

Fonte: Elaborada pelo autor.

Figura 40 – Modelo de Características dos Módulos Educacionais para o Ensino Informal.

Fonte: Elaborada pelo autor.

São por meio de tais grupos das *features* (Figuras 39 e 40) que a ferramenta provê meios para acompanhar os alunos nas diferentes interações realizadas nas aplicações *m-learning* criadas na LPS.

O processo de ensino e aprendizagem formal, só é possível quando os alunos são devidamente identificados e suas interações registradas. Neste contexto, nas aplicações *m-learning*, a identificação dos alunos ocorre pela atribuição de credenciais de acesso (usuário e senha). É por meio destas credenciais que os *feedbacks* de interação com a interface e de interações com os módulos educacionais serão fornecidos, além de indicar-se os resultados tanto para os professores quanto para os alunos, permitindo que possam, então, direcionar seus esforços para a superação de possíveis limitações no ensino e aprendizagem, justificando a obrigatoriedade de tais *features*.

As *features* de usuário são necessárias, ainda, para que as *features* de avaliação, de apoio ao ensino e aprendizagem e de monitoramento e desempenho de aprendizagem funcionem corretamente. Essas *features*, ainda que possam estar presentes em aplicações que apoiem o ensino informal, são partes integrantes dos processos de ensino e aprendizagem na modalidade formal.

Os alunos identificados nas aplicações, ao realizarem uma atividade educacional integrada com a *feature* de avaliação, recebem uma nota atribuída às respostas de suas atividades. O resultado obtido será fornecido, por sua vez, por *features* específicas de *feedback* e resultados. Já as *features* de apoio ao ensino e aprendizagem necessitam da identificação dos alunos para permitir troca de mensagens entre estes e com seus professores, o envio e recebimento de respostas para dúvidas, entre outras. Finalmente, o monitoramento e desempenho de aprendizagem podem ser enriquecidos com a *feature* de avaliação, ou utilizar apenas a quantificação de acertos e erros obtidos na realização de atividades de aprendizagem para fornecer meios de se acompanhar o desempenho dos alunos pelos seus professores que podem, então, fornecer apoio necessário a superar limitações e resolver problemas de aprendizagem de seus alunos.

Já o ensino informal apresenta especificidades significativas, uma vez que o aluno se torna o principal responsável pela aquisição de conhecimento. Neste sentido, os software educacionais devem fornecer mecanismos para auxiliar os alunos. Logo, não há mais a obrigatoriedade de se manter credenciais de acesso, como pode-se notar na Figura 40, bem como, em certos casos, não é necessário fornecer *feedback* e resultados, quando os alunos objetivam realizar atividades práticas livres.

A seguir, cada oito grupos de *features* (Figuras 39 e 40), que correspondem a oito das nove categorias de requisitos adotadas no catálogo (Capítulo 3), são descritos no que tange a sua disponibilização e configuração na ferramenta. A nona categoria corresponde aos requisitos não funcionais.

- **Conteúdo de Aprendizagem:** referem-se ao grupo de *features* que disponibilizam conteúdos educacionais que permitirão a aprendizagem de conceitos e teorias sobre tópicos determinados pelos professores, além de permitir a recapitulação de conceitos apresentados em sala de aula. Os conteúdos educacionais são compostos por um ou mais tipos de representação, sendo os principais: arquivos do tipo *Portable Document Format* (PDF), texto, figuras, links e vídeos. Estes podem ser integrados de modo a compor unidades de aprendizagem mais extensas, porém, conforme indicado por teorias educacionais e no catálogo de requisitos *m-learning*, tais conteúdos devem ser curtos. Por exemplo, vídeos não devem ultrapassar 15 minutos de duração. Ademais, no cadastro dos conteúdos, deve-se indicar o tópico no qual tal conteúdo integrará, para sua devida apresentação e acesso na estrutura da aplicação.

As *features* de monitoramento e desempenho de aprendizagem, bem como *feedback*

e avaliação educacional, quando selecionadas e integradas aos conteúdos, permitem o monitoramento e identificação das interações realizadas em tais conteúdos, e.g., rolagem da tela até o final do texto, visualização do vídeo até seu término, tempo de permanência na interface de conteúdo. Assim, é possível fornecer apoio aos professores no processo de avaliação, juntamente com os demais módulos educacionais.

- **Atividades de Aprendizagem:** as atividades são compostas por uma ou mais questões. As questões podem ser criadas durante a criação de uma atividade ou antecipadamente.

No processo de criação de uma questão, uma ou mais *tags* devem ser selecionadas. Cada uma dessas *tags* referem-se às categorias de problemas no ensino de programação (Figura 41).

Figura 41 – Página de Criação das Questões e Seleção das *Tags*.

Alternativa 3: `cout<< (altura>= 170.0 ? "Aprovado": "Reprovado");`

Alternativa 4: `cout<< (if (altura<= 170.0) << "Aprovado": << "Reprovado");`

Alternativa 5: `cout<< "Reprovado": cout<< "Aprovado"? if (altura<= 170.0);`

Tag(s) da Questão:

Incluir: Problemas no Ensino e Aprendizagem de Programação:

- ☐ P0 - Geral
- ☒ P1 - Aprendizagem de conceitos de programação.
- ☒ P2 - Aplicação de conceitos de programação.
- ☒ P3 - Compreensão de programas.
- ☐ P4 - Fatoração e refatoração de programas.
- ☐ P5 - Motivação
- ☐ P6 - Dificuldades relacionadas aos professores.

Atualizar Cancelar

Fonte: Elaborada pelo autor.

Apesar de ser uma estratégia simples, quando integrada a uma *feature* de monitoramento e desempenho de aprendizagem, é possível identificar o problema que o aluno enfrenta. Por exemplo, o professor cadastra uma questão que tem como objetivo identificar a compreensão de um programa. Se a questão for objetiva, com múltipla escolha, e o aluno acerta parcialmente a questão, isso indica algum possível problema em relação à compreensão do que foi ensinado. Caso seja uma questão objetiva com resposta única, ou subjetiva (corrigida pelo professor), e tiver sido respondida incorretamente, as *tags* relacionadas indicam ao professor em quais grupos de problemas o aluno apresentou dificuldades de aprendizagem, permitindo ao professor tomar decisões ou traçar estratégias para investigar o motivo do erro e mitigar a limitação do aluno. Logo, as *tags* auxiliam no processo de melhora nos processos de ensino e aprendizagem.

Finalmente, tal como nos conteúdos de aprendizagem, ao selecionar *features* como *feedback* e resultados, monitoramento e desempenho e avaliação educacional, que se complementam, uma atividade pode ser definida como avaliação e, sendo esta somativa, por exemplo, pode compor uma nota final. Essa nota pode ser monitorada pelo professor e, posteriormente, pode ser utilizada como *feedback* ao aprendiz e, até mesmo, para possíveis mecanismos na aplicação que possam vir a apoiar o processo de mitigação de limitações na retenção e compreensão de tópicos de estudo de modo automático, e.g., tutor inteligente.

- **Feedback e Resultados:** as *features* de *feedback* dividem-se em dois grupos: *feedback* da aplicação, em relação às interações dos usuários com a mesma (professores ou alunos); e *feedback* no processo de ensino e aprendizagem. Este último poderá ser aprimorado se outras características, como monitoramento e desempenho de aprendizagem e avaliação educacional tiverem sido selecionadas. Na ausência da mesma, este *feedback* exibirá a resposta correta das questões respondidas, ou mensagens personalizadas e definidas pelo professor no cadastro das atividades, de questões que as compõem, e dos conteúdos educacionais.

Quando integrado com outras *features*, tal *feedback* será aprimorado, exibindo tempo de acesso; desempenho de aprendizagem considerando outras atividades realizadas e também o término das lições (conteúdos educacionais); evolução da aprendizagem; auxílio na retomada de tópicos que apresentarem dificuldades considerando, para isso, os resultados das atividades, as *tags* e o conteúdo educacional relacionados e cadastrados para um mesmo tópico.

- **Monitoramento e Desempenho de Aprendizagem:** estas *features* realizam o monitoramento das interações dos alunos em relação às respostas das atividades e visualização de conteúdos educacionais. O desempenho está relacionado ao resumo dos resultados das respostas corretas e incorretas das questões ou, quando na seleção da *feature* de avaliação educacional, os resultados das avaliações realizadas. Ambos são mantidos e podem ser visualizados pelos professores em área específica, não sendo exibidos para os alunos se a *feature* de *feedback* e resultados forem configuradas para não exibirem tais informações. Quando configuradas para fornecer tal informação, o *feedback* e resultados permitem que o aluno saiba de seu desempenho e busque sanar dúvidas com o professor ou revise um determinado tópico que já tenha estudado.
- **Mecanismos para Ensino de Programação:** em tal grupo concentram-se mecanismos que podem ser um editor de código, depurador, compilador, entre outros, que são utilizados em diferentes tipos de questões que compõem uma atividade educacional, ou para a condução de atividades livres. Apesar de algumas dessas *features* não serem utilizadas no contexto de fundamentos de programação, a necessidade de apoiar o ensino informal e atividades que utilizem tais recursos levaram à disponibilização dos mesmos.

O isolamento de *features* de programação em um grupo específico foi proposta para permitir e facilitar o processo de integração de outros domínio educacionais na possível evolução da linha. Apesar de ter sido desenvolvida considerando requisitos específicos do domínio de programação, novos domínios educacionais podem ser investigados a substituir ou integrar a LPS, favorecendo a interdisciplinaridade.

Finalmente, por estarem integrados às atividades, essas *features* também podem ser utilizadas por outras *features*, como *feedback* e resultados, monitoramento e desempenho de aprendizagem e avaliação educacional.

- **Avaliação Educacional:** a avaliação educacional é uma *feature* integrada às atividades de aprendizagem, permitindo que a aplicação forneça um conceito ou nota para as questões das atividades que forem aplicadas e respondidas. Serve ainda, como meio de prover dados para o monitoramento e desempenho de aprendizagem. É importante destacar que as atividades e suas questões permitem identificar número de erros e acertos, sem a necessidade de inclusão deste conjunto de características. Contudo, a inclusão da avaliação aprimora tal processo, incluindo o registro, cálculo de média, relação conceito e tempo para resposta de uma atividade, entre outros.
- **Apoio ao Ensino e Aprendizagem:** este conjunto de *features* concentra módulos e funcionalidades que apoiam o ensino e aprendizagem, e.g., *blog*, fórum, *chat*, questionários diversos. Essas *features* são independentes das demais, mas buscam possibilitar a comunicação e colaboração por meio da troca de informações entre alunos e entre alunos e professores.
- **Usuários:** esta é uma das únicas categorias de *features* que refletem também requisitos não funcionais configuráveis, relacionados a segurança. Dentre tais configurações, tem-se a possibilidade de troca de mensagens de modo anônimo, mesmo que o aluno esteja identificado na aplicação; a ativação de *logs* de interação na aplicação; registro de tempo; entre outros. Adicionalmente, é uma *feature* obrigatória, quando selecionado o ensino formal, para permitir o acompanhamento da retenção de aprendizagem pelos professores e fornecer *feedback* aos alunos.

Para a definição das diferentes *features* e de seus respectivos módulos educacionais, apoiando a mitigação de problemas no ensino e aprendizagem de programação, como já mencionado, o catálogo de requisitos foi integrado a permitir apoio no estabelecimento desses módulos. A Figura 42 exemplifica essa integração por meio da tela de criação do módulo educacional das atividades de aprendizagem. Nela, requisitos que apoiam a configuração dessa *feature* podem ser consultados por meio do botão verde de interrogação.

Figura 42 – Página de Criação das Atividades.

Configuração das Atividades de Aprendizagem ⓘ ?

Configure os itens educacionais presentes na permissão selecionada na etapa anterior.

Título:

Atividade 1

Descrição:

Atividade aplicada aos alunos do primeiro ano para reforço dos conteúdos.

Deseja reutilizar essa atividade educacional em outras aplicações?

Não

Tópico:

Funções

Questões:

O trecho de código em C++ mostrado em seguida foi extraído de um programa...

Prévia da Lista de Atividades Educacionais na Aplicação

✓ Atividade: Atividade 1

Fonte: Elaborada pelo autor.

Além da exibição dos requisitos pertinentes a apenas a *feature* que corresponde à interface exibida no *browser*, a versão completa do catálogo pode ser acessada por meio da opção “Catálogo de Requisitos”, na barra de navegação.

A Figura 43 apresenta o painel principal da aplicação, no qual as principais ações da ferramenta TANGRAM podem ser acessadas. É na ação de “Nova Configuração” que as *features* apresentadas anteriormente são selecionadas para compor a aplicação. Após criar uma configuração, essa deve ser utilizada para seguir com a ação para “Nova Aplicação” e, então, a configuração dos módulos educacionais.

A ferramenta, além de permitir a criação dos diferentes módulos educacionais, como conteúdos, atividades e a sequenciação dos mesmos em tópicos, criando uma estrutura semelhante a um índice de livro, permite o compartilhamento de tais módulos com outros professores e também com outras aplicações que possam vir a desenvolver. O objetivo é aplicar reuso também para os diferentes módulos educacionais que serão utilizados pelos estudantes para aprender fundamentos de programação.

Figura 43 – Tela Principal da TANGRAM.

Fonte: Elaborada pelo autor.

Ademais, a ferramenta TANGRAM também possui cadastros básicos cujo os dados cadastrados estarão envolvidos nos processos administrativos (e.g., cadastro de professores, aprendizes, turma, matrícula); e de aprendizagem, (e.g., cadastro de questões, de tópicos educacionais).

Assim, a ferramenta apoia diferentes usuários, considerando seus respectivos cenários de atuação: (i) professores, que são auxiliados nos processos de ensino e aprendizagem, por meio das aplicações e também do acompanhamento das ações de seus alunos nestas aplicações possibilitando, ainda, a interação com estes alunos com *features* específicas; (ii) desenvolvedores, por meio do cadastro e gerenciamento das *features* existentes na LPS, fornecendo apoio para a condução, principalmente, da evolução da linha, por meio da integração de novas *features*; e (iii) indiretamente, os alunos, por meio das aplicações e de seus diferentes módulos educacionais, a apoiar a aprendizagem de fundamentos de programação.

6.3 Avaliação de Usabilidade com Professores

Usabilidade pode ser definida como a facilidade de uso e aceitabilidade de um sistema para uma classe específica de usuários. A facilidade de uso afeta o desempenho dos usuários e sua satisfação, enquanto a aceitabilidade afeta se o produto pode ser utilizado (HOLZINGER, 2005; BANGOR; KORTUM; MILLER, 2009). Neste sentido, a TANGRAM busca promover e facilitar diferentes tarefas relacionadas à criação das aplicações da LPS e de seus módulos educacionais, além de fornecer apoio para mitigar problemas no domínio educacional que aborda.

Com o objetivo de identificar se a ferramenta TANGRAM consegue atingir seus obje-

tivos na perspectiva de professores (potenciais usuários), uma avaliação de usabilidade com a ferramenta foi realizada.

6.3.1 Planejamento, Metodologia e Execução

Com o objetivo de aplicar um teste de usabilidade que reduza a fadiga e permita uma avaliação consistente da ferramenta, optou-se pela escolha do Sistema de Escala de Usabilidade – *System Usability Scale* (SUS), que permite a avaliação de produtos como páginas da web, dispositivos móveis, sistemas de resposta a interações via voz e outros, por meio de dez questões, que devem ser respondidas por meio de uma escala Likert (LIKERT, 1932), com ranque de 1 a 5, variando de “concorda totalmente” a “discorda totalmente”, respectivamente (BANGOR; KORTUM; MILLER, 2008; BANGOR; KORTUM; MILLER, 2009). Os critério do SUS apoiam a avaliação quantitativa de: (a) efetividade, no que tange se os usuários conseguem completar seus objetivos; (b) eficiência, em relação à quantidade de esforço e recursos necessários para realização de determinada tarefa; e (c) satisfação, no que diz respeito se a experiência foi satisfatória.

A instrumentação para a condução desta avaliação foi desenvolvida por meio de um questionário *online*, no qual o participante (professor); deveria escolher entre duas tarefas similares e realizá-la na TANGRAM. Para garantir que a instrumentação atendia ao escopo e objetivos da avaliação, uma execução piloto foi realizada com dois alunos de doutorado, levando a ajustes na descrição das tarefas e suas sub-tarefas.

6.3.2 Descrição e Análise dos Resultados

Em relação à caracterização dos 12 participantes, no que diz respeito à formação, um deles é especialista, três possuem mestrado incompleto, um mestrado completo, cinco possuíam doutorado incompleto e dois doutorado completo.

O gráfico da Figura 44 apresenta o nível de conhecimento técnico e teórico no tópico de aplicações educacionais web e o nível de familiarização em avaliações de usabilidade.

Figura 44 – Níveis de Conhecimento dos Participantes.

Fonte: Dados da pesquisa.

A quantidade de participantes em nível intermediário e proficiente em conhecimento sobre aplicações educacionais móveis educacionais (8) ultrapassa o total de nível básico e intermediário (5). O mesmo ocorre para aqueles com conhecimento em avaliação de usabilidade (8). O motivo para questionar sobre o nível de conhecimento em avaliações de usabilidade deve-se à necessidade de identificar possíveis problemas no processo de realização da mesma, caso seja necessário, permitindo traçar ainda uma relação entre possíveis *outliers* e a familiarização do participante com avaliações de usabilidade.

O gráfico da Figura 45 apresenta o total de respostas para a pergunta que objetiva identificar se o participante já utilizou alguma ferramenta para criação de conteúdos, módulos ou aplicações educacionais móveis para o ensino de fundamentos de programação. Dois participantes indicaram que sim e complementaram a resposta indicando a criação de vídeos e outro a criação de curso na plataforma Udeemy¹ em domínio diferente de programação. Ou seja, nenhum deles utilizou alguma ferramenta que permitisse a criação de elementos educacionais no domínio da LPS proposta. Esse resultado fornece indícios de que a utilização de ferramentas que permitam a criação de software, não apenas aplicações *m-learning*, é uma prática que ainda carece de maior atenção e esforços para seu crescimento, principalmente em contexto nacional.

Figura 45 – Total Participantes que Já Utilizaram Ferramentas Educacionais.

Fonte: Dados da pesquisa.

Para a aplicação do questionário do SUS, duas tarefas foram disponibilizadas para que os participantes selecionassem uma delas e a realizassem na ferramenta, sendo tais tarefas conjuntos de passos a serem realizados na ferramenta TANGRAM. Quatro selecionaram a Tarefa 1 e oito a Tarefa 2. Após a realização da sua respectiva tarefa, os participantes eram convidados a responder as dez questões do SUS. A Tabela 7 apresenta os resultados e a estatística descritiva.

¹ <www.udemy.com/>

Tabela 7 – Resultados Coletados e Estatística Descritiva do SUS.

ID	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Contagem SUS	Total Média	Desvio Padrão	Intervalo de Confiança
#1	4	3	2	4	4	1	4	3	4	3	80,00	8,00	2,00	1,24
#2	3	3	3	1	4	4	3	4	1	3	72,50	7,25	1,90	1,18
#3	3	2	3	2	3	3	3	4	3	2	70,00	7,00	1,20	0,74
#4	4	3	4	4	4	4	4	2	3	4	90,00	9,00	1,40	0,87
#5	3	2	3	3	3	0	4	3	4	2	67,50	6,75	2,05	1,27
#6	2	4	3	4	2	3	3	3	3	3	75,00	7,50	1,00	0,62
#7	4	3	2	3	4	1	4	4	4	2	77,50	7,75	2,25	1,39
#8	4	3	4	3	4	3	1	3	2	2	72,50	7,25	1,85	1,15
#9	2	0	4	2	4	0	2	4	4	4	65,00	6,50	3,50	2,17
#10	3	4	3	3	3	4	2	3	3	2	75,00	7,50	1,00	0,62
#11	4	2	3	2	3	4	4	2	2	3	72,50	7,25	1,80	1,12
#12	4	2	2	3	4	3	3	3	3	4	77,50	7,75	1,35	0,84
Total Média											74,58	7,46	21,30	1,10

Fonte: Dados da pesquisa.

Para calcular a pontuação apresentada nas colunas correspondentes a cada uma das dez questões, identificadas pela letra “Q” e o número da questão (Tabela 7), foram utilizados os cálculos indicados no SUS (BANGOR; KORTUM; MILLER, 2008; BANGOR; KORTUM; MILLER, 2009):

- Para marcação de respostas com valores ímpares na escala Likert (1, 3, 5), deve-se subtrair 1 da pontuação indicada pelo participante;
- Para marcação de respostas pares, na escala (2 e 4), deve-se subtrair a resposta de 5;
- Ao final, todos os valores devem ser somados e então multiplicados por 2,5. O valor final poderá ir de 0 a 100.

A linha “Totais” apresenta os totais para cada um dos itens somatórios e médias, obtidos por meio das respostas calculadas considerando as instruções do SUS. A coluna “Total Média” resulta no valor de 74,58. A média do SUS é de 68 pontos. Logo, se o valor obtido em tal coluna for maior que 68, há indícios de que a ferramenta apresenta um grau de efetividade, eficiência e satisfação aceitáveis.

Devido ao tamanho reduzido da amostra (12), o intervalo de confiança foi calculado, resultado no valor de 1,27. Assim, o valor do SUS de 74,58, deve variar entre $74,58 - 1,10 = 73,48$ a $74,58 + 1,10 = 75,68$. Logo, como tal intervalo está acima da média indicada no SUS, considerou-se resultado como relevante e positivo em relação à usabilidade da ferramenta, obteve-se boas evidências de que TANGRAM é efetiva, eficiente e satisfatória em termos de usabilidade. A Figura 46 apresenta os *box plots* para cada uma das questões do SUS. A versão interativa do mesmo pode ser acessada em <<https://plot.ly/~andersonmarcolino/1/>>.

É importante mencionar que, como indicado anteriormente, os resultados do SUS variam de 0 a 100. Logo, o valor obtido de 74,50, apesar de estar acima da média de tal teste, expõe

possíveis problemas na interface da ferramenta. Para averiguar que tipos de problemas foram enfrentados, havia uma questão direcionada a identificar sugestões e críticas dos participantes em relação à ferramenta.

O Quadro 22 apresenta as sugestões e críticas quanto à ferramenta e seu uso. A principal crítica refere-se à necessidade de um exemplo ou tutorial que explique como proceder no processo de criação da aplicação. Contudo, a ferramenta possui uma página de ajuda que indica os passos para criação das aplicações e dos módulos educacionais.

Figura 46 – Box Plot das Respostas para as Questões do SUS.

A Figura 47 é uma imagem extraída da tela de ajuda, e tem como principal objetivo guiar o usuário na criação da configuração e, em seguida, na criação da aplicação. Adicionalmente, outros itens de navegação na ferramenta estão presentes na mesma interface, auxiliando no seu uso. No entanto, com o objetivo de facilitar ainda mais o processo de criação da configuração e aplicação, vídeo que apresentem a gravação das interações na ferramenta podem ser incluídos em versões futuras.

Outro ponto indicado refere-se ao uso do mesmo rótulo em telas diferentes e também títulos divergentes dos descritos nas tarefas. Estes receberam as devidas correções para evitar ambiguidades. Os demais problemas, em geral de menor importância, foram analisados e melhorias e correções foram aplicadas.

6.3.3 Ameaças à Validade da Avaliação

Em relação à avaliação de validade, destacam-se as seguintes ameaças:

Ameaças à Validade de Conclusão: a principal ameaça em relação à avaliação SUS é o tamanho da amostra, que deve ser incrementada em estudos futuros.

Ameaças à Validade de Construção: os participantes foram convidados a participar podendo, a qualquer momento, optar por não finalizar a avaliação. Acredita-se que, com esta

Quadro 22 – Sugestões e Críticas por Participante.

ID	Classificação	Sugestões/Críticas	Adotaria a Ferramenta?
#1	80,00	-	Sim
#2	72,50	-	Sim
#3	70,00	"A aplicação é muito interessante e achei bem útil. Entretanto, algumas coisas poderiam ser melhor explicadas para o usuário, já que muitas coisas são repetitivas. Por exemplo, o uso do rótulo "Título" em diversas telas, sem especificar a que o título pertence. Especificar quais são as etapas para a criação da App também ajudaria o usuário a entender quais são os passos que ele deverá seguir para completar a tarefa. Essa informação tem que ser dada antes do usuário começar a criar o seu aplicativo. O vai e vem de uma tela para a outra pode confundir."	Sim
#4	90,00	"Sugiro Cases explicativos para ilustrar com um layout"	Sim
#5	67,50	"Os botões abaixo do QRCode estavam desabilitados ao final da aplicação. Poderia diferenciar melhor o que é uma Unidade e um Conteúdo educacional"	Não
#6	75,00	"A respeito da aplicação não, mas as atividades da tarefa 1 me deixaram um pouco confuso. Mais especificamente a atividade 10, sua descrição não confere com o título da página."	Sim
#7	77,50	"Eu só consegui realizar até o passo 10. Ai o conteúdo desaparece e a tela fica em branco."	Sim
#8	72,50	"A unica sugestão é na mensagem exibida na edição do Conteúdo de Aprendizagem: "Tem certeza que deseja cancelar a criação de seu aplicativo?". Além do pequeno erro de português, achei a mensagem incoerente, pois acredito que apenas os dados não salvos seriam perdidos, certo? Na dúvida acabei salvando mesmo sem alterar nada."	Sim
#9	65,00	"Está muito confuso os termos, está com layout agradável porém dá aplicar UX"	Não
#10	75,00	"Poderia ter algum tutorial de como utilizar a aplicação. Algo tipo um passo a passo de forma gráfica."	Sim
#11	72,50	"Achei um pouco confuso os passos para cadastrar as atividades, para mim não ficou claro o que estava cadastrando a cada passo."	Não
#12	77,50	"Sem os itens que eu deveria seguir para realizar a atividade (listados antes), creio que eu levaria um tempo maior para aprender a usá-la. Precisaria ler algum manual ou algo do tipo."	Sim

Fonte: Dados da pesquisa.

Figura 47 – Excerto da Ajuda da Ferramenta TANGRAM.

Fonte: Elaborada pelo autor.

decisão, os participantes responderam prontamente à avaliação, sem o viés ocasionado pela possível obrigatoriedade de se responder o questionário em sua totalidade.

Ameaças à Validade Interna:

- **Diferença entre participantes:** a experiência dos professores participantes pode interferir nos resultados coletados. Contudo, esta ameaça não pode ser mitigada, visto que a existência de grupos homogêneos de participantes é de difícil solução; e
- **Efeitos de Fadiga:** por meio da condução de um estudo piloto, o tempo para responder

tal avaliação foi de 30 minutos. Neste sentido, efeitos de fadiga não foram considerados relevantes. Adicionalmente, o SUS foi escolhido para permitir a execução da avaliação em menor tempo.

Ameaça à Validade Externa:

- **Instrumentação:** para garantir que o questionário não apresentasse possíveis limitações, um estudo piloto foi conduzido. O estudo permitiu que modificações fossem feitas quanto às tarefas a serem realizadas na ferramenta, garantindo adequações na avaliação como um todo.

Na próxima seção são discutidas questões relacionadas às aplicações *m-learning* criadas na ferramenta, considerando as diferentes *features* da LPS, bem como algumas das características mais relevantes desenvolvidas.

6.4 Aplicações para o Ensino e Aprendizagem de Programação

Como discutido, as aplicações educacionais móveis da LPS proposta neste trabalho são desenvolvidas por meio da ferramenta TANGRAM. Ao indicar a presença de determinado grupo de *features*, estas estarão disponíveis para sua definição a compor um módulo educacional e, ao final, a aplicação móvel a ser compartilhada com os alunos.

A aplicação móvel desenvolvida considera, para essa primeira versão da LPS, apenas a plataforma móvel Android, por ser a mais difundida no mercado². A tecnologia adotada foi o *framework React Native*, que é uma abstração que permite o desenvolvimento de aplicações móveis híbridas³ (MALAVOLTA *et al.*, 2015).

Uma aplicação móvel híbrida é uma aplicação que é desenvolvida utilizando tecnologias *web*, porém, é projetada para ser instalada e utilizada em dispositivos móveis (MALAVOLTA *et al.*, 2015). Seu funcionamento, para o usuário final, é o mesmo de uma aplicação nativa: (i) a aplicação pode ser distribuída por uma loja de aplicativos, (ii) a aplicação pode ser instalada no dispositivo da mesma forma que qualquer aplicação nativa, e (iii) sua utilização não requer nenhuma ação diferente por parte do usuário.

Deste modo, a aplicação é entregue em sua totalidade, obtendo os dados definidos na TANGRAM para renderizar as interfaces em tempo de execução, exibindo apenas as *features* definidas em tal aplicação, por meio da TANGRAM. Assim, uma única instalação da aplicação

² <<https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/>>

³ <<https://facebook.github.io/react-native/>>

no *smartphone* do usuário permitirá o acesso a uma ou mais aplicações definidas por um ou mais professores. O acesso a tais aplicações ocorrerá somente se o usuário possuir permissão. Esta permissão, por sua vez, é fornecida na TANGRAM por meio da seleção de turmas e seus respectivos alunos.

Como proposta de um mínimo produto viável, foram desenvolvidas nesta aplicação algumas das principais *features* classificadas como prioritárias pelos professores na avaliação do catálogo de requisitos *m-learning* (Capítulo 4). São elas: (1) identificação de usuário; (2) apresentação de conteúdos e atividades educacionais; (3) editor de código com apoio de um conjunto de gestos; e (4) *feedback* de interação. Cada uma dessas *features* auxilia na mitigação de alguma das categorias de problemas de programação, discutidas especialmente no Capítulo 1.

As *features* de identificação de usuário colaboram com o processo de identificação dos alunos e suas interações nas aplicações. Logo, correspondem ao apoio na mitigação de problemas de programação gerais, envolvendo todas as categorias. Já as *features* de conteúdos educacionais buscam apoiar os problemas relacionados, principalmente, à categoria de aprendizagem de conceitos de programação, utilizando-se de representações diversas, como vídeos, textos e figuras. Já as *features* de atividades educacionais, provêm meios de se aplicar os conceitos de programação aprendidos por meio dos conteúdos; para identificar a compreensão de programação e condução de fatoração e refatoração. As *features* referentes aos mecanismos de programação concentram-se principalmente no editor de código, que colabora também com a aplicação de conceitos aprendidos, integrando questões de atividades educacionais.

Para todas essas *features*, o grupo de *feedback* e resultados busca guiar e auxiliar, por meio de instruções e resultados das questões, o processo de aprendizagem. Adicionalmente, o professor pode acompanhar na TANGRAM os resultados dos alunos, em relação à realização das atividades propostas, o que fornece apoio aos professores, mitigando problemas de acompanhamento.

A diversidade destas *features* e a possibilidade de integração das mesmas, ainda que em um grupo reduzido, quando comparado às diferentes funcionalidades descritas nos 97 requisitos do catálogo *m-learning*, possibilitam a investigação da mitigação de problemas nas seis categorias descritas e discutidas no decorrer deste trabalho (SOUZA; BATISTA; BARBOSA, 2015; MARCOLINO; BARBOSA, 2017a). A integração de tais *features* difere-se ainda, quando comparadas às especificações dos software identificados no MS discutido no Capítulo 3 que, para a maioria das propostas, acabam mitigando problemas pontuais ou para um grupo menor de problemas no ensino e aprendizagem de programação, além de, em sua maioria, apoiarem o ensino informal.

Já no contexto de aplicações *m-learning* e, mais especificamente, nas características exclusivas de hardware dos dispositivos móveis, buscou-se desenvolver uma *feature* que pudesse reduzir limitações de tal plataforma, além de buscar motivar seu uso pelos alunos, como discutido previamente no Capítulo 5. Nesta perspectiva, um conjunto de gestos apresentado como um catálogo foi proposto para: (i) permitir a substituição dos teclados nativos das plataformas

móveis, que podem dificultar a realização de tarefas de codificação mais extensas; e (ii) motivar os alunos a utilizarem tal plataforma, considerando a especificidade de interação obtida com as telas sensíveis ao toque.

A Figura 48 apresenta os 12 gestos do catálogo selecionados por meio de uma avaliação conduzida com 69 estudantes de graduação de cursos de computação e relacionados. Cada um dos 12 gestos buscou representar uma estrutura de código específica para permitir a sua inserção em um editor de código, em atividades de programação, reduzindo a necessidade do uso do teclado nativo dos dispositivos móveis. Como tal catálogo é integrado na LPS como uma *feature*, este pode ser ou não utilizado nas aplicações móveis educacionais, de acordo com a definição do professor durante a criação da aplicação na ferramenta TANGRAM.

Figura 48 – Catálogo de Gestos *M-learning* e Trechos de Código que Representam.

Fonte: Elaborada pelo autor.

Após a definição de uma versão inicial do catálogo, uma nova avaliação foi conduzida. Nesta avaliação os gestos foram utilizados de modo prático, por meio de aplicativo (*Gesture Tool*) que permite testar gestos em dispositivos móveis. Ao todo 40 alunos de cursos de computação e relacionados, participaram. A Figura 49 exibe a tela de cadastro dos gestos no aplicativo, utilizadas para a inserção dos gestos. Após o participante realizar a inserção correta, uma mensagem com o nome da estrutura de código correspondente era exibida.

Ao final do uso prático dos gestos, três questões foram respondidas, para identificar: (a) se os participantes usariam os gestos para substituir teclados nativos; (b) o quão satisfatória foi a experiência no uso dos gestos; e (c) opinião se o catálogo pode vir a motivar os estudantes a aprender por meio de aplicações móveis.

Figura 49 – *Gesture Tool* e Usuário Inserindo Gesto durante a Segunda Avaliação.

Fonte: Elaborada pelo autor.

Todas as respostas das perguntas foram positivas. Contudo, considerando a amostra reduzida e a dificuldade de generalização, elas são consideradas evidências preliminares de que o catálogo pode ser adotado como meio para substituir os teclados dos dispositivos de modo satisfatório podendo, ainda, ser um meio motivante para os aprendizes conduzirem suas tarefas em dispositivos móveis. O protocolo completo para o estabelecimento do catálogo estão disponíveis com maiores detalhes no Apêndice C.

6.4.1 Desenvolvimento da Aplicação Móvel e Integração dos Gestos como Serviços

As funcionalidades da aplicação foram desenvolvidas como componentes *stateless* e contêineres *statefulls*, ambos conceitos relativos ao *React* e especificam módulos que possuem gerenciamento de estado da aplicação e os que apenas provêm interfaces, respectivamente. Essa possibilidade facilitou e impulsionou o desenvolvimento monolítico da aplicação que facilita, ainda, o processo de atualização e disponibilização aos usuários.

Já quanto ao desenvolvimento do catálogo de gestos como uma *feature* a integrar a LPS, foi necessário adotar o algoritmo *1\$ Recognizer* (WOBBROCK; WILSON; LI, 2007), a garantir o reconhecimento adequado dos gestos, quando inseridos nas telas dos dispositivos.

O algoritmo *1\$ Recognizer* é um algoritmo para realizar o reconhecimento de gestos (WOBBROCK; WILSON; LI, 2007), assemelhando-se ao algoritmo de *K Nearest Neighbors* (BEZDEK; CHUAH; LEEP, 1986). Dado um conjunto de pontos de entrada, o algoritmo os compara a pontos associados a gestos já cadastrados previamente, e retorna o gesto ao qual o

conjunto de pontos de entrada mais se assemelha.

Após a realização do reconhecimento do gesto, um determinado código deve ser adicionado ao código contido em um editor, também integrado como uma *feature* à LPS. Utilizou-se, assim, a biblioteca *CodeMirror*⁴ para a implementação do editor de código, apresentado na Figura 50. A escolha ocorreu pelo desempenho satisfatório em testes e pelo suporte da biblioteca à numeração em linhas, *highlight* de acordo com a sintaxe da linguagem utilizada, e uma API para realizar diversas ações no editor.

Figura 50 – Editor *CodeMirror* contendo código *HTML* e *JavaScript*.

```
1 <!-- Create a simple CodeMirror instance -->
2 <link rel="stylesheet" href="lib/codemirror.css">
3 <script src="lib/codemirror.js"></script>
4 <script>
5     var editor = CodeMirror.fromTextArea(myTextarea, {
6         lineNumbers: true
7     });
8 </script>
```

Fonte: Extraído de <<https://codemirror.net/>>.

A Figura 51 exibe a tela de criação de atividades, na qual os professores podem selecionar ou não a *feature* de gestos por meio do item “Deseja integrar o catálogo de gestos nas atividades de programação?”. Ao indicar sim, o módulo educacional de atividades integra o catálogo de gestos, sendo esta escolha salva na base de dados da TANGRAM. Quando um aluno realizar o *login* na aplicação, as atividades de programação que utilizam editor de código exibirão a área para inserção dos gestos (Figura 51 (a)). Se o professor indicar não, o registro no banco indicará a não inclusão da *feature* de gestos e a aplicação exibirá o teclado nativo do dispositivo para permitir a inserção de código nas atividades de programação (Figura 51 (b)).

⁴ <<https://codemirror.net/>>

Figura 51 – Utilização da TANGRAM para Seleção da *feature* de Gestos. a) Com a *feature* de gestos.b) Sem a *feature* de gestos.

Fonte: Elaborada pelo autor.

Finalmente, o único fator limitante quanto as versões iniciais da aplicação refere-se à necessidade de as manter conectadas, via rede móvel ou *wi-fi*, para que as definições realizadas na TANGRAM possam ser renderizadas de acordo com as especificações e escolhas feito pelo professor. A proposição do catálogo, as duas avaliações e seu desenvolvimento resultaram em publicações nacionais (SCHAEFER; MARCOLINO; BARBOSA, 2018; SANTOS; MARCOLINO; BARBOSA, 2018) e internacionais (MARCOLINO *et al.*, 2018).

6.5 Considerações Finais

Neste capítulo foi apresentada a ferramenta TANGRAM, uma aplicação *web* que possibilita o gerenciamento das *features* da LPS, por meio da definição de configurações a serem utilizadas na criação das aplicações educacionais móveis. Durante a criação das aplicações, além de *features* específicas a serem disponibilizadas, o professor define os módulos educacionais a serem utilizados para seus alunos e turmas no processo de aprendizagem. O objetivo da ferramenta é fornecer um mecanismo facilitador para o processo de criação das aplicações, reduzindo a carga técnica necessária para a criação das aplicações, permitindo um modo interativo para a criação dos módulos educacionais que podem, ainda, serem reutilizados pelos usuários da ferramenta. Neste contexto, a ferramenta foi avaliada quanto a sua usabilidade. Os resultados apresentaram evidências preliminares de que a ferramenta é útil para a adoção no contexto da criação de aplicações móveis.

Na sequência, discutiu-se brevemente sobre o desenvolvimento da aplicação a ser instalada nos dispositivos móveis dos alunos, a escolha da tecnologia adotada e das *features* candidatas, desenvolvidas para uma primeira versão da aplicação a ser avaliada e comparada com outras aplicações. Apresentou-se ainda a relação de tais *features* com as categorias de problemas de programação que visam mitigar. Em especial, foi apresentada a proposta de um catálogo de gestos, seu estabelecimento e duas avaliações, concluindo-se com o seu desenvolvimento, de modo a integrar à LPS. O objetivo do catálogo é permitir a disponibilização de um modo mais interativo com as aplicações, no contexto de inserção de trechos de código nas tarefas de codificação. Os resultados da avaliação com alunos, levaram a identificação de indícios positivos para a adoção de tais gestos como meios para motivar o uso de aplicações educacionais móveis.

No próximo capítulo são apresentadas as conclusões, contribuições e limitações da pesquisa, bem como trabalhos futuros e publicações associadas.

CONCLUSÕES

A busca pela melhoria no processo de ensino e aprendizagem tem sido alvo de pesquisas em diferentes áreas, cursos e disciplinas (KHALID; BUUS, 2013; COLLINS; HALVERSON, 2018; MARCOLINO; BARBOSA, 2015a; JOHNSON; ADAMS; CUMMINS, 2016; HAWKRIDGE; VINCENT; HALES, 2018). Neste contexto, destaca-se a área de computação, na qual disciplinas que envolvem aspectos teóricos e práticos de programação vêm ganhando cada vez mais importância em nível mundial (BORDINI *et al.*, 2016; AURELIANO; TEDESCO, 2012; DUNCAN; BELL; TANIMOTO, 2014). Contudo, apesar de sua difusão e importância, uma série de limitações ainda são enfrentadas (SOUZA; BATISTA; BARBOSA, 2015; MARCOLINO; BARBOSA, 2015b).

Neste contexto, o uso de software para tentar sanar tais limitações e melhorar os processos de ensino e aprendizagem no domínio de programação é facilmente percebido (MARCOLINO; BARBOSA, 2015b; AURELIANO; TEDESCO, 2012). Porém, com o avanço de tecnologias da informação e comunicação, principalmente os dispositivos móveis, seu barateamento e aumento no poder computacional, surge uma plataforma ainda pouco investigada em relação a propostas de melhorias no domínio de fundamentos de programação no ensino formal (MARCOLINO; BARBOSA, 2015b; CROMPTON, 2013; BARAN, 2014; SOAD *et al.*, 2017a; IEA, 2018).

Em uma perspectiva diferente, mas relacionada, o processo para concepção de software educacionais, em geral, é custoso e complexo, devido às *features* envolvidas (KHALID; BUUS, 2013; COLLINS; HALVERSON, 2018). O estabelecimento de tais soluções envolvem questões do domínio de negócio, e.g., socioculturais, econômicas, legais e didático-pedagógicas; e devem considerar o hardware disponível para sua execução (MARCOLINO; BARBOSA, 2015b). Tais especificidades de hardware são ainda mais evidentes no contexto dos dispositivos móveis (CROMPTON, 2013; BARAN, 2014; SOAD *et al.*, 2017a; IEA, 2018).

Assim, para permitir a redução de custo e um processo de desenvolvimento facilitado, a Engenharia de Software provê diferentes estratégias e metodologias que, além de reduzir custos

e melhorar o processo de desenvolvimento de soluções que necessitem mais esforços, como os software educacionais, permitem um nível maior de reutilização de artefatos e, consequentemente, de funcionalidades que muitas vezes já estão presentes em software adotados. Dentre as metodologias da Engenharia de Software baseada em Reúso, destacam-se as linhas de produto de software (POHL; BOCKLE; LINDEN, 2005; GOMAA, 2004; GUANA; CORREAL, 2013; FIORI *et al.*, 2012; DHUNGANA *et al.*, 2010).

Destarte, considerando: (i) os problemas de ensino e aprendizagem de programação, (ii) os dispositivos móveis, (iii) linhas de produto, e (iv) as lacunas relacionadas a incipiência de aplicações educacionais móveis desenvolvidas por meio da metodologia LPS no domínio de programação, esta pesquisa de doutorado procurou estabelecer uma infraestrutura de apoio ao desenvolvimento de aplicações educacionais móveis para o ensino de fundamentos de programação (MARCOLINO; BARBOSA, 2015b; MARCOLINO; BARBOSA, 2015a).

O estabelecimento de uma infraestrutura composta pela linha de produtos, seus ativos centrais (*features*), e a ferramenta de apoio TANGRAM, tem como objetivo principal criar meios para meios para sanar, ou ao menos mitigar, os problemas no ensino e aprendizagem de fundamentos de programação, considerando aplicações educacionais móveis. Essas aplicações são criadas por meio da seleção de diferentes *features* da LPS, com o apoio da ferramenta TANGRAM, que mantém, ainda, o registro dessas *features*, auxiliando na evolução sistêmica da LPS.

Assim, a ferramenta TANGRAM permite a criação facilitada de aplicações educacionais móveis para os professores, reduzindo a complexidade e necessidade de conhecimentos técnicos avançados, quando considerado o desenvolvimento direto por meio das *features* da LPS. A ferramenta auxilia, também, o acompanhamento da aprendizagem dos alunos, visto que possui integração com diferentes APIs e *features* utilizadas nas aplicações móveis.

Na perspectiva de desenvolvedores, tanto o projeto arquitetural quanto as decisões tecnológicas buscam facilitar o processo de integração de novas *features*. Ademais, os registros das *features* na TANGRAM buscam mapear as interfaces que permitem a integração de tais *features* com as aplicações, bem como as APIs e outros serviços e microsserviços que os consomem, o que facilita o gerenciamento, reúso e inclusão de novas funcionalidades.

Ao final, os produtos da linha podem mitigar problemas no ensino e aprendizagem de programação, principalmente pelo uso de funcionalidades específicas da plataforma móvel, como as telas sensíveis ao toque, e a criação de meios mais motivantes para se realizar tarefas de codificação em aplicações móveis.

Para a criação, sobretudo da LPS que compõe a infraestrutura, o arcabouço de engenharia de linhas de produto (SPLE) foi adotado. Os diferentes sub-processos do SPLE foram realizados no decorrer da pesquisa. Na medida em que cada sub-processo do SPLE era concluído e novos artefatos eram definidos (e.g., catálogo de requisitos, modelos arquiteturais, ferramenta), avalia-

ções eram executadas. Estas avaliações, principalmente qualitativas, auxiliaram na investigação e estabelecimento dos artefatos, utilizados como entrada para cada sub-processo subsequente do arcabouço, até a avaliação de usabilidade referente ao uso da ferramenta na concepção das aplicações educacionais móveis.

A Figura 52 resume as três atividades conduzidas nesta pesquisa e as próximas três atividades, correspondendo aos trabalhos futuros.

Figura 52 – Resumo das Atividades e Resultados da Pesquisa.

Fonte: Elaborada pelo autor.

A etapa 1 indica a principal motivação para a condução desta pesquisa (i.e., problemas no ensino e aprendizagem de programação), em conjunto com a possibilidade de impulsionar o uso de dispositivos móveis nos processos de ensino e aprendizagem de fundamentos de programação. Na etapa 2, o catálogo de requisitos foi estabelecido, permitindo então a criação da LPS e seu núcleo. Com a LPS desenvolvida, passou-se a etapa 3, que se refere ao desenvolvimento da ferramenta TANGRAM que apoia o desenvolvimento das aplicações, utilizando, para isso, uma configuração que se refere às *features* selecionadas na LPS. A partir da etapa 4, são apresentadas as atividades futuras. Na etapa 4, as aplicações educacionais móveis serão investigadas em relação: (a) ao uso dos dispositivos móveis e *features* específicas, considerando o hardware de tais dispositivos; (b) à variação de diferentes *features* para promover e mitigar problemas no ensino e aprendizagem de fundamentos de programação; e (c) a outros software no mesmo domínio. Na etapa 5, os resultados do uso de tais aplicações serão avaliados na perspectiva de professores. Ainda na etapa 5, novas *features* para a interação de alunos com professores serão

integradas, permitindo a avaliação também da infraestrutura, no sentido de permitir a evolução da LPS. Finalmente, na etapa 6, as *features* de acompanhamento de desempenho dos alunos e turmas também serão avaliadas por professores.

7.1 Contribuições da Pesquisa

As principais contribuições desta pesquisa de doutorado são sumarizadas a seguir.

- **Caracterização do estado da arte de software educacionais para o ensino de programação:** um MS foi conduzido para caracterizar o cenário atual de software educacionais para o ensino de programação. Este MS permitiu uma visão geral dos software educacionais para o ensino e aprendizagem de fundamentos de programação em evidência na literatura.
- **Caracterização do estado da arte de linhas de produtos de software educacionais gerais e para o ensino de programação:** um MS foi conduzido para identificar as linhas de produto de software educacionais de uso geral e bem como aquelas específicas para o domínio de programação. O MS permitiu identificar as diferentes estratégias para o estabelecimento das linhas e a incipiência em termos de decisões de projeto e escolhas tecnológicas para o projeto e concepção de LPS educacionais.
- **Identificação de problemas de ensino e aprendizagem de programação no Brasil:** utilizando a classificação de problemas no domínio de ensino e aprendizagem de programação (SOUZA; BATISTA; BARBOSA, 2015), uma pesquisa foi conduzida com professores e alunos de instituições públicas e privadas no Brasil. Esta pesquisa buscou identificar se os problemas classificados com base em MS conduzido com estudos primários em inglês correspondem aos problemas enfrentados no Brasil. Tal pesquisa possibilitou a identificação dos problemas a serem enfrentados e também de soluções que estão sendo adotadas, permitindo também sua adoção por outros professores.
- **Estabelecimento de um catálogo de requisitos para aplicações educacionais móveis no domínio de ensino e aprendizagem de programação:** com base no MS de software educacionais para o ensino de programação, um catálogo de requisitos para o desenvolvimento de aplicações educacionais móveis no domínio de ensino de programação foi proposto.

O catálogo procurou, além de integrar e fornecer subsídios para a adaptação de funcionalidades, *features* e módulos educacionais para a plataforma móvel, o respaldo e integração com diferentes teorias educacionais e também com a classificação de problemas no ensino e aprendizagem de programação, identificados em pesquisas anteriores. O objetivo é fornecer apoio complementar às especificações definidas por *stakeholders*, criando meios

adicionais para a definição e criação das aplicações necessárias para se atingir objetivos educacionais definidos pelos professores.

- **Estabelecimento de uma arquitetura de linha de produtos de software para aplicações m-learning no domínio de programação:** com a utilização do catálogo de requisitos para aplicações educacionais móveis no domínio de programação, a arquitetura da LPS foi proposta. Esta engloba conceitos de serviços e microserviços facilitando o processo de reúso e integração de novas *features* a LPS.

Para se chegar ao modelo arquitetural desenvolvido, a arquitetura passou pelo processo de definição de um modelo conceitual, com maior grau de abstração. Esse modelo foi avaliado por especialistas e professores. Em seguida, um novo modelo foi proposto, com apoio de dois desenvolvedores especialistas em soluções híbridas. Esse modelo foi então novamente avaliado e, após evidências de sua factibilidade, serviu como artefato base para o desenvolvimento da LPS. Além disso, as decisões tecnológicas também foram realizadas com base em resultados das avaliações, a permitir sua adoção em projetos similares.

- **Desenvolvimento da LPS e da ferramenta TANGRAM:** considerando o catálogo de requisitos e os modelos arquiteturas avaliados, a LPS e a ferramenta TANGRAM foram desenvolvidas em paralelo. A ferramenta apoia, principalmente, os professores no processo de configuração e definição das aplicações para seus alunos, além de permitir o acompanhamento da aprendizagem e também o gerenciamento de ativos da LPS. Essa última funcionalidade é voltada, principalmente, para os desenvolvedores.
- **Catálogo de gestos para aplicações educacionais móveis:** com o intuito de mitigar problemas relacionados à inserção de textos mais longos nos dispositivos móveis, um catálogo de gestos para facilitar as tarefas de codificação foi estabelecido. Esse catálogo, além de prover, como serviço, uma *feature* para integrar as aplicações móveis da LPS, visa investigar melhorias no engajamento e na motivação dos aprendizes em relação ao uso dos dispositivos móveis como ferramenta para tarefas de codificação.

Deste modo, foi possível responder à questão de pesquisa principal no contexto deste trabalho de doutorado:

- *Uma infraestrutura, composta por uma LPS para criação de aplicações educacionais móveis, e uma ferramenta de apoio para a criação destas aplicações, estabelece meios para investigar a mitigação de diferentes problemas no ensino e aprendizagem de fundamentos de programação?*

De acordo com resultados provenientes das avaliações: (1) do catálogo de requisitos, utilizado como artefato principal para o desenvolvimento dos ativos da LPS e da ferramenta TANGRAM; (2) dos modelos arquiteturas, na perspectiva de professores de programação

e desenvolvedores; e (3) de usabilidade da ferramenta TANGRAM, foram identificadas evidências positivas ainda que preliminares, quanto ao estabelecimento de uma infraestrutura capaz de fornecer meios de apoiar professores, por meio de aplicações educacionais móveis, para mitigar problemas no ensino e aprendizagem de fundamentos programação. É importante destacar que ainda se faz necessária, a realização de novas avaliações, principalmente no que tange aos produtos da LPS, para averiguar o potencial das mesmas na mitigação e atenuação dos problemas no domínio alvo. Esta etapa já está em condução, como parte dos trabalhos futuros (Seção 7.3).

7.2 Limitações da Pesquisa

Entre as limitações da pesquisa conduzida, destacam-se:

- **Avaliações experimentais e tamanhos da amostras:** considerando o domínio de ensino e aprendizagem de programação, fazem-se necessárias avaliações com professores e alunos. Contudo, a disponibilidade destes para participar das avaliações mostrou-se menor que a esperada. Logo, replicações e outros estudos no contexto real de ensino devem ser conduzidos para ampliar a amostra e permitir a generalização dos resultados. Adicionalmente, estudos experimentais que busquem (i) comparar a ferramenta TANGRAM com outras ferramentas, ainda que em domínios de LPS diferentes; e (ii) avaliar as aplicações móveis educacionais, devem ser conduzidos.
- **Desenvolvimento de mais ativos para a LPS:** para que a LPS possa ser adotada e atenda melhor aos professores e aprendizes, mais ativos devem ser desenvolvidos e, consequentemente, avaliados de acordo com os requisitos definidos no catálogo de requisitos para o domínio da LPS. A inclusão de novos ativos permitirá, ainda, a identificação de evidências no que refere-se a sua capacidade de facilitar o processo de desenvolvimento e integração de artefatos, na perspectiva de desenvolvedores.

7.3 Trabalhos Futuros

Baseado nas limitações discutidas previamente, identificou-se diversas possibilidades para a condução de trabalhos futuros no contexto desta pesquisa, além da evolução e integração de novas *features* a LPS. A seguir são apresentados os trabalhos mais relevantes e prioritários:

- **Desenvolvimento de mais ativos para a LPS:** considerando o catálogo de requisitos, mais ativos devem ser desenvolvidos, bem como mais artefatos relacionados aos testes e modelos da LPS, de modo a auxiliar e facilitar sua evolução. Nesta perspectiva, a evolução da linha permite, ainda, a avaliação de sua facilidade no que se refere ao desenvolvimento e integração de novos ativos, na perspectiva de desenvolvedores.

- **Adoção de linguagem de padrões móveis:** conduzir a evolução da aplicação móvel monolítica, considerando a linguagem de padrões **MLearning-PL**([FIORAVANTI; BARBOSA, 2018b](#); [FIORAVANTI; BARBOSA, 2018a](#)), de modo a melhorar a qualidade da aplicação e dos seus módulos educacionais renderizados.
- **Condução de mais avaliações da ferramenta TANGRAM:** conduzir avaliações comparativas para a ferramenta TANGRAM, ainda que com ferramentas similares em outros domínios, e também, as aplicações móveis em contextos reais de ensino. Tais aplicações devem ser avaliadas, principalmente, no que diz respeito à capacidade de melhorar o ensino e aprendizagem, se comparadas com aplicações já existentes.
- **Finalização da ferramenta TANGRAM:** finalização da nova interface da ferramenta e melhor integração da mesma com os ativos da linha, mais especificamente, fornecendo melhor controle sobre requisitos de qualidade e avaliá-los com respeito à usabilidade (*tarefa em curso*). Adicionalmente, pesquisas em relação ao uso de tecnologias nativas para o desenvolvimento de um maior número de aplicações móveis como produtos da linha devem ser consideradas.
- **Compartilhamento da infraestrutura:** após a integração de mais ativos na LPS, realização de melhorias na TANGRAM, e a condução de avaliações de ambos, a infraestrutura deverá ser compartilhada com instituições e professores que desejarem a adotar para prover apoio nos processos de ensino e aprendizagem de programação.

Cabe salientar que todos os documentos, código fonte, questionários e procedimentos para avaliação da infraestrutura já estão disponíveis no link <https://andersonmarcolino.gitlab.io/tangram/>, podendo ser reutilizados em novas pesquisas, sob licença *creative commons* (*Atribuição-NãoComercial 4.0 Internacional*).

A Figura 53 resume, na forma de um diagrama, as principais atividades referentes aos trabalhos futuros.

Figura 53 – Resumo das Atividades Futuras.

Fonte: Elaborada pelo autor.

7.4 Publicações Resultantes

As principais publicações resultantes das atividades conduzidas nesta pesquisa de doutorado são apresentadas a seguir:

- MARCOLINO, A. S.; BARBOSA, E. F. **software Educacionais para o Ensino de Programação: Um Mapeamento Sistemático**. Em: Simpósio Brasileiro de Informática na Educação. p. 190-199, 2015.
- MARCOLINO, A. S.; BARBOSA, E. F. **Linhas de Produto de Software no Domínio Educacional: Um Mapeamento Sistemático**. Em: Simpósio Brasileiro de Informática na Educação. p. 239-249, 2015.
- MARCOLINO, A. S.; BARBOSA, E. F. **A Survey on Problems Related to the Teaching of Programming in Brazilian Educational Institutions**. In: 2017 IEEE Frontiers in Education Conference (FIE)., p. 1–9, Out. 2017.
- MARCOLINO, A. S.; BARBOSA, E. F. **Towards an M-learning Requirements Catalog for the Development of Educational Applications for the Teaching of Programming**. In: 2016 IEEE Frontiers in Education Conference (FIE), p. 1–5, Out. 2016.
- MARCOLINO, A. S.; BARBOSA, E. F. **Towards a Software Product Line Architecture to Build M-learning Applications for the Teaching of Programming**. In: Hawaii International Conference on System Sciences, 2017, Waikoloa (HI), 2017.
- MARCOLINO, A. S.; SANTOS, A. S.; SCHAEFER, M. A. T.; BARBOSA, E. F. **Towards a Catalog of Gestures for M-learning Applications for the Teaching of Programming**. In: 48th Annual Frontiers in Education Conference (FIE 2018), 2018, San Jose (CA) - EUA. 48th Annual Frontiers in Education Conference (FIE 2018), 2018. p. 1-7.

Complementarmente, outros trabalhos indiretamente relacionados a esta pesquisa foram publicados:

- MARCOLINO, ANDERSON; OLIVEIRA, EDSON; GIMENES, ITANA; BARBOSA, ELLEN F. **Empirically Based Evolution of a Variability Management Approach at UML Class Level**. In: 2014 IEEE 38th Annual Computer Software and Applications Conference (COMPSAC), 2014, Vasteras. 2014 IEEE 38th Annual Computer Software and Applications Conference. p. 354-363.
- MARCOLINO, A. S. ; OLIVEIRA JUNIOR, E. A. ; GIMENES, I. M. S.; BARBOSA, E. F. **Variability Resolution and Product Configuration with SMarty: An Experimental Study on UML Class Diagrams**. JOURNAL OF COMPUTER SCIENCES, v. 13, p. 211-226, 2017.

- SOAD, GUSTAVO WILLIANS ; FIORAVANTI, MARIA LYDIA; FALVO, VENILTON; MARCOLINO, ANDERSON ; FILHO, NEMESIO FREITAS DUARTE; BARBOSA, ELLEN FRANCINE . **ReqML-catalog: The road to a requirements catalog for mobile learning applications**. In: 2017 IEEE Frontiers in Education Conference (FIE), 2017, Indianapolis. 2017 IEEE Frontiers in Education Conference (FIE), 2017. p. 1-9.
- SCHAEFER, M. A. T. ; MARCOLINO, A. S. ; BARBOSA, E. F. . **Desenvolvimento de Serviços Educacionais Móveis para uma Linha de Produtos de Software no Domínio de Fundamentos de Programação**. In: 26o. Simpósio Internacional de Iniciação Científica e Tecnológica da USP (SIICUSP 2018), 2018, São Carlos (SP). 26o. Simpósio Internacional de Iniciação Científica e Tecnológica da USP (SIICUSP 2018), 2018. p. 1-1.
- SANTOS, A. S. ; MARCOLINO, A. S. ; BARBOSA, E. F. . **Proposta e Validação de um Vocabulário de Gestos para Aplicações Móveis Educacionais para o Ensino de Fundamentos de Programação**. In: 26o. Simpósio Internacional de Iniciação Científica e Tecnológica da USP (SIICUSP 2018), 2018, São Carlos (SP). 26o. Simpósio Internacional de Iniciação Científica e Tecnológica da USP (SIICUSP 2018), 2018. p. 1-1.

Por fim, encontra-se em avaliação a seguinte publicação:

- MARCOLINO, A. S.; FALVOJR V.; DE PAULA R. J.; and BARBOSA, E. F. **Evolving an M-learning Software Product Line in an Industry Practitioner Perspective**. In: Information and Software Technology, 2019. No prelo.

REFERÊNCIAS

AALST, W. V. D.; HEE, K. M. V. **Workflow management: models, methods, and systems**. [S.l.]: MIT press, 2004. Citado na página 63.

AARRENIEMI-JOKIPELTO, P. T-learning Model for Learning via Digital TV. In: **Conf. on Innovation in Education for Electrical and Information Engineering (EIE)**. Lappeenranta: [s.n.], 2005. Citado nas páginas 42 e 43.

ABACHI, H. R.; MUHAMMAD, G. The impact of m-learning technology on students and educators. **Computers in human behavior**, Elsevier, v. 30, p. 491–496, 2014. Citado na página 24.

ABDEL-BASSET, M.; MOHAMED, M.; CHANG, V. Nmcda: A framework for evaluating cloud computing services. **Future Generation Computer Systems**, Elsevier, v. 86, p. 12–29, 2018. Citado na página 41.

ADEWOJO, A. A.; BASS, J. M. Evaluating the effect of multi-tenancy patterns in containerized cloud-hosted content management system. In: IEEE. **Parallel, Distributed and Network-based Processing (PDP), 2018 26th Euromicro International Conference on**. [S.l.], 2018. p. 278–282. Citado na página 113.

AJILA, S. A.; TIERNEY, P. J. The Foom Method-modeling Software Product Lines in Industrial Settings. In: CITESEER. **Proceedings of the Int. Conf. on Software Engineering Research and Practice (SERP02)**. Las Vegas, Nevada, USA: CSREA Press. [S.l.], 2002. Citado na página 49.

ALAVI, S.; ARSENAULT, D.; WHITEHEAD, A. Quaternion-based gesture recognition using wireless wearable motion capture sensors. **Sensors**, Multidisciplinary Digital Publishing Institute, v. 16, n. 5, p. 605, 2016. Citado na página 201.

ALMEIDA, M. E. B. Tecnologia na Escola: Criação de Redes de Conhecimentos. **Integração das Tecnologias na Educação, Brasília, DF**, p. 70–73, 2005. Citado na página 40.

ALRASHEEDI, M.; CAPRETZ, L. F. Determination of critical success factors affecting mobile learning: A meta-analysis approach. **arXiv preprint arXiv:1801.04288**, 2018. Citado nas páginas 25 e 40.

ANDERSON, T.; DRON, J. Integrating learning management and social networking systems. **Italian Journal of Educational Technology**, Edizioni Menabò-Menabò srl, v. 25, n. 3, p. 5–19, 2018. Citado na página 24.

ANTHONY, L.; BROWN, Q.; NIAS, J.; TATE, B. Examining the need for visual feedback during gesture interaction on mobile touchscreen devices for kids. In: **Proceedings of the 12th International Conference on Interaction Design and Children**. New York, NY, USA: ACM, 2013. (IDC '13), p. 157–164. Citado nas páginas 201 e 206.

- AROYO, L.; DICHEVA, D. The new challenges for e-learning: The educational semantic web. **Journal of Educational Technology & Society**, JSTOR, v. 7, n. 4, 2004. Citado na página 24.
- ATKINSON, C.; BAYER, J.; MUTHIG, D. Component-based Product Line Development: the KobrA Approach. In: **Software Product Lines**. [S.l.]: Springer, 2000. p. 289–309. Citado na página 49.
- AURELIANO, V. C. O.; TEDESCO, P. C. de A. R. Ensino e Aprendizagem de Programação para Iniciantes: uma Revisão Sistemática da Literatura focada no SBIE e WIE. **Simpósio Brasileiro de Informática na Educação**, 2012. Citado nas páginas 26, 31, 32, 39, 66, 157, 187 e 194.
- BACICH, L.; NETO, A. T.; TREVISANI, F. de M. **Ensino híbrido: personalização e tecnologia na educação**. [S.l.]: Penso Editora, 2015. Citado na página 24.
- BALA, M. Use of ict in higher education. **Multidisciplinary Higher Education, Research, Dynamics & Concepts: Opportunities & Challenges For Sustainable Development**, v. 1, n. 1, p. 368–376, 2018. Citado na página 40.
- BALAJI, B. P. An integrative review of web 3.0 in academic libraries. **Library Hi Tech News**, Emerald Publishing Limited, v. 35, n. 4, p. 13–17, 2018. Citado na página 41.
- BALALAIE, A.; HEYDARNOORI, A.; JAMSHIDI, P.; TAMBURRI, D. A.; LYNN, T. Micro-services migration patterns. **Software: Practice and Experience**, Wiley Online Library, v. 48, n. 11, p. 2019–2042, 2018. Citado nas páginas 114 e 116.
- BANGOR, A.; KORTUM, P.; MILLER, J. Determining what individual sus scores mean: Adding an adjective rating scale. **Journal of usability studies**, Usability Professionals' Association, v. 4, n. 3, p. 114–123, 2009. Citado nas páginas 144, 145 e 147.
- BANGOR, A.; KORTUM, P. T.; MILLER, J. T. An empirical evaluation of the system usability scale. **Intl. Journal of Human–Computer Interaction**, Taylor & Francis, v. 24, n. 6, p. 574–594, 2008. Citado nas páginas 145 e 147.
- BANO, M.; ZOWGHI, D.; KEARNEY, M.; SCHUCK, S.; AUBUSSON, P. Mobile learning for science and mathematics school education: A systematic review of empirical evidence. **Computers & Education**, Elsevier, v. 121, p. 30–58, 2018. Citado nas páginas 43, 44 e 45.
- BARAN, E. A review of research on mobile learning in teacher education. **Journal of Educational Technology & Society**, JSTOR, v. 17, n. 4, p. 17, 2014. Citado nas páginas 25, 42, 43, 44, 45 e 157.
- BARKER, A.; KRULL, G.; MALLINSON, B. A Proposed Theoretical Model for M-learning Adoption in Developing Countries. In: **Proceedings of mLearn**. [S.l.: s.n.], 2005. Citado na página 44.
- BARREIRO, P. S.; GARCIA-SAIZ, D.; PANTALEON, M. Z. Building families of software products for e-learning platforms: A case study. **Tecnologias del Aprendizaje, IEEE Revista Iberoamericana de**, v. 9, n. 2, p. 64–71, May 2014. Citado na página 60.
- BARTHOLO, V.; AMARAL, M.; CAGNIN, M. Uma Contribuição para a Adaptabilidade de Ambientes Virtuais de Aprendizagem para Dispositivos Móveis. **Revista Brasileira de Informática na Educação**, v. 17, n. 02, 2009. Citado na página 25.

- BASILI, V. R.; SELBY, R. W.; HUTCHENS, D. H. Experimentation in Software Engineering. **IEEE Trans. Software Eng.**, v. 12, n. 7, p. 733–743, 1986. Citado na página 88.
- BASS, L.; CLEMENTS, P.; KAZMAN, R. **Software architecture in practice**. [S.l.]: Addison-Wesley Professional, 2003. Citado na página 101.
- BAYER, J.; FLEGE, O.; KNAUBER, P.; LAQUA, R.; MUTHIG, D.; SCHMID, K.; WIDEN, T.; DEBAUD, J.-M. PuLSE: A Methodology to Develop Software Product Lines. In: **Proceedings of the Symposium on Software Reusability**. New York, NY, USA: ACM, 1999. (SSR '99), p. 122–131. Citado na página 49.
- BELLANCA, J. A. **21st century skills: Rethinking how students learn**. [S.l.]: Solution Tree Press, 2010. Citado na página 24.
- BEZDEK, J. C.; CHUAH, S. K.; LEEP, D. Generalized k-nearest neighbor rules. **Fuzzy Sets and Systems**, v. 18, n. 3, p. 237 – 256, 1986. Dedicated to the memory of Richard E. Bellman. Citado na página 153.
- BHULLAR, M. A New Method of Learning: M-Learning (Mobile Learning). In: **Int. Conf. on Computer Science Education (ICCSE)**. [S.l.: s.n.], 2014. p. 322–325. Citado nas páginas 43, 44 e 45.
- BIOLCHINI, J.; MIAN, P. G.; NATALI, A. C. C. **Systematic Review in Software Engineering**. Rio de Janeiro, RJ, Brasil, 2005. Citado nas páginas 58 e 66.
- BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **The Unified Modeling Language user guide**. Redwood City, CA, USA: Addison Wesley Longman Publishing Co., Inc., 1999. Citado na página 47.
- BORDINI, A.; AVILA, C. M. O.; WEISSHAHN, Y.; CUNHA, M. M. da; CAVALHEIRO, S. A. da C.; FOSS, L.; AGUIAR, M. S.; REISER, R. H. S. Computação na educação básica no brasil: o estado da arte. **Revista de Informática Teórica e Aplicada**, v. 23, n. 2, p. 210–238, 2016. Citado nas páginas 39, 157 e 194.
- BRAGA, P. H. C.; CARDOSO, A.; JR., E. A. L. **Desenvolvimento de Gestos Personalizados para Criação e Navegação em Ambientes de Realidade Virtual e Aumentada associada a Dispositivos Móveis (In Portuguese)**. 2012. Disponível em <https://www.researchgate.net/publication/303920129_Desenvolvimento_de_Gestos_Personalizados_para_Criacao_e_Navegacao_em_Ambientes_de_Realidade_Virtual_e_Aumentada_associada_a_Dispositivos_Moveis>. Acessado em 18, Fev., 2019. Citado na página 201.
- BROWN, J. S.; COLLINS, A.; DUGUID, P. Situated Cognition and the Culture of Learning. **Educational Researcher**, v18, n. 1, p. p. 32–42, Jan-Fev 1989. Citado na página 42.
- CABADA, R. Z.; ESTRADA, M. L. B.; HERNÁNDEZ, F. G.; BUSTILLOS, R. O.; REYES-GARCÍA, C. A. An affective and web 3.0-based learning environment for a programming language. **Telematics and Informatics**, v. 35, n. 3, p. 611 – 628, 2018. ISSN 0736-5853. SI: EduWeb of Data. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0736585316304907>>. Citado na página 41.
- CAPILLA, R.; BOSCH, J.; KANG, K. **Systems and Software Variability Management: Concepts, Tools and Experiences**. [S.l.]: Springer, 2013. (SpringerLink : Bücher). ISBN 9783642365836. Citado na página 47.

- CAPILLA, R.; BOSCH, J.; TRINIDAD, P.; RUIZ-CORTÁZAR, A.; HINCHEY, M. An overview of dynamic software product line architectures and techniques: Observations from research and industry. **Journal of Systems and Software**, v. 91, n. 0, p. 3 – 23, 2014. Citado na página 26.
- CARVALHO, T. b.; ARAÚJO, B. c. d. de; VALENTIM, R. e.; JUNIOR, J. f. D.; TOURINHO, F. f. g.; DINIZ, R. f. h. Ise-spl: A software product line approach applied to automatic generation of systems for medical education in e-learning platform. **Revista Brasileira de Engenharia Biomedica**, v. 29, n. 4, p. 434–439, 2013. Citado nas páginas 53 e 60.
- CERNY, T.; DONAHOO, M. J.; PECHANEC, J. Disambiguation and comparison of soa, micro-services and self-contained systems. In: **ACM. Proceedings of the International Conference on Research in Adaptive and Convergent Systems**. [S.l.], 2017. p. 228–235. Citado na página 123.
- CERNY, T.; DONAHOO, M. J.; TRNKA, M. Contextual understanding of microservice architecture: current and future directions. **ACM SIGAPP Applied Computing Review**, ACM, v. 17, n. 4, p. 29–45, 2018. Citado nas páginas 102, 105 e 123.
- CGI. Comitê Gestor da Internet no Brasil. Pesquisa Sobre o Uso das Tecnologias de Informação e Comunicação no Brasil: TIC Domícilios e Empresas 2013. **São Paulo**, 2014. Citado na página 40.
- CHAVES, M. *et al.* **Annotation driven representational state transfer (REST) web services**. [S.l.]: Google Patents, 2018. US Patent 9,934,029. Citado na página 105.
- CHIMALAKONDA, S.; NORI, K. Accelerating educational technologies using software product lines. In: **IEEE Int. Conf. on Tech. Enhanced Education (ICTEE), 2012**. [S.l.: s.n.], 2012. p. 1–4. Citado na página 60.
- _____. A software engineering perspective for accelerating educational technologies. In: **IEEE 12th Int. Conf. on Advanced Learning Tech. (ICALT), 2012**. [S.l.: s.n.], 2012. p. 754–755. Citado na página 60.
- _____. What makes it hard to apply software product lines to educational technologies? In: . [S.l.: s.n.], 2013. p. 17–20. Citado na página 60.
- CHIMALAKONDA, S.; NORI, K. V. A family of software product lines in educational technologies. **arXiv preprint arXiv:1802.05173**, 2018. Citado na página 25.
- CLEMENTS, P.; NORTHROP, L. **Software Product Lines: Practices and Patterns**. [S.l.]: Addison-Wesley, 2002. (SEI Series in Software Engineering). Citado na página 54.
- COCHRANE, T. mLearning: Why? What? Where? How? In: **Proceedings of the Australasian Society for Computers in Learning in Tertiary Education**. [S.l.: s.n.], 2011. Citado na página 41.
- COCKBURN, A.; BRYANT, A. Do it This Way: Equal Opportunity Programming for Kids. In: **Computer-Human Interaction, 1996. Proceedings., Sixth Australian Conference on**. [S.l.: s.n.], 1996. p. 246–251. Citado na página 32.
- COCKBURN, A.; WILLIAMS, L. Extreme programming examined. In: SUCCI, G.; MARCHESE, M. (Ed.). Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2001. cap. The Costs and Benefits of Pair Programming, p. 223–243. Citado na página 32.

COLLINS, A.; HALVERSON, R. **Rethinking education in the age of technology: The digital revolution and schooling in America**. [S.l.]: Teachers College Press, 2018. Citado na página 157.

CONFORTO, D.; CAVEDINI, P.; MIRANDA, R.; CAETANO, S. Pensamento computacional na educação básica: interface tecnológica na construção de competências do século xxi. **Revista Brasileira de Ensino de Ciências e Matemática**, v. 1, n. 1, 2018. Citado na página 194.

COSTA, C. J.; APARICIO, M. Evaluating Success of a Programming Learning Tool. In: **Proceedings of the Int. Conf. on Information Systems and Design of Communication**. New York, NY, USA: [s.n.], 2014. p. 73–78. Citado na página 32.

CRAIG, A.; COLDWELL-NEILSON, J.; GOOLD, A.; BEEKHUYZEN, J. A Review of E-learning Technologies—Opportunities for Teaching and Learning. In: **Int. Conf. on Computer Supported Education (CSEDU)**. [S.l.: s.n.], 2012. p. 29–41. Citado na página 24.

CROMPTON, H. A historical overview of m-learning. In: **Handbook of mobile learning**. [S.l.]: Routledge, 2013. p. 41–52. Citado nas páginas 25 e 157.

DAGIENE, V.; GUDONIENE, D.; BARTKUTE, R. The integrated environment for learning objects design and storing in semantic web. **International Journal of Computers, Communications & Control**, v. 13, n. 1, 2018. Citado na página 107.

DELIMITROU, C.; KOZYRAKIS, C. Uncovering the security implications of cloud multi-tenancy with bolt. **IEEE Micro**, IEEE, v. 38, n. 3, p. 86–97, 2018. Citado na página 113.

DHUNGANA, D.; GRÄNBACHER, P.; RABISER, R.; NEUMAYER, T. Structuring the modeling space and supporting evolution in software product line engineering. **Journal of Systems and Software**, v. 83, n. 7, p. 1108 – 1122, 2010. {SPLC} 2008. Citado nas páginas 25 e 158.

DIAS, A. de O.; GUERRA, E. M.; SILVEIRA, F. F.; SILVA, T. S. da. Mapping dynamic behavior between different object models in aom. In: SPRINGER. **International Conference on Computational Science and Its Applications**. [S.l.], 2018. p. 233–249. Citado na página 114.

DUARTE FILHO, N. F.; BARBOSA, E. F. A Requirements Catalog for Mobile Learning Environments. In: **Proceedings of the Annual ACM Symposium on Applied Computing**. New York, NY, USA: [s.n.], 2013. p. 1266–1271. Citado nas páginas 25, 43, 44, 63, 75, 78, 81 e 82.

DUNCAN, C.; BELL, T.; TANIMOTO, S. Should your 8-year-old learn coding? In: **Proceedings of the 9th Workshop in Primary and Secondary Computing Education**. New York, NY, USA: ACM, 2014. (WiPSCE '14), p. 60–69. Disponível em: <<http://doi.acm.org/10.1145/2670757.2670774>>. Citado nas páginas 23, 24, 31, 32 e 157.

EL-HUSSEIN, M. O. M.; CRONJE, J. C. Defining Mobile Learning in the Higher Education Landscape. **Educational Technology & Society**, v. 13, n. 3, p. 12–21, 2010. Citado nas páginas 42, 44 e 45.

ESPER, S.; FOSTER, S. R.; GRISWOLD, W. G. CodeSpells: Embodying the Metaphor of Wizardry for Programming. In: . Canterbury, United kingdom: [s.n.], 2013. p. 249 – 254. ISSN 1942647X. Citado na página 24.

- FALVO JUNIOR, V.; DUARTE FILHO, N. F.; OLIVEIRA JR, E.; BARBOSA, E. F. Towards the Establishment of a Software Product Line for Mobile Learning Applications. **Int. Conf. on Software Engineering and Knowledge Engineering (SEKE)**, v. 1, p. 678–683, 2014. Citado nas páginas 25, 28, 60 e 102.
- FAROOQ, U.; SCHAFER, W.; ROSSON, M. B.; CARROLL, J. M. M-education: Bridging the Gap of Mobile and Desktop Computing. In: IEEE. **Proceedings in IEEE Int. Workshop on Wireless and Mobile Technologies in Education**. [S.l.], 2002. p. 91–94. Citado na página 41.
- FELDER, R. M.; SILVERMAN, L. K. *et al.* Learning and teaching styles in engineering education. **Engineering education**, v. 78, n. 7, p. 674–681, 1988. Citado nas páginas 37 e 192.
- FERNANDES, G.; FERREIRA, C. Dificuldades Durante a Produção de e Conteúdos para Formação em Ciência e Tecnologia: quais são e como Solucioná-las? **Revista Brasileira de Informática na Educação**, v. 19, n. 03, 2011. Citado na página 40.
- FILHO, N. F. D.; BARBOSA, E. F. A contribution to the establishment of reference architectures for mobile learning environments. **IEEE Revista Iberoamericana de Tecnologias del Aprendizaje**, v. 10, n. 4, p. 234–241, Nov 2015. Citado na página 108.
- FIORAVANTI, M.; BARBOSA, E. Mlearning-pl: a pedagogical pattern language for mobile learning applications. In: . [S.l.: s.n.], 2018. p. 64. Citado nas páginas 28 e 163.
- FIORAVANTI, M. L.; BARBOSA, E. F. A catalog of pedagogical patterns for learning applications. In: **Proceedings of the 48th Annual Frontiers in Education Conference (FIE 2018), San Jose, USA**. [S.l.: s.n.], 2018. Citado nas páginas 28 e 163.
- _____. Mlearning-pl: a pedagogical pattern language for mobile learning applications. In: **Anais dos Workshops do Congresso Brasileiro de Informática na Educação**. [S.l.: s.n.], 2018. v. 7, n. 1, p. 64. Citado na página 44.
- FIORI, D. R.; GIMENES, I. M. d. S.; MALDONADO, J. C.; OLIVEIRA JUNIOR, E. A. Variability Management in Software Product Line Activity Diagrams. In: International Conference on Distributed Multimedia Systems. **Proceedings of the International Conference on Distributed Multimedia Systems, Skokie, IL: Knowledge Systems Graduate School**, v. 1, p. 89–94, 2012. Citado nas páginas 25 e 158.
- FOFONCA, E.; BRITO, G. da S.; ESTEVAM, M.; CAMAS, N. P. V. **Metodologias pedagógicas inovadoras: contextos da educação básica e da educação superior**. Curitiba: Editora IFPR, 2018. v. 1. Citado nas páginas 24 e 25.
- _____. **Metodologias pedagógicas inovadoras: contextos da educação básica e da educação superior**. Curitiba: Editora IFPR, 2018. v. 2. Citado na página 24.
- FREITAS, E. R. N. de; BICALHO, M. G. P.; SOUZA, M. C. R. F. de; NETTO, C. M. Informática e Educação no Ensino Superior: Reflexões Sobre a Relação com o Saber de Estudantes de Cursos da Área de Computação. **Revista Brasileira de Informática na Educação**, v. 20, p. 69–78, 2012. Citado na página 40.
- FU, Q.-K.; HWANG, G.-J. Trends in mobile technology-supported collaborative learning: A systematic review of journal publications from 2007 to 2016. **Computers Education**, v. 119, p. 129 – 143, 2018. ISSN 0360-1315. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0360131518300046>>. Citado nas páginas 25 e 40.

- GACKENHEIMER, C. What is react? In: _____. **Introduction to React**. Berkeley, CA: Apress, 2015. p. 1–20. Citado nas páginas 128 e 131.
- GALAFASSI, F. P.; GLUZ, J. ao C.; GALAFASSI, C. A Interação e Apropriação Tecnológica de Pessoas com Deficiência Através de Ferramenta Multimídia Acessível. **Revista Brasileira de Informática na Educação**, v. 21, p. 41–52, 2013. Citado na página 40.
- GEORGIEVA, E.; SMRIKAROV, A.; GEORGIEV, T. A General Classification of Mobile Learning Systems. In: **Int. Conf. on Computer Systems and Technologies CompSysTech**. [S.l.: s.n.], 2005. v. 8. Citado nas páginas 43, 44, 68, 75, 78 e 81.
- GOMAA, H. **Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures**. Redwood City, CA, USA: Addison Wesley Longman Publishing Co., Inc., 2004. Citado nas páginas 25, 47, 49, 50 e 158.
- GOMES, A.; MENDES, A. J. An Environment to Improve Programming Education. In: . Bulgária: [s.n.], 2007. v. 285. Citado na página 24.
- GUANA, V.; CORREAL, D. Improving software product line configuration: A quality attribute-driven approach. **Information and Software Technology**, v. 55, n. 3, p. 541 – 562, 2013. Special Issue on Software Reuse and Product Lines Special Issue on Software Reuse and Product Lines. Citado nas páginas 25 e 158.
- GUTIERREZ, M. **Análise de Métodos para Criação de Vocabulário de Gestos para Aplicações em Superfície Multi-Toque (In Portuguese)**. Dissertação (Mestrado) — Escola Politécnica da Universidade de São Paulo, 2015. Citado na página 201.
- HADDAD, W. D. Icts for education. **A Reference Handbook**, 2002. Citado na página 40.
- HARASIM, L. **Learning theory and online technology**. [S.l.]: Routledge, 2012. Citado na página 87.
- HARTANTO, B. **Incorporating Anchored Learning in a C# Intelligent Tutoring System**. Tese (Doutorado) — Queensland University of Technology, 2014. Citado na página 32.
- HASSAN, S.; AZIZ, O. S. Flexible scenarios to build a service oriented architecture on demand. **Journal of Fundamental and Applied Sciences**, University of El Oued, v. 10, n. 5S, 2018. Citado na página 102.
- HAWKRIDGE, D.; VINCENT, T.; HALES, G. **New information technology in the education of disabled children and adults**. [S.l.]: Routledge, 2018. Citado nas páginas 23 e 157.
- HAYTHORNTHWAITE, C.; ANDREWS, R. **E-learning theory and practice**. [S.l.]: Sage Publications, 2011. Citado na página 89.
- HEFLIN, H.; SHEWMAKER, J.; NGUYEN, J. Impact of mobile technology on student attitudes, engagement, and learning. **Computers and Education**, v. 107, p. 91 – 99, 2017. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0360131517300064>>. Citado nas páginas 25 e 40.
- HIPPCHEN, B.; GIESSLER, P.; STEINEGGER, R.; SCHNEIDER, M.; ABECK, S. Designing microservice-based applications by using a domain-driven design approach. **International Journal on Advances in Software (1942-2628)**, v. 10, p. 432 – 445, 12 2017. Citado na página 123.

- HOLZINGER, A. Usability engineering methods for software developers. **Communications of the ACM**, ACM, v. 48, n. 1, p. 71–74, 2005. Citado na página 144.
- HUANG, Y.-M.; HUANG, T.-C.; HSIEH, M.-Y. Using Annotation Services in a Ubiquitous Jigsaw Cooperative Learning Environment. **Educational Technology & Society**, v. 11, n. 2, p. 3–15, 2008. Citado na página 42.
- IBGE. **Instituto Brasileiro de Geografia e Estatística. Características gerais dos domicílios e dos moradores : 2017**. 2018. Disponível em <<https://biblioteca.ibge.gov.br/index.php/biblioteca-catalogo?view=detalhes&id=2101566>> Último acesso em: 13 nov. 2018. Citado na página 40.
- IEA. **Diagnósticos e Propostas para a Educação Básica Brasileira**. São Paulo, SP, Brasil, 2018. Último acesso em: 07 nov. 2018. Citado nas páginas 25 e 157.
- IEEE. IEEE Std. for Learning Object Metadata. **Std 1484.12.1-2002**, p. i–32, 2002. Citado na página 61.
- ILLERIS, K. **Contemporary Theories of Learning: Learning Theorists ... In Their Own Words**. [S.l.]: Taylor & Francis, 2018. Citado na página 24.
- ISO/IEC JTC1/SC 7. **ISO/IEC 25010: Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models**. Disponível em: <https://www.iso.org/standard/35733.html>, 2011. Acessado em 19 Out. 2018. Citado nas páginas 78 e 81.
- ISO/IEC/IEEE. **ISO/IEC/IEEE Systems and Software Engineering - System Life Cycle Processes**. [S.l.], 2008. c1-84 p. Citado nas páginas 78 e 81.
- IVANOVI, M.; XINOGALOS, S.; KOMLENOV, A. Usage of technology enhanced educational tools for delivering programming courses. **International Journal of Emerging Technologies in Learning (iJET)**, International Association of Online Engineering, Kassel, Germany, v. 6, n. 4, p. 23–30, December 2011. Citado nas páginas 13, 79 e 80.
- JACOBSON, I.; GRISS, M.; JONSSON, P. **Software Reuse: Architecture, Process and Organization for Business Success**. [S.l.]: Addison-Wesley, 1997. Citado na página 47.
- JAGUŠT, T.; BOTIČKI, I.; SO, H.-J. A review of research on bridging the gap between formal and informal learning with technology in primary school contexts. **Journal of Computer Assisted Learning**, Wiley Online Library, 2018. Citado na página 93.
- JALIL, A.; BEER, M.; CROWTHER, P. Pedagogical requirements for mobile learning: A review on mobile learning task model. **Journal of Interactive Media in Education**, Ubiquity Press, v. 2015, n. 1, 2015. Citado nas páginas 75, 78, 81, 87 e 88.
- JAMIESON-PROCTOR, R. Transforming learning with information and communication technologies: Insights from three decades of research. Australian Council for Educational Research (ACER), 2018. Citado na página 25.
- JOHNSON, L.; ADAMS, S.; CUMMINS, M. **NMC Horizon Report: 2016 Higher Education Edition**. [S.l.], 2016. Disponível em: <http://cdn.nmc.org/media/2016-personalized-learning-student-success-summit-report_public.pdf>. Citado nas páginas 23 e 157.

JONASSEN, D. **Computadores x Ferramentas Cognitivas: Desenvolver o Pensamento Crítico nas Escolas**. [S.l.]: Porto, 2007. Citado na página 71.

JUNIOR, E. A. O.; GIMENES, I. M. S.; MALDONADO, J. C. Systematic Management of Variability in UML-based Software Product Lines. **Journal of Universal Computer Science**, v. 16, n. 17, p. 2374–2393, 2010. Disponível em: <<http://dblp.uni-trier.de/db/journals/jucs/jucs16>>. Citado nas páginas 25, 49 e 63.

KAMAL, A. **Enabling Techniques to support Reliable Smartphone-Based Motion Gesture Interaction**. Dissertação (Mestrado), 2015. Disponível em: <<http://hdl.handle.net/10012/9030>>. Citado na página 201.

KANG, K. C.; COHEN, S. G.; HESS, J. A.; NOVAK, W. E.; PETERSON, A. S. **Feature-Oriented Domain Analysis (FODA) Feasibility Study**. [S.l.], 1990. Citado nas páginas 48, 49, 63 e 101.

KANG, K. C.; KIM, S.; LEE, J.; KIM, K.; SHIN, E.; HUH, M. FORM: A Feature-oriented Reuse Method with Domain-specific Reference Architectures. **Ann. Softw. Eng.**, Red Bank, NJ, USA, v. 5, p. 143–168, 1998. Citado na página 49.

KARLESKY, M.; WILLIAMS, G.; BEREZA, W.; FLETCHER, M. Mocking the embedded world: Test-driven development, continuous integration, and design patterns. **ESC**, v. 413, 05 2007. Citado nas páginas 125 e 127.

KEARNEY, M.; SCHUCK, S.; BURDEN, K.; AUBUSSON, P. Viewing mobile learning from a pedagogical perspective. **Research in learning technology**, v. 20, 2012. Citado na página 81.

KELLEHER, C. Barriers to programming engagement. **Advances in gender and education**, v. 1, n. 1, p. 5–10, 2009. Citado nas páginas 37 e 192.

KHADDAGE, F.; CHRISTENSEN, R.; LAI, W.; KNEZEK, G.; NORRIS, C.; SOLOWAY, E. A Model Driven Framework to Address Challenges in a Mobile Learning Environment. **Education and Information Technologies**, Springer US, p. 1–16, 2015. Citado nas páginas 43, 44 e 45.

KHALID, M. S.; BUUS, L. A theoretical framework mapping barriers of integrating and adopting educational technology. **Research and Practice in Technology Enhanced Learning**, Submitted, 07 2013. Citado na página 157.

KIM, S.; THIESSEN, P. A.; CHENG, T.; YU, B.; BOLTON, E. E. An update on pug-rest: Restful interface for programmatic access to pubchem. **Nucleic acids research**, 2018. Citado na página 105.

KITCHENHAM, B.; CHARTERS, S. **Guidelines for performing Systematic Literature Reviews in Software Engineering**. [S.l.], 2007. Citado nas páginas 58 e 66.

KLASSEN, A.; EIBRINK-LUNZENAUER, M.; GLOGGLER, T. Requirements for mobile learning applications in higher education. In: IEEE. **Multimedia (ISM), 2013 IEEE International Symposium on**. [S.l.], 2013. p. 492–497. Citado na página 81.

KONG, J. S.-L.; KWOK, R. C.-W.; FANG, Y. The effects of peer intrinsic and extrinsic motivation on mmog game-based collaborative learning. **Information & Management**, Elsevier, v. 49, n. 1, p. 1–9, 2012. Citado na página 89.

- KOOLE, M. L. A Model for Framing Mobile Learning. **Mobile learning: Transforming the delivery of education and training**, v. 1, n. 2, p. 25–47, 2009. Citado nas páginas 42, 43 e 44.
- KRUEGER, C. Easing the transition to software mass customization. In: **Software Product-Family Engineering**. [S.l.]: Springer Berlin Heidelberg, 2002. v. 2290, p. 282–293. Citado nas páginas 49, 52, 62, 65 e 125.
- KRUEGER, C.; CLEMENTS, P. Feature-based systems and software product line engineering with gears from biglever. In: **ACM. Proceedings of the 22nd International Conference on Systems and Software Product Line-Volume 2**. [S.l.], 2018. p. 1–4. Citado nas páginas 53, 65 e 77.
- KURGUN, H.; KURGUN, O. A.; AKTAŞ, E. What does web 4.0 promise for tourism ecosystem? a qualitative research on tourism ecosystem stakeholders' awareness. **Journal of Tourism and Hospitality Management**, v. 6, n. 1, p. 55–65, 2018. Citado na página 41.
- LAHTINEN, E.; ALA-MUTKA, K.; JÄRVINEN, H.-M. A Study of the Difficulties of Novice Programmers. **ACM Special Interest Group on Computer Science Education**, New York, NY, USA, v. 37, n. 3, p. 14–18, 2005. Citado na página 32.
- LEONG, F.; AUSTIN, J. **The Psychology Research Handbook: A Guide for Graduate Students and Research Assistants**. SAGE Publications, 2005. Disponível em: <<https://books.google.com.br/books?id=tb5yAwAAQBAJ>>. Citado na página 185.
- LIAO, Y. C.; BRIGHT, G. W. Effects of Computer Programming on Cognitive Outcomes: a Meta-Analysis. **Journal of Educational Computing Research**, p. 251–266, 1991. Citado na página 32.
- LIKERT, R. A Technique for the Measurement of Attitudes. **Archives of Psychology**, v. 22, n. 140, p. 1–55, 1932. Citado na página 145.
- LINDEN, F. J.; SCHMID, K.; ROMMES, E. **Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering**. Berlin: Springer, 2007. Citado nas páginas 47, 53, 100, 101 e 116.
- LIU, Y.; HAN, S.; LI, H. Understanding the Factors Driving M-learning Adoption: a Literature Review. **Campus-Wide Information Systems**, Emerald Group Publishing Limited, v. 27, n. 4, p. 210–226, 2010. Citado nas páginas 44 e 45.
- LIYANAGE, M. P. P.; GUNAWARDENA, K. L.; HIRAKAWA, M. Using learning styles to enhance learning management systems. **ICTer**, v. 7, n. 2, 2014. Citado nas páginas 37 e 192.
- LÓPEZ, A. R.; GARCÍA-PEÑALVO, F. J. Relationship of knowledge to learn in programming methodology and evaluation of computational thinking. In: **ACM. Proceedings of the Fourth International Conference on Technological Ecosystems for Enhancing Multiculturality**. [S.l.], 2016. p. 73–77. Citado na página 32.
- MALAVOLTA, I.; RUBERTO, S.; SORU, T.; TERRAGNI, V. Hybrid mobile apps in the google play store: An exploratory investigation. In: **Proceedings of the Second ACM International Conference on Mobile Software Engineering and Systems**. Piscataway, NJ, USA: IEEE Press, 2015. (MOBILESoft '15), p. 56–59. Disponível em: <<http://dl.acm.org/citation.cfm?id=2825041.2825051>>. Citado na página 150.

MARCOLINO, A.; JUNIOR, E. A. O.; GIMENES, I. M. S.; MALDONADO, J. C. Towards the Effectiveness of a Variability Management Approach at Use Case Level. **Int. Conf. on Software Engineering & Knowledge Engineering**, v. 1., p. 214–219, 2013. Citado nas páginas [49](#) e [51](#).

MARCOLINO, A.; OLIVEIRAJR, E.; GIMENES, I. M. de S. Variability Identification and Representation in Software Product Line UML Sequence Diagrams: Proposal and Empirical Study. In: **2014 Brazilian Symposium on Software Engineering, Maceió, Brazil, Set. 28 - October 3, 2014**. [S.l.: s.n.], 2014. p. 141–150. Citado nas páginas [51](#) e [101](#).

MARCOLINO, A.; OLIVEIRAJR, E.; GIMENES, I. M. de S.; BARBOSA, E. F. Empirically Based Evolution of a Variability Management Approach at UML Class Level. In: **IEEE 38th Annual Computer Software and Applications Conference, COMPSAC, Vasteras, Sweden, Julho 21-25**. [S.l.: s.n.], 2014. p. 354–363. Citado na página [101](#).

MARCOLINO, A. S.; BARBOSA, E. Towards an m-learning requirements catalog for the development of educational applications for the teaching of programming. In: **2016 IEEE Frontiers in Education Conference (FIE)**. [S.l.: s.n.], 2016. p. 1–5. Citado nas páginas [96](#) e [101](#).

MARCOLINO, A. S.; BARBOSA, E. F. Linhas de Produto de Software no Domínio Educacional: Um Mapeamento Sistemático. **Simpósio Brasileiro de Informática na Educação**, v. 1, p. 239–249, 2015. Citado nas páginas [26](#), [46](#), [65](#), [76](#), [78](#), [101](#), [115](#), [157](#) e [158](#).

_____. Softwares Educacionais para o Ensino de Programação: Um Mapeamento Sistemático. **Simpósio Brasileiro de Informática na Educação**, v. 1, p. 190–199, 2015. Citado nas páginas [25](#), [26](#), [37](#), [39](#), [73](#), [77](#), [157](#), [158](#) e [194](#).

_____. Towards an m-learning requirements catalog for the development of educational applications for the teaching of programming. In: **2016 IEEE Frontiers in Education Conference (FIE), 2016, Erie (PA)**, p. 1–5, 2016. Citado nas páginas [24](#) e [25](#).

_____. A survey on problems related to the teaching of programming in brazilian educational institutions. In: **2017 IEEE Frontiers in Education Conference (FIE)**, p. 1–9, Oct 2017. Citado nas páginas [23](#), [24](#), [32](#), [39](#), [77](#) e [151](#).

MARCOLINO, A. S.; BARBOSA, E. F. Towards a Software Product Line Architecture to Build m-learning Applications for The Teaching of Programming. In: **Proceedings of the 50th Hawaii International Conference on System Sciences**. [S.l.: s.n.], 2017. Citado na página [112](#).

MARCOLINO, A. S.; OLIVEIRAJR, E. A. Comparing SMarty and PLUS for Variability Identification and Representation at Product-Line UML Class Level: A Controlled Quasi-Experiment, *Journal = JCS*, volume = 13, number = 11, pages = 617–632, year = 2017. Citado nas páginas [51](#) e [102](#).

MARCOLINO, A. S.; OLIVEIRAJR, E. A.; GIMENES, I. M. de S.; BARBOSA, E. F. Variability resolution and product configuration with smarty: An experimental study on uml class diagrams. **JCS**, v. 13, p. 307–319, 2017. Citado na página [51](#).

MARCOLINO, A. S.; SANTOS, A. S.; SCHAEFER, M. A. T.; BARBOSA, E. F. Towards a catalog of gestures for m-learning applications for the teaching of programming. In: **Proceedings of the 48th Annual Frontiers in Education Conference (FIE 2018), San Jose, USA**. [S.l.: s.n.], 2018. Citado na página [156](#).

- MEDEIROS, A. C. S. **Processo de Desenvolvimento de Gestos para Interface de Usuário (In Portuguese)**. Dissertação (mathesis), maio 2015. Disponível em: <<http://tede.biblioteca.ufpb.br:8080/handle/tede/7863>>. Citado na página 201.
- METZGER, A.; POHL, K. Software product line engineering and variability management: Achievements and challenges. In: **Proceedings of the on Future of Software Engineering**. New York, NY, USA: ACM, 2014. (FOSE 2014), p. 70–84. Citado nas páginas 25 e 101.
- MICCIOLO, M.; ARROYO, I.; HARRISON, A.; HULSE, T.; OTTMAR, E. The wearable learning cloud platform for the creation of embodied multiplayer math games. In: SPRINGER. **International Conference on Artificial Intelligence in Education**. [S.l.], 2018. p. 220–224. Citado na página 40.
- MICROSOFT. **The Repository Pattern**. <https://msdn.microsoft.com/en-us/library/ff649690.aspx>: [s.n.], 2017. Disponível em: <<https://msdn.microsoft.com/en-us/library/ff649690.aspx>>. Citado na página 114.
- MOREIRA, F.; ROCHA, Á. **A Special Issue on Disruption of higher education in the 21st century due to ICTs**. [S.l.]: Elsevier, 2018. Citado nas páginas 40, 42, 43, 44 e 45.
- MURRAY, T.; BLESSING, S.; AINSWORTH, S. **Authoring Tools for Advanced Technology Learning Environments: Toward Cost-Effective Adaptive, Interactive and Intelligent Educational Software**. Springer, 2003. ISBN 9781402017728. Disponível em: <<https://books.google.com.br/books?id=iLcMaweQoxQC>>. Citado na página 24.
- NADAREISHVILI, I.; RONNIE, M.; MCLARTY, M.; AMUNDSEN, M. **Microservice Architecture**. [S.l.]: O'Really, 2016. Citado na página 115.
- NEWMAN, S. **Building Microservices**. 1st. ed. [S.l.]: O'Reilly Media, Inc., 2015. ISBN 1491950358, 9781491950357. Citado nas páginas 114, 115 e 116.
- NG, P. H.; CHIU, M. W. Where programming skills meet the social needs. In: **Service-Learning for Youth Leadership**. [S.l.]: Springer, 2019. p. 165–184. Citado na página 31.
- NORTHROP, L. M. **Software Product Lines Essentials**. 2008. Acessado em 07 out. 2013. Disponível em: <<http://www.sei.cmu.edu/library/assets/spl--essentials.pdf>>. Citado nas páginas 48, 49 e 54.
- NORTHROP, L. M.; CLEMENTS, P. C. **A Framework for Software Product Line Practice, Version 5.0**. 2009. Acessado em 07 out. 2013. Disponível em: <<http://www.sei.cmu.edu/productlines/tools/framework/>>. Citado nas páginas 46, 47, 48 e 50.
- OBERWEIS, A.; PANKRATIUS, V.; STUCKY, W. Product lines for digital information products. **Information Systems**, v. 32, n. 6, p. 909 – 939, 2007. Citado na página 60.
- OLIVEIRAJR, E. A.; GIMENES, I. M. S.; MALDONADO, J. C. Systematic Evaluation of Software Product Line Architectures. **Journal of Universal Computer Science**, v. 19, n. 1, p. 25–52, jan 2013. Disponível em: <http://www.jucs.org/jucs/_19_1/systematic_evaluation_of_software>. Citado nas páginas 51 e 115.
- PAHL, C.; JAMSHIDI, P.; ZIMMERMANN, O. Architectural principles for cloud software. **ACM Transactions on Internet Technology (TOIT)**, ACM, v. 18, n. 2, p. 17, 2018. Citado na página 102.

PANKRATIUS, V.; STUCKY, W.; VOSSEN, G. Aspect-oriented re-engineering of e-learning courseware. **Learning Organization**, v. 12, n. 5, p. 457–470, 2005. Citado na página 60.

PAPADAKIS, S.; KALOGIANNAKIS, M.; ORFANAKIS, V.; ZARANIS, N. The appropriateness of scratch and app inventor as educational environments for teaching introductory programming in primary and secondary education. In: **Early Childhood Development: Concepts, Methodologies, Tools, and Applications**. [S.l.]: IGI Global, 2019. p. 797–819. Citado nas páginas 24 e 31.

PARSONS, D.; RYU, H.; CRANSHAW, M. A Study of Design Requirements for Mobile Learning Environments. In: **Int. Conf. on Advanced Learning Technologies and Technology-enhanced Learning (ICALT)**. [S.l.: s.n.], 2006. p. 96–100. Citado na página 44.

_____. A design requirements framework for mobile learning environments. **Journal of Computers**, v. 2, n. 4, p. 1–8, 2007. Citado nas páginas 75, 78 e 81.

PATTERNS, M. **Microsoft Application Architecture Guide**. 2nd. ed. [S.l.]: Microsoft Press, 2009. Citado na página 102.

PEA, R. D.; KURLAND, D. On the Cognitive Effects of Learning Computer Programming. **New Ideas in Psychology**, v. 2, n. 2, p. 137 – 168, 1984. Citado na página 32.

PEARS, A.; SEIDMAN, S.; MALMI, L.; MANNILA, L.; ADAMS, E.; BENNEDSEN, J.; DEVLIN, M.; PATERSON, J. A Survey of Literature on the Teaching of Introductory Programming. **ACM Special Interest Group on Computer Science Education (SIGCSE)**, v. 39, p. 204–223, dez. 2007. Citado na página 70.

PERERA, K.; PERERA, I. A rule-based system for automated generation of serverless-microservices architecture. In: IEEE. **2018 IEEE International Systems Engineering Symposium (ISSE)**. [S.l.], 2018. p. 1–8. Citado na página 123.

PETERSEN, K.; FELDT, R.; MUJTABA, S.; MATTSSON, M. Systematic mapping studies in software engineering. In: **12th Int. Conf. on Evaluation and Assessment in Software Engineering**. Swinton, UK, UK: British Computer Society, 2008. (EASE'08), p. 68–77. Disponível em: <<http://dl.acm.org/citation.cfm?id=2227115.2227123>>. Citado na página 58.

PETTICREW, M.; ROBERTS, H. **Systematic Reviews in the Social Sciences: A Practical Guide**. [S.l.]: Blackwell Publishing, 2006. Citado nas páginas 58 e 67.

PIMMER, C.; MATEESCU, M.; GRÖHBIEL, U. Mobile and ubiquitous learning in higher education settings. a systematic review of empirical studies. **Computers in Human Behavior**, v. 63, p. 490 – 501, 2016. Citado nas páginas 25 e 40.

PITEIRA, M.; COSTA, C. Computer Programming and Novice Programmers. In: **Proceedings of the Workshop on Information Systems and Design of Communication**. New York, NY, USA: [s.n.], 2012. p. 51–53. Citado na página 32.

_____. Learning Computer Programming: Study of Difficulties in Learning Programming. In: **Proceedings of the Int. Conf. on Information Systems and Design of Communication**. [S.l.: s.n.], 2013. Citado na página 32.

POHL, K.; BOCKLE, G.; LINDEN, F. **Software Product Line Engineering Foundations, Principle, and Techniques**. [S.l.]: Secaucus, NJ, USA: Springer-Verlag, 2005. Citado nas páginas 25, 28, 49, 50, 51, 53, 101, 125, 135 e 158.

- POHL, K.; METZGER, A. Software product lines. In: _____. **The Essence of Software Engineering**. Cham: Springer International Publishing, 2018. p. 185–201. Citado na página 118.
- POPAT, S.; STARKEY, L. Learning to code or coding to learn? a systematic review. **Computers & Education**, Elsevier, v. 128, p. 365–376, 2019. Citado nas páginas 24 e 31.
- PRESS, I. C. S.; PRESS, A. **Joint Task Force for Computing Curricula**. 2005. Disponível em <<https://www.acm.org/education/curricula-recommendations>> Último acesso em: 13 Jan. 2019. Citado nas páginas 23, 31 e 32.
- PRESSMAN, R. **Software Engineering: A Practitioner's Approach**. 8. ed. New York, NY, USA: McGraw-Hill, Inc., 2015. Citado nas páginas 24, 46 e 116.
- RABISER, R.; SCHMID, K.; BECKER, M.; BOTTERWECK, G.; GALSTER, M.; GROHER, I.; WEYNS, D. A study and comparison of industrial vs. academic software product line research published at splc. In: ACM. **Proceedings of the 22nd International Conference on Systems and Software Product Line-Volume 1**. [S.l.], 2018. p. 14–24. Citado nas páginas 53 e 65.
- REINHARTZ-BERGER, I.; KEMELMAN, M. Extracting core requirements for software product lines. **Requirements Engineering**, Springer, p. 1–19, 2019. Citado na página 53.
- REIS, H. M. **Concepção de um software de geometria interativa utilizando interfaces gestuais para dispositivos móveis (In Portuguese)**. Dissertação (mathesis) — Universidade São Paulo, out. 2014. Citado na página 201.
- REIS, H. M.; BORGES, S. S.; GRIFFITHS, P. A.; MORO, L. F.; ISOTANI, S. Interação em interfaces de softwares de geometria interativa: Um mapeamento sistemático. In: **Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)**. [S.l.: s.n.], 2013. v. 24, n. 1, p. 255. Citado na página 201.
- RITTER, S.; MURRAY, R. C.; HAUSMANN, R. G. Educational software design: Education, engagement, and productivity concerns. In: **End-User Considerations in Educational Technology Design**. [S.l.]: IGI Global, 2018. p. 35–51. Citado na página 24.
- ROBINS, A.; ROUNTREE J. AN ROUNTREE, N. Learning and teaching programming: A review and discussion. **Computer Science Education**, p. 137–172, 2003. Citado na página 32.
- ROSCHELLE, J.; PEA, R. A walk on the WILD Side: How Wireless Handhelds May Change CSCL. In: **Proceedings of the Conf. on Computer Support for Collaborative Learning: Foundations for a CSCL Community**. [S.l.]: International Society of the Learning Sciences, 2002. p. 51–60. Citado nas páginas 44 e 45.
- ROSE, L.; PAIGE, R.; KOLOVOS, D.; POLACK, F. The epsilon generation language. In: SCHIEFERDECKER, I.; HARTMAN, A. (Ed.). **Model Driven Architecture – Foundations and Applications**. [S.l.]: Springer Berlin Heidelberg, 2008. v. 5095, p. 1–16. Citado na página 63.
- ROSHAN, A.; DARYUSH, N.; BAHAREH, M. Mobile learning: The Introduction of Online and Offline Mobile Learning Systems Based on Cellular Phones. In: **Fourth International Conference on E-Learning and E-Teaching (ICELET)**. [S.l.: s.n.], 2013. p. 36–39. Citado nas páginas 43, 44 e 45.

RUBINE, D. Specifying gestures by example. **SIGGRAPH Comput. Graph.**, ACM, New York, NY, USA, v. 25, n. 4, p. 329–337, jul. 1991. ISSN 0097-8930. Citado na página 201.

SALES, C. G.; DANTAS, V. F. ProGame: um jogo para o ensino de algoritmos e programação. **Simpósio Brasileiro de Informática na Educação**, 2010. Citado na página 24.

SANCHEZ, P.; GARCIA-SAlZ, D.; ZORRILLA, M. Software product line engineering for e-learning applications: A case study. In: **Int. Symp. on Comp. in Education**. [S.l.: s.n.], 2012. p. 1–6. Citado na página 60.

SANDOVAL, K. **Building a Backend for Frontend (BFF) For Your Microservices**. 2017. Disponível em <<https://nordicapis.com/building-a-backend-for-frontend-shim-for-your-microservices/>>, Último acesso em 11, Dez. 2018. Disponível em: <<https://nordicapis.com/building-a-backend-for-frontend-shim-for-your-microservices/>>. Citado na página 123.

SANTOS, A. S.; MARCOLINO, A. S.; BARBOSA, E. F. Proposta e Validação de um Vocabulário de Gestos para Aplicações Educacionais Móveis para o Ensino de Fundamentos de Programação. **26o. Simpósio Internacional de Iniciação Científica e Tecnológica da USP (SIICUSP 2018)**, 2018. Citado na página 156.

SANTOSA, C. P.; SEGATTO, R.; TEIXEIRA, Y. P.; WILLERS, T. F. **Interface Natural na Educação: Um Estudo de Caso no Ensino de Geografia (In Portuguese)**. 2014. Availble in <<http://br-ie.org/pub/index.php/wie/article/view/3106>> Acessado em April, 19th, 2018. Citado na página 201.

SARRAB, M. M-learning in Education: Omani Undergraduate Students Perspective. **Procedia - Social and Behavioral Sciences**, v. 176, p. 834 – 839, 2015. Int. Educational Technology Conference IETC Chicago, IL, USA. Citado nas páginas 40, 41, 43, 44, 45 e 93.

SBC. **Sociedade Brasileira de Computaç. Currículo de Referência da Sociedade Brasileira de Computação para Cursos de Graduação em Computação e Informática**. 2003. Disponível em <<http://www.sbc.org.br/index>> Último acsso em: 17 de Dez. de 2013. Citado nas páginas 23 e 31.

_____. **Currículo de Referência da Sociedade Brasileira de Computação para Cursos de Graduação em Computação e Informática**. 2015. Disponível em <<http://www.sbc.org.br/documentos-da-sbc/summary/131-curriculos-de-referencia/760-curriculo-de-referencia-cc-ec-versao2005>> Último acsso em: 13 de Jan. de 2019. Citado na página 32.

SCHAEFER, M. A. T.; MARCOLINO, A. S.; BARBOSA, E. F. Desenvolvimento de Serviços Educacionais Móveis para uma Linha de Produtos de Software no Domínio de Fundamentos de Programação. **26o. Simpósio Internacional de Iniciação Científica e Tecnológica da USP (SIICUSP 2018)**, 2018. Citado na página 156.

SCHLÜNZEN, E. T. M. Escola Inclusiva e as Novas Tecnologias. **Integração das Tecnologias na Educação, Brasília, DF**, p. 80–84, 2005. Citado na página 40.

SEEHORN, D.; CAREY, S.; FUSCHETTO, B.; LEE, I.; MOIX, D.; O'GRADY-CUNNIFF, D.; OWENS, B. B.; STEPHENSON, C.; VERNON, A. Csta k–12 computer science standards: Revised 2011. ACM, 2011. Citado na página 32.

SEI. **Software Engineering Institute: Hall of Fame**. 2013. <<http://www.splc.net/fame.html>>. Último acesso em: 07 out. 2013. Disponível em: <<http://www.splc.net/fame.html>>. Citado na página 26.

SHARMA, S. K.; KITCHENS, F. L. Web Services Architecture for M-learning. **Electronic Journal on e-learning**, Citeseer, v. 2, n. 1, p. 203–216, 2004. Citado nas páginas 41 e 44.

SHESHASAYEE, A.; BEE, M. N. E-learning: Mode to improve the quality of educational system. In: **Smart Computing and Informatics**. [S.l.]: Springer, 2018. p. 559–566. Citado na página 40.

SINCLAIR, J.; AHO, A.-M. Experts on super innovators: understanding staff adoption of learning management systems. **Higher Education Research & Development**, Taylor & Francis, v. 37, n. 1, p. 158–172, 2018. Citado na página 24.

SOAD, G.; FILHO, N. F. D.; BARBOSA, E. Quality evaluation of mobile learning applications. In: . [S.l.: s.n.], 2016. p. 1–8. Citado na página 28.

SOAD, G. W.; FIORAVANTI, M. L.; FALVO, V.; MARCOLINO, A.; FILHO, N. F. D.; BARBOSA, E. F. Reqml-catalog: The road to a requirements catalog for mobile learning applications. In: IEEE. **Frontiers in Education Conference (FIE)**. [S.l.], 2017. p. 1–9. Citado nas páginas 25, 28, 75 e 157.

SOAD, G. W.; FIORAVANTI, M. L.; FALVO, V.; MARCOLINO, A.; FILHO, N. F. D.; BARBOSA, E. F. Reqml-catalog: The road to a requirements catalog for mobile learning applications. In: **2017 IEEE Frontiers in Education Conference (FIE)**. [S.l.: s.n.], 2017. p. 1–9. Citado nas páginas 43 e 44.

SOMMERVILLE, I. **Software Engineering**. 9. ed. Harlow, England: Addison-Wesley, 2010. Citado nas páginas 24, 25, 46, 75, 76 e 127.

SOUZA, D. M.; BATISTA, M. H. S.; BARBOSA, E. F. Problemas e Dificuldades no Ensino de Programação: Um Mapeamento Sistemático. In: **Revista Brasileira de Informática na Educação**. [S.l.: s.n.], 2015. Citado nas páginas 23, 24, 28, 32, 33, 37, 38, 76, 79, 151, 157, 160, 185 e 193.

SOUZA, J. S. de; LOPES, A. S. B. Estimulando o pensamento computacional e o raciocínio lógico no ensino fundamental por meio da obi e computação desplugada. In: **Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)**. [S.l.: s.n.], 2018. v. 29, n. 1, p. 1893. Citado na página 194.

STRAUB, E. T. Understanding technology adoption: Theory and future directions for informal learning. **Review of educational research**, SAGE Publications Sage CA: Los Angeles, CA, v. 79, n. 2, p. 625–649, 2009. Citado na página 93.

STRAY, V.; MOE, N. B.; SJOBERG, D. I. Daily stand-up meetings: Start breaking the rules. **IEEE Software**, IEEE, 2018. Citado na página 88.

THÜM, T.; KÄSTNER, C.; BENDUHN, F.; MEINICKE, J.; SAAKE, G.; LEICH, T. Featureide: An extensible framework for feature-oriented software development. **Science of Comp. Prog.**, Elsevier, v. 79, p. 70–85, 2014. Citado na página 100.

TILKOV, S.; VINOSKI, S. Node.js: Using javascript to build high-performance network programs. **IEEE Internet Computing**, v. 14, n. 6, p. 80–83, Nov 2010. ISSN 1089-7801. Citado na página [127](#).

TING, R.-L. Mobile learning: Current Trend and Future Challenges. In: IEEE. **IEEE Int. Conf. on Advanced Learning Technologies (ICALT)**. [S.l.], 2005. p. 603–607. Citado na página [44](#).

TOTEJA, R.; KUMAR, S. Usefulness of M-Devices in Education: A Survey. **Procedia - Social and Behavioral Sciences**, v. 67, n. 0, p. 538 – 544, 2012. ISSN 1877-0428. Citado nas páginas [43](#), [44](#) e [45](#).

TRAXLER, J. Defining, discussing and evaluating mobile learning: The moving finger writes and having writ.... **The International Review of Research in Open and Distributed Learning**, v. 8, n. 2, 2007. Citado na página [42](#).

UHLIG, R.; NEIGER, G.; RODGERS, D.; SANTONI, A. L.; MARTINS, F. C.; ANDERSON, A. V.; BENNETT, S. M.; KAGI, A.; LEUNG, F. H.; SMITH, L. Intel virtualization technology. **Computer**, IEEE, v. 38, n. 5, p. 48–56, 2005. Citado na página [41](#).

VARGHESE, B.; BUYYA, R. Next generation cloud computing: New trends and research directions. **Future Generation Computer Systems**, Elsevier, v. 79, p. 849–861, 2018. Citado na página [41](#).

VATAVU, R.-D.; CRAMARIUC, G.; SCHIPOR, D. M. Touch interaction for children aged 3 to 6 years: Experimental findings and relationship to motor skills. **International Journal of Human-Computer Studies**, Elsevier, v. 74, p. 54–76, 2015. Citado na página [201](#).

WANGENHEIM, C. G. von; NUNES, V. R.; SANTOS, G. D. dos. Ensino de computação com SCRATCH no ensino fundamental—um estudo de caso. **Revista Brasileira de Informática na Educação**, v. 22, n. 3, p. 115–125, 2014. Citado nas páginas [32](#), [39](#) e [194](#).

WATSON, C.; LI, F. W. Failure rates in introductory programming revisited. In: ACM. **Proceedings of the 2014 conference on Innovation & technology in computer science education**. [S.l.], 2014. p. 39–44. Citado na página [31](#).

WATSON, C.; LI, F. W. B.; LAU, R. W. H. Learning programming languages through corrective feedback and concept visualisation. In: **Proceedings of the 10th International Conference on Advances in Web-Based Learning**. Berlin, Heidelberg: Springer-Verlag, 2011. (ICWL'11), p. 11–20. ISBN 978-3-642-25812-1. Citado nas páginas [13](#) e [80](#).

WEISS, D. M.; LAI, C. T. R. **Software Product-line Engineering: a Family-based Software Development Process**. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999. ISBN 0-201-69438-7. Citado nas páginas [47](#) e [49](#).

WESTERMAN, W. C. **Gesture learning**. [S.l.]: Google Patents, 2016. US Patent 9,311,528. Citado na página [201](#).

WILSON, J. **Node.js 8 the Right Way: Practical, Server-side Javascript that Scales**. [S.l.]: Pragmatic Bookshelf, 2018. Citado na página [127](#).

WING, J. M. Computational thinking and thinking about computing. **Philosophical transactions of the royal society of London A: mathematical, physical and engineering sciences**, The Royal Society, v. 366, n. 1881, p. 3717–3725, 2008. Citado na página [32](#).

WOBBROCK, J. O.; WILSON, A. D.; LI, Y. Gestures without libraries, toolkits or training: A \$1 recognizer for user interface prototypes. In: **Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology**. New York, NY, USA: ACM, 2007. (UIST '07), p. 159–168. Citado nas páginas [153](#) e [201](#).

ZHAI, S.; KRISTENSSON, P. O.; APPERT, C.; ANDERSON, T. H.; CAO, X. *et al.* Foundational issues in touch-surface stroke gesture design: an integrative review. **Foundations and Trends® in Human–Computer Interaction**, Now Publishers, Inc., v. 5, n. 2, p. 97–205, 2012. Citado nas páginas [201](#) e [206](#).

ZHOU, D.-D.; ZHANG, H.-M.; ZHONG, S.-C.; ZHANG, Z.; MA, X.-C. The design of codeless e-learning animation materials developing platform. In: **Int. Conf. on Machine Learning and Cybernetics**. [S.l.: s.n.], 2009. v. 6, p. 3585–3590. Citado na página [60](#).

ZORRILLA, M. **Business Intelligence Applications and the Web: Models, Systems and Technologies: Models, Systems and Technologies**. [S.l.]: Business Science Reference, 2011. (Advances in Business Information Systems and Analytics:). Citado na página [64](#).

QUESTIONÁRIO SOBRE PROBLEMAS NO ENSINO E APRENDIZAGEM DE PROGRAMAÇÃO

A.1 Questionário sobre Problemas no Ensino e Aprendizagem de Programação

Com o objetivo de identificar se as categorias de problemas internacionais especificadas no mapeamento de [Souza, Batista e Barbosa \(2015\)](#) correspondem aos problemas enfrentados no Brasil, um questionário *online* foi conduzido com 59 professores das redes pública e privada, de diferentes localidades do Brasil. Além de solicitar que os participantes indicassem quais das categorias de problemas correspondem aos mais evidentes em suas instituições, os mesmos foram questionados quanto a estratégias e softwares que utilizam em sala, e se enfrentam outros problemas relevantes. De modo complementar, para buscar compreender as dificuldades por parte dos alunos, um questionário específico com estes também foi aplicado. O planejamento, metodologia, aplicação e análise são detalhados a seguir.

As seguintes tarefas foram realizadas para a elaboração do questionário: (i) definição dos objetivos a serem alcançados e definição das questões seguindo as diretrizes propostas por [Leong e Austin \(2005\)](#); (ii) validação das questões com pesquisadores do laboratório de Computação Aplicada à Educação e Tecnologia Social Avançada (CAEd) e do laboratório de Engenharia de Software (LabES), ambos da Universidade de São Paulo (ICMC-USP) e (iii) execução de uma aplicação piloto.

Após a aplicação do questionário para membro do laboratório que não havia participado da validação das questões, o questionário foi disponibilizado aos professores por meio de lista

de e-mails da Sociedade Brasileira de Computação¹. Já o questionário destinado aos alunos foi aplicado em dois momentos, também online, como pré e pós-teste em um curso de fundamentos de programação para web realizado na Universidade de São Paulo por meio da iniciativa *Codifique*². Este curso aborda tópicos elementares sobre programação web com JavaScript³ e é oferecido anualmente para a comunidade em geral.

A decisão pela divisão da aplicação do questionário aos alunos teve como objetivo avaliar questões que possam ser influenciadas antes e depois do contato com os tópicos de programação.

O questionário para os professores (QP) foi composto por três questões. A primeira refere-se à classificação das categorias dos problemas de programação, segundo os problemas que eles encontram em suas aulas. A segunda procurou saber estratégias e/ou soluções para tais problemas. A terceira e última questão procurou identificar quais outros possíveis problemas são enfrentados pelos mesmos nas disciplinas de programação.

Já o questionário para os alunos foi composto por cinco questões no pré-teste e oito no pós-teste, sendo que apenas quatro questões do pós-teste foram utilizadas nesta pesquisa, visto que as demais referiam-se especificamente a tópicos ministrados no curso. As cinco questões aplicadas no pré-teste (QEA) procuraram identificar os motivos pelos quais os alunos estavam realizando o curso de programação; a importância de tal disciplina para o futuro deles; se adotariam tecnologias móveis para aprender a programar; possíveis estratégias e/ou abordagens que gostariam de ver em aulas de programação; e que outros meios seriam indicados para apoiar o processo de aprendizagem dos mesmos.

No que tange às quatro questões aplicadas no pós-teste (QEB), uma delas procurou reconfirmar as respostas dadas pelos alunos no pré-teste no que diz respeito ao uso de tecnologias móveis; uma buscou identificar as dificuldades que possuíam ou adquiriram no decorrer do curso; uma buscou identificar os motivos de tais dificuldades e a última delas procurou identificar sugestões para aperfeiçoamento do curso oferecido.

O esquema apresentado na Figura 54 apresenta um panorama geral do questionário aplicado. A Figura 54 (a) apresentam de quê regiões os professores participantes são; a Figura 54 (a) apresenta o perfil dos professores em relação ao seu grau de escolaridade; a Figura 54 (c) apresenta as atividades realizadas na execução da pesquisa e; a Figura 54 (d) apresenta o perfil dos alunos participantes.

A.1.1 *Análise das Respostas do Questionário com Professores*

Um total de 59 professores responderam voluntariamente o questionário. Destes, 6 são do nordeste do Brasil, 7 do centro oeste, 31 do sul e 15 do sudeste. Não houve participantes da região

¹ <<http://www.sbc.org.br/>>

² <<http://cursocodifique.weebly.com/>>

³ <https://www.w3schools.com/js/js_es6.asp>

Figura 54 – Informações sobre o Questionário: a) Regiões do Professores Participantes, b). Perfil dos Professores, c). Atividades do Questionário e d). Perfil dos alunos.

Fonte: Elaborada pelo autor.

norte. Sudeste e sul tiveram o maior número de respondentes. Tais regiões correspondem com as que mais apresentam iniciativas relacionadas ao ensino de programação para iniciantes, de acordo com mapeamento conduzido por [Aureliano e Tedesco \(2012\)](#), e correspondem à primeira e à terceira posição das áreas brasileiras de maior densidade demográfica, segundo o Instituto Brasileiro de Geografia e Estatística (IBGE)⁴

A escolaridade dos professores participantes correspondem a 19 alunos de doutorado, 13 mestres, 11 alunos de mestrado e 16 são especialistas. A média de tempo de experiência no ensino de programação é de 7,2 anos.

As respostas coletadas com os professores são discutidas a seguir.

QP1. *O quão precisas são as categorias de problemas relacionados ao ensino de programação (Quadro 1), quando analisadas na perspectiva dos problemas que você tem encontrado durante suas aulas? Classifique-as.*

A Tabela 8 apresenta o percentual de indicações em cada um dos cinco níveis de incidência.

⁴ http://ftp.ibge.gov.br/Estimativas_de_Populacao/Estimativas_2016/

Tabela 8 – Categorias de Problemas de Programação e Níveis de Incidência

Id	Categorias	% por Nível de Incidência					
		1	2	3	4	5	Total
P1	Aprendizagem de conceitos de programação	32,20%	32,20%	15,25%	15,25%	5,10%	100,00%
P2	Aplicação de conceitos de programação	35,60%	28,80%	16,95%	13,55%	5,10%	100,00%
P3	Compreensão de programas	15,25%	35,60%	42,38%	6,77%	0,00%	100,00%
P4	Fatoração e refatoração de programas	15,25%	23,72%	37,31%	16,95%	6,77%	100,00%
P5	Motivação	23,72%	25,42%	22,04%	20,33%	8,49%	100,00%
P6	Dificuldades relacionadas aos professores	15,25%	8,49%	33,90%	23,72%	18,64%	100,00%

Fonte: Dados da pesquisa.

A classificação realizada pelos participantes indicam a categoria **Aplicação de conceitos de programação** (P2) como a mais incidente, correspondendo a 35,6%. As categorias classificadas com incidência 2 e 3, correspondendo a 35,6% e 42,38% respectivamente e referem-se à **Compreensão de programas** (P3). Finalmente, a categoria **Dificuldades relacionadas aos professores** (P6) corresponde as incidências 4 e 5, 23,72% e 18,64% respectivamente. O empate entre os níveis de incidência deve-se à indicação dos participantes, que podiam classificar uma categoria com um mesmo nível de incidência.

Ao comparar-se os percentuais por categorias de problemas e percentuais atribuídos na pesquisa, nota-se que o problema mais incidente é atribuído a categoria **Aplicação de conceitos de programação** (P2), ultrapassando o percentual do mapeamento, que era liderado pela categoria **Aprendizagem de conceitos de programação** (P1). Como segunda categoria mais incidente tem-se categoria **Compreensão de programas** (P3), que no mapeamento é considerada a categoria de quarta posição. Finalmente, como categoria menos incidente, tem-se a referente às **Dificuldades relacionadas aos professores** (P6), que no mapeamento foi considerada também a categoria com menor incidência em relação ao número de estudos que atacavam tal problema.

Destarte, nota-se a diversidade de limitações e principalmente as duas mais significativas, que envolvem a aprendizagem dos conceitos de programação (P1) e a aplicação de tais conceitos (P2). A primeira categoria forma a base necessária para a evolução e continuidade no processo de aprendizagem de linguagens de programação específicas, ou seja, leva à aplicação dos conceitos de modo efetivo. A Figura 55 apresenta um gráfico com o ranque de posições segundo o mapeamento sistemático, na cor azul, e do questionário com os professores, na cor laranja.

QP2. *Você tem adotado estratégias e/ou soluções de software para mitigar os problemas enfrentados no processo de ensino de programação? Descreva-as.*

Além de buscar identificar as possíveis soluções e estratégias adotadas, a questão visa permitir que tais soluções possam ser adotadas por outros professores que enfrentam problemas similares. As principais estratégias e soluções adotadas foram resumidas e são elencadas a seguir:

- Adoção de linguagens simplificadas de programação, i.e., linguagens com poucas estruturas complexas, variáveis fracamente tipadas e um conjunto menor de palavras reservadas,

Figura 55 – Ranque de Resultados por Categoria de Problemas.

Fonte: Dados da pesquisa.

como a linguagem Python.

- Adoção de ideias e conteúdos de aprendizagem provenientes de cursos de programação online, como os da Udacity ⁵ e Coursera ⁶;
- Adoção de diferentes abordagens para motivar a aprendizagem, i.e., aprendizagem baseada em problemas, condução de várias atividades de aprendizagem, como questões e desafios e competição de programação, como as maratonas de programação;
- Adoção de software como complemento às atividades de aprendizagem desenvolvidas em sala de aula, como o Scratch ⁷, Greenfoot ⁸, VisualLog ⁹, Ambientes de Desenvolvimento Integrado (do inglês, *Integrated Development Environments* (IDEs), Coding DOJO ¹⁰, BlueJ ¹¹ e URI Online Judge ¹² e;
- Adoção de conteúdos de aprendizagem abordando diferentes meios de representação, como vídeos, apresentações de slides e uso de ilustrações divertidas e atraentes, na concepção dos alunos.

Considerando as diferentes estratégias de aprendizagem indicadas, a que apresentou maiores indicações corresponde à aplicação de diversas atividades de aprendizagem, principalmente listas de exercícios para motivar os alunos a aplicar conceitos aprendidos de modo prático para que, assim, seja possível identificar mais rapidamente dúvidas e dificuldades. Finalmente, as soluções de software e de aplicações indicadas são adotadas como complemento, permitindo e facilitando a aplicação dos conhecimentos aprendidos em sala.

⁵ <<https://www.udacity.com/>>

⁶ <<https://www.coursera.org/>>

⁷ <<https://scratch.mit.edu/>>

⁸ <www.greenfoot.org/>

⁹ <www.apoioinformatica.inf.br/produtos/visualg>

¹⁰ <<http://dojopuzzles.com/>>

¹¹ <www.bluej.org/>

¹² <<https://www.urionlinejudge.com.br/>>

QP3. *Há outros problemas relacionados ao ensino de programação que não tenham sido considerados nas seis categorias de problemas analisadas?*

Para evitar possível viés relacionado à categorização dos problemas no ensino e aprendizagem de programação ou à ausência de possíveis problemas enfrentados nas instituições brasileiras, esta terceira pergunta foi incluída no questionário.

A maior parte dos professores concordaram que as seis categorias de problemas é completa o suficiente para representar os principais problemas relacionados ao ensino e aprendizagem de programação. Contudo, tais resultados não podem ser generalizados para todas as instituições brasileiras. Logo, as respostas apresentam evidências preliminares de que os problemas de programação encontrados em nível internacional também são refletidos nas instituições de ensino em nosso país. Por outro lado, mesmo com a limitação da amostra, esta pode ser incrementada com futuras replicações do questionário.

Finalmente, dentre os principais problemas destacados pelos professores, elencam-se:

- “Os alunos não estão preparados para programar, e os seus conhecimentos em matemática e raciocínio lógico não são profundos”.
- “Os alunos não sabem a diferença entre o uso de uma linguagem de programação e a aplicação lógica de conceitos. Programar deve ser pensado na aplicação de conhecimentos representados por meio de uma linguagem específica de programação, já fundamentos de programação deve manter o foco no pensamento lógico e algoritmos, sem a aplicação específica de uma linguagem”.
- “Os alunos não estão engajados. Eles preferem usar dispositivos móveis ou navegar na Internet do que realizar atividades de aprendizagem” e “com a disponibilização e alcance de diferentes tecnologias (e.g. jogos, redes sociais, etc.), a concentração dos mesmos é prejudicada”.
- “Os alunos não estão interessados em realizar suas tarefas de aprendizagem. Todas as tarefas são consideradas difíceis, principalmente as que demandam mais tempo, e.g., assistir vídeo aulas com duração superior a 15 minutos”.
- “Os alunos copiam as respostas (plágio). Eles preferem copiar as respostas das atividades de seus colegas do que resolver por si próprios”.
- “Ferramentas de programação fornecem poucas simulações visuais”.
- “Os alunos têm problemas ao expressar suas dúvidas”.
- “Os alunos tem dificuldades em interpretar atividades de aprendizagem, e.g., questões e desafios, que são redigidos em mais de quatro linhas (analfabetismo funcional)”.

- “Os alunos não conseguem pesquisar informações por si só. Há falta de autonomia e iniciativa”.
- “Os alunos não gostam de trabalhar em grupo. Eles apresentam dificuldades em formar grupos”.
- “Os alunos frequentemente desistem de seus cursos”.
- “Programas de disciplinas são frequentemente fixos e, de algum modo, desatualizados”.
- “Ferramentas de código aberto (do inglês *open source*), e ferramentas gratuitas são limitadas”.
- “Tópicos aos quais os alunos já aprenderam começam a desmotivá-los quando aplicados em atividades de aprendizagem. A customização e criação de exercícios seria uma boa solução”.
- “Materiais ou conteúdos educacionais não permitem a simulação de funções, e.g., métodos e estruturas de repetição, em disciplinas introdutórias de programação”.

Neste contexto, notou-se que a maioria dos relatos estão relacionados: (i) aos conceitos básicos necessários para que os alunos possam entender os conceitos básicos de programação envolvidos, (ii) desmotivação e (iii) distração com outras tecnologias. No que se refere as ferramentas e software de apoio, estes não são completos o suficiente para atender as necessidades dos professores. E, em alguns casos, tem-se a questão de currículos desatualizadas, que não adotam novas soluções para apoiar tanto professores, quanto alunos em sala.

A.1.2 Análise das Respostas do Questionário com alunos

Um total de 40 alunos estavam matriculados para o curso introdutório de programação web, todos da cidade de São Carlos-SP. Todos os participantes foram convidados a responder o questionário. Contudo, apenas 18 participaram. Destes, 10 responderam o pré-teste e o pós-teste, 4 responderam apenas o pré-teste e 4 responderam apenas o pós-teste, resultando no total de 18 respondentes.

Em relação ao grau de escolaridade dos participantes do pré-teste, dois respondentes estavam matriculados no primeiro ano do ensino médio, quatro no segundo, cinco no terceiro e três já haviam finalizado o ensino médio. Quanto ao perfil dos respondentes do pós-teste, dois estavam no primeiro ano do ensino médio, seis no segundo, quatro no terceiro e dois já haviam finalizado seus estudos, também no ensino médio.

As respostas coletadas com os alunos no pré-teste (QEA) e pós-teste (QEB) são discutidas a seguir.

QEA1. *Por que você quer aprender a programar?*

A maioria dos alunos indicaram que pretendem seguir carreira na área e que a tecnologia está presente em, basicamente, tudo. Assim indicaram que, se eles souberem programar, poderão criar novas facilidades, melhorando o ambiente que os rodeia.

Esta resposta unânime destaca a preocupação dos alunos e a importância em saber programar. Devido ao tamanho restrito da amostra, esta evidência inicial sugere que os alunos sabem da importância de aprender a programar. Logo, isso demonstra a importância da inclusão de disciplinas de computação e programação no currículo de escolas primárias e secundárias.

QEA2. *Por que aprender a programar é importante para o seu futuro?*

Esta pergunta foi aplicada para reforçar a resposta da questão **QEA1**. Como esperado, ela foi respondida de modo similar pelos alunos, que indicaram que aprendendo a programar, eles podem entender as tecnologias que os rodeiam, além de ser crucial para suas carreiras.

QEA3 e QEB1. *Se pudesse, você adotaria tecnologias móveis para aprender a programar?*

Esta pergunta foi incluída pois nota-se a incipiência de soluções para apoiar o ensino de programação por meio de dispositivos móveis. Neste contexto, buscou-se identificar se os respondentes utilizariam dispositivos móveis para aprender a programar.

Ao todo, 13 alunos indicaram que adotariam dispositivos móveis para aprender a programar no pré-teste, principalmente por poder ser utilizado em qualquer lugar e a qualquer hora. Apenas um participante disse não, justificando que computadores fornecem ambientes de desenvolvimento mais completos. Em relação às respostas do pós-teste, novamente 13 alunos indicaram que utilizariam dispositivos móveis para aprender a programar e um disse que não, sem apresentar justificativa. Como quatro participantes responderam exclusivamente um dos testes, considerando as respostas coletadas no pré-teste, das 13 respostas da possível adoção, 10 foram reconfirmadas no pós-teste.

QEA4. *Quais estratégias e/ou abordagens você gostaria usar em software que objetivam ensinar programação?*

O objetivo desta pergunta foi identificar as preferências de estratégias e abordagens que, indiretamente, acabam por traçar estilos de aprendizagem (FELDER; SILVERMAN *et al.*, 1988; LIYANAGE; GUNAWARDENA; HIRAKAWA, 2014) e consequentemente meios para melhor engajar os alunos a aprender a programar (KELLEHER, 2009).

Dentre as respostas obtidas, quatro responderam vídeos; dois jogos; oito a mesclagem de vídeos, animações, textos figuras e jogos e um respondeu estratégias que utilizem poucos textos e textos curtos, justificando que seriam mais engajantes que textos longos.

QEA5. *Que outros meios você gostaria que fossem utilizados para apoiar o processo de aprendizagem de programação?*

Seis alunos demonstraram interesse em aplicações móveis, quatro em aprendizagem de programação com robôs, dois por meio de cursos online, dois por jogos e dois por meio de software para computadores. Nota-se assim uma diversidade de opções a serem adotadas em disciplinas e cursos de programação.

QEB2. *Quais tópicos você têm ou teve mais dificuldade em aprender?*

Entre as respostas, um aluno indicou estruturas de repetição, um estruturas condicionais, sete vetores e matrizes, sete funções e um indicou não possuir dificuldades nos tópicos especificamente ministrados no curso. As dificuldades encontradas foram identificadas por meio da próxima questão.

QEB3. *Por que você acredita ter tais dificuldades nesses tópicos especificamente?*

A principal dificuldade relatada pelos respondentes correspondeu em seguir a teoria ensinada, dois deles indicaram o nível de dificuldade das atividades e três não indicaram as razões.

Com base na respostas das questões QEB3 e QEB4 foi possível identificar que tais considerações são abrangidas nas seis categorias de problemas propostas por [Souza, Batista e Barbosa \(2015\)](#).

Quanto às atividades aplicadas, os alunos recebiam algumas tarefas a serem resolvidas em casa, sem qualquer ferramenta ou material de apoio, além do fornecido durante as aulas presenciais. As dúvidas eram todas sanadas em sala de aula pelo instrutor. Um outra consideração refere-se a dificuldades do instrutor em identificar as limitações do aluno, uma vez que era necessário e esperado que cada aluno reportasse tais dificuldades ao instrutor no decorrer das aulas, o que não ocorreu com frequência.

QEB4. *Você possui alguma sugestão para melhorar o curso de programação que realizou?*

Duas sugestões foram indicadas: (1) necessidade de ensinar mais detalhadamente fundamentos de programação e (2) adoção de outras linguagens de programação, além de JavaScript.

Conceitos básicos de programação foram ensinados por meio de JavaScript. Todavia, os alunos consideraram os tópicos um tanto quanto avançados para a base de conhecimento que possuíam. Nesta perspectiva, eles expressaram a necessidade de aprenderem melhor os fundamentos de programação antes de uma linguagem, como JavaScript.

A.1.3 Desafios no Ensino e Aprendizagem de Programação

Com base nos problemas categorizados e no questionário conduzidos com professores e aprendizes elencam-se os principais desafios a serem investigados e superados nesta área:

- Direcionar esforços para a proficiência em matemática e raciocínio lógico, para impulsionar a qualidade no processo de ensino e aprendizagem de programação.
- Conceber soluções de software mais flexíveis, que agreguem meios de melhorar o processo de *feedback*; identificação de limitações e resolução de dúvidas; conteúdos e atividades de aprendizagem com duração e descrições curtas; estratégias e simulações visuais, e.g., ilustrações, objetos 3D; e interações colaborativas e cooperativas.
- Conceber ferramentas para apoiar os professores durante e depois das aulas, no que refere-se ao monitoramento do desempenho de seus alunos;
- Adoção de soluções motivacionais para os alunos;
- Integração de funcionalidades e facilidades que os alunos encontram em aplicações do cotidiano; e.g. aplicações de redes sociais, jogos.
- Criação de conteúdos por meio de diferentes estratégias de apresentação, e.g., considerando a separação de conceitos de lógica e algoritmos de conceitos específicos de linguagens de programação e;
- Adoção de dispositivos móveis para apoiar o ensino e aprendizagem.

Nesta perspectiva, observa-se que estes desafios estão relacionados ao domínio de programação e não estão restritos ao Brasil (MARCOLINO; BARBOSA, 2015b). Porém, em comparação com outros países e suas iniciativas para impulsionar o setor tecnológico, i.e., inclusão de disciplinas de programação no currículo de cursos elementares, muitas mudanças são necessárias. Entre elas, destacam-se a necessidade de melhor preparar os professores para que criem o hábito de utilizar tecnologias inovadoras e que possam enriquecer suas aulas. Tais professores, principalmente das escolas públicas, necessitam de mais motivação e treinamento (AURELIANO; TEDESCO, 2012; WANGENHEIM; NUNES; SANTOS, 2014; BORDINI *et al.*, 2016), além de necessitarem do respaldo dos órgãos educacionais federais, estaduais e municipais, no que tange a atualização de currículos e planos de ensino que integrem o pesamento computacional e programação em suas aulas. O que, como pode-se observar em algumas iniciativas, é uma tendência para um futuro próximo (CONFORTO *et al.*, 2018; SOUZA; LOPES, 2018).

LPS EDUCACIONAIS E SOFTWARES PARA O ENSINO E APRENDIZAGEM DE PROGRAMAÇÃO: UMA NOVA ITERAÇÃO

B.1 LPSs no Domínio Educacional e no Domínio de Programação

Esta seção apresenta os estudos primários recuperados entre 2014 a 2018, para complementar o MS conduzido de 2004 a 2014, com o objetivo de identificar linhas de produto de software no domínio educacional e de ensino e aprendizagem de programação.

A primeira versão deste mapeamento está disponível em <http://www.br-ie.org/pub/index.php/sbie/article/view/5160> e o protocolo adicional está disponível em <https://goo.gl/trRMBI>. Estes foram considerados para a condução desta nova iteração.

A Tabela 9 apresenta a quantidade de estudos recuperados e o total final dos artigos após a execução das fases de seleção, de acordo com o protocolo do mapeamento.

Tabela 9 – Estudos Primários Recuperados por Base de Busca.

#	Base	Qtd.	Descartados (Seleção Preliminar)	Seleção Preliminar	Descartados (Seleção Final)	Seleção Final
1	ACM	16	14	2	2	0
2	Compedex	56	44	12	10	2
3	IEEE	1	1	0	0	0
4	RBIE	3	0	3	0	3
5	ScienceDirect	314	298	16	16	0
Total		390	357	33	28	5

A Tabela 23 apresenta os cinco estudos selecionados, ordenados por ano de publicação. Incluí-se o título do estudo, *venue*, e sua referência. Apesar de considerar a janela de tempo de

2014 até 2018, a base da Revista Brasileira de Informática na Educação (RBIE) retornou um estudo de 2012 e de 2013 que não havia sido identificado no MS anterior. Por ser considerado uma contribuição importante, foi incluída nos resultados deste mapeamento. Adicionalmente, o artigo de Id 4 é resultado de nova publicação relacionada a estudo primário já considerado nos resultados do MS anterior. Deste modo, foi também considerado nos resultados desta nova iteração para identificar possíveis novas considerações.

Quadro 23 – Estudos Primários Recuperados.

Id	Título	Referência	Ano	Venue
1	Uma Linha de Produto de Software baseada na Web Semântica para Sistemas Tutores Inteligentes	[1]	2012	RBIE
2	Sobre o Desenvolvimento de Software Educacional: proposta de uma Linha de Produto de Software para Módulos de Aprendizagem Interativa	[2]	2013	RBIE
3	Building Families of Software Products for e-Learning Platforms: A Case Study	[3]	2014	IEEE REVISTA IBEROAMERICANA DE TECNOLOGIAS DEL APRENDIZAJE,
4	A Contribution to the Adoption of Software Product Lines in the Development of Mobile Learning Applications	[4]	2014	IEEE
5	Uma Linha de Produto de Software para Desenvolvimento de Jogos Educativos	[5]	2018	RBIE

Fonte: Dados da pesquisa.

As referências indicadas no Quadro 23 estão disponíveis em <https://drive.google.com/open?id=167ihY_VYq52Bm-dKxmCUceEiZGBelD2o>.

Em [1], os autores propõem uma LPS para sistemas tutores inteligentes, criando um estudo de cada no domínio de programação para avaliar em nível de modelos a LPS proposta. Em [2] os autores propõem uma LPS para módulos de aprendizagem interativa (iMA). Por meio da LPS desenvolveram um iMA denominado iVProg, sendo este um aplicativo para ensino de introdução à programação empregando um modelo visual, no qual o aluno pode escolher comandos a partir de ícones. Este aplicativo foi avaliado por meio de estudo de caso. E criaram, ainda, o o iMA iTangram, utilizado em uma prova de conceitos. O nome iTangram forneceu a ideia para a ferramenta de apoio à LPS desta pesquisa, considerando que o quebra-cabeças Tangram permite o desenvolvimento de diferentes figuras pela combinação de suas 7 peças, tal como pode-se criar diferentes produtos em uma linha, por meio da seleção de suas *features*.

Em [3], é discutida uma LPS para aplicações educacionais móveis de domínio geral, sendo conduzido um estudo de caso para avaliar o processo de criação de produtos por meio da LPS. Em [4], uma LPS foi desenvolvida por meio de artefatos extraídos de uma aplicação *e-Learning* chamada *E-Learning Web Miner* (ElWM) para o desenvolvimento de outros ambientes educacionais eletrônicos, considerando as *features* extraídas.

Finalmente, em [5] é proposta uma LPS para a criação de jogos educacionais. Uma das propostas foi o desenvolvimento de um jogo considerando *features* de um jogo educacional para o ensino de programação chamado RoboCode. O produto gerado foi um jogo com suporte a multijogadores.

B.2 Softwares para o Ensino e Aprendizagem de Programação

Esta seção apresenta os estudos primários recuperados entre 2014 a 2018, para complementar o MS conduzido de 2004 a 2014, com o objetivo de identificar softwares educacionais para o ensino e aprendizagem de programação.

A primeira versão deste mapeamento está disponível em <http://www.br-ie.org/pub/index.php/sbie/article/view/5150> e o protocolo adicional está disponível em <https://goo.gl/4hu1gZ>. Estes foram considerados para a condução desta nova iteração.

A Tabela 10 apresenta a quantidade de estudos recuperados e o total final dos artigos, após a execução das fases de seleção, de acordo com o protocolo do mapeamento.

Tabela 10 – Estudos Primários Recuperados por Base de Busca.

#	Base	Qtd.	Descartados (Seleção Preliminar)	Seleção Preliminar	Descartados (Seleção Final)	Seleção Final
1	ACM	494	343	151	140	11
2	Compedex	772	749	23	20	3
3	IEEE	53	11	42	32	10
4	RBIE	26	20	6	3	3
5	ScienceDirect	31	15	16	9	7
Total		1376	1138	238	204	34

Fonte: Dados da pesquisa.

O Quadro 24 apresenta os 34 estudos selecionados, ordenados por ano de publicação. Inclui-se o título do estudo, país de origem, base de busca no qual o estudo foi recuperado e sua referência.

As referências indicadas no Quadro 24 estão disponíveis em <https://drive.google.com/file/d/1G8YMoOPE8Q1jZRN-ChyQrJvJigIAidsc/view?usp=sharing>.

Em relação aos softwares encontrados, concentrou-se esforços nos estudos que correspondiam a aplicações móveis para o ensino e aprendizagem de programação. Nesta perspectiva, o estudo [22] e [31] considerou o *App Inventor* (AI), desenvolvido pelo MIT¹ e já identificado na primeira execução do MS, como meio para promover a aprendizagem de programação para alunos iniciantes, considerando dispositivos móveis com robôs. Em [15], os autores especificam a proposta de uma aplicação que busca ensinar de lógica de programação de modo similar ao Scratch.

¹ <http://appinventor.mit.edu/explore/>

Quadro 24 – Estudos Primários Recuperados.

Id	Título	Referência	Ano	Venue	País	Base de Busca
1	Design and evaluation of a web-based dynamic algorithm visualization environment for novices	[1]	2013	Conferência	Grécia	ScienceDirect
2	Student perception and usage of an automated programming assessment tool	[2]	2013	Periódico	Espanha	ScienceDirect
3	Integração de uma Metodologia de Ensino Presencial de Programação com um Sistema Tutor Inteligente	[3]	2014	Periódico	Brasil	RBIE
4	Ensino de Computação com SCRATCH no Ensino Fundamental –Um Estudo de Caso	[4]	2014	Periódico	Brasil	RBIE
5	Salespoint: A Java framework for teaching object-oriented software development	[5]	2014	Periódico	Reino Unido	ScienceDirect
6	Developing a self-regulated oriented online programming teaching and learning system	[6]	2014	Conferência	China	IEEE
7	A Platform for Teaching Logic Programming using Virtual Worlds	[7]	2014	Conferência	Grécia	IEEE
8	Guess My Object – An 'Objects First' Game on Objects' Behavior and Implementation with BlueJ	[8]	2014	Conferência	Alemanha	ACM
9	CaptainTeach: Multi-stage, In-flow Peer Review for Programming Assignments	[9]	2014	Conferência	EUA	ACM
10	Enki: A Pedagogical Services Aggregator for Learning Programming Languages	[10]	2014	Conferência	Portugal	ACM
11	Defining and Evaluating Conflicting Animations for Programming Education: The Case of Jeliot ConAn	[11]	2014	Conferência	Finlândia	ACM
12	Múltiplas Representações Externas no Suporte à Aquisição de Conhecimento em Programação de Computadores	[12]	2015	Periódico	Brasil	RBIE
13	A Programming Environment for Visual Block-Based Domain-Specific Languages	[13]	2015	Conferência	Japão	ScienceDirect
14	Programming assistance tools to support the learning of IT programming in South African secondary schools	[14]	2015	Periódico	África do Sul	ScienceDirect
15	Proposal of programming creation application using road signs by smartphones	[15]	2015	Conferência	Japão	IEEE
16	Programming Education for Primary Schoolchildren Using a Textual Programming Language	[16]	2015	Conferência	Japão	IEEE
17	Intelligent Assistant for Helping Students to Learn Programming	[17]	2015	Conferência	Croácia	IEEE
18	A Desktop VR-based HCI Framework for Programming Instruction	[18]	2015	Conferência	EUA	IEEE
19	Enhancing Robot Programming with Visual Feedback and Augmented Reality	[19]	2015	Conferência	Suíça	ACM
20	Tangible gimmick for programming education using RFID systems	[20]	2016	Conferência	Japão	ScienceDirect
21	Teaching Introductory Robotics Programming Learning to Program with National Instruments' LabVIEW	[21]	2016	Periódico	EUA	Compendex
22	Using App Inventor 2 in A Summer Programming Workshop: Improvements Over Previous Years	[22]	2016	Conferência	Arábia Saudita	IEEE
23	An effective visual programming tool for learning and using robotics middleware	[23]	2016	Simpósio	Japão	IEEE
24	Jimbo: A Collaborative IDE with Live Preview	[24]	2016	Workshop	EUA	IEEE
25	From Alice to Python. Introducing Text-based Programming in Middle Schools	[25]	2016	Conferência	Qatar	ACM
26	Parallel programming with pictures is a Snap!	[26]	2017	Periódico	EUA	ScienceDirect
27	The Effects of a Robot Game Environment on Computer Programming Education for Elementary School Students	[27]	2017	IEEE	Coreia do Sul	Compendex
28	RECT: A Cloud-Based Learning Tool for Graduate Software Engineering Practice Courses With Remote Tutor Support	[28]	2017	Periódico	China	Compendex
29	Utilizing Programming Education Support Tool pgtracer in an Actual Programming Course	[29]	2017	Congresso	Japão	IEEE
30	CodeWorkout: Short Programming Exercises with Built-in Data Collection	[30]	2017	Conferência	EUA	ACM
31	Improved Mobile Robot Programming Performance through Real-time Program Assessment	[31]	2017	Conferência	Suíça	ACM
32	Eliph: Effective Visualization of Code History for Peer Assessment in Programming Education	[32]	2017	Conferência	Coreia	ACM
33	Exploring Instructional Support Design in an Educational Game for K-12 Computing Education	[33]	2018	Conferência	EUA	ACM
34	Scratch Microworlds: Designing Project-Based Introductions to Coding	[34]	2018	Conferência	EUA	ACM

Fonte: Dados da pesquisa.

Todos os estudos foram analisados a fim de permitir melhorias no catálogo de requisitos e, diferentemente do primeiro MS, as especificações técnicas não foram analisadas, visto que a proposta e tecnologias para o desenvolvimento da LPS já haviam sido definidas. 14 desses estudos apoiam o ensino presencial, sendo softwares utilizados em *desktops*; 15 apoiam o ensino

presencial e a distância, sendo softwares que estão disponíveis *online*; 3 indicam seu apoio ao ensino a distância, utilizando-se de ambientes *online*; e 2 não especificam o tipo de ensino apoiado. No contexto das aplicações móveis, dos três estudos identificados, um deles se mantém em nível de proposta de uma aplicação, e os outros dois utilizam-se do *App Inventor* sendo que, para um deles, a necessidade de estar integrado com robôs retira a possibilidade de sua adoção no ensino a distância.

Finalmente, alguns estudos específicos para outros domínios foram retornados, como é o caso do estudo [28]. A justificativa para inclusão de estudos nesta perspectiva deve-se ao fato de proporem atividades de programação assistidas por softwares. Nesse sentido, tais softwares foram considerados para análise no contexto deste MS.

B.3 Conclusões

Deste modo, notou-se que, poucos estudos que adotam a modalidade *m-learning* no contexto de LPSs. Contudo, os estudos identificados se mostram interessantes nos contextos tecnológico e educacional que estão inseridos. Sendo, portanto, considerado para complementar os estudos primários identificados na primeira iteração do MS.

Quando aos softwares para ensino e aprendizagem de programação, apesar da quantidade expressiva de soluções (34), apenas três adotam dispositivos móveis. Este cenário motiva a condução de maiores investigações no domínio de aplicações móveis, tal como foi identificado na primeira iteração do MS, para explorar melhor identificar seu potencial como meio de promover o ensino e aprendizagem.

CATÁLOGO DE GESTOS PARA AS APLICAÇÕES EDUCACIONAIS MÓVEIS

C.1 Catálogo de Gestos para as Aplicações Educacionais Móveis

Dentre os objetivos de propor e desenvolver o catálogo de gestos como uma *feature* para a LPS estão: (1) identificar evidências iniciais quanto ao uso de gestos como meio de atrair aprendizes para o uso de dispositivos móveis na realização de atividades de programação; (2) identificar se o catálogo pode vir a motivar aprendizes a aprender programação utilizando aplicações educacionais móveis; e (3) investigar o quão satisfatória pode ser a experiência do usuário substituindo os teclados convencionais das plataformas móveis por gestos.

Para possibilitar a criação do catálogo para interfaces naturais de gestos, estudos primários que propõem a criação de gestos foram investigados (ZHAI *et al.*, 2012; ANTHONY *et al.*, 2013; RUBINE, 1991; WOBBROCK; WILSON; LI, 2007; WESTERMAN, 2016; ANTHONY *et al.*, 2013; BRAGA; CARDOSO; JR., 2012; GUTIERREZ, 2015; MEDEIROS, 2015; SANTOSA *et al.*, 2014; KAMAL, 2015; VATAVU; CRAMARIUC; SCHIPOR, 2015; ALAVI; ARSENAULT; WHITEHEAD, 2016; REIS, 2014; REIS *et al.*, 2013). Apesar de diferentes propostas, nenhum estudo primário recuperado considerou o desenvolvimento de gestos para o ensino e aprendizagem de programação por meio de aplicações educacionais móveis, o desenvolvimento do catálogo, cuja concepção é discutida a seguir.

C.1.1 Planejamento e Concepção

O planejamento e concepção do catálogo foi dividido em três tarefas, a saber: (i) mapeamento de conceitos de programação e seleção de linguagem de programação a serem representadas por meio de tais gestos; (ii) identificação de estruturas (excertos) de código das linguagens

de programação candidatas a serem integradas nas aplicações; e (iii) proposição dos gestos de interação para as respectivas estruturas de código, baseadas nos estudos primários discutidos anteriormente (Seção C.1).

Na primeira tarefa, diversos planos de aula de disciplinas de programação de diferentes cursos de computação da Universidade de São Paulo foram analisados. Como resultado, duas estruturas com conceitos de programação foram modeladas: uma cobrindo aspectos procedurais, como apresentado na Figura 56, como apresentado na Figura 49, e outra estrutura cobrindo aspectos do paradigma orientado a objetos, apresentado na Figura 57.

Figura 56 – Diagrama do Paradigma Procedural.

Fonte: Elaborada pelo autor.

Figura 57 – Diagrama do Paradigma Orientado a Objetos.

Fonte: Elaborada pelo autor.

Tais modelos foram adotados para permitir a seleção de duas linguagens de programação e, em consequência, a definição de excertos de códigos ou estruturas a serem representadas pelos

gestos a serem estabelecidos na próxima tarefa. As linguagens de programação selecionadas foram C e Python. A primeira representando os aspectos do paradigma procedural e a segunda representando o paradigma orientado a objetos, respectivamente. Ambas foram selecionadas considerando os currículos analisados previamente.

Na segunda tarefa as estruturas de código foram mapeadas. Tais estruturas foram escolhidas por meio da análise do material educacional e cursos das linguagens C e Python, considerando os fundamentos de cada linguagem. O Quadro 25 apresenta as 12 estruturas de código consideradas no catálogo. Em um primeiro momento, apenas estruturas básicas das duas linguagens foram consideradas.

Quadro 25 – Estrutura de Códigos em C e Python.

Item	Estrutura	Código C	Código Python
1	Condicional If	if() { }	if :
2	Condicional If then Else	if() { } else { }	if : else:
3	Laço While	while() { }	while :
4	Laço Do While	do(condição) { } while();	-
5	Condicional Switch Case	switch() { default: break; }	-
6	Operador Ternário	? : ;	-
7	Laço For	for(;;) { }	for in :
8	Comentário em bloco (I)	/* * */	''' '''
9	Comentário em linha (II)	//	#
10	Função	-	def () :
11	Exceção	-	try : except () []:
12	Classe	-	class :

Fonte: Dados da pesquisa.

Na terceira tarefa, cada uma das estruturas foram analisadas, sendo propostos de quatro a cinco gestos para cada uma dessas estruturas. Para a criação de tais gestos os trabalhos discutidos anteriormente, sobre métodos para concepção de gestos, foram adotados. Para reduzir o número de alternativas, para duas ou três opções para cada uma das estruturas (Quadro 25), a usabilidade dos gestos e sua relação intuitiva com o código, e.g., uma letra, uma palavra, uma referência ao comportamento durante a execução da estrutura, foram considerados. De posse de um grupo mais seleto de gestos, uma primeira avaliação foi conduzida.

C.1.2 Primeira Avaliação do Catálogo de Gestos

Esta primeira avaliação foi conduzida com o objetivo de identificar qual o gesto candidato a ser desenvolvido para cada uma das estruturas de código, ou seja, definir uma primeira versão do catálogo de gestos. Para isso, o processo de definição e condução da avaliação foi composto

por seis etapas: (i) planejamento e criação da instrumentação por meio de um questionário para a seleção dos gestos; (ii) execução piloto da aplicação do questionário com dois pesquisadores ; (iii) aplicação de melhorias no questionário e alguns dos gestos avaliados, com base na aplicação piloto; (iv) revisão do questionário com especialista; (v) aplicação do questionário em sessão presencial; e (vi) resumo e análise dos resultados.

Assim, o conjunto de gestos estabelecido após a condução das três tarefas de proposição foi incluído em um questionário. Cada uma das estruturas correspondeu a uma questão do mesmo. Os gestos foram identificados por letras, e os participantes eram convidados a selecionar o gesto mais pertinente de acordo com a estrutura de código representada. Na inexistência de um gesto candidato, os participantes tinham a possibilidade de desenhar um novo gesto e justificar suas propostas. Esse campo extra foi disponibilizado também para cada uma das 12 questões. O questionário final está disponível neste link <<https://goo.gl/Kusyvr>>.

A aplicação do questionário ocorreu presencialmente com 69 estudantes do curso de Ciência da Computação da Universidade de São Paulo, em uma única sessão. O questionário foi distribuído de modo impresso. Todos os participantes pertenciam ao mesmo ano no curso. O único requisito eram que os participantes tivessem algum conhecimento prévio com programação. Todos eram voluntários, podendo participar ou não.

C.1.2.1 Análise dos Resultados

O tempo necessário para responder ao questionário foi, em média, de sete minutos.

O Quadro 26 resume os resultados em percentual. O gesto com maior número de seleções (*Dominance of gesture most selected*) variou entre 45% a 78% de aceitação. O percentual de gestos propostos pelos participantes (coluna *Rejection*), e a soma do percentual dos demais gestos selecionados (coluna *Other Gestures*), não ultrapassaram o percentual total dos gestos com os maiores percentuais de aceitação. Logo, os gestos propostos sistematicamente seguindo estudos primários como os de [Zhai et al. \(2012\)](#) [Anthony et al. \(2013\)](#) foram os que receberam maiores indicações.

A taxa de rejeição dos gestos propostos, obtidos com base no número de gestos desenhados, variou de 4% a 16% (10% em média). O total de sugestões de novos gestos para as 12 questões foram de 78. 3% das questões foram deixadas em branco. Os gestos que tiveram respostas em branco foram o *switch-case*, *ternary-operator*, *functions*, *exception* e *class*, sendo esses últimos referentes à linguagem Python e ao paradigma orientado a objetos.

O gesto com nível mais alto de rejeição foi o referente a estrutura condicional *if()*. Consequentemente, foi o que apresentou menor índice de aceitação (45%). No entanto, a soma do percentual de rejeição e de seleção dos outros gestos não ultrapassou o percentual de aceitação. Neste sentido, este gesto tornou-se um dos candidatos a receber adaptações.

Quadro 26 – Resultados da Primeira Avaliação do Catálogo de Gestos.

Code Sstructure		Gesture				Dominance of		Other Gestures	Rejection		No answer
#Id	Name	a	b	c	d	no answer	gesture most selected		(drew their own getures)		
1	Conditional If	18	9	31	11	0	45%	39%	16%	0%	
2	Conditional If Then Else	17	9	33	10	0	48%	38%	14%	0%	
3	Loop While	40	20	9		0	58%	29%	13%	0%	
4	Do While	39	21	9		0	57%	30%	13%	0%	
5	Conditional Switch Case	34	29	5		1	49%	42%	7%	1%	
6	Ternary Operator	44	18	3		4	64%	26%	4%	6%	
7	Loop For	46	16	7		0	67%	23%	10%	0%	
8	Block Commentary (I)	44	18	7	0	64%	26%	10%	0%		
9	Line Commentary (II)	54	11	4	0	78%	16%	6%	0%		
10	Function	32	28	7	2	46%	41%	10%	3%		
11	Exception	23	34	3	9	49%	33%	4%	13%		
12	Class	39	19	3	8	57%	28%	4%	12%		

Fonte: Dados da pesquisa.

A Figura 58 apresenta os gestos selecionados por meio da primeira avaliação.

Figura 58 – Gestos Selecionados na Primeira Avaliação.

Fonte: Elaborada pelo autor.

Com base nestes gestos, uma nova avaliação foi planejada e conduzida com o objetivo principal de avaliar o catálogo no que diz respeito à satisfação no uso prático dos gestos pelos aprendizes.

C.1.3 Segunda Avaliação do Catálogo de Gestos

Nesta segunda avaliação, três aspectos principais foram verificados: (1) a aceitação do catálogo; (2) a experiência dos usuários ao utilizarem os gestos em dispositivos móveis; (3) evidências se a adoção dos gestos e aplicações móveis podem vir a motivar os estudantes na realização de tarefas de codificação em dispositivos móveis.

O planejamento e execução da segunda avaliação seguiu cinco etapas: (i) investigação de seleção de aplicação para permitir que os gestos propostos fossem testados de modo prático em dispositivos móveis; (ii) criação de questionário para avaliação da usabilidade dos gestos; (iii) aplicação piloto da instrumentação; (iv) execução da aplicação; e (v) resumo e análise dos resultados.

Por se tratar de uma avaliação prática, o aplicativo *Gesture Tool*¹ foi adotado para ser utilizado no processo de avaliação dos gestos. Assim, os gestos apresentados na Figura 58 foram incluídos em um questionário. Neste questionário cada gesto possuía sua estrutura de código e seu respectivo gesto. Os participantes tinham três tentativas de inserção do gesto na aplicação, antes de seguir para o próximo gesto. Ao final, três questões foram incluídas para identificar: (a) se os participantes usariam os gestos para substituir teclados nativos; (b) o quão satisfatória foi a experiência no uso dos gestos; e (c) opinião se o catálogo pode vir a motivar os estudantes a aprender por meio de aplicações móveis.

O questionário passou, então, por uma execução piloto com dois alunos de graduação, não tendo sido identificadas melhorias a serem realizadas na instrumentação. A versão final do questionário pode ser acessado neste link: <<https://goo.gl/gYD4kP>>.

A aplicação do questionário ocorreu de modo presencial, em diversas sessões, devido à necessidade de se usar dispositivos móveis. Todos os participantes eram alunos da Universidade de São Paulo e eram convidados pelo aplicador a participar nos intervalos das aulas, sendo todos voluntários. O aplicador também foi responsável por instruir os alunos na execução e por marcar e controlar o número de tentativas realizadas para cada gesto, bem como sanar demais problemas que pudessem vir a surgir.

A Figura 59 exibe a tela de cadastro dos gestos no aplicativo *Gesture Tool*, utilizadas para a inserção dos gestos. Após o participante realizar a inserção correta, uma mensagem com o nome da estrutura de código correspondente era exibida.

C.1.3.1 Análise dos Resultados

Um total de 40 participantes avaliaram o catálogo; destes, cinco eram estudantes de mestrado, 11 de doutorado e 24 estudantes de graduação, todos de cursos da área de ciência da computação. Como o tempo não era um fator relevante e cada participante poderia levar um período de tempo específico para visualizar e desenhar o gesto, este não foi considerado.

¹ <<https://play.google.com/store/apps/details?id=com.davemac327.gesture.tool>>

Figura 59 – *Gesture Tool* e Usuário Inserindo Gesto durante a Segunda Avaliação.

Fonte: Elaborada pelo autor.

Um dos aspectos registrados durante a execução da avaliação foi o número de tentativas para cada um dos gestos, apresentado no gráfico da Figura 60. Essas tentativas correspondem a não identificação do gesto pelo aplicativo *Gesture Tool*, geralmente ocasionado por erro no gesto inserido pelo participante ou falha da aplicação. No entanto, apesar do número elevado de tentativas para o gesto line comment e ternary operator, os aplicadores informaram que os erros cometidos não foram devido a falhas na aplicação utilizada, mas sim por erros do gesto inserido pelos participantes. Logo, os totais de tentativas foram resultado da falta de atenção dos participantes, principalmente para os gestos mais complexos.

Figura 60 – Número de Tentativas por Estrutura de Código.

Fonte: Dados da pesquisa.

Em relação às respostas às questões, na primeira delas, que tinha como objetivo identificar se os participantes adotariam tais gestos para substituir os teclados nativos, 34 participantes

responderam “sim”, e 6 “não”.

Na segunda questão, relacionada à satisfação em utilizar tais gestos, os participantes eram convidados a assinalar uma das cinco opções em uma escala de satisfação: “totalmente satisfeito”, “muito satisfeito”, “satisfeito”, “pouco satisfeito” ou “insatisfeito”. O gráfico da Figura 61 apresenta os três níveis de satisfação da escala que foram assinalados.

Figura 61 – Satisfação dos Participantes na Utilização dos Gestos.

Fonte: Dados da pesquisa.

Metade dos participantes indicaram estarem muito satisfeitos (50%); 35% indicaram estarem satisfeitos e 15% indicaram estarem totalmente satisfeitos.

Em relação à terceira questão, que buscava identificar se os gestos apresentados podem motivar estudantes a aprender a programar utilizando dispositivos móveis, 34 participantes responderam “sim”, ou seja, eles acreditam que o catálogo de gestos podem motivar os estudantes; e apenas quatro participantes responderam “não”.

Todas as respostas das perguntas foram positivas. Contudo, considerando a amostra reduzida e a dificuldade de generalização, elas são consideradas evidências preliminares de que o catálogo pode ser adotado como meio para substituir os teclados dos dispositivos de modo satisfatório podendo, ainda, ser um meio motivante para os aprendizes conduzirem suas tarefas em dispositivos móveis.

C.1.4 Ameaças à Validade das Avaliações

Ameaças da Validade de Conclusão: para ambas avaliações, a maior preocupação refere-se ao tamanho da amostra, que deve ser ampliada em estudos futuros.

Ameaças à Validade de Construção: em ambas as avaliações, todos os participantes eram estudantes e voluntários, reduzindo o viés da coleta das informações, já que não tinham a obrigatoriedade de responder aos questionários das avaliações.

Ameaças à Validade Interna:

- **Efeitos de Fadiga:** como a média de tempo para responder a primeira avaliação foi

de 7 minutos, e 10 minutos para a segunda avaliação, os efeitos de fadiga não foram considerados relevantes.

Ameaças à Validade Externa:

- **Instrumentação:** para ambas as avaliações, uma execução piloto foi realizada para permitir ajustes e garantir que a instrumentação estivesse correta, buscando mitigar essa ameaça.
- **Participantes:** alunos de graduação foram selecionados, sendo estes os potenciais usuários das aplicações educacionais móveis da LPS proposta. Na primeira avaliação, estes não foram selecionados de modo aleatório, mas não havia obrigatoriedade em participar. Na segunda avaliação, os alunos foram convidados aleatoriamente. Contudo, nenhuma delas permite a generalização dos resultados, necessitando de replicações para permitir maior capacidade de generalização.