Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Coordenação de Pós-Graduação em Ciência da Computação

Feedback em Ambientes Educacionais no Domínio de Programação

Priscylla Maria da Silva Sousa

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação, da Universidade Federal de Campina Grande - Campus I, como parte dos requisitos necessários para obtenção do grau de Mestre em Ciência da Computação.

Área de Concentração: Ciência da Computação
Linha de Pesquisa: Modelos Cognitivos e Computacionais

Joseana Macêdo Fechine Régis de Araújo
Evandro de Barros Costa
(Orientadores)

Campina Grande, Paraíba, Brasil

S725f Sousa, Priscylla Maria da Silva.
Feedback em ambientes educacionais no dom°nio da programa´²o / Priscylla Maria da Silva Sousa. - Campina Grande, 2018.
98 f.: il. color.

Disserta´²o (Mestrado em Ci¸ncia da Computa´²o) - Universidade Federal de Campina Grande, Centro de Engenharia El¶trica e Inform®tica, 2018.
"Orienta´²o: Prof.Ł Dr.Ł Joseana Mac¸do Fechine R¶gis de Araßjo, Prof. Dr. Evandro de Barros Costa."
Refer¸ncias.

1. Feedback. 2. Programa´²o. 3. Ambientes Educacionais. 4. Aprendizagem de programa´²o. I. Araßjo, Joseana Mac¸do Fechine R¶gis de. II. Costa, Evandro de Barros. III. T°tulo.

CDU 004.42(043)

FICHA CATALOGRΞFICA ELABORADA PELA BIBLIOTECΞRIA MARIA ANTONIA DE SOUSA CRB 15/398

"FEEDBACK EM AMBIENTES EDUCACIONAIS NO DOMÍNIO DE PROGRAMAÇÃO"

PRISCYLLA MARIA DA SILVA SOUSA

DISSERTAÇÃO APROVADA EM 08/10/2018

JOSEANA MACÊDO FECHINE RÉGIS DE ARAÚJO, Dra., UFCG
Orientador(a)

EVANDRO DE BARROS COSTA, Dr., UFAL
Orientador(a)

LEANDRO DIAS DA SILVA, Dr., UFAL
Examinador(a)

PATRICIA CABRAL DE AZEVEDO RESTELLI TEDESCO, Dra., UFPE
Examinador(a)

CAMPINA GRANDE - PB

Resumo

O uso de ambientes de aprendizagem *online* - cujo objetivo é auxiliar no processo de ensino-aprendizagem – tem apresentado um rápido crescimento nos últimos anos. Um dos principais componentes que constituem esses ambientes é o *feedback*. Considerado um dos fatores mais significativos no processo de aprendizagem, o *feedback* é essencial para que o aluno obtenha um retorno sobre o seu desempenho e aprendizagem. Fornecer *feedback* aos estudantes em ambientes de aprendizagem online é um desafio, visto que diversos fatores e configurações devem ser levados em consideração ao se especificar um *feedback*, como questões de conteúdo, forma de apresentação e nível de personalização. Dentro desse contexto, nesta pesquisa são discutidos os desafios da criação de *feedback* voltados para ambientes educacionais no domínio de programação, em que os estudantes precisam praticar programação por meio da submissão de algoritmos que solucionem problemas. A maioria desses sistemas provê um *feedback* de baixa qualidade, informando ao estudante apenas se sua solução está correta ou não. Muitos estudantes não conseguem, a partir desse simples *feedback*, progredir em seu aprendizado. Nessa perspectiva, esta pesquisa propõe um mecanismo de *feedback* para ser utilizado em ambientes educacionais de programação, que auxilia o estudante no processo de elaboração da sua solução, fornecendo dicas sobre os próximos passos e correção de erros. Para que o mecanismo fosse criado, foi elaborado um modelo conceitual para especificação do *feedback* que seria utilizado. A validação da pesquisa foi levada a efeito por meio da aplicação de um quase-experimento com um grupo de estudantes de uma disciplina introdutória de programação. Os resultados mostraram que a prática de exercícios de programação com o auxílio de mecanismo de *feedback* criado teve um impacto positivo na aprendizagem dos estudantes.

Abstract

The use of online learning environments - whose goal is to aid in the teaching-learning process - has shown rapid growth in recent years. One of the critical components of these environments is feedback. The feedback is considered one of the most significant factors in the learning process, and it is essential for the student to obtain a return on his/her performance and learning. Providing feedback to students in online learning environments is challenging since there is some factors and settings should be taken into account for specifying feedback, such as the content of questions, presentation form, and the level of customization. In this context, this work discusses the challenges of creating feedback for educational environments in the programming domain, where students need to practice programming through the submission of algorithms that solve problems. Most of the learning systems provide poor quality feedback, only informing the student if his/her solution is correct or not. Many students can not, from this simple feedback, progress in their learning. In this perspective, this work proposes a feedback mechanism, to be used in educational programming environments, which assists the student in the process of elaborating his solution, providing tips on the next steps and correction of errors. For the mechanism to be created, a conceptual model was developed to specify the feedback that would be used. The validation of the work was carried out through the application of a quasi-experiment with a group of students of an introductory programming course. The results showed that the practice of programming exercises with the help of the feedback mechanism created had a positive impact on student learning.

Dedicatória

À minha mãe, Antônia, pelo apoio incondicional em todos os momentos, mesmo desejando que eu permanecesse com os pés firmes no chão, ela permitiu que eu criasse asas para alçar voos cada vez mais altos.

Agradecimentos

Agradeço sempre a Deus, pelas maravilhas que ele tem realizado em minha vida e por sempre conduzir meus caminhos durante as dificuldades e percalços da vida.

À minha família, em especial à minha mãe (Antônia), meu pai (Sebastião) e meus irmãos (João, André e Filipe), pelo constante incentivo e por terem compreendido minhas ausências e momentos difíceis no decorrer dessa caminhada do mestrado.

Aos meus orientadores, Joseana Fechine e Evandro Costa, pelo apoio, pela confiança em mim depositada, a dedicação de tempo, os conselhos e sobretudo pela amizade.

À todos os professores que tive durante o mestrado, por todo conhecimento que me proporcionaram. A equipe da COPIN, por toda a ajuda, paciência e esclarecimentos em todas as ocasi oes que precisei.

Aos amigos que sempre me incentivam, desde a época do ensino médio, Alana, Anderson, Carol, Gilton, Uziel, Wilson, Isabela, Jaqueline e Paulo. Aos amigos de grupo de pesquisa, por todas as discussões enriquecedoras que tivemos, Jonathas, Cristina, Heitor, Marlos, Cleyton, Emanuele e Roberth. Também agradeço as irmãs que ganhei durante minha estadia em Campina Grande, Eduarda e Nicole, essa jornada não teria sido a mesma sem o apoio de vocês.

Conteúdo

Lista de Siglas e Abreviaturas

AUC - *Answer-Until-Correct (Responda até Acertar)*

EF - *Elaborated Feedback (Feedback Elaborado)*

IDE - *Integrated Development Environment (Ambiente de Desenvolvimento Integrado)*

ITS - *Intelligent Tutoring System (Sistema Tutor Inteligente)*

KCR - *Knowledge of the Correct Response (Conhecimento da Resposta Correta)*

KP - *Knowledge of Performance (Conhecimento de Performance)*

KR - *Knowledge of Result/Response (Conhecimento do Resultado/Resposta)*

MOOC - *Massive Open Online Course (Curso Online Aberto e Massivo)*

MTF - *Multiple-Try-Feedback (Feedback de Múltiplas Tentativas)*

AST - *Abstract Syntax Tree (Árvore Sintática Abstrata*

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Neste capítulo, é apresentado o contexto na qual a pesquisa está inserida, buscando elucidar as motivações que levaram ao surgimento e desenvolvimento desta dissertação. Em seguida, é apresentada a caracterização do problema de pesquisa. Posteriormente, os objetivos da pesquisa são expostos e, na sequência, é apresentada uma síntese da solução proposta e dos resultados obtidos. Por fim, a última seção deste capítulo presta-se a apresentar a estrutura da dissertação.

1.1 Contextualização e Motivação

Atualmente, a aplicação de tecnologias em educação se tornou de grande interesse da comunidade acadêmica e empresarial. Desde o advento da Web, muitos sistemas educacionais *online* têm surgido com o objetivo de auxiliar o processo de ensino-aprendizagem. Diversos trabalhos na literatura relatam a criação e o uso de sistemas educacionais para auxiliar tanto o ensino presencial quanto a distância, sendo possível encontrar sistemas voltados para diversos níveis de escolaridade e domínios do conhecimento (HEFFERNAN; HEFFERNAN, 2014; PRICE; DONG; LIPOVAC, 2017).

Nos últimos anos, ocorreu a popularização dos chamados Cursos *Online* Abertos e Massivos, do inglês *Massive Open Online Course* (MOOC), que são cursos realizados em ambientes virtuais de aprendizagem para turmas de tamanho elevado (SMITH; ENG, 2013). Tais ambientes se tornaram um sucesso acadêmico, oferecendo cursos em escala mundial (MAAS et al., 2014). Um tipo de ambiente que pode ser utilizado para a oferta de MOOCs são os

Sistemas Tutores Inteligentes, do inglês *Intelligent Tutoring System* (ITS). Já conhecidos há anos, são caracterizados por serem sistemas instrucionais baseados em computador com modelos de conteúdo instrucional que especificam "o que" ensinar, e estratégias de ensino que especificam "como" ensinar (WENGER, 1987).

A popularização desses sistemas ocorre em função dos grandes benefícios proporcionados, tanto para os estudantes, quanto para os professores. Os alunos têm a oportunidade de estudar a qualquer hora e lugar, enquanto os professores diminuem sua carga de trabalho, como, por exemplo, na correção de exercícios e o fornecimento de *feedback* aos estudantes. Em se tratando de *feedback*, este é um dos principais diferenciais do uso desses sistemas, dado que os mesmos podem oferecer um *feedback* imediato às ações do estudante.

O *feedback* é considerado um dos principais fatores que influencia o processo de aprendizagem, em especial no contexto de ambientes educacionais *online* (NARCISS, 2013). Segundo Narciss (2007), no contexto educacional, o termo *feedback* se refere a todas as informações pós-resposta que informam aos estudantes sobre o seu estado real de aprendizagem ou desempenho, com o intuito de guiar o processo de aprendizagem na direção necessária.

Os ambientes educacionais *online* podem possuir diferentes formas de *feedback*. Segundo Hattie e Timperley (2007), o *feedback* fornecido nesses sistemas auxilia na redução da discrepância entre o estado atual dos estudantes e os resultados de aprendizagem pretendidos.

O uso desses sistemas em disciplinas de cursos consideradas complexas e com alto índice de reprovação pode auxiliar os estudantes em seu processo de aprendizado, aumentando suas chances de êxito. Nos cursos da área de computação, algumas das disciplinas que frequentemente apresentam um alto grau de reprovação são as disciplinas introdutórias de programação (TENNYSON; BECK, 2018).

O ensino de programação constitui um conhecimento basilar para cursos de computação, desde cursos técnicos até de nível superior. Dada a sua importância, é fundamental que os alunos, além de compreender os conceitos básicos de programação, também concluam a disciplina com competência para construir soluções algorítmicas de qualidade para os problemas com os quais eles podem se deparar (KEUNING; HEEREN; JEURING, 2014). A dificuldade dos alunos na aprendizagem de programação acarreta tanto a evasão dos alunos dos cursos quanto algumas situações indesejadas, como, por exemplo, os alunos avançarem

nessas disciplinas ainda com dificuldades e com um domínio precário da prática de programação (LAHTINEN; ALA-MUTKA; JäRVINEN, 2005).

Para evitar os problemas citados, os professores costumam adotar em suas aulas uma metodologia conhecida como "*learning by doing*" (ANZAI; SIMON, 1979a). Ela estabelece que o aprendizado ocorre a partir da prática constante: quanto mais o aluno pratica a construção de soluções, maior a base de conhecimento que ele obtém para a elaboração de novas soluções. Os conhecimentos adquiridos e aplicados em soluções anteriores são utilizados para resolver os próximos problemas. Deste modo, o aluno exercita e aprimora seu raciocínio lógico e matemático.

Entretanto, o uso da metodologia "*learning by doing*" acarreta em sobrecarga de trabalho para o professor. Quanto mais exercícios práticos de programação os alunos realizarem, maior o volume de correções e *feedback* que o professor deve realizar. Outro agravante é que, quanto maior a quantidade de alunos por turma, maior será o volume de trabalho do professor. Apesar disso, a prática dos alunos e o recebimento de *feedback* são fundamentais para a evolução dos seus conhecimentos.

O problema da sobrecarga de volume de trabalho do professor, somado à necessidade de os alunos praticarem e receberem *feedback*, já vem sendo abordado na literatura (BERGIN; REILLY, 2005; CHAVES et al., 2014). Nesse contexto, os sistemas educacionais *online* vêm sendo utilizados para que os estudantes possam praticar programação e receber um *feedback* automático (DOUCE; LIVINGSTONE; ORWELL, 2005). Desta forma, para cada exercício o estudante pode submeter soluções e receber do sistema um *feedback* informando se estão corretas ou não (CHAVES et al., 2014). Este é o tipo mais comum de *feedback* encontrado em sistemas educacionais de programação (KYRILOV; NOELLE, 2016), o qual indica somente se a solução está correta baseada em um conjunto de testes. Esse tipo de *feedback* é fornecido somente após o estudante completar uma solução e submetê-la para avaliação. Ainda são raros, no contexto de programação, os sistemas que fornecem auxílio ao estudante durante a elaboração da solução (KEUNING; HEEREN; JEURING, 2014).

Ainda assim, as propostas de sistemas educacionais voltados para programação apresentam dificuldades em fornecer um *feedback* adequado e útil para os estudantes. Segundo Kyrilov e Noelle (2016), a maioria dos sistemas entrega um *feedback* de baixa qualidade. Em sua pesquisa, realizada com estudantes de programação, percebeu-se que a maioria dos

estudantes não consegue submeter uma solução correta em sua primeira tentativa, e destes, poucos tentam uma segunda vez. Kyrilov e Noelle (2016) sugerem que um *feedback* mais rico e detalhado seja oferecido ao estudante; neste caso, uma possível solução é o fornecimento de *feedback* durante o processo de elaboração da solução, como dicas e sugestões passo a passo.

Fornecer *feedback* ao estudante para auxiliá-lo durante a construção do algoritmo não é uma tarefa trivial. Caso o *feedback* não seja adequado, o estudante poderá passar a ignorar qualquer tipo de ajuda proposta pelo sistema. O auxílio também deve ser condizente com o estado atual da solução do estudante. Além disso, existem diversos recursos educacionais que podem compor o conteúdo do *feedback*, tais como: videoaulas, dicas textuais, soluções modelo, explicações do enunciado do problema, simulações, dentre outros (LAHTINEN; ALA-MUTKA; JäRVINEN, 2005).

1.2 Caracterização do Problema de Pesquisa

A propagação da criação de ambientes computadorizados voltados para a educação produziu novas oportunidades de pesquisa, assim como gerou novos desafios (JAIN et al., 2016). Um dos desafios existentes é a criação de mecanismos de *feedback* integrados a tais sistemas, o que possibilita ao estudante submeter atividades e receber um *feedback* intermédio.

Com base nos problemas apresentados na Seção 1.1, sobre as dificuldades dos estudantes na aprendizagem de programação, em especial no que diz respeito à prática de exercícios em sistemas educacionais, surgiu a seguinte questão: *como propor uma solução de feedback no domínio de programação?*

Segundo Keuning, Jeuring e Heeren (2018), a maioria das soluções de *feedback* existentes para o domínio de programação oferece *feedback* pós-submissão, ou seja, aguardam o estudante finalizar a solução de um problema para oferecer uma mensagem de *feedback*. De acordo com Kyrilov e Noelle (2016), esse tipo de *feedback* não é útil para a maioria dos estudantes, que possuem dificuldades na elaboração da solução.

Sendo assim, este trabalho visa contribuir para a seguinte questão de pesquisa:

QP. ***Como auxiliar o estudante durante a resolução de exercícios de programação?***

Para auxiliar o estudante na resolução de exercícios de programação, esta pesquisa apre-

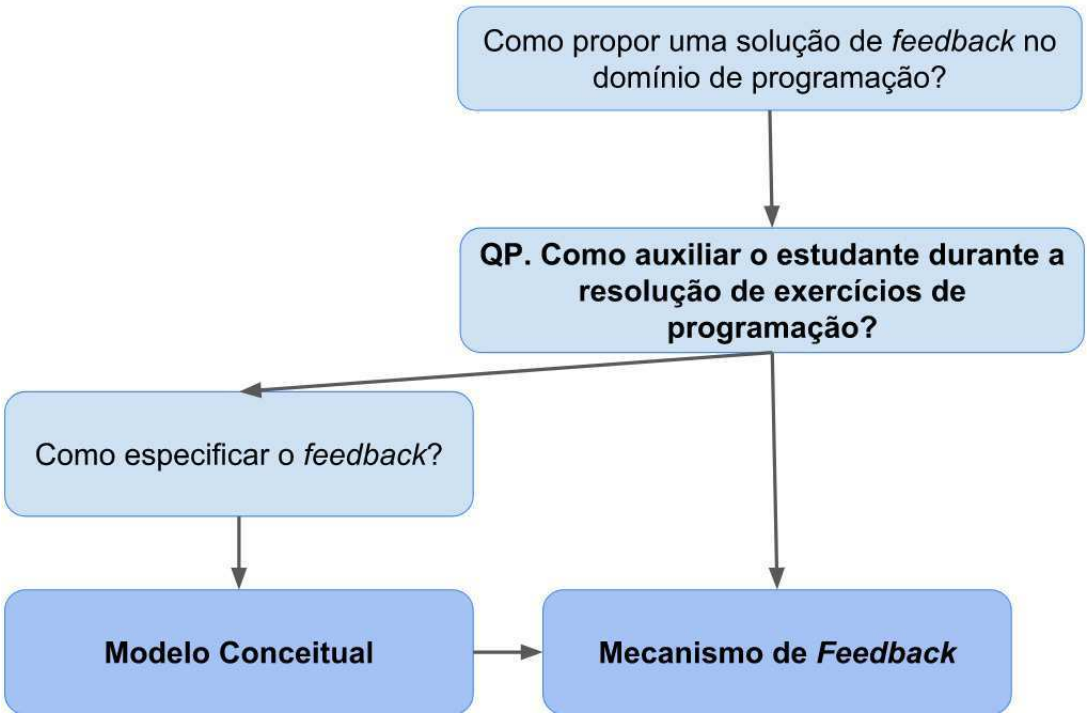
senta como solução a criação de um mecanismo de *feedback* que oferece dicas de correção de erros e ajuda na construção da solução.

Para que o mecanismo de *feedback* fosse criado, primeiro foi necessário realizar a especificação do *feedback* que seria utilizado, o que gerou uma subquestão de pesquisa: *como especificar o feedback?*

Para solução desta subquestão, foram revisados trabalhos da literatura e análise de sistemas educacionais *online* para programação. Por fim, foi proposta uma solução de modelo conceitual para especificação dos componentes do *feedback*. Este modelo foi utilizado para a especificação do mecanismo de *feedback* criado nesta pesquisa.

Na Figura 1.1 é apresentado um esquema da questão de pesquisa abordada nesta pesquisa, bem como as soluções que surgiram a partir dela.

Figura 1.1. Problema de pesquisa.



Fonte: autoria própria.

1.3 Objetivos da Pesquisa

Nesta seção serão descritos os principais objetivos da pesquisa apresentada.

1.3.1 Objetivo Geral

O objetivo geral da pesquisa é propor e validar um mecanismo de *feedback* que auxilie o estudante durante a resolução de exercícios de programação.

1.3.2 Objetivos Específicos

Os objetivos específicos da pesquisa são delineados nesta seção, a saber:

- Elaborar um modelo para especificação de mecanismos de *feedback* em sistemas educacionais no contexto de programação;
- Criar uma especificação para ser utilizada por mecanismos de *feedback* focados no domínio de programação;
- Criar um mecanismo de *feedback*para ser utilizado no domínio de programação;
- Integrar o mecanismo de *feedback* criado a um sistema educacional existente.

1.4 Organização da Dissertação

Este documento está organizado da seguinte forma: no Capítulo 2, são apresentados os conceitos teóricos necessários para compreensão da pesquisa realizada. No Capítulo 3, são discutidas as pesquisas correlatas que oferecem outras abordagens relacionadas à modelagem de *feedback*. No Capítulo 4, é apresentado o modelo conceitual para especificação de *feedback*. No Capítulo 5, é apresentada a instância do arcabouço criada para o domínio de programação e detalhado o processo de implementação do mecanismo de *feedback*, além de serem descritos a validação e os experimentos realizados. Por fim, no Capítulo 6, são apresentadas as considerações finais com a síntese das contribuições e propostas para pesquisas futuras.

Capítulo 2

Fundamentação Teórica

Neste capítulo, será apresentado o conceito de *feedback*, bem como suas principais características. Em seguida, é apresentada uma visão geral dos principais fatores que impactam a eficácia e a eficiência do *feedback* no processo de aprendizagem. Depois será apresentada uma visão geral do uso de *feedback* no ensino de programação, domínio para o qual o mecanismo de *feedback* desta pesquisa foi desenvolvido. Então, será apresentada uma seção sobre ambientes de apoio ao ensino de programação e, por fim, será realizada uma explanação sobre métodos de análise de soluções de programação para geração de *feedback*.

2.1 Conceitualização e Características do *Feedback*

O *feedback* tem sido considerado um dos elementos fundamentais para tornar o processo de aprendizagem mais eficiente (NARCISS, 2007), podendo ser encontrado em diversos trabalhos na literatura, sob uma variedade de configurações (MORY, 2003).

De acordo com Mory (2003), no contexto instrucional, o *feedback* pode ser descrito como qualquer comunicação ou procedimento com o aprendiz que forneça informações sobre a precisão de sua resposta, geralmente associada a uma pergunta instrucional. Ao se trabalhar com sistemas computadorizados de apoio à aprendizagem, o *feedback* pode ser considerado como qualquer informação apresentada ao estudante após qualquer entrada, com o objetivo de moldar as suas percepções, ou seja, auxiliar o estudante não apenas no seu processo de aprendizagem, mas na compreensão do próprio processo.

Embora alguns trabalhos considerem como *feedback* apenas as informações sobre a cor-

reção de uma atividade após o estudante finalizá-la, ele pode ser composto por outros tipos de informação, tais como: precisão da resposta, orientações de resolução, mensagens motivacionais, dicas de passo-a-passo e análises críticas sobre a qualidade das soluções dos alunos (MORY, 2003). Considerando que o conteúdo do *feedback* pode ser composto por recursos de suporte ao estudante na resolução de exercícios, vale a pena destacar o termo "*feedback* formativo", que pode ser definido como toda informação comunicada ao estudante com o objetivo de alterar o seu pensamento ou comportamento, tendo como resultado a promoção da melhoria da aprendizagem do estudante (SHUTE, 2008).

De acordo com Shute (2008), há uma série de variáveis que devem ser consideradas para que o *feedback* formativo cumpra seu objetivo de melhorar a aprendizagem do estudante, como, por exemplo, as características individuais dos estudantes e das atividades propostas. Além disto, existe uma variedade de conteúdos possíveis para o *feedback*, tais como explicações da atividade, sugestões de resoluções e exemplos de soluções. Neste contexto, cada tipo de conteúdo de *feedback* pode ser associado a características dos estudantes e particularidades dos exercícios. Além disto, diversos fatores podem influenciar na eficiência do *feedback*, tais como: a complexidade da atividade, fatores internos do estudante (ex: conhecimento a priori, habilidade de processar informação, vontade de superar erros, entre outros), o objetivo pedagógico do *feedback* (metas pretendidas que estão relacionadas ao desenvolvimento de competências e habilidades dos estudantes), o procedimento de diagnóstico (capacidade do sistema em avaliar o estudante, identificar os problemas e selecionar o *feedback* adequado) e a qualidade do *feedback* (NARCISS, 2007).

Narciss (2007) considera o *feedback* como uma entidade multidimensional composta por três entidades, a saber:

- **Funcionalidade**: O *feedback* pode ter funções cognitivas, metacognitivas e motivacionais. O *feedback* cognitivo tem a função de auxiliar o estudante na aquisição de conhecimento do conteúdo, estratégias e procedimentos que devem ser utilizados na resolução de problemas a fim de se chegar a uma solução correta. O *feedback* metacognitivo tem a função de abordar estratégias metacognitivas e fornecer ajuda para o estudante monitorar e avaliar seus objetivos e estratégias de aprendizagem. A função do *feedback* motivacional é manter o esforço, persistência e intensidade do estudante durante a resolução de atividades;

- **Conteúdo**: Refere-se aos aspectos semânticos relacionados ao conteúdo contido na mensagem de *feedback*. Exemplos de conteúdo: dicas, explicações, informar se as soluções estão corretas, analogias, simulações, entre outros;

- **Apresentação**: Diz respeito à forma e ao modo como a mensagem de *feedback* é apresentada ao estudante. Neste aspecto, são discutidas questões sobre quando oferecer *feedback* (imediato ou não), quantidade de tentativas que o estudante pode realizar para solucionar a tarefa, se será utilizada alguma técnica para personalização do *feedback* ou se todos os estudantes receberão o mesmo *feedback* independente de suas características individuais, ou se alguma opção de escolha será fornecida ao estudante, como, por exemplo, escolher entre uma mensagem em forma de texto ou vídeo, etc.

Cada uma destas entidades deve ser definida de forma clara na implementação de mecanismos de *feedback*, uma vez que sua combinação influenciará diretamente na eficiência do mecanismo.

2.2 Eficiência do *Feedback*

Narciss (2013) cita três fatores que podem afetar a eficiência do *feedback*: i) complexidade da tarefa; ii) fatores internos do estudante (ex: conhecimento a priori, vontade de superar os erros, habilidade de interpretação, etc); iii) fatores externos (ex: acurácia do diagnóstico e identificação dos erros, seleção das ações corretivas e qualidade do *feedback*).

Percebe-se que tais fatores são de difícil mensuração, em especial os fatores internos do estudante. Com relação a estes, o conhecimento *a priori* pode ser verificado com aplicações de pré-testes, entretanto a vontade do estudante em superar seus erros e melhorar seu aprendizado é algo complexo de ser determinado, podendo-se optar pela declaração explícita do aluno.

Com relação à complexidade da tarefa, aquelas com um nível de complexidade maior podem ser de difícil compreensão para o estudante, exigindo um *feedback* mais detalhado. Tal nível pode ser determinado de forma estática por um professor ou ser inferido a partir do percentual de erros e acertos dos estudantes.

Os fatores externos que influenciam o *feedback* podem ser medidos de forma manual ou

automática. Por exemplo, pode-se armazenar o *log* de mensagens de *feedback* escolhidas pelo sistema e exibidas ao estudante e, posteriormente, verificar se, dado o conjunto de mensagens disponíveis, o *feedback* exibido foi a escolha mais adequada.

A eficiência do *feedback* é tratada como o efeito mais benéfico no desempenho do estudante (STRIJBOS; PAT-EL; NARCISS, 2010). Entretanto, é extremamente complexa de ser medida, pois depende de um número variado de fatores, sendo, assim, necessária a elaboração de experimentos bem controlados para isolar cada um desses fatores e verificar a influência deles (STRIJBOS; PAT-EL; NARCISS, 2010).

Em função da complexidade no isolamento e controle de tantos fatores, as pesquisas mais recentes na área têm aplicado quase-experimentos (PETERS; KÖRNDLE; NARCISS, 2018), que visam estimar o impacto causal de uma intervenção (COOK; CAMPBELL; SHADISH, 2002).

2.3 *Feedback* na Aprendizagem de Programação

A temática do processo de ensino-aprendizagem de programação tem sido discutida e abordada em diversos trabalhos ao longo dos anos. Frequentemente, as turmas de programação costumam ser grandes e heterogêneas, o que torna difícil para o professor criar abordagens e materiais didáticos que beneficiem as particularidades de cada estudante (LAHTINEN; ALA-MUTKA; JäRVINEN, 2005). Entre essas particularidades, encontram-se o estilo de aprendizagem e o perfil de codificação do estudante, ou seja, o modo como o estudante se comporta e as ações que ele executa enquanto codifica a solução para um problema.

Com relação ao perfil de codificação dos estudantes, Blikstein (2011) realizou um estudo no qual analisou o processo de elaboração de soluções algorítmicas de estudantes com o objetivo de identificar perfis de codificação. Ao analisar o processo de criação das soluções dos estudantes, Blikstein constatou a existência de comportamentos distintos entre os participantes do estudo. Foi observado que, durante a codificação, muitos dos estudantes observavam exemplos de soluções semelhantes à do exercício proposto durante a codificação, outros costumavam realizar buscas pelo uso correto da sintaxe dos comandos. Em suas conclusões, o estudo indica que recursos de suporte aos estudantes durante o processo de elaboração da solução e estratégias de ensino precisam ser projetadas para atender a diversidade de estilos

de codificação e o perfil de variados tipos de estudantes.

Levando em consideração o estudo de Blikstein, pode-se constatar que os estudantes têm o hábito de buscar auxílio durante a resolução dos exercícios de programação. Neste sentido, um sistema computadorizado poderia auxiliá-los e fornecer ajuda em forma de *feedback*. Também é possível constatar que diferentes tipos de *feedback* podem ser usados para ajudar estudantes com perfis de codificação distintos.

Um dos primeiros trabalhos a analisar o impacto do *feedback* em sistemas de ensino de programação foi realizado por Corbett e Anderson (2001). Eles analisaram qual o momento mais apropriado para oferecer *feedback* ao estudante e qual o nível de controle que o sistema deve ter sobre o *feedback*. No estudo foram usados dois tipos de recursos no *feedback*: indicações de erros e dicas para construção da solução. Eles constataram que o comportamento dos estudantes variava de acordo com cada configuração de *feedback*, e que, no geral, quanto maior o controle do sistema sobre o *feedback*, menor o tempo de resolução e a taxa de erros.

Em sua maioria, os sistemas de apoio ao ensino de programação oferecem *feedback* aos estudantes apenas após a submissão de uma solução completa (KEUNING; HEEREN; JEURING, 2014). Além da informação sobre a corretude da solução, o tipo mais comum de *feedback* é a apresentação dos testes para os quais o algoritmo do aluno informou a saída de forma correta ou incorreta (PAES et al., 2013). Outro *feedback* comum é a exibição das mensagens de erro de compilação e execução do código (ALVES; JAQUES, 2014).

Existem ainda trabalhos que possuem mecanismos de *feedback* não automatizados, a partir dos quais o estudante deve esperar pelo *feedback* que será produzido por um professor, monitor ou colega de turma (POLITZ; KRISHNAMURTHI; FISLER, 2014; HYYRYNEN et al., 2010). Nesses casos, também é necessário que o estudante submeta uma solução completa para receber posteriormente alguma mensagem de *feedback* sobre a mesma.

2.4 Ambientes de Apoio ao Ensino de Programação

Professores de disciplinas introdutórias de programação tendem a buscar recursos e ferramentas computacionais para auxiliá-los em suas atividades (ALVES; JAQUES, 2014). Tais ferramentas visam atender necessidades específicas e bastante diversificadas. Marcelino, Mihaylov e Mendes (2008) apresentam alguns tipos de recursos e ferramentas criados como

suporte para disciplinas introdutórias de programação, a saber:

- Linguagens de programação simplificadas (ROBERTS, 2001);
- Ambientes de desenvolvimento controlados (KÖLLING et al., 2003);
- Ferramentas para testar as soluções dos estudantes (SANTOS; RIBEIRO, 2011);
- Ferramentas gráficas para gerar animações e simulações dos algoritmos (BEN-ARI et al., 2002);

Existem ainda ferramentas que integram mais de um dos tipos citados (MANSO; MARQUES; DIAS, 2010).

Outra ferramenta comumente usada são os Juízes *Online*, que são plataformas concebidas para serem usadas em competições de programação. A principal funcionalidade dessas plataformas é a avaliação do código submetido pelos competidores, na qual é fornecido um conjunto de entradas de testes para verificar se o código submetido retorna as saídas esperadas para cada entrada.

Embora úteis, os Juízes *Online* não foram concebidos com objetivos pedagógicos; foram criados para serem usados em competições e, posteriormente, pelos professores para auxiliar na prática de programação dos alunos, uma vez que assim eles poderiam ter um rápido *feedback* sobre a corretude de seus algoritmos. Dada a utilidade dos Juízes *Online*, algumas pesquisas objetivam acrescentar a eles novos recursos com fins pedagógicos, tais como o JOnline (SANTOS; RIBEIRO, 2011) e o MOJO (CHAVES et al., 2014).

As formas de submissão de soluções nestes juízes não possuem grandes diferenças entre si, o usuário insere o código da solução ou precisa anexar o arquivo com o código, em seguida, indica em qual linguagem de programação a solução foi elaborada (ou o sistema pode inferir a linguagem de forma automática a partir do arquivo). A adição de mais elementos pedagógicos às funcionalidades dos Juízes *Online* culminou no surgimento de novos ambientes com o foco em oferecer melhor suporte aos estudantes e professores, agregando o uso de *feedback* mais sofisticado e elaborado, com o intuito de ajudar o estudante a progredir no seu aprendizado.

2.5 Análise das Soluções em Ambientes de Apoio ao Ensino de Programação

Existem duas abordagens possíveis para realizar avaliação automática de algoritmos, a abordagem dinâmica e a abordagem estática (ALA-MUTKA, 2005a). Ambas as abordagens são amplamente utilizadas na literatura em ambientes de apoio ao ensino de programação (DOUCE; LIVINGSTONE; ORWELL, 2005). A abordagem dinâmica possui como requisito a execução do algoritmo; para isso, ele deve estar completo e sintaticamente correto. Um exemplo desta abordagem é a avaliação utilizando casos de testes. Neste tipo de avaliação, o algoritmo do estudante é executado com a entrada do caso de teste; ao fim da execução, o algoritmo do estudante irá fornecer uma saída que será comparada com a saída esperada do caso. Se elas forem iguais, o algoritmo é considerado correto.

A abordagem estática não requer que o algoritmo seja executado, sendo assim, ela pode ser utilizada mesmo quando o algoritmo do estudante possui erros sintáticos ou quando está incompleto. Um exemplo desta abordagem é o uso de analisadores de similaridade de algoritmos, nos quais a solução algorítmica do estudante é comparada com soluções modelo criadas pelo professor.

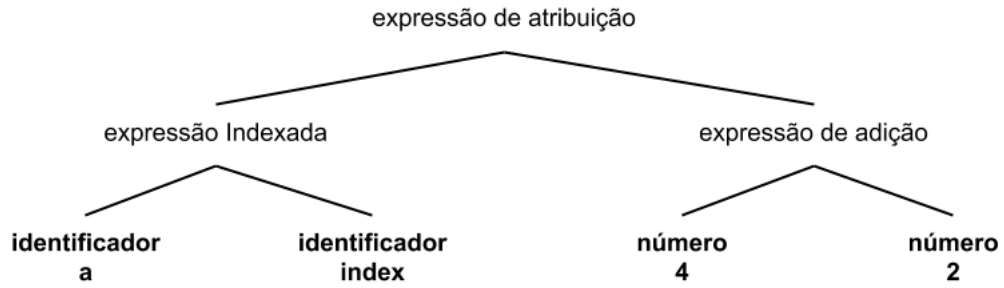
A abordagem estática também pode ser usada na análise do algoritmo para fornecer mensagens de *feedback* ao estudante, visto que ela não precisa que o algoritmo esteja finalizado ou correto.

Os analisadores de similaridade podem utilizar diversas técnicas para calcular a similaridade entre as soluções modelo e a solução do estudante. Elas se baseiam na comparação da diferença entre as soluções. Essa comparação pode ser realizada por meio de técnicas de comparação entre grafos ou entre *strings*. Na comparação usando grafos, a estrutura dos algoritmos é representada em grafos e suas diferenças estruturais são comparadas. Na comparação usando *strings*, cada algoritmo é representado por uma *strings* e aplicado algum método para verificar as diferenças entre elas, como, por exemplo, o cálculo de distância de Levenshtein.

Ambas as técnicas de comparação grafos e *string* podem fazer uso de árvores sintáticas abstratas, do inglês *abstract syntax tree* (AST). Uma AST é uma estrutura de dados que representa os elementos estruturais de um algoritmo e seus relacionamentos (LOUDEN,

2004). A Figura 2.1 apresenta um exemplo de árvore sintática abstrata para a expressão na linguagem C: $a[index] = 4 + 2$.

Figura 2.1. Exemplo de AST gerada para a expressão $a[index] = 4 + 2$ na linguagem C.



Fonte: adaptada de Louden (2004).

2.6 Considerações Finais do Capítulo

Neste capítulo, foram descritos os conceitos básicos necessários para uma melhor compreensão da pesquisa, entre eles a definição de *feedback* e de suas principais características. Foi discutido que o *feedback* engloba não apenas a mensagem apresentada ao estudante após a submissão de uma atividade, mas também outros tipos de informação, como mensagens motivacionais e orientação na resolução de problemas. Também foi discutido sobre a complexidade envolvida na determinação da eficiência do *feedback* no contexto educacional. Por fim, foram apresentados tipos de ambientes usados no ensino de programação e como eles realizam a análise de soluções.

No próximo capítulo, serão apresentadas as principais pesquisas correlatas sobre propostas de modelagem do *feedback* e mecanismos de *feedback* criados com o intuito de ajudar o estudante durante a resolução de exercícios.

Capítulo 3

Pesquisas Correlatas

Este capítulo está dividido em duas seções. A seção 3.1 tem como objetivo apresentar as propostas encontradas na literatura que visam representar, modelar ou caracterizar o *feedback* em um contexto educacional. A busca e seleção dos trabalhos foi realizada nas seguintes bases de dados: ACM Digital Library, IEEE Xplore, Google Scholar, Scopus, Web of Science e Portal de Periódicos da Capes. Para se obter um conjunto de pesquisas mais atualizadas com o contexto educacional atual, as buscas de publicações foram limitadas às divulgadas a partir do ano de 2010. A seção 3.2 tem como objetivo abordar as pesquisas correlatas de mecanismos de *feedback* cuja proposta está focada no fornecimento de *feedback* aos estudantes durante o processo de elaboração da solução no domínio de programação.

Ao final de cada seção, é apresentada uma comparação crítica entre as características da pesquisa desenvolvida e as demais pesquisas correlatas.

3.1 Pesquisas Correlatas do Modelo Conceitual para Especificação de *Feedback*

Nesta seção, serão apresentadas as pesquisas correlatas que propõem uma modelagem ou especificação para o *feedback* usado em ambientes educacionais *online*. Na seção 3.1.1 será apresentada uma pesquisa cujo foco é propor uma modelagem de *feedback* por meio de padrões, a fim de auxiliar o desenvolvedor de mecanismos de *feedback*. Na seção 3.1.2 é apresentada uma abordagem com foco no professor, cujo objetivo é ajudá-lo a especificar

o *feedback* que deseja utilizar. A seção 3.1.3 contém a apresentação de uma abordagem que tem por objetivo ser utilizada para avaliação e comparação entre *feedbacks* usados em diferentes ambientes *online*. Por fim, a seção 3.1.4 apresenta uma análise comparativa entre as abordagens apresentadas e o modelo conceitual proposto neste trabalho para especificação de *feedback* pedagógico em sistemas educacionais *online*.

3.1.1 Abordagem Voltada ao Desenvolvedor

Em um conjunto de artigos publicados a partir do ano de 2014, Inventado e Scupelli propõem a criação de um conjunto de padrões de projeto para o desenvolvimento de sistemas de aprendizagem *online*. Tais padrões tem por objetivo auxiliar professores e desenvolvedores na criação desses sistemas. O modelo apresentado em Inventado et al. (2017) constitui a versão mais recente publicada, o qual é composto por cinco padrões de projeto gerais: Problemas, Domínio de Aprendizagem, Motivação, Aprendizagem Personalizada e *Feedback* de Aprendizagem. Cada padrão diz respeito a uma característica/especificação, descritos a seguir.

1. **Problemas (*Problems*)**: Diz respeito à criação de problemas em sistemas de aprendizagem *online*.

2. **Domínio de Aprendizagem (*Mastery Learning*)**: Visa abordar os desafios voltados a garantir aos estudantes o domínio de um conceito ou habilidade.

3. **Motivação (*Motivation*)**: Contém os padrões de projeto que ajudam a manter a motivação do estudante enquanto ele aprende.

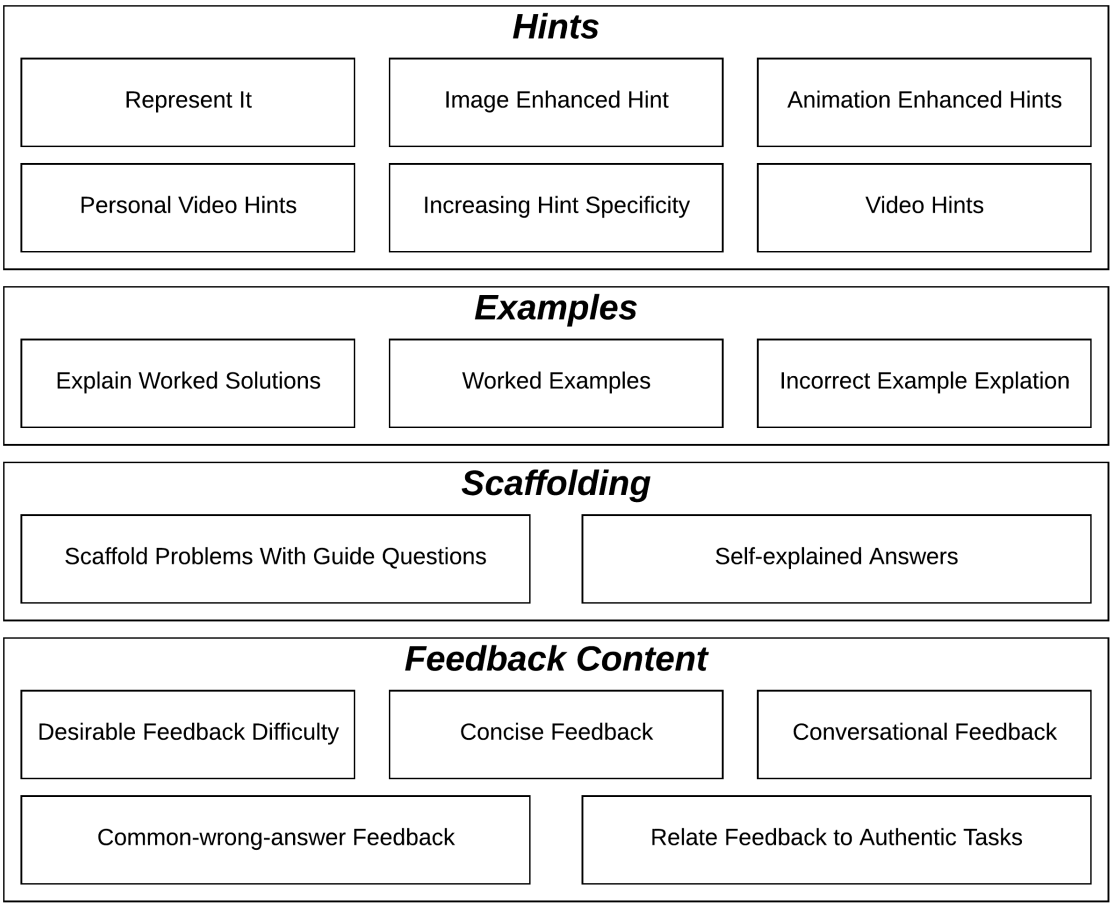
4. **Aprendizagem Personalizada (*Personalized Learning*)**: Contém padrões de projeto que fazem com que seja possível ao sistema se adaptar às características dos estudantes, tais como seu estado de conhecimento, seu emocional e sua personalidade.

5. ***Feedback* de Aprendizagem (*Learning Feedback*)**: Contém padrões de projeto que abordam desafios relacionados à geração de *feedback* para os estudantes.

Para a criação dos padrões de projeto, os autores utilizaram uma metodologia de mineração de dados chamada 3D2P (INVENTADO; SCUPELLI, 2016a), baseada nos dados

de um ambiente de ensino de matemática denominado ASSISTments (HEFFERNAN; HEFFERNAN, 2014). Para cada um dos padrões gerais existem os subpadrões de projetos. Para os efeitos da pesquisa, serão considerados e discutidos somente os padrões referentes ao *Feedback* de Aprendizagem, que são apresentados na Figura 3.1.

Figura 3.1. Modelo de Padrões de Projeto para *Feedback* de Aprendizagem.



Fonte: Adaptado de Inventado et al. (2017).

De acordo com Inventado et al. (2017), projetar um *feedback* efetivo para sistemas de aprendizagem *online* é uma tarefa não trivial, dado que existem diversos fatores que precisam ser considerados, como o ambiente de aprendizagem, o histórico do estudante, as diferenças entre os estudantes, entre outros. Em seu modelo, o *feedback* de aprendizagem se divide em quatro componentes, sendo os três primeiros: Dicas (*Hints*), Exemplos (*Examples*) e *Scaffolding*. Estes componentes constituem diferentes estratégias para a geração de *feedback*. O quarto componente é o Conteúdo de *Feedback* (*Feedback Content*), cujo foco está no conteúdo do *feedback* que será utilizado pelas três estratégias citadas.

As estratégias são um conjunto de padrões que indicam como o sistema deve proceder na interação com o estudante durante o fornecimento de *feedback*. Cada padrão indica o que deve ser usado, fornecido ou pedido ao estudante. Por exemplo, um padrão de estratégia pode determinar que o estudante pode requisitar ajuda ou se o sistema irá realizar algum tipo de detecção e provimento automático.

Na Tabela 3.1, são apresentados os padrões de estratégias de Exemplos (*Examples*). O detalhamento desses padrões pode ser encontrado em (INVENTADO et al., 2017; INVENTADO; SCUPELLI, 2016b, 2015). Na Tabela 3.2, são apresentados os padrões de estratégias de *feedback Scaffolding* (INVENTADO et al., 2017; INVENTADO; SCUPELLI, 2016b).

Tabela 3.1. Padrões de Estratégias por Exemplos Propostos por Inventado & Scupelli (2017).

Padrão	**Descrição**
Explanation Worked Solutions	Fornecer aos alunos soluções trabalhadas, claramente explicadas quando esses não puderem responder os problemas corretamente, apesar de receberem suporte.
Worked Examples	Fornecer aos alunos um exemplo semelhante ao problema que eles precisam resolver, para que possam entender como resolver o problema, sem revelar a resposta.
Incorrect Example Explanation	Pedir para os estudantes explicarem um exemplo incorreto, para ajudá-los a entender e evitar erros comuns e conceitos errados.

Tabela 3.2. Padrões de Estratégias por *Scaffolding* propostos por Inventado & Scupelli (2017).

Padrão	**Descrição**
Scaffold Problems with Guide Questions	Pedir aos alunos que respondam a perguntas de orientação que possam ajudá-los a lembrar de conhecimentos anteriores ou fazer inferências necessárias para resolver um problema.
Self-explanined Answers	Pedir aos alunos que expliquem sua resposta, para garantir que eles a compreendam, para ajudar a reforçar seu entendimento e incentivá-los a fazer generalizações a partir de suas soluções.

Na Tabela 3.3, são apresentados os padrões de projeto para estratégias de fornecimento de dicas. As especificações dessas estratégias podem ser encontradas em (INVENTADO et al., 2017; INVENTADO; SCUPELLI, 2016a, 2016c).

Os padrões de conteúdo de *feedback* se referem a técnicas que devem ser usadas para a criação do conteúdo. Na Tabela 3.4 são apresentados os cinco padrões propostos até o momento, dos quais apenas o primeiro da tabela já foi concluído. Este padrão propõe que

o criador do conteúdo do *feedback* realize uma análise de problemas para os quais já existe uma base de soluções submetidas pelos estudantes. A partir dos erros comuns encontrados nessas soluções, deve-se criar o conteúdo de *feedback* abordando esses erros e ensinando-os a corrigi-los ou evitar que sejam cometidos.

Tabela 3.3. Padrões de Estratégias por Dicas Propostos por Inventado & Scupelli (2017).

Padrão	**Descrição**
Represent it	Incentivar os alunos a externar seus pensamentos usando representações, para ajudá-los a entender melhor o problema e descobrir a resposta.
Animation Enhanced Hint	Incorporar animações em dicas, para quebrar a monotonia, capturar a atenção e fornecer maneiras alternativas de analisar ou resolver o problema.
Increasing Hint Specificity	Permitir que os estudantes requisitem dicas elaboradas de forma progressiva, em que a última dica contém a resposta correta.
Image Enhanced Hint	Esclarecer dicas para problemas de matemática, adicionando imagens que ajudam a desambiguar termos e explicações confusas.
Personal Video Hint	Usar vídeos pessoais para tornar as dicas mais significativas para os alunos.
Video Hint	Usar vídeo para apresentar o feedback que ajudará os estudantes a visualizar o problema, capturar sua atenção e minimizar a tendência de ignorar o feedback.

Tabela 3.4. Padrões de Conteúdo de *Feedback* propostos por Inventado & Scupelli (2017).

Padrão	**Descrição**
Common-wrong-answer feedback	Identificar respostas erradas comuns para um determinado problema e construir um feedback para abordar os erros e equívocos identificados. Sendo os erros considerados deslizes cometidos por desconhecimento e os equívocos considerados como falhas devido a má interpretação ou uma confusão de entendimento.
Concise feedback	Evitar explicações de feedback irrelevantes e desnecessariamente longas, para que os alunos possam se concentrar nas informações necessárias para resolver o problema.
Conversational feedback	Usar um estilo de conversação na comunicação de ideias com os alunos, para tornar o material mais envolvente. Como por exemplo, simular um diálogo com o estudante por meio de mensagens de texto ou fazendo uso de um *chatbot*.
Desirable feedback difficulty	Considerar o que os estudantes já sabem, para prover um feedback que os desafiará.
Relate feedback to authentic tasks	Usar exemplos baseados em configurações do mundo real, para ajudar os alunos a entender o valor da habilidade ensinada.

3.1.2 Abordagem Voltada ao Professor

O modelo proposto por Narciss (2013) tem como objetivo fornecer uma base teórica e empírica para orientar a concepção e avaliações de estratégias de feedback. Em seu trabalho, são especificados tipos de feedback, e também é apresentada uma conceitualização de feedback como uma entidade multidimensional. Em síntese, o trabalho busca explicitar os principais fatores que influenciam o feedback e seus efeitos nos estudantes.

Como foi apresentado anteriormente no Capítulo 2 (Fundamentação Teórica), Narciss (2013) define feedback em um contexto educacional, como uma informação pós-resposta que é oferecida aos estudantes para informar o seu estado atual de conhecimento ou a sua performance. Em seu trabalho, são expostos diversos fatores que afetam a eficiência de feedback. Narciss afirma que mesmo o feedback mais sofisticado é inútil se os estudantes não o utilizarem ou não estiverem dispostos a investir tempo e esforço para melhorar e corrigir seus erros. Além da força de vontade, os estudantes precisam das habilidades necessárias para satisfazer os requisitos relacionados à correção de seus erros.

O trabalho de Narciss (2013) catalogou os tipos de *feedback* descritos a seguir:

- **Conhecimento de Performance (do inglês, *Knowledge Of Performance*, KP)**
  Este tipo de feedback contém informações sobre o desempenho do estudante após o mesmo ter submetido uma tarefa ou um conjunto de tarefas ao sistema, como, por exemplo, o percentual de questões solucionadas de forma correta.

- **Conhecimento de Resultado/Resposta (do inglês, *Knowledge Of Result/Response*, KR)**
  Este tipo de feedback contém apenas uma informação binária indicando se a atividade do estudante está correta ou incorreta.

- **Conhecimento da Resposta Correta (do inglês, *Knowledge Of The Correct Response*, KCR)**
  Após a submissão de uma solução, este tipo de feedback oferece ao estudante a visualização de uma resposta correta para a atividade.

- **Responda até Acertar (do inglês, *Answer-Until-Correct*, AUC)**
  Este feedback oferece ao estudante o feedback KR, juntamente com a oportunidade

de realizar uma nova tentativa para solucionar a atividade corretamente. A quantidade de feedback é fornecida de forma ilimitada, até o estudante acertar a resolução da atividade.

- **Feedback de Múltiplas Tentativas (do inglês, *Mutiple-Try Feedback*, MTF)**
  Este feedback oferece ao estudante o feedback KR, juntamente com a oportunidade de realizar novas tentativas. Entretanto, diferente do feedback AUC, existe um número limitado de tentativas.

- **Feedback Elaborado (do inglês, *Elaborated Feedback*, EF)**
  Neste tipo de feedback, o estudante recebe informações adicionais, além de KR ou KCR. Por exemplo, dicas de como solucionar a atividade ou indicações de erros em sua solução.

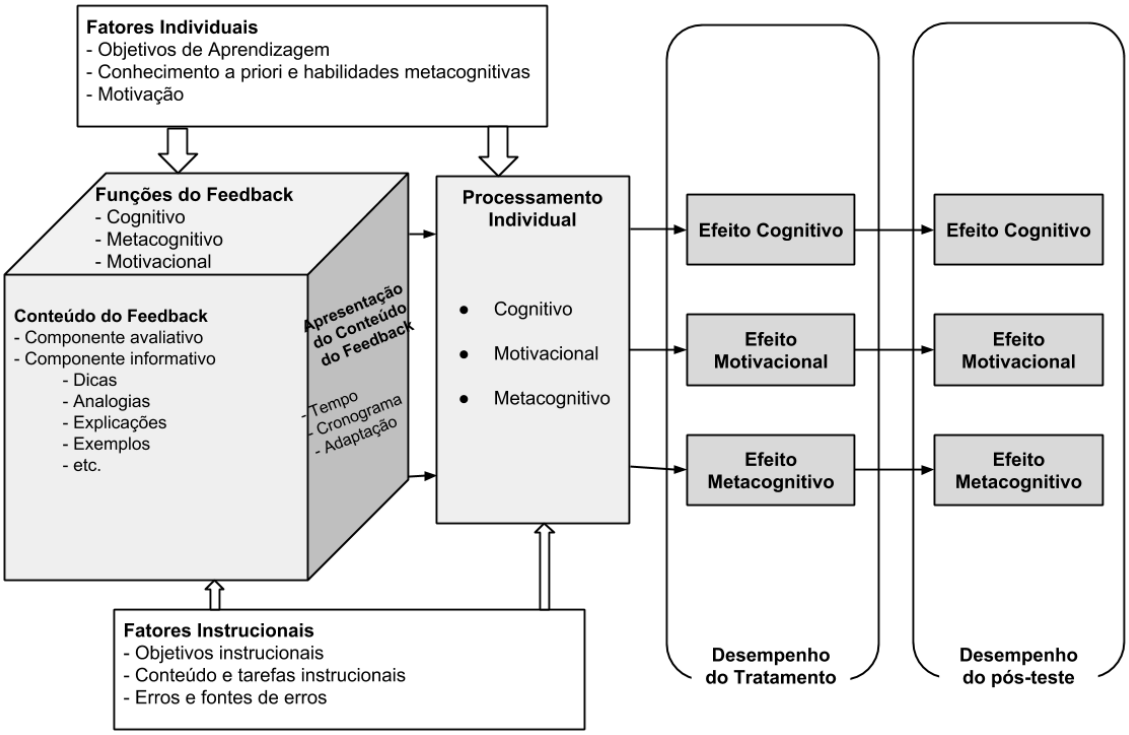
ParaNarciss (2013), em um contexto educacional, é necessário diferenciar o feedback fornecido ao estudante por fontes externas e internas. O feedback de fonte interna é gerado e consumido pelo próprio estudante, são suas percepções pessoais que ocorrem durante a realização de atividades; como, por exemplo, perceber que estava aplicando um conceito de forma equivocada. O feedback fornecido por fontes de informação externas pode ser proveniente de diversas fontes, como o professor, um sistema computacional, um colega de turma, entre outros. O modelo dos fatores e influências do feedback externo proposto por Narciss é apresentado na Figura 3.2.

Para Narciss (2007), o feedback externo pode ser visto como uma entidade multidimensional, possuindo três facetas: função, conteúdo e apresentação de conteúdo.

No que se refere à função do feedback externo, esta pode ser dividida em três, descritas a seguir:

- **Função Cognitiva**: a função cognitiva do feedback diz respeito aos elementos de conhecimento relacionados ao conteúdo, procedimentos e estratégias que o estudante precisa saber para chegar a uma solução correta. Pode ser feita a seguinte distinção entre os as funções cognitivas de feedback relacionadas às respostas incorretas: a) *função de informação*, casos nos quais o estudante desconhece o tipo, localização ou motivo dos seus erros na solução; em programação, por exemplo, esquecer um

Figura 3.2. Modelo de *feedback* externo proposto por Narciss (2007).



Fonte: Adaptado de Narciss (2007).

ponto e vírgula ao final de uma linha do programa na linguagem C; b) *função de conclusão*, em que o erro é causado pela falta de conhecimento do estudante, por exemplo, uma atividade de programação exigir o uso de uma estrutura de repetição, mas o estudante não inseri-la na solução; c) *função corretiva*, em que o erro é atribuído a um conhecimento errado do estudante, por exemplo, uma atividade de programação exigir o uso de uma estrutura de repetição e o estudante utilizá-la de forma incorreta; d) *função de diferenciação*, em que o erro é atribuído a um conhecimento impreciso, por exemplo, o estudante definir de forma errada os tipos das variáveis do programa; e) *função de restruturação*, em que o erro é oriundo de conexões errôneas entre os conteúdos.

- **Função Metacognitiva**: esta função do feedback diz respeito ao monitoramento e avaliação das metas do estudante, em conjunto com sua motivação para gerar suas próprias informações e automonitoramento. A função metacognitiva pode ser aplicada em casos nos quais os estudantes precisam ser encorajados a gerar seus próprios critérios de

monitoramento ou avaliação. Por exemplo, fazendo perguntas sobre as estratégias de resolução, solução criada ou ações do estudante.

- **Função Motivacional**: a maioria dos estudos sobre feedback é focada no desempenho do estudante nas atividades propostas, negligenciando o impacto motivacional do feedback. O feedback motivacional pode ser explorado no feedback EF das seguintes formas: a) função de incentivar o estudante a progredir e continuar suas atividades; b) função de facilitar o processo de aprendizagem do estudante, por meio de mensagens que o façam superar suas dificuldades; c) função de melhorar a auto-eficácia, processo no qual o estudante, após cometer erros em uma atividade, consegue solucioná-la com sucesso; e d) função de reatribuição.

Quanto ao conteúdo da mensagem de feedback, este pode ser constituído por dois componentes básicos: avaliação e informação. A avaliação diz respeito à correção das atividades e ao desempenho do estudante. A informação complementa a avaliação por meio de detalhes sobre o conteúdo da atividade, erros e soluções. A combinação desses dois componentes pode gerar uma variedade de conteúdos de feedback. Nesse sentido, o feedback elaborado (EF) é o que apresenta melhores resultados, porém também é o que demanda mais esforço. O conteúdo do feedback deve ser adaptado ao tipo de atividade que o estudante deve executar.

A apresentação do conteúdo diz respeito à forma e ao modo como o feedback é apresentado ao estudante. O feedback pode ser fornecido após a submissão de uma solução ou durante a sua elaboração. Neste último caso, ele tem a função de orientar o estudante para a conclusão da atividade com sucesso. A apresentação pode ocorrer de forma imediata ou posterior. Na apresentação imediata, o estudante recebe o feedback durante a elaboração da solução ou imediatamente após a sua submissão. Deste modo, ele não tem que esperar um tempo indeterminado para saber se sua solução está correta, e caso não esteja, pode imediatamente iniciar sua correção e tentar novamente. Esta forma de apresentação pode interferir na reflexão do estudante e na forma como ele lida com uma solução incorreta ou incompleta.

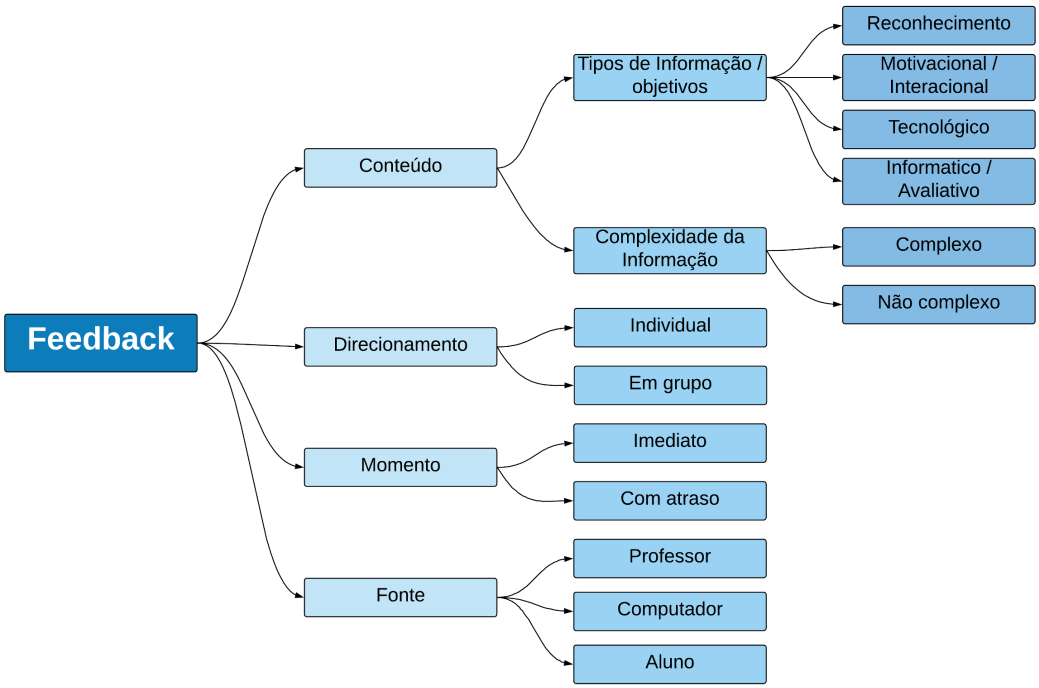
No feedback posterior, o estudante recebe informações sobre sua solução algum tempo após a sua submissão. Seu uso é recomendado quando a avaliação automática é custosa ou para trabalhar habilidades de metacognição com o estudante. A forma de apresentação também pode ser adaptativa ou não. Nos casos em que a apresentação adaptativa é utilizada,

deve-se refletir sobre quais características do estudante devem ser usadas para a adaptação e como a adaptação será usada para determinar a forma do conteúdo e a sequência do feedback. Além disto, também deve-se determinar o momento no qual o feedback será apresentado, se ele será requisitado pelo estudante ou se o sistema irá detectar quando o estudante precisa de ajuda.

3.1.3 Abordagem Voltada à Avaliação e Comparação

Cardoso (2011) propõe uma classificação dos tipos de feedback para ambientes *online*. Em seu trabalho, o feedback é tratado sob os seguintes aspectos: o seu conteúdo, o seu direcionamento, o momento em que ele é fornecido e a fonte do feedback. Uma visão geral pode ser vista na Figura 3.3.

Figura 3.3. Categorização de *feedback* proposta por Cardoso.



Fonte: Adaptado de Cardoso (2011).

De acordo com Cardoso (2011), o conteúdo do feedback é dividido em duas categorias: tipos de informação e nível de complexidade de informação. Os tipos de informação são subdivididos em:

- **Reconhecimento**: fornece uma confirmação ao estudante de que um evento ocorreu; por exemplo, que sua atividade foi submetida com sucesso;
- **Motivacional/Interacional**: tipo de conteúdo relacionado às reações emocionais do estudante ao interagir com o ambiente *online*; por exemplo, uma mensagem incentivando o estudante a continuar tentando responder uma atividade;
- **Tecnológico**: este tipo de conteúdo está relacionado às informações sobre a utilização do ambiente *online* utilizado;
- **Informativo/Avaliativo**: fornece informações sobre a avaliação do aluno; por exemplo, se a solução do estudante para uma atividade está correta ou incorreta.

Quanto ao nível de complexidade do conteúdo, o mesmo pode ser dividido em conteúdos complexos e não complexos. Mensagens complexas são mais longas e complicadas para o entendimento do aluno, enquanto as não complexas são mais curtas e objetivas, tendendo a ser mais eficazes.

O direcionamento do feedback indica para quem a mensagem de feedback é exibida, a qual pode ser direcionada a um indivíduo ou a todos os estudantes de uma turma/curso. O direcionamento da mensagem pode estar vinculado ao tipo de conteúdo de feedback pretendido. É possível vincular, por exemplo, o feedback de conteúdo motivacional para ser direcionado a todo o grupo. Outro exemplo pode ser a relação de feedback avaliativo, sendo direcionado individualmente para cada aluno.

O momento do fornecimento do feedback pode ser distinguido entre imediato ou com atraso. O feedback imediato é síncrono, a partir do qual o estudante pode instantaneamente receber mensagens sobre a resolução de suas atividades ou de seu desempenho, ou até mesmo ajuda para solucionar uma questão. O feedback com atraso é assíncrono, a partir do qual o estudante pode receber a mensagem com minutos, horas ou dias após a realização de atividades no ambiente *online*. Diversos estudos buscam descobrir qual das duas opções é mais eficaz. Entretanto, considera-se que o tempo entre a submissão da atividade e a entrega do feedback não é o principal fator que influencia na eficácia do feedback, mas sim a natureza da tarefa e a capacidade do estudante.

No que se refere à fonte do feedback, Cardoso (2011) difere três delas em ambientes *online*: professor, computador e alunos. O professor, como fonte de feedback, elabora pessoal-

mente o conteúdo da mensagem. O computador, como fonte, refere-se à criação de feedback automático. O uso de outros alunos, como fonte, faz referência a situações de fórum ou de avaliação por pares.

3.1.4 Análise Comparativa entre as Abordagens de Especificação de *Feedback*

Com o intuito de posicionar o modelo conceitual proposto em relação às demais pesquisas apresentadas, realizou-se uma análise comparativa. Na Tabela 3.5 são apresentadas as pesquisas abordadas, além da pesquisa proposta.

Tabela 3.5. Comparação entre pesquisas sobre caracterização e modelagem de *feedback*.

Pesquisa	Voltada ao Professor	Voltada ao Desenvolvedor	Voltada à Avaliação e Comparação
Inventado et al. (2017)	Sim	Sim	Não
Narciss (2013)	Sim	Não	Sim
Cardoso (2011)	Não	Não	Sim
Proposta de Modelo Conceitual desta Pesquisa	**Sim**	**Sim**	**Sim**

Das pesquisas correlatas, o trabalho de Inventado et al. (2017) é o mais recente, entretanto possui algumas limitações. A forma de construção dos padrões foi baseada numa metodologia de mineração de dados em um ambiente de matemática, e os próprios autores informam que não há garantias de que os padrões descobertos podem ser generalizados para além de ambientes no contexto de matemática. Além disto, seu foco é guiar desenvolvedores na criação de feedbacks para esses ambientes, especificando como devem ser implementadas as estratégias de entrega de feedback e criação do conteúdo, não tratando de outros fatores relacionados ao feedback, como, por exemplo, seu caráter pedagógico.

Como apresentado, o modelo de Narciss (2007, 2013) tem o intuito de orientar a concepção e avaliação de estratégias de feedback. Em seu modelo, o papel pedagógico do feedback é destacado, porém não possui o comprometimento de fornecer auxílio no desenvolvimento de mecanismos de feedback. Seu foco maior é na determinação de aspectos do feedback vinculados às estratégias pedagógicas que o professor irá adotar.

Por outro lado, o modelo de Cardoso (2011) tem o enfoque nos requisitos funcionais que podem ser utilizados para classificar os tipos de feedback usados em ambientes *online*, não

tratando do mérito pedagógico (embora faça relações entre os requisitos do feedback e como este pode ser usado nas estratégias pedagógicas do professor).

A partir desta análise comparativa entre as pesquisas, reforça-se o diferencial da pesquisa realizada em comparação às outras do estado da arte. O arcabouço aqui proposto integra tanto a visão pedagógica quanto técnica que envolve o feedback, buscando um compromisso entre amplitude e especificidade, considerando diferentes fatores, e visando-se contribuir para analisar soluções apresentadas nos ambientes computacionais, além de fornecer subsídios aos que se encarregam de propor soluções de feedback para os ambientes interativos de aprendizagem.

3.2 Pesquisas Correlatas do Mecanismos de *Feedback*

Nesta seção, serão apresentadas as pesquisas correlatas cujo objetivo é fornecer feedback para os estudantes durante o processo de resolução de exercícios de programação. Foram excluídas as pesquisas cujo intuito era prover feedback após a submissão da solução. Na seção 3.2.1, são apresentadas as pesquisas correlatas, e na seção 3.2.2, é realizada uma análise comparativa entre o mecanismo de feedback proposto neste trabalho e os correlatos.

A habilidade de programação é algo que é aprendido na prática, porém uma alta carga de resolução de exercícios por parte dos alunos resulta na sobrecarga de trabalho para o professor (ALVES; JAQUES, 2014). A partir desse problema, surgiram os ambientes de apoio ao ensino de programação, a partir dos quais os estudantes podem solucionar exercícios e receber um feedback imediato para as suas soluções. Entretanto, a maioria desses sistemas oferece somente um feedback posterior à submissão da solução do estudante, não possuindo mecanismos que auxiliem o estudante na elaboração do algoritmo (KEUNING; JEURING; HEEREN, 2018).

O suporte durante o processo de construção da solução é importante, pois podem ocorrer cenários nos quais o estudante não consiga finalizar a elaboração de um exercício, ou não consiga sequer iniciá-la (KYRILOV; NOELLE, 2016). Isso pode ocorrer por diversos fatores, tais como: dificuldade na compreensão do enunciado do problema, desconhecimento das estruturas (*statements*) necessárias para a solução, dificuldade em representar, na forma de código, o seu pensamento, entre outros. Todos esses fatores atrapalham o processo de apren-

dizado do estudante. Partindo dessas constatações, surge o desafio de auxiliar o estudante durante o processo de construção da solução.

O uso de feedback para auxiliar o estudante durante a resolução de exercícios tem sido usado em diversos contextos. Em ambientes que trabalham assuntos no domínio da matemática existem sistemas que oferecem dicas, explicações e videoaulas para os estudantes superarem suas dificuldades e ajudá-los a concluir a elaboração das respostas (SEFFRIN et al., 2012; KYRILOV; NOELLE, 2016).

Ostrow e Heffernan (2014) em seu trabalho integraram um sistema de feedback a um ambiente de ensino de matemática chamado ASSISTments (HEFFERNAN; HEFFERNAN, 2014). Um dos objetivos foi analisar o uso de diferentes tipos de conteúdo de feedback (vídeos e texto) e seu impacto na aprendizagem dos estudantes. Esse trabalho pode ser aplicado também no domínio de programação.

Considerando que os estudantes apresentam diferentes perfis de codificação (BLIKSTEIN, 2011) e de estilos de aprendizagem, os sistemas de apoio ao ensino de programação podem recorrer a diferentes conteúdos de feedback para cada perfil de estudante, de modo que o conteúdo mais adequado seja escolhido para ser apresentado ao estudante durante o processo de resolução de exercícios de programação. Nesse sentido, Corbett e Anderson (2001) trabalharam com três conteúdos de feedback em seu sistema tutor para a linguagem de programação Lisp: mensagens informando a existência de erros, dicas e explicações (estas eram fornecidas com trechos dos códigos corretos para a solução). Outros trabalhos, como o de Singh, Gulwani e Solar-Lezama (2013), fornecem dicas como único conteúdo de feedback.

Em sua maioria, os sistemas de apoio ao ensino de programação oferecem feedback aos estudantes apenas após a submissão de uma solução completa (KEUNING; JEURING; HEEREN, 2018). Além da informação sobre se a solução está correta, o tipo mais comum de feedback consiste na apresentação dos testes para os quais o algoritmo do aluno informou a saída de forma correta ou incorreta (PAES et al., 2013). Outro feedback comum é a exibição das mensagens de erro de compilação e execução de código (ALVES; JAQUES, 2014).

Existem ainda trabalhos que possuem mecanismos de feedback não automatizados, a partir dos quais o estudante deve esperar pelo feedback que será produzido por um professor, monitor ou colega de turma (POLITZ; KRISHNAMURTHI; FISLER, 2014; HYYRYNEN

et al., 2010). Nesses casos, também é necessário que o estudante submeta uma solução completa para receber posteriormente algum feedback sobre a mesma.

3.2.1 Mecanismos de *Feedback* no Domínio de Programação

Diante do exposto na seção anterior, na revisão bibliográfica, foram considerados como correlatos os trabalhos que ofereciam feedback de forma automática durante a resolução de exercícios de programação. Algumas das características observadas em cada um dos trabalhos correlatos foi o conteúdo do feedback e sua forma de seleção.

O trabalho de Keuning, Heeren e Jeuring (2014) apresenta um protótipo de um sistema tutor para programação imperativa. O raciocinador de domínio usado no sistema provê facilidades para geração de feedback durante a resolução dos exercícios. Cada problema cadastrado no sistema é associado a uma ou mais soluções modelos, criadas especificamente com diferentes estratégias de resolução do problema. As soluções modelo são anotadas por um instrutor e mensagens específicas podem ser associadas a trechos de código. As principais contribuições desse trabalho são: i) suporte ao desenvolvimento passo a passo da solução; ii) geração de diagnóstico sobre o caminho da solução que está sendo desenvolvida; iii) feedback derivado usando a descrição dos exercícios e as soluções modelo. A principal desvantagem da pesquisa consiste na criação das soluções modelo e das suas anotações, cujo trabalho fica sob responsabilidade de um instrutor. O volume de trabalho manual pode ser considerado grande, a depender da quantidade de exercícios que serão inseridos no sistema. Além disso, fica a cargo também do instrutor pensar nas estratégias de solução para cada problema, o que pode ocasionar em situações nas quais o estudante surpreenda com estratégias não pensadas pelo instrutor.

Corbett e Anderson (2001) foram um dos primeiros trabalhos a analisar o uso de feedback em sistemas de ensino de programação. Eles utilizaram três tipos de conteúdo de feedback: indicações de erros, dicas e explicações. O feedback foi adicionado a um sistema para o ensino da linguagem de programação Lisp. Nesse trabalho foram analisadas diferentes configurações de feedback e ao final do trabalho Corbett e Anderson (2001) concluíram que a definição de estratégias pedagógicas deveria ser usada para determinar a configuração de feedback usada no sistema.

No trabalho de Hong (2004) foi desenvolvido um mecanismo de feedback para ser uti-

lizado em um sistema tutor para a linguagem Prolog. Segundo Hong (2004), classes de programas Prolog utilizam a mesma técnica de programação, constituindo um padrão comum de código (solução). A partir dessa constatação, foi criado um conjunto de regras para cada padrão. Tais regras são usadas para selecionar as dicas que serão apresentadas aos estudantes.

Singh, Gulwani e Solar-Lezama (2013) em seu trabalho apresentam um sistema de geração automática de feedback usado em um ambiente para programação em Python. O conteúdo do feedback tem como objetivo ajudar o estudante a corrigir erros contidos em seu algoritmo. O sistema foi testado com milhares de submissões de estudantes durante o decorrer de suas disciplinas de introdução à programação. Em 64% dos casos, o sistema conseguiu oferecer o feedback correto para a correção dos erros das soluções. O principal da pesquisa consiste no mecanismo de feedback capaz de ajudar o estudante somente ao serem identificados erros em suas soluções, ou seja, o sistema não pode auxiliar o estudante nos casos em que ele não saiba como começar a resolução ou tenha dúvidas durante a elaboração do algoritmo.

O trabalho de Rivers e Koedinger (2017) apresenta o ITAP (*Intelligent Teaching Assistant for Programming*). Esse sistema utiliza dados históricos de soluções de exercícios para gerar de forma automática dicas personalizadas para os estudantes. O sistema analisa o estado atual da solução do estudante e o caminho de construção da solução. Para realizar a seleção e apresentação das cias são necessários dois componentes: i) pelo menos uma solução modelo e ii) um mecanismo de teste automático de algoritmos. Esse trabalho também possui a limitação de auxiliar os estudantes somente ao identificar erros nas soluções.

3.2.2 Análise Comparativa entre os Mecanismos de *Feedback*

Na Tabela 3.6, é apresentada uma síntese dos trabalhos relacionados e uma comparação com o mecanismo de *feedback* apresentado nesta pesquisa. Entre os trabalhos apresentado na Tabela 3.6, Hong (2004) e Corbett e Anderson (2001) são os que mais se diferenciam do mecanismo criado nesta pesquisa, por abordarem soluções no escopo do paradigma funcional e no paradigma de programação em lógica. Entretanto, uma das semelhanças entre eles e esta pesquisa está no uso de mais de um tipo de *feedback*.

Keuning, Heeren e Jeuring (2014), Singh, Gulwani e Solar-Lezama (2013) e Rivers e

Koedinger (2017), assim como esta pesquisa, trabalham no escopo do paradigma funcional, entretanto possuem como limitação em comum o uso de um único tipo de *feedback*. Outra limitação é que nenhuma delas oferece características de personalização baseadas no estudante, seja ela manual (o estudante informando as preferências) ou automática (por meio de inferência).

Tabela 3.6. Síntese das Pesquisas Correlatas e comparação com o Mecanismos de *Feedback* Proposto.

Pesquisa	**Linguagem de Programação**	**Conteúdo do *Feedback***	**Seleção do *Feedback***
Keuning, Heeren e Jeuring (2014)	Linguagens imperativas	Dicas para construção da solução	Utiliza soluções modelo para identificar as estratégias de desenvolvimento adotadas pelo estudante. Seleciona o *feedback* a partir dessas informações.
Corbett e Anderson (2001)	Lisp	Indicação de erros, dicas e explicações	Utiliza soluções modelo para geração das dicas e estratégias pedagógicas para selecioná-las.
Hong (2004)	Prolog	Dicas e *templates* de soluções	Utiliza regras baseadas em padrões de programação em Prolog.
Singh, Gulwani e Solar-Lezama (2013)	Python	Dicas para correção de erros	Utiliza soluções modelo para gerar regras para correção dos erros.
Rivers e Koedinger (2017)	Python	Dicas para correção de erros	Utiliza soluções modelo para identificar o estado da solução e o caminho de elaboração. Seleciona o *feedback* a partir dessas informações.
Mecanismo Proposto	C/C++	Dicas para construção da solução, próximos passos, correção de erros e *templates* de soluções.	Utiliza soluções modelo para identificar o estado da solução e seleciona o *feedback* a partir dessa informação e das preferências do estudante.

O mecanismo de *feedback* criado nesta pesquisa será apresentado no Capítulo 5. Ele diferencia-se dos demais por oferecer mais de um conteúdo de *feedback* e diversas formas de apresentação. O mecanismo fornece dicas para o estudante iniciar a solução, dicas sobre os próximos passos a serem realizados a partir do estado atual da sua solução, dicas para

correção de erros e *templates* de solução em formato de fluxograma. Além disso, tais conteúdos possuem mais de uma forma de apresentação, texto, vídeo e imagem, de forma que os estudantes podem selecionar a forma de apresentação de acordo com suas preferências e seu perfil. Esta característica é considerada importante dado que estudantes distintos podem ter diferentes preferências na forma como o *feedback* é apresentado.

Capítulo 4

Um Modelo Conceitual para *Feedback* Pedagógico

Para realização da criação de um mecanismo de *feedback*, primeiramente foi necessário criar uma especificação. No Capítulo 3 (Pesquisas Correlatas), foram apresentados modelos de *feedback* existentes na literatura, sendo também realizada uma análise comparativa no que diz respeito à utilização de cada um deles.

Em função das restrições discutidas no Capítulo 3 sobre os modelos existentes na literatura, foi realizada a criação de um Modelo Conceitual para especificação de mecanismos de *feedback* em ambientes educacionais computadorizados.

Neste capítulo, será apresentado o Modelo Conceitual para especificação de *Feedback* Pedagógico, criado para ser utilizado na especificação do mecanismo de *feedback* proposto nesta pesquisa. Na seção 4.1 será apresentada a metodologia de desenvolvimento do Modelo. A seção 4.2 contém a apresentação do Modelo Conceitual e a descrição dos seus principais componentes. Na seção 4.3, é realizada uma discussão sobre a forma de utilização do Modelo com a apresentação de um mapa conceitual.

4.1 Metodologia de Criação do Modelo Conceitual

Em função das restrições discutidas no Capítulo 3 sobre as modelagens de *feedback* educacional existentes na literatura, foi realizada a criação de um Modelo Conceitual para especificação de mecanismos de *feedback* em ambientes educacionais computadorizados. Este

modelo foi utilizado para especificar o mecanismo de *feedback* proposto nesta pesquisa, cujo intuito é auxiliar alunos na resolução de exercícios de programação.

O processo de criação do Modelo Conceitual foi divido em três etapas:

- Revisão da literatura sobre a modelagem de *feedback* educacional;
- Análise de ambientes de programação que possuíam mecanismos de *feedback*;
- Criação de uma versão inicial do Modelo para discussão e refinamento.

A primeira etapa a ser realizada nesta pesquisa foi a busca na literatura por propostas de modelagens de *feedback*. Como apresentado no Capítulo 3, foram encontrados três trabalhos diretamente correlatos, entretanto percebeu-se a complexidade de sua utilização para serem aplicados para a especificação do mecanismo de *feedback* pretendida na pesquisa. O modelo de Inventado et al. (2017) é focado em matemática e limitado a esse domínio. O modelo de Cardoso (2011) tem o intuito apenas de comparação, portanto não se compromete com questões tecnológicas necessárias ao se especificar um mecanismo de *feedback* para ambientes educacionais computadorizados. Por fim, o modelo de Narciss (2013) seria o mais propenso a ser utilizado, porém, apesar de possuir um rico detalhamento sobre o conteúdo do *feedback*, as demais características como função e apresentação são abstratas e pouco especificadas. A partir destas constatações, foi proposta a criação de um Modelo Conceitual para especificação do *feedback*. Estes três modelos foram tomados como ponto de partida para a identificação dos principais componentes do *feedback*: dimensão, direcionamento da mensagem, momento do provimento, fonte provedora, conteúdo e apresentação.

Em sequência foi realizada a segunda etapa, composta pela análise de ambientes de programação que possuíam mecanismos de *feedback*. Essa análise foi realizada por meio de uma revisão da literatura e pela utilização de alguns ambientes disponíveis de forma gratuita para teste ou utilização na Web. Detalhes sobre esta análise focada em ambientes brasileiros pode ser acessada em Silva et al. (2015). Após esta análise, foi realizado um processo de refinamento do Modelo Conceitual, que, em sua terceira etapa de construção, foi apresentado na Jornada de Atualização em Informática na Educação (JAIE) por meio de um minicurso e publicação nos anais da Jornada como capítulo de livro(COSTA et al., 2016).

Após a apresentação no JAIE, o Modelo Conceitual passou por modificações que resultaram na criação de um sétimo componente de personalização. Ressalta-se aqui que o

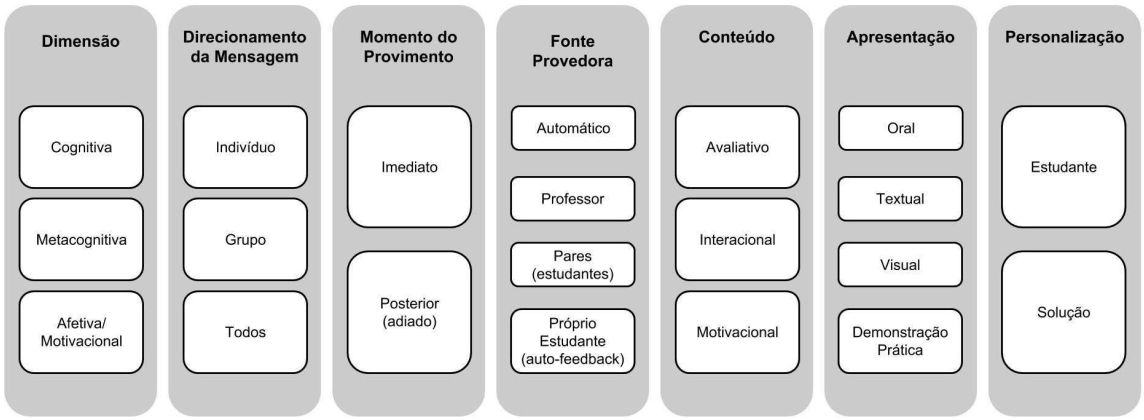
Modelo não tem a intenção de abranger a totalidade de fatores que compõem o *feedback*; sua versão final apresentada nesta pesquisa pode ainda sofrer futuras modificações por meio da execução de trabalhos futuros apresentados no Capítulo 6.

O processo de criação do Modelo foi realizado buscando um compromisso com a amplitude que o conceito de *feedback* possui e com as especificidades geradas pelos ambientes mais recentes concebidos com ideias de personalização e adaptação do *feedback*.

4.2 Modelo Conceital para *Feedback*

Na Figura 4.1, é apresentado o Modelo Conceitual elaborado na pesquisa, formado por seis componentes básicos, sendo estes: dimensão, direcionamento da mensagem, momento do provimento, fonte provedora, conteúdo e apresentação. Cada um destes componentes possui desdobramentos que, ao serem selecionados e combinados, compõem a especificação de um *feedback*. A seguir, é apresentado o detalhamento de cada um dos componentes do Modelo.

Figura 4.1. Modelo Conceitual para *Feedback* Pedagógico.



Fonte: autoria própria.

4.2.1 Dimensão do *Feedback*

A dimensão do *feedback* corresponde ao objetivo pedagógico da mensagem enviada ao aluno. Neste contexto, uma mensagem pode possuir dimensões cognitivas, metacognitivas e afetivas/motivacionais.

A **dimensão cognitiva** diz respeito à construção do conhecimento do aluno de forma crítica e reflexiva. Uma mensagem de *feedback* com dimensão cognitiva pode ter como objetivo

ajudar o aluno na assimilação do conteúdo, no processo de entendimento e na elaboração da solução para uma atividade, reforçar o conhecimento do aluno, corrigir erros e verificar a corretude das soluções do aluno.

O objetivo de mensagens de *feedback* com **dimensão metacognitiva** é fazer com que o aluno reflita sobre o seu próprio conhecimento. Mensagens de *feedback* com dimensão metacognitivas ajudam o aluno a compreender melhor como ele pode utilizar seus conhecimentos no entendimento e resolução de atividades, em determinar quais as melhores práticas de aprendizagem para ele e como aprender a partir de seus erros e acertos. A metacognição pode ser compreendida como o "pensar sobre o pensamento" (FLAVEL, 1979). Em um contexto educacional, ela se refere ao conhecimento que o aluno possui sobre o seu próprio conhecimento, ou seja, sobre sua cognição e atividade de aprendizagem (MARINI, 2006).

O *feedback* com **dimensão afetiva/motivacional** possui mensagens com estímulos aos alunos, com o objetivo de mantê-los interessados, ativos e confiantes. É importante frisar que uma mensagem de *feedback* pode englobar todas as dimensões aqui descritas, não estando restrita a uma única dimensão.

4.2.2 Direcionamento da Mensagem

O componente de direcionamento da mensagem refere-se ao destinatário do *feedback*. Em um ambiente de sala de aula tradicional, ao apresentar um problema para ser resolvido pelos alunos em sala de aula, o professor pode ser chamado por um aluno para esclarecer alguma dúvida sobre o problema, neste caso o *feedback* é individualizado para o aluno. Caso o professor resolva pedir a atenção da turma para explicar melhor o problema, estará dando um *feedback* para toda a turma e, em casos em que os estudantes estejam trabalhando em pares ou grupos para encontrar uma solução, o professor pode acompanhar o progresso e fornecer *feedback*s diferenciados para cada grupo. Neste cenário estão apresentados os três subcomponentes do direcionamento da mensagem: individual, todos e grupo.

Transpondo o cenário anterior para um ambiente de aprendizagem *online*, técnicas de personalização podem fazer uso da modelagem do usuário para prover *feedback* **individualizado** para cada um dos estudantes. Outro benefício do uso da tecnologia é a possibilidade de agrupamentos por meio de técnicas de mineração de dados, sendo possível diferenciar o *feedback* para **grupos** de estudantes com características distintas. Por fim, pode-se também

oferecer um *feedback* único para **todos** os estudantes que utilizam o ambiente sem considerar aspectos individuais.

4.2.3 Momento do Provimento

O componente de momento do provimento diz respeito a quando fornecer o *feedback*. Sendo assim, pode-se classificá-lo como síncrono ou assíncrono, ou em outras palavras, imediato ou adiado. Segue-se aqui a caracterização intuitiva dada por Shute (2008), que diz: o *feedback* **imediato** é aquele que o estudante recebe logo após responder uma questão ou realizar uma tarefa ou uma ação. O *feedback* **adiado** é aquele que o estudante recebe algum tempo após a realização de uma tarefa. Aqui vale uma ressalva particular para a situação específica na qual é necessário considerar a emissão de *feedback*s para a tarefa que tem sua execução em andamento; esse tipo de atividade pode fazer uso de forma mais eficiente do *feedback* imediato, pois pode oferecer mensagens para auxiliar o aluno durante o processo de elaboração da solução da atividade.

O *feedback* imediato pode ainda ser dividido em dois tipos: requisitado e automático. No primeiro caso, o estudante explicitamente solicita o *feedback* e, no segundo caso, o instante em que o *feedback* é fornecido é controlado inteiramente pelo sistema (ou pelo professor). Por exemplo, no Portugol Studio (NOSCHANG et al., 2014), IDE para programação em Portugol, caso o estudante tente utilizar uma variável sem inicializá-la, a IDE irá sublinhá-la em vermelho e exibir uma mensagem de que a variável não foi inicializada (fornecendo também dicas de como inicializar). Neste caso, o aluno recebe um *feedback* imediato sem ter feito explicitamente nenhum pedido de verificação de código ou ajuda.

4.2.4 Fonte Provedora

Quanto ao componente denominado de fonte provedora, distingue-se aqui o professor, o sistema computacional (*feedback* automático), os estudantes (pares), ou ainda o *autofeedback*.

O **professor** é uma das principais fontes de *feedback* no contexto educacional, em especial na educação *online*. É importante destacar que o professor é colocado aqui como um especialista do domínio, na categoria de professor. Podem se enquadrar aqui outras possibilidades oriundas de diferentes modalidades de ensino, como, por exemplo, o tutor

da educação a distância ou um monitor da educação presencial. Como fonte provedora de *feedback*, o professor pode elaborá-lo mobilizando todos os demais componentes do Modelo apresentado, em um ambiente de educação *online*, por exemplo. Por meio de ferramentas de autoria, o professor pode inserir o conteúdo da mensagem de *feedback* que será exibida, determinar a quem direcioná-la (por exemplo, alunos que acertaram o exercício recebem um *feedback* diferenciado dos que erraram), o momento que a mensagem deve ser enviada, a forma como deve ser exibida, além de, particularmente, poder expressar aspectos específicos sobre as dimensões cognitivas, metacognitivas e afetivas/motivacionais.

O *feedback* provido por **estudantes (pares)** é uma atividade social, em que se espera a contribuição dos pares para melhorar o processo de aprendizagem dos envolvidos, provendo tanto benefícios cognitivos, quanto afetivos e sociais. Em um contexto presencial, esta é uma fonte de *feedback* bastante utilizada pelos alunos, ao procurarem ajuda de colegas de turma para auxiliar na resolução de exercícios ou para esclarecer dúvidas. Em um contexto *online*, este é um recurso utilizado muitas vezes para complementar o *feedback* dado pelo professor, ou para suprir a falta de recursos automáticos do sistema. Neste contexto, os alunos podem, por exemplo, fornecer *feedback* em resposta a uma mensagem de um determinado estudante, ou ainda, por meio de avaliação por pares (*peer review*), os alunos podem se avaliar mutuamente.

O *feedback* **automático** é aquele caracterizado por sua geração e emissão ser realizada por um sistema computacional. Neste sentido, a qualidade do conteúdo da mensagem de *feedback* é diretamente dependente da técnica utilizada para sua geração automática. Por exemplo, este tipo de *feedback* é mais simples para um sistema que trabalha com questões bem definidas, como questões objetivas, na qual o sistema precisa apenas emitir uma mensagem com indicações de "correta" ou "incorreta". Tal mensagem pode ainda ser incrementada com justificativas e um parecer sobre os erros. Também se enquadram neste aspecto questões de resolução aberta em domínios com regras bem definidas, como expressões aritméticas, por possuir regras de resolução claras e específicas. Este é um domínio no qual o sistema pode oferecer automaticamente *feedback*s mais ricos, contendo até mesmo informações sobre a qualidade das soluções do aluno. Entretanto, para domínios que possuem problemas abertos, a geração automática de *feedback* é um pouco mais complexa, principalmente devido à imprevisibilidade de respostas. Quanto a isto, a principal alternativa utilizada tem sido a

geração de *feedback* baseada em técnicas de inteligência artificial, particularmente, de processamento de linguagem natural, representação semântica de conhecimento e mineração de dados (RIVERS; KOEDINGER, 2017), permitindo um bom nível de análise de soluções discursivas.

O *feedback* do **próprio estudante (*autofeedback*)** está relacionado ao que Narciss (2013) denomina de *feedback* interno, ou seja, são informações perceptíveis pelo próprio estudante quanto ao seu desempenho e conhecimento; por exemplo, a confiança do estudante em sua própria resposta ou a percepção de que ele está seguindo os passos corretos para chegar a solução correta. O próprio estudante, como fonte provedora de *feedback*, está fortemente relacionado a atividades da dimensão de metacognição. Uma forma de ambientes *online* aproveitarem esse tipo de *feedback* consiste em solicitar que o estudante faça autoavaliações, ou até mesmo reveja soluções antigas para análise e comentários.

4.2.5 Conteúdo do *Feedback*

No que diz respeito ao conteúdo do *feedback*, este possui três subcomponentes: avaliativo, interacional e motivacional. As mensagens de *feedback* com **conteúdo avaliativo** contêm informações sobre a avaliação do desempenho do estudante, seja essa feita pelo professor ou pelo sistema. Tais mensagens geralmente possuem informações sobre o resultado da correção de questões individuais ou de listas de exercícios, podendo conter informações sobre notas ou qualidade das respostas.

O **conteúdo interacional** se refere às mensagens cujo objetivo é fornecer conselhos práticos e específicos sobre como o estudante deve proceder em seu processo de aprendizagem, podendo incluir mensagens de ajuda para indicar quais passos o estudante deve realizar para concluir uma atividade ou como evitar erros cometidos anteriormente (DENNIS; MASTHOFF; MELLISH, 2016). Um *feedback* interacional tem como principal função auxiliar o estudante, podendo até mesmo fazer recomendações de materiais de estudos.

O **conteúdo motivacional** visa oferecer um suporte emocional ao estudante durante ou após a resolução de atividades. Este é um tipo de conteúdo de *feedback* que está bastante presente no ensino tradicional, em situações como, por exemplo, o professor incentiva seus alunos a tentar solucionar os exercícios em vez de esperar que ele mesmo forneça a resolução correta. Em sistemas computacionais, este componente pode estar vinculado a elementos de

gamificação.

É importante destacar que uma mesma mensagem de *feedback* pode conter todos os três subcomponentes, não havendo aqui relação de exclusividade entre mensagem e característica do conteúdo. Inclusive, recomenda-se que além de informações binárias (certo ou errado) inclua-se também no *feedback* mensagens com justificativas para avaliação binária, ou ainda, produzir um diálogo com o estudante, e lhe pedir uma justificativa de sua solução. No caso particular de *feedback* para resposta certa, pode-se ainda fornecer informações adicionais, dando conta, por exemplo, de aspectos de qualidade da resposta. Para respostas incorretas, convém levar em conta aspectos motivacionais, recorrendo-se a mensagens com conteúdo encorajador, tais como, "Não desista, tente novamente!". No mais, o *feedback* pode ainda sugerir ao estudante algum recurso pedagógico com propósito de reforço ou complementação.

4.2.6 Apresentação do *Feedback*

Quanto a componentes de apresentação da mensagem de *feedback*, destacam-se quatro formas de exibição da mensagem: textual, visual, oral e demonstração prática. **Apresentações textuais** são compostas essencialmente por texto, como dicas e explicações, e o texto pode ainda possuir elementos de interação, como palavras destacadas ou *hiperlinks*. No domínio de programação, por exemplo, a exibição de um código fonte se enquadra na categoria de apresentação discursiva.

A **apresentação visual** contém elementos como vídeos, imagens e animações. Seu objetivo é exibir ao aluno diferentes elementos para reflexão. Por exemplo, no domínio de operações de frações, a exibição do equivalente às frações numéricas em formato de fatias de pizza é um tipo de apresentação visual. A **apresentação oral** tem o objetivo de prover um *feedback* no formato de áudio, esse tipo de apresentação é comumente utilizado em sistemas educacionais de línguas estrangeiras.

Por fim, a **demonstração prática** é um tipo de apresentação formada por animações ou jogos com os quais o estudante pode interagir.

4.2.7 Personalização do *Feedback*

O componente de personalização diz respeito à possibilidade de prover um *feedback* individualizado para os estudantes. Este componente pode ser dividido em personalização baseada na solução e personalização baseada no estudante. Quando **baseada na solução**, a personalização utiliza o estado atual da solução do estudante para prover uma mensagem de *feedback* que o auxilie a corrigir erros ou a melhorar de acordo com as características da solução. Por exemplo, em exercícios de programação existem várias soluções possíveis, porém igualmente corretas para um mesmo exercício. No entanto, na maioria dos casos, podem ser detectados padrões de soluções. Utilizando esses padrões, podem ser geradas mensagens de *feedback* personalizadas baseada na detecção do padrão da solução do estudante.

Já a personalização **baseada no estudante** utiliza características individuais do estudante para prover mensagem de *feedback*, essas características podem ser: informações pessoais (como, por exemplo, sexo e idade), preferências (que podem ser declaradas pelo estudante ao se cadastrar no sistema ou inferidas de forma automática), personalidade [traços de personalidade do estudante (AGUIAR; FECHINE; COSTA, 2015)], estilo de aprendizagem [(AGUIAR et al., 2017)], nível de conhecimento (estado atual do conhecimento do estudante ou seu histórico de evolução) e histórico de interações com sistema educacional (análise do comportamento do estudante na resolução de exercícios no sistema, por exemplo, o tempo de resolução ou quantidade de *feedback* solicitado).

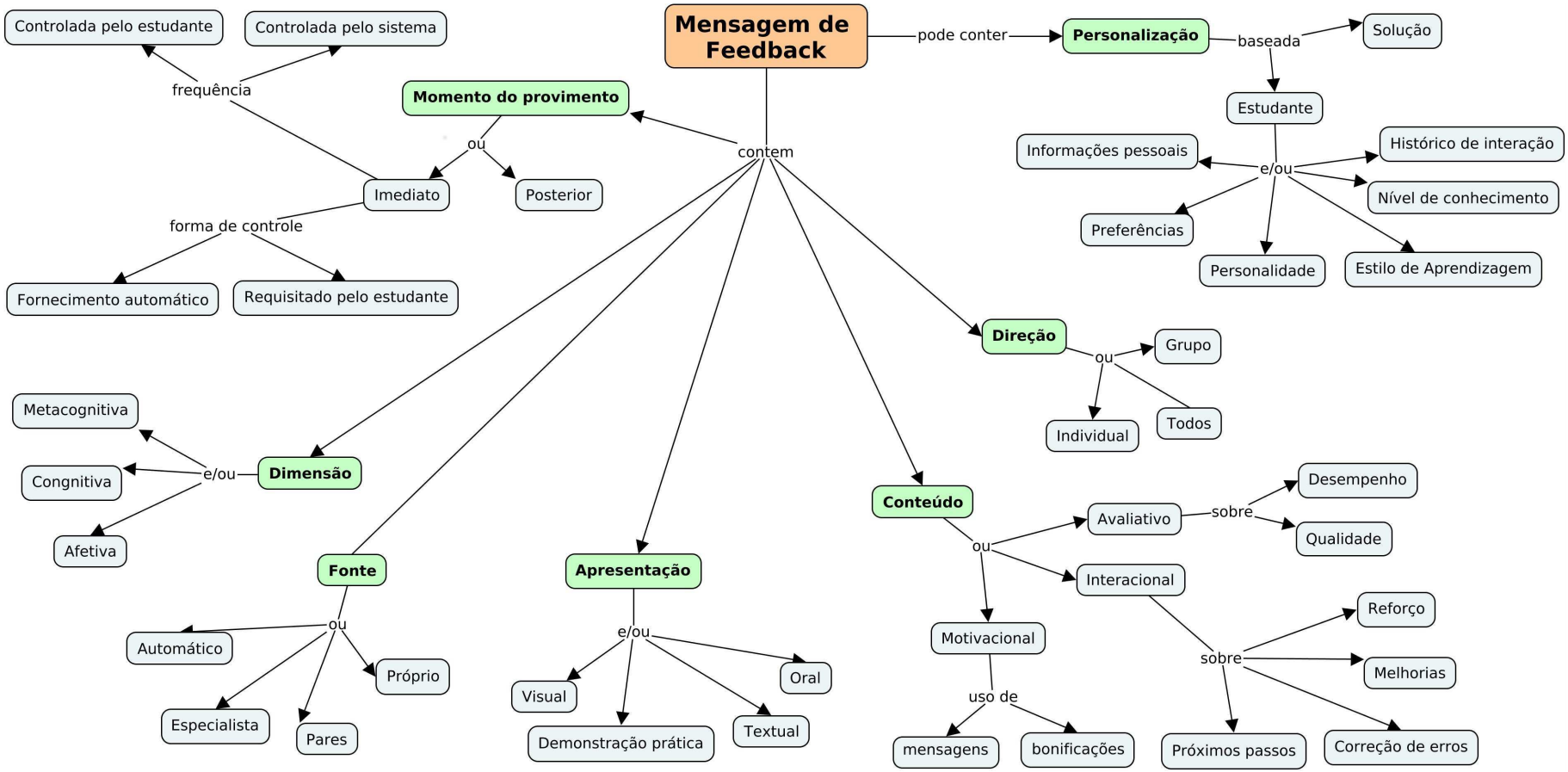
4.3 Mapa Conceitual para Utilização do Modelo

Para facilitar a utilização do modelo proposto neste capítulo, esta seção apresenta um mapa conceitual que pode ser utilizado na orientação da especificação de novos mecanismos de *feedback* e na análise de mecanismos já existentes.

O mapa é apresentado na Figura 4.2. Os componentes do *feedback* são apresentados em verde e seus subcomponentes em azul. Uma mensagem de *feedback* deve conter obrigatoriamente os componentes momento do provimento, dimensão, fonte, apresentação, conteúdo e direção. O componente de personalização é o único opcional, uma vez que nem todos os sistemas possuem necessidade ou capacidade de prover mensagens personalizadas.

Os componentes momento do provimento, fonte, conteúdo e direção possuem seus sub-

Figura 4.2. Mapa Conceitual para Especificação da Mensagem de *Feedback*.



Fonte: autoria própria.

componentes ligados por um ou exclusivo, ou seja, uma mensagem de *feedback* possui apenas um dos seus subcomponentes. Por exemplo, ao enviar uma mensagem de *feedback* ao estudante, o momento do provimento obrigatoriamente tem que ser imediato ou posterior, e não os dois ao mesmo tempo. Isso não impossibilita que, para um mesmo exercício, o estudante receba uma mensagem de *feedback* imediata e outra algum tempo após a submissão da solução, entretanto, neste caso, são duas mensagens de *feedback* distintas, ou seja, possuem especificações distintas.

Os componentes dimensão e apresentação possuem seus subcomponentes conectados por um ou inclusivo (e/ou), indicando que uma mensagem de *feedback* pode possuir mais de um dos subcomponentes, ou até mesmo todos.

4.4 Considerações Finais do Capítulo

Este capítulo teve por objetivo apresentar a proposição do Modelo para a especificação de *feedback* pedagógico. Na Figura 4.1 está sintetizada a proposta de Modelo, contendo os componentes: dimensão, direcionamento da mensagem, momento do provimento, fonte provedora, conteúdo, apresentação e personalização.

O Modelo tem o intuito de ser utilizado no processo de especificação de requisitos para o desenvolvimento de mecanismos de *feedback*. Um mapa conceitual foi apresentação na seção 4.3 para ajudar na utilização do Modelo com este propósito.

Em síntese, convém ressaltar que há uma variedade de possibilidades de mensagens de *feedback*, que podem englobar diversos componentes existentes no Modelo. Por exemplo, após a solução correta de uma lista de exercícios, o estudante pode receber um *feedback* adiado, preparado pelo professor, com conteúdo motivacional e que seja apresentado utilizando elementos textuais e de demonstração prática.

No próximo capítulo, será apresentada a criação do mecanismo de *feedback* proposto nesta pesquisa. O mecanismo foi desenvolvido para o domínio de programação, tendo como objetivo fornecer *feedback* para estudantes durante o processo de resolução de problemas de programação.

Capítulo 5

Mecanismo de *Feedback* no Domínio de Programação

Inicialmente, neste capítulo, é apresentada a metodologia aplicada no planejamento e criação do mecanismo de *feedback* no domínio de programação. Em seguida, a especificação do mecanismo é descrita utilizando o modelo conceitual visto no Capítulo 4. Posteriormente, é realizada uma descrição do modo de funcionamento do mecanismo junto com a apresentação de cenários de utilização do mesmo. Na sequência, a seção 5.4 apresenta dois estudos preliminares realizados para validação do mecanismo de *feedback*. O mecanismo foi avaliado por meio da aplicação de um quase-experimento, cujo planejamento, resultados e discussões são apresentados na seção 5.5.

5.1 Metodologia de Criação do Mecanismo de *Feedback*

A principal contribuição deste trabalho é a criação de um mecanismo de *feedback* para auxiliar estudantes na resolução de exercícios de programação. A primeira foi a criação da definição do domínio no qual o mecanismo seria desenvolvido. Em sequência, foi determinado o público-alvo do mecanismo. Com a definição do domínio e do público-alvo, foi necessário determinar para qual linguagem de programação o mecanismo seria criado. Embora ele tenha sido desenvolvido com a possibilidade de expansão para outras linguagens de programação estruturadas, inicialmente foi desenvolvido para trabalhar com exercícios na linguagem de programação C/C++. A próxima etapa realizada foi a seleção do ambiente de

programação no qual o mecanismo seria integrado. Nas próximas seções serão descritas e justificadas as escolhas realizadas em cada uma dessas etapas.

5.1.1 Definição do Domínio

Podem ser encontrados diversos trabalhos na literatura que mostram que estudantes iniciantes em programação apresentam uma série de dificuldades em seu aprendizado. A técnica mais utilizada por professores para que os alunos consigam assimilar melhor o conteúdo e aprender a programar é a prática constante de exercícios de programação, conhecida como *learning by doing* (ANZAI; SIMON, 1979b). Os exercícios de programação consistem na criação de soluções algorítmicas para problemas propostos. Entretanto, os estudantes, além de praticar a criação de algoritmos, também precisam receber um retorno do professor sobre suas soluções, o que em geral resulta na sobrecarga de trabalho do professor, em função também do número de alunos em suas turmas. Para resolver este problema, surgiram as ferramentas computadorizadas de apoio ao ensino de programação, como, por exemplo, IDEs específicas para serem usadas por iniciantes (NOSCHANG et al., 2014), uso de juízes *online* (PETIT; GIMÉNEZ; ROURA, 2012; ALA-MUTKA, 2005b), micromundos (XINOGALOS; SATRATZEMI; DAGDILELIS, 2006) e sistemas tutores inteligentes (RIVERS; KOEDINGER, 2017). Em geral, essas ferramentas possuem um público-alvo específico, são vinculadas a linguagens ou paradigmas de programação específicos e possuem seus próprios mecanismos de provimento de *feedback* para os estudantes.

O domínio de programação foi escolhido para a criação do mecanismo de *feedback* desta pesquisa devido às dificuldades, já citadas anteriormente, dos estudantes no aprendizado de programação e na necessidade do professor de fornecer *feedback* aos alunos. Sendo assim, o modelo conceitual apresentado no Capítulo 4 foi utilizado para especificar um mecanismo de *feedback* para o domínio de programação. Por meio desta especificação, foi desenvolvido o mecanismo cuja função é auxiliar os estudantes durante o processo de resolução de exercícios de programação.

5.1.2 Público-Alvo

O aprendizado de programação exige o domínio de conceitos e habilidades que, no geral, não são abordados durante o período escolar (pré-universitário) dos alunos. Embora existam iniciativas para o ensino de programação ainda no ensino médio (SANTOS et al., 2017; SILVA, 2017; PANTALEÃO; AMARAL; SILVA, 2017), na maioria dos casos isto acontece apenas em instituições de ensino que integram o ensino médio com um curso técnico profissionalizante na área de informática ou em projetos de extensão universitários.

Assim como nos cursos de graduação, também são observados nos cursos técnicos de nível médio altos índices de reprovação e evasão oriundos das disciplinas de introdução à programação (AMORIM et al., 2016; SOBRINHA et al., 2016). Uma consequência negativa desses índices é a desmotivação dos alunos, que poderiam no futuro seguir na área de computação, porém desistem ainda jovens no primeiro contato com a área devido às suas dificuldades. Sendo assim, o mecanismo de *feedback* foi criado tendo em vista que o seu público-alvo são alunos de nível médio em seu primeiro contato com programação, em especial os alunos de cursos técnicos de nível médio.

5.1.3 Linguagem de Programação

Diversas abordagens podem ser aplicadas para ensinar uma pessoa que nunca teve contato com programação, como, por exemplo, ensinar a lógica de programação por meio de jogos como Scratch (MALONEY et al., 2010), usar fluxogramas, linguagens de pseudocódigo ou linguagens de programação reais. Em cursos técnicos de informática, os professores costumam optar por utilizar linguagens de programação utilizadas no mercado, tais como Python, C/C++ ou Java. O mecanismo de *feedback* proposto nesta pesquisa foi criado e validado para prover mensagens de ajuda durante a resolução de exercícios da linguagem C/C++, porém, pela forma como ele foi desenvolvido, é possível realizar extensões para dar suporte a outras linguagens de programação.

5.1.4 Ambiente de Programação

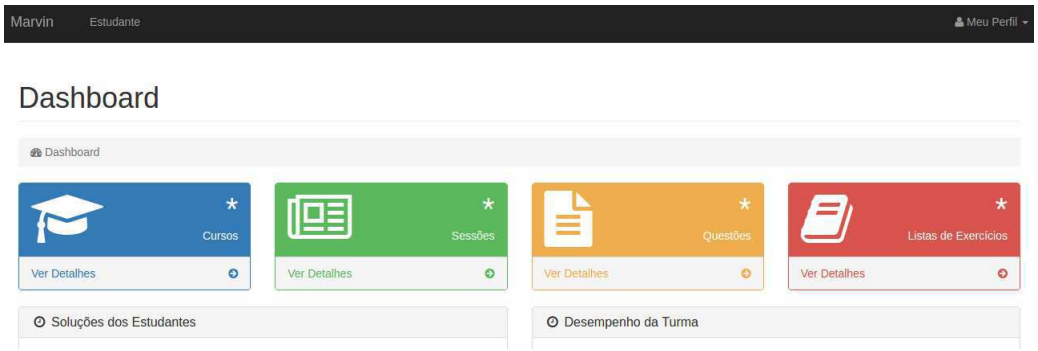
Ao trabalhar com o ensino de programação para iniciantes, podem ser utilizados dois tipos de ambientes; *online* ou *desktop*. Nesta pesquisa, optou-se por trabalhar com ambientes *online*,

dado que ao se trabalhar com ferramentas *desktop*, sempre que novas funcionalidades são adicionadas os estudantes precisam baixar e instalar uma nova versão. Além disto, a coleta de dados se torna mais difícil, o que complica sua análise. Ao trabalhar com ambientes *online*, ainda existe a vantagem de, caso o aluno não possua um computador pessoal, poder interagir com o ambiente por meio de dispositivos móveis. Em função disto, o mecanismo de *feedback* criado foi integrado a um sistema educacional computadorizado disponível *online* para o ensino de programação.

5.1.5 Marvin: Sistema de Aprendizagem de Programação

Marvin é um ambiente *online* para aprendizagem de programação voltado para o ensino de programação estruturada para iniciantes. O sistema possui funcionalidades disponíveis para professores e estudantes. O professor pode criar e gerenciar turmas, como também cadastrar questões de programação para serem solucionadas pelo estudante. Na Figura 5.1 é apresentada a página principal exibida ao professor após *login* no sistema.

Figura 5.1. Página Principal do Sistema Marvin para o Professor.



Fonte: autoria própria.

Os estudantes podem se cadastrar no Marvin, mas somente terão acesso ao login se algum professor os vincularem a uma turma existente. O estudante pode visualizar a lista de questões criada pelo professor e submeter soluções ao sistema. Na Figura 5.2 é apresentada a página principal do Marvin exibida ao estudante após entrar no sistema.

A avaliação das soluções submetidas pelo sistema é realizada de forma manual. Quando um estudante submete uma solução, ela é encaminhada para o professor, que, via sistema, realiza a avaliação informando se a solução está correta ou incorreta. O estudante pode

visualizar o seu histórico de soluções já avaliadas e as pendentes de avaliação.

Figura 5.2. Página Principal do Sistema Marvin para o Estudante.

Fonte: autoria própria.

O mecanismo de *feedback* criado nesta pesquisa foi desenvolvido em Java e integrado ao sistema Marvin. Com isso muitas funcionalidades novas foram adicionadas. Na primeira, o professor pode criar mensagens de *feedback* para as questões de programação para ajudar os estudantes na elaboração de suas soluções (detalhes na seção 5.3.2). Na segunda, voltada para o estudante, ele pode requisitar ajuda ao sistema para questões que possuem mensagens de *feedback* disponíveis (detalhes na seção 5.3.4).

5.2 Especificação do Mecanismo de *Feedback* Para o Domínio de Programação

Para a criação do mecanismo de *feedback* proposto nesta pesquisa, inicialmente foi realizada a especificação com base no Modelo Conceitual descrito no Capítulo 4. Nesta seção, são descritas as decisões relacionadas a esta especificação, bem como sua exposição.

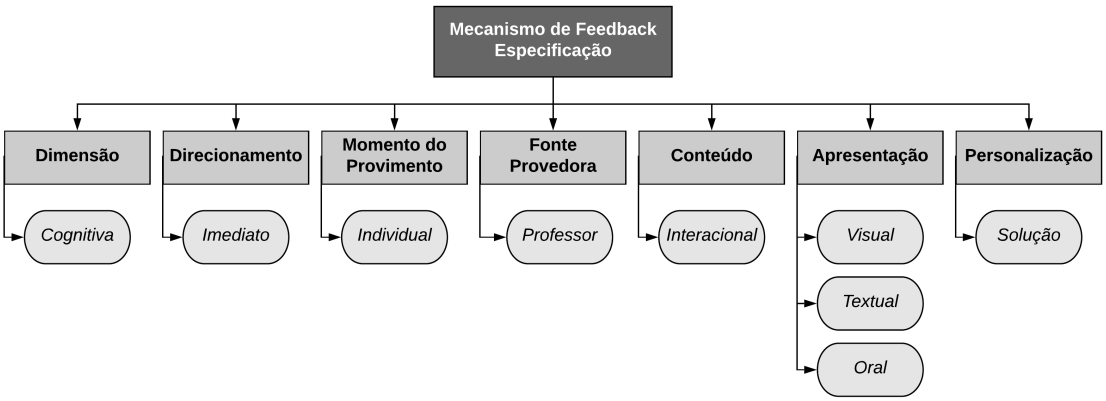
A maioria dos ambientes *online* destinados à prática de programação é baseada em juízes *online* usados em competições de programação (CAMPOS; E., 2004). Nesses ambientes, o aluno submete a solução para um problema e, em seguida, o sistema executa um conjunto de testes e compara a saída da solução do aluno com a saída esperada para o conjunto de testes. Se todas as saídas da solução forem iguais às saídas esperadas, a solução é considerada correta, caso contrário o sistema retorna ao aluno uma mensagem de solução incorreta.

Observando esse cenário, fica evidente que o único *feedback* que o aluno receberá desse tipo de sistema é uma mensagem binária, do tipo "correta" ou "incorreta", para a sua solução. Além disto, o aluno receberá esta informação somente se conseguir concluir seu raciocínio, finalizar a solução e submetê-la no sistema. Dada esta condição, muitas alunos ficam sem assistência, pois podem não conseguir sequer concluir uma solução minimamente correta para submeter. Isto pode ocorrer caso eles não compreendam o enunciado do problema, ou iniciem a solução, mas não saibam como fazê-la funcionar para todas as situações pedidas no problema ou tenham dúvidas sobre quais estruturas utilizar. Nestes casos, o estudante tende a desistir da resolução do problema, e se isto acontecer com frequência, o estudante pode se sentir desmotivado a continuar.

Com o intuito de auxiliar os alunos nesta situação em particular, foi definido para o mecanismo ter como objetivo fornecer *feedback* para o aluno durante a construção da solução para o problema, a fim de que ele consiga finalizar uma solução minimamente correta antes que a mesma seja submetida para apreciação do sistema. A solução submetida é avaliada manualmente pelo professor. Não faz parte do escopo da pesquisa fornecer *feedback* automático após as submissões.

A representação gráfica da especificação do mecanismo pode ser vista na Figura 5.3.

Figura 5.3. Especificação do mecanismo de *feedback* para domínio de programação.



Fonte: autoria própria.

O *feedback* trabalhado possui **dimensão** cognitiva, dado que é composto por mensagens com o objetivo de fazer o aluno refletir e melhorar seu entendimento sobre programação por meio de dicas para a resolução de exercícios de codificação.

O **direcionamento** da mensagem de *feedback* é individual. O **momento** do provimento

é imediato, com o estudante controlando as requisições de *feedback* e a frequência de recebimento, ou seja, será responsabilidade do estudante pedir ajuda ao sistema (por meio de botões na interface gráfica do sistema), sendo que o mesmo também terá controle sobre a frequência de recebimento, com isso ele poderá pedir ajuda sempre que desejar sem que o sistema controle a quantidade ou o intervalo de tempo entre os pedidos.

A **fonte** provedora é o professor, que deverá criar, para cada exercício, um conjunto de recursos de suporte (mensagens de *feedback*) que possam auxiliar o estudante durante a resolução do exercício.

O **conteúdo** do *feedback* é interacional, dado que os recursos criados pelo professor têm como objetivo oferecer conselhos práticos para serem aplicados pelo estudante na elaboração da sua solução. O conteúdo das mensagens de *feedback* criadas para o mecanismo são sobre a correção de erros e os próximos passos que devem ser realizados para avançar na solução.

As mensagens de *feedback* possuem três formas de **apresentação**: textual, oral e visual. Sendo compostas por dicas e informações em formato de texto e vídeos, exemplos de códigos da solução e imagens de fluxogramas.

O mecanismo possui **personalização** baseada no contexto e *status* corrente da solução do estudante no momento em que ele solicita ajuda. Ou seja, para a seleção da mensagem de *feedback* mais adequada, o mecanismo se baseia no estado atual da solução do estudante, esteja ela inacabada ou finalizada.

5.3 Funcionamento do Mecanismo de *Feedback* no Domínio de Programação

O mecanismo de feedback foi implementado como um componente para o sistema de aprendizagem Marvin, sendo assim, ele faz uso das funcionalidades do sistema apresentadas na Seção 5.1.5. Nesta seção é apresentado o funcionamento do mecanismo de *feedback*, sendo descrita a modelagem da mensagem de *feedback*, sua forma de criação e algoritmo de escolha das mensagens que devem ser exibidas ao estudante após uma requisição.

São apresentados também dois cenários de utilização. O primeiro cenário a ser exposto é o de criação de mensagens de *feedback*. Nele, será descrito como o professor cria uma mensagem de *feedback* para uma questão específica já cadastrada no sistema Marvin. Também

será apresentado como o mecanismo trata internamente das mensagens cadastradas.

O segundo cenário de utilização do mecanismo é o de requisição de *feedback* pelo estudante. Neste cenário será descrita a forma como o estudante utiliza o mecanismo para pedir ajuda durante a resolução dos exercícios e como internamente o sistema decide qual mensagem de *feedback* deve ser selecionada e exibida ao estudante.

5.3.1 Criação das Mensagens de *Feedback*

Como foi exposto no Capítulo 2 (Fundamentação Teórica), sistemas criados para auxiliar o aluno no processo de aprendizagem proveem, tradicionalmente, três tipos de *feedback*: avaliativo (indicando se a solução está certa ou errada), indicação de erros (explicitando os erros da solução do estudante ou destacando o local do erro na solução) e dicas para construção da solução. É neste último tipo de *feedback* que o mecanismo criado na pesquisa está concentrado, ou seja, em fornecer *feedback* que ajude o estudante a progredir na elaboração de sua solução e/ou corrigir eventuais erros.

No geral, o comportamento de um professor ao fornecer *feedback* aos alunos durante aplicação de exercícios em sala de aula é fazer com que os alunos se lembrem de algo que eles já sabem (viram durante as aulas) ou fazer com que os estudantes tenham novos *insights* sobre como solucionar o problema, ou seja, fazer com que eles tenham a compreensão de como solucionar o problema por meio de captação mental dos elementos necessários e de como relacioná-los para chegar até uma solução correta. O *feedback* criado foi elaborado com o intuito de captar a essência deste tipo de *feedback*. Três formas de *feedback* são usadas: textos, vídeos e imagens.

O texto é composto por dicas e sugestões para indicar ao aluno como progredir na solução e/ou corrigir erros; o *feedback* em forma de vídeo foi elaborado com as mesmas mensagens dos *feedback*s de texto, sendo ditas em áudio com a imagem de uma IDE exibindo trechos de código citados no texto. Os *feedbacks* em forma de imagem são fluxogramas com um esqueleto de uma solução modelo correta para servir de base para o aluno construir sua própria solução.

Era necessário garantir a qualidade das mensagens de *feedback* para cada uma das formas (vídeos, textos e fluxogramas). Para tanto, foi utilizada a abordagem de "anotação humana". Essa abordagem consiste em seres humanos especialistas criarem anotações em dados pre-

viamente coletados. Essas anotações servirão como conteúdo para o *feedback*. Uma desvantagem do seu uso está no esforço humano necessário para a criação dos *feedback* por meio de anotações nas soluções coletadas; entretanto, a sua principal vantagem está no fato do *feedback* ser criado e checado por um especialista humano, o que na pesquisa era necessário para garantir qualidade às mensagens.

5.3.2 Cenário de Criação das Mensagens de *Feedback*

Esta seção descreve o cenário de funcionamento do mecanismo que permite ao professor ou especialista cadastrar mensagens de *feedback* para exercícios de programação. Primeiramente será apresentado o cenário de utilização do cadastro do ponto de vista do professor utilizando a interface gráfica do sistema.

Posteriormente, haverá uma discussão sobre o funcionamento interno do mecanismo após o professor efetuar o cadastro da mensagem de *feedback*.

O professor pode criar mensagens de *feedback* para cada questão de programação cadastrada no Marvin. Para isso, basta o professor selecionar a questão desejada e em seguida escolher a opção de criação de *feedback* para a questão. Após isso, o professor deve preencher o formulário exibido na Figura 5.4.

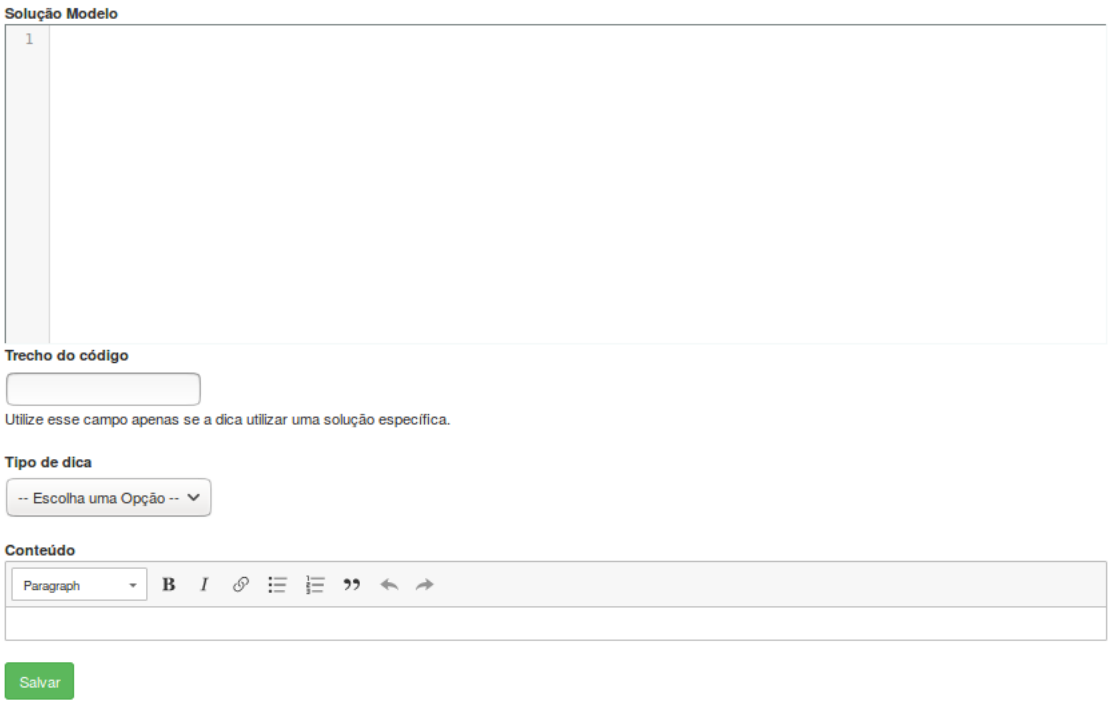
O cadastro de uma mensagem de *feedback* possui quatro campos: solução modelo, trecho do código, tipo de dica e conteúdo. Apenas os campos tipo de dica e conteúdo são obrigatórios, sendo assim, as seguintes configurações de cadastro estão disponíveis:

- **Preencher todos os campos.** Neste caso, o professor irá inserir uma solução modelo e, em seguida, indicar no campo "trecho de código" a parte da solução que a mensagem de *feedback* ficará associada. Isto é, o conteúdo da mensagem refere-se especificamente apenas a este trecho de código. No campo "tipo de dica" deve ser selecionada a opção de apresentação: vídeo, fluxograma ou texto. No campo "conteúdo" deverá ser inserido o texto da mensagem, a imagem do fluxograma ou o *link* para um vídeo.

- **Preencher apenas os campos tipo de dica e conteúdo.** As mensagens de *feedback* cadastradas dessa forma são denominadas de mensagens gerais e são formadas por dicas de interpretação do enunciado do problema ou sugerindo como iniciar uma solução.

- **Preencher os campos solução modelo, tipo de dica e conteúdo.** Neste caso, como nenhum trecho de código da solução modelo é indicado, a mensagem de *feedback* ficará associada a todo o código da solução modelo, indicando que o conteúdo da mensagem se refere a solução inteira.

Figura 5.4. Tela do sistema para criação de uma nova mensagem *feedback*.



Fonte: autoria própria.

O professor pode inserir várias mensagens de *feedback* com diferentes soluções modelo para uma mesma questão de programação. Ele também pode repetir uma mesma solução modelo e criar mensagens para diferentes trechos de código desta solução.

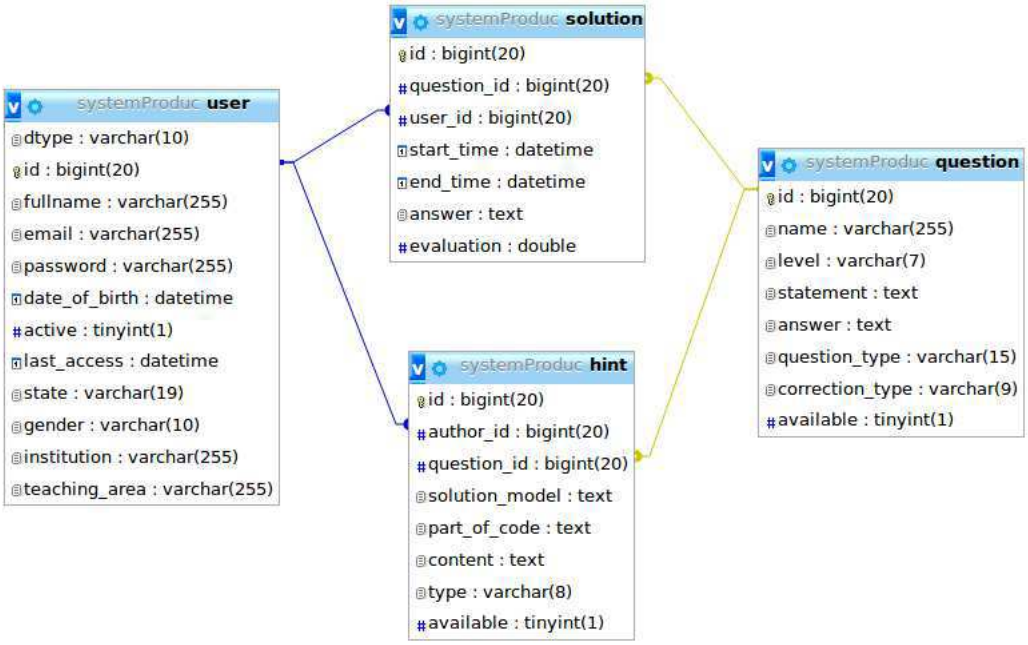
5.3.3 Modelagem da Mensagem de *Feedback* no Sistema

Para armazenar o *feedback* criado pelo professor no banco de dados, foi criada a entidade *hint*, que se conecta com as entidades já existentes no banco de dados do sistema Marvin: *user*, *question* e *solution*, como pode ser visto na Figura 5.5.

Os atributos da tabela *hint* são descritos a seguir.

- **id** - Identificador único do *feedback*.

Figura 5.5. Modelagem do *feedback* no banco de dados do sistema.



Fonte: autoria própria.

- **author_id** - Identificador do usuário autor do *feedback*.
- **question_id** - Identificador da questão à qual o *feedback* está relacionado.
- **solution_model** - Modelo de solução para a questão criada pelo professor.
- **part_of_code** - Parte do código da solução modelo para a qual o *feedback* foi criado. Pode ter valor nulo.
- **content** - Conteúdo da mensagem de *feedback*, pode ser um texto, o *link* para uma imagem ou o *link* para um vídeo.
- **type** - Tipo de conteúdo do *feedback*, pode ser: *VIDEO*, *TEXT* ou *IMAGE*.
- **available** - Valor booleano que indica se o *feedback* está liberado para ser exibido aos estudantes ou não.

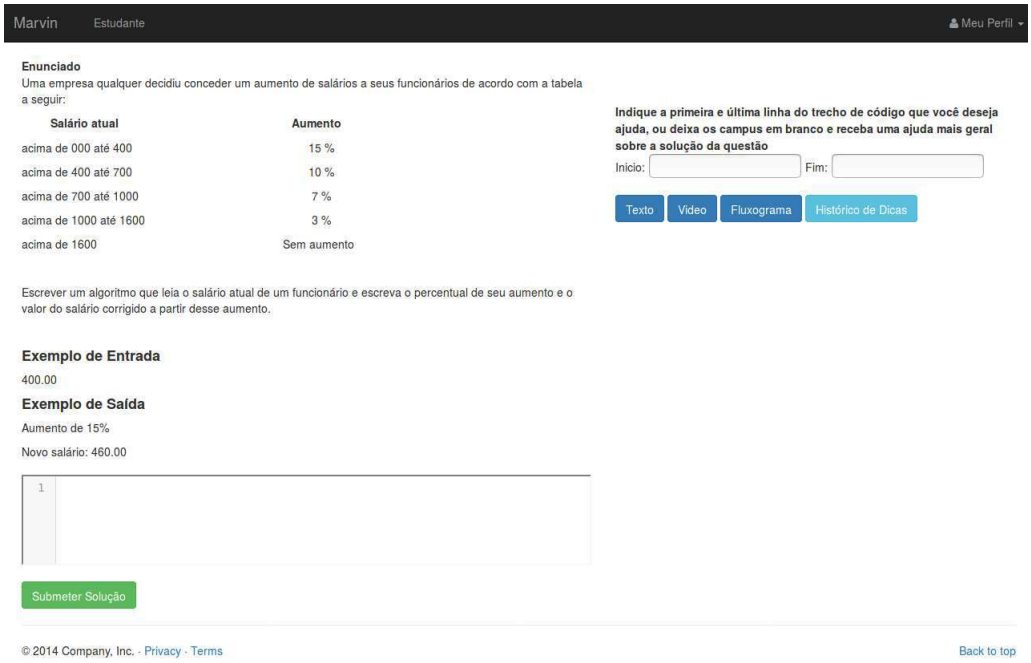
5.3.4 Cenário de Requisição e Entrega de Mensagens de *Feedback*

Esta seção descreve o cenário de utilização do mecanismo de *feedback* por parte do estudante, indicando o passo a passo de como o estudante requisita e recebe mensagens de *feedback* do

mecanismo.

O sistema Marvin possui uma série de questões de programação para as quais existem mensagens de *feedback* associadas. A Figura 5.6 apresenta um exemplo da interface de resolução de questões no sistema Marvin. A interface é dividida em duas partes, do lado esquerdo estão o enunciado e informações sobre a questão que deve ser solucionada e um campo de edição onde o estudante deve construir ou inserir a solução que será submetida ao sistema. O estudante pode programar diretamente na interface gráfica do sistema ou em uma IDE externa e inserir o código na interface.

Figura 5.6. Tela de resolução de questões exibida ao estudante.



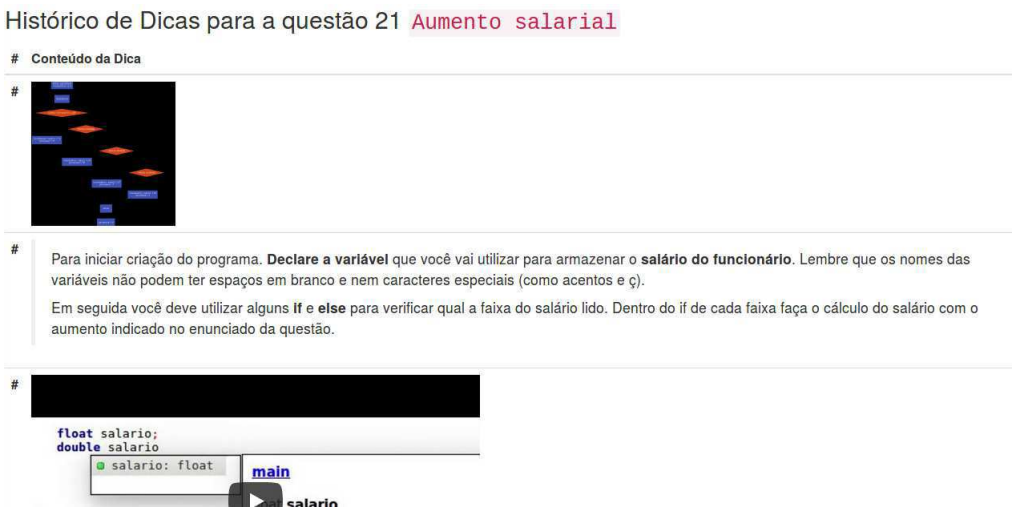
Fonte: autoria própria.

O lado direito da Figura 5.6 contém a área de *feedback*, na qual o estudante pode selecionar o tipo de mensagem que deseja (vídeo, texto ou fluxograma) e visualizar as mensagens. Para requisitar um *feedback* ao sistema, o estudante possui seis componentes de interface disponíveis, quatro botões e dois campos editáveis. Os botões disponíveis são: “Texto”, “Vídeo”, “Fluxograma” e “Histórico de dicas”, os três primeiros servem para o estudante requisitar um tipo de mensagem específica, se ele desejar receber uma mensagem textual ele selecionará o primeiro botão, se desejar um vídeo o segundo botão. Selecionando o terceiro botão o estudante receberá um fluxograma com um exemplo de uma solução modelo para a

questão.

O botão "Histórico de dicas" oferece ao estudante a possibilidade de visualizar o histórico de dicas recebidas até o momento para a questão que o estudante está solucionando. Um exemplo do histórico de dicas é exibido na Figura 5.7.

Figura 5.7. Exemplo de histórico de mensagens *feedback* recebidas pelo estudante.



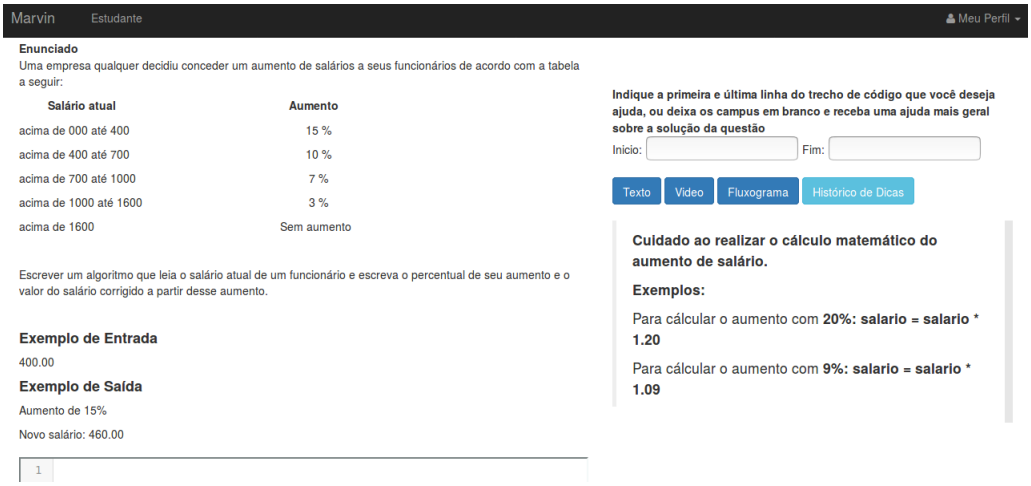
Fonte: autoria própria.

A área de *feedback* possui ainda dois campos editáveis, denominados de "Início" e "Fim". Eles servem para o estudante indicar um trecho da sua solução para o qual ele deseja ajuda, isto é realizado indicando o número das linhas em que o trecho inicia e termina. Após indicar o trecho o estudante requisita ajuda por meio dos botões descritos anteriormente. Em seguida, uma mensagem de *feedback* é selecionada pelo mecanismo e exibida na tela, conforme apresentado na Figura 5.8.

5.3.5 Funcionamento da Seleção da Mensagem de *Feedback*

Uma série de questões estão envolvidas ao trabalhar com *feedback* em ambientes educacionais, uma dessas é o momento do provimento do *feedback* (como exposto no capítulo 4). No mecanismo criado, foi utilizada uma abordagem de *feedback* imediato, fornecida por meio da requisição do aluno. Alguns estudos mostram que esse tipo de abordagem *on-demand*, em que o *feedback* precisa ser requisitado, apresenta resultados melhores do que abordagens automáticas, nas quais o sistema decide o momento do provimento do *feedback* (RAZZAQ;

Figura 5.8. Exibição de conteúdo de *feedback* para o estudante.



Fonte: autoria própria.

HEFFERNAN, 2010). Para solicitar um *feedback* ao sistema, o estudante precisará executar as ações descritas na seção 5.3.4, que expõem os cenários de requisição de mensagens de *feedback*. De forma sucinta, o estudante deve:

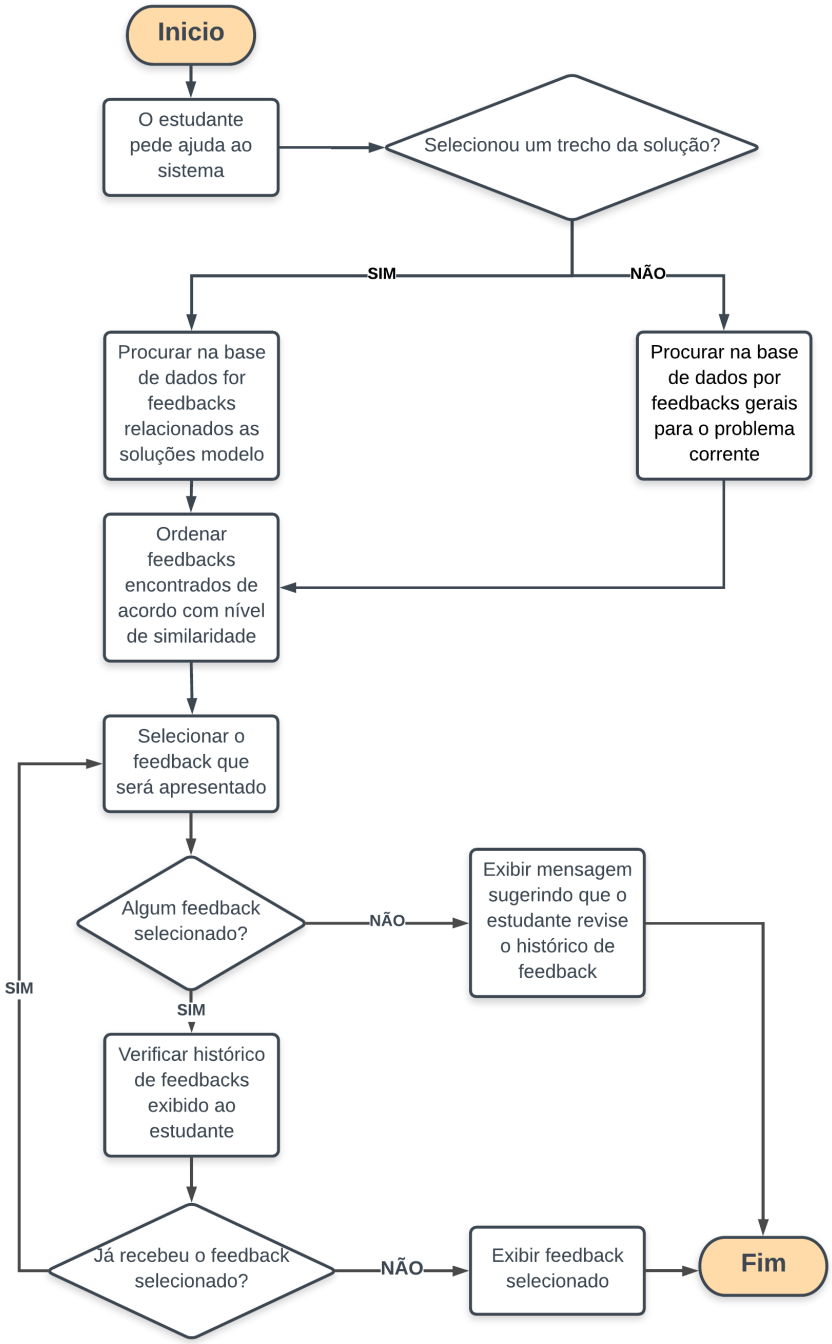
1. Indicar o trecho de código para o qual precisa de ajuda. Esse passo é opcional e será usado pelo algoritmo de seleção de *feedback*. Caso o estudante não marque nenhum trecho específico, o *feedback* será selecionado utilizando como base todo o código elaborado até o momento, que pode ser inclusive vazio, caso o aluno não tenha iniciado a elaboração da solução.

2. Pressionar um dos botões de ajuda (três estão disponíveis para requisitar ajuda em forma de texto, fluxograma ou vídeo).

A partir do momento em que o estudante seleciona o botão de um dos tipos de ajuda, começa um processo interno no mecanismo de *feedback*, exibido em formato de fluxograma na Figura 5.9.

Inicialmente, o mecanismo verifica se o estudante inseriu para qual trecho da solução ele deseja receber ajuda. Caso nenhum trecho tenha sido especificado, o mecanismo irá utilizar o estado atual da solução do estudante para procurar por *feedbacks* gerais, que são mensagens de *feedback* cujo conteúdo foi criado e associado a uma solução modelo completa ou que

Figura 5.9. Fluxograma de funcionamento interno do mecanismo de *feedback*.



Fonte: autoria própria.

foi criado sem estar associado a nenhuma solução modelo (ver detalhes sobre a criação das mensagens de *feedback* na seção 5.3.2).

Caso o estudante tenha especificado trecho da solução para o qual ele deseja receber ajuda, então o sistema irá procurar por *feedbacks* específicos, que são mensagens de *feedback* cujo conteúdo foi criado e associado a trechos de soluções modelo.

O sistema realiza o processo de busca seguindo os seguintes passos:

- Recupera todas as mensagens de *feedback* vinculadas à questão que o estudante está solucionando;
- Extrai o trecho de código especificado do estado atual da solução do estudante. Caso o trecho não seja especificado, a solução inteira será extraída;
- Extrai os trechos de código das soluções modelo cujas mensagens de *feedback* estão associadas. No caso de *feedback* gerais, a solução inteira é extraída;
- Realiza um cálculo de similaridade entre o trecho de código da solução do estudante e os trechos extraídos das soluções modelo das mensagens de *feedback*;
- Retorna as mensagens cujo cálculo de similaridade seja diferente de zero.

Após o processo de busca, o mecanismo cria uma lista das mensagens recuperadas em ordem decrescente de acordo com o grau de similaridade. Na sequência, a primeira mensagem de *feedback* é selecionada e comparada ao histórico de dicas já recebidas pelo estudante para a questão. Caso a mensagem já tenha sido apresenta ao estudante no decorrer da resolução da questão, a próxima mensagem da lista é selecionada e o processo se repete até que alguma mensagem nova seja exibida ao estudante, ou, caso o estudante já tenha recebido todas as mensagens de *feedback* contidas na lista, o mecanismo irá recomendar que o estudante revise seu histórico de *feedback* para tentar progredir na elaboração da sua solução.

Conforme mencionado, o mecanismo de *feedback* faz uso de um cálculo de similaridade para determinar qual mensagem de *feedback* deve ser apresentada ao estudante. O mecanismo realiza o cálculo de similaridade entre o trecho de código fornecido pelo estudante e os trechos das soluções modelo. Para realização deste cálculo é utilizado o ANTLR (ANTRL, 2018), para gerar um analisador para construir e analisar árvores sintáticas abstratas,

do inglês *abstract syntax tree* (AST), que são representações da estrutura de um código fonte escrito em uma linguagem de programação (para mais detalhes ver Capítulo 2).

No processo de cálculo de similaridade, primeiro é criada uma representação da AST dos trechos de código das soluções modelo e do trecho da solução do estudante, cuja representação é fornecida em formato *string*. Em seguida, as *strings* são comparadas utilizando o algoritmo de distância de Levenshtein (GILLELAND, 2009), que é realizada por meio de verificações de quantas adições, remoções ou substituições são necessárias para transformar uma *string* na outra. A comparação que resultar no menor número de modificações é considerada a mais similar.

5.4 Estudos Prévios para Validação do Mecanismo de *Feedback*

O mecanismo de *feedback* para o domínio de programação foi criado baseado na especificação apresentada na seção 5.2 utilizando a linguagem de programação Java e integrado como um componente do sistema Marvin. Para que fosse possível realizar a validação do mecanismo, foram realizados dois estudos prévios. O primeiro foi com o objetivo de elaborar as mensagens de *feedback* que seriam cadastradas no sistema. O segundo foi baseado no requisito do mecanismo de *feedback* de que o estudante indique para qual trecho da solução ele precisa de ajuda; sendo assim, o estudo teve como objetivo verificar se os estudantes eram capazes de identificar tais trechos.

Nas próximas seções, ambos os estudos serão apresentados. Os estudos foram realizados com alunos de uma turma de iniciantes da disciplina de introdução a programação de um curso técnico de Informática para Internet. Ambos os estudos foram aplicados pelo desenvolvedor do mecanismo de *feedback* e pelo professor da disciplina cujos alunos faziam parte.

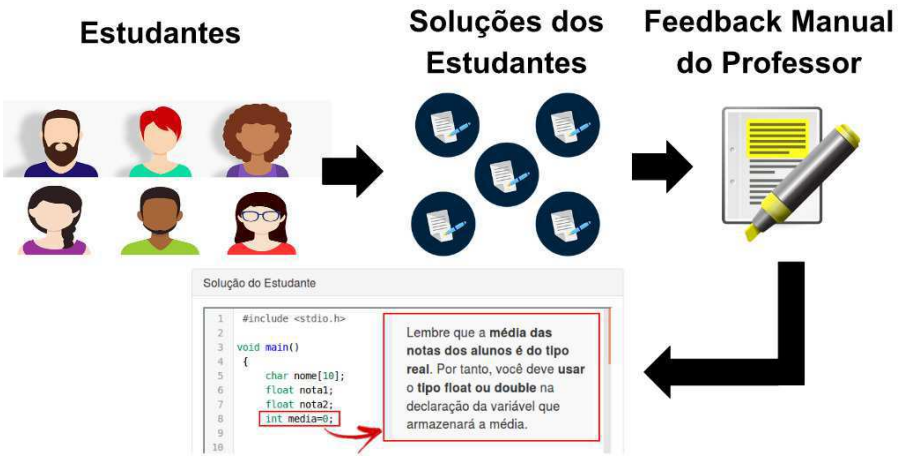
5.4.1 Estudo para Elaboração do Conteúdo das Mensagens de *Feedback*

O estudo descrito nesta seção teve como objetivo elaborar as mensagens de *feedback* que seriam cadastradas no sistema Marvin para serem utilizadas pelo mecanismo de *feedback*. Como foi descrito na seção 5.3.1, o mecanismo de *feedback* criado nesta pesquisa utiliza o método de "anotação humana" para criação das mensagens. Sendo assim, era necessário coletar um conjunto de soluções, para que as mesmas pudessem ser analisadas por especialistas, para a criação das anotações.

Tais anotações são vinculadas a diferentes trechos das soluções e contêm a mensagem de *feedback* que será apresentada aos alunos. Para a realização dessa coleta, um conjunto de seis alunos voluntários usaram um ambiente de resolução de problemas de programação (Marvin) para solucionar um conjunto de questões de programação selecionadas previamente. A linguagem C foi utilizada para programação das soluções das questões.

Este estudo seguiu o procedimento apresentado na Figura 5.10. Primeiramente foram indicadas cinco questões de programação para o seis alunos resolverem (questões disponíveis no Apêndice A).

Figura 5.10. Procedimento realizado no Estudo para Criação das Mensagens de *Feedback*.



Fonte: autoria própria.

Cada aluno estava em um computador individual e utilizou uma IDE *desktop* para programar as soluções das questões na linguagem C. Após finalizar as soluções, os alunos tinham acesso ao sistema Marvin para submetê-las.

Ao submeter uma solução, a mesma era avaliada por um professor de programação que estava presente no mesmo local e tinha acesso ao sistema. Para cada submissão dos alunos, ele realizava uma avaliação e fornecia uma mensagem de *feedback* aos alunos via sistema. Caso as soluções estivessem incorretas, os alunos realizavam alterações e submetiam novamente até chegar a uma solução correta, sempre recebendo uma mensagem de *feedback* do professor para cada submissão incorreta. As mensagens de *feedback* fornecidas pelo professor foram todas registradas no sistema.

Os alunos tiveram 4 horas para solucionar as questões, tendo sido submetidas ao total 54 soluções no sistema. Um resumo das submissões é apresentado na Tabela 5.1, na qual é possível observar os dados de submissão para cada uma das cinco questões.

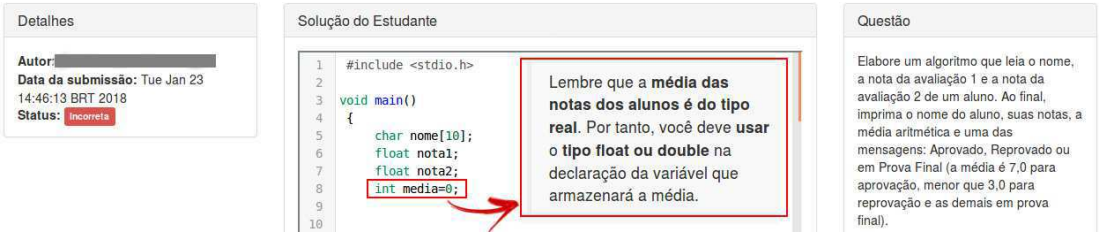
Tabela 5.1. Dados as submissões dos alunos utilizadas para criação do *feedback*.

	Total	**Q01**	**Q02**	**Q03**	**Q04**	**Q05**
Total de soluções	54	16	10	12	10	6
Soluções corretas	26 (48%)	6	6	6	5	3
Soluções incorretas	28 (52%)	10	4	6	5	3

Em um momento posterior, o registro de mensagens fornecidas pelo professor e as soluções a que elas estavam associadas foram submetidas a um processo de análise e aprimoramento. A partir desse registro, foram criadas as mensagens de *feedback* que foram cadastradas no sistema para serem utilizadas na aplicação de experimentos para validação do mecanismo de *feedback*.

As mensagens de *feedback* geradas deste processo foram cadastradas como mensagens do tipo "texto" no sistema e associadas aos trechos de código das soluções das quais elas se referiam. Na Figura 5.11 é apresentado um exemplo de submissão incorreta de um aluno, com o destaque para o trecho de código anotado e a mensagem de *feedback* associada. Para a criação das mensagens de *feedback* em forma de vídeo, foi utilizado o mesmo texto exposto de forma oral com a gravação de uma demonstração do passo a passo para programar o que foi sugerido. O *feedback* em forma de fluxograma foi criado usando como base o esqueleto das soluções mais comuns submetidas pelos alunos.

Figura 5.11. Exemplo de uma solução incorreta e a mensagem de *feedback* associada.



Fonte: autoria própria.

5.4.2 Estudo para Analisar Procedimento de Requisição de *Feedback*

Esta seção apresenta um estudo baseado no requisito do mecanismo de *feedback* de que o estudante indique para qual trecho da solução ele precisa de ajuda. Dessa forma, o estudo teve como objetivo verificar se os estudantes eram capazes de identificar tais trechos. Ressalta-se, ainda, que a indicação do trecho da solução não é obrigatória para que o estudante receba um *feedback*, porém é altamente recomendada.

A marcação do trecho do código para o qual o aluno deseja receber um *feedback* parte do princípio de que o aluno é capaz de identificar e explicitar o trecho da solução que pode estar errado ou que ele não consegue desenvolver. Porém, surge um questionamento: se o aluno é capaz de encontrar o trecho, ele também não seria capaz de solucioná-lo? Para verificar esta questão, foi realizado um estudo com 19 alunos iniciantes em programação. Tais alunos pertencem à mesma turma da disciplina de introdução a programação do estudo realizado na seção 5.4.1.

O objetivo deste estudo foi verificar se os alunos eram capazes de identificar trechos de código com erros, e caso conseguissem, se também eram capazes de corrigi-los.

Para a realização do estudo, foi elaborado um questionário contendo 10 questões de programação com suas respectivas soluções (questionário disponível no Apêndice B). Cada solução possuía um erro semântico, que poderia ser desde a troca de um operador lógico até a falta de estruturas de repetição ou seleção necessárias para solução.

Para cada uma das 10 questões, os estudantes deveriam informar o trecho de código com problema e modificar a solução fornecida para torná-la correta. O questionário foi aplicado da seguinte forma: os estudantes receberam um arquivo de documento de texto contendo as questões e deveriam respondê-las no próprio documento e submetê-lo de forma *online* ao

final de um período de tempo de 4 horas.

Os resultados deste estudo são apresentados na Tabela 5.2. Apenas três alunos conseguiram responder todas as 10 questões, 79% dos alunos responderam mais da metade das questões e 21% responderam menos da metade.

Tabela 5.2. Dados da avaliação das respostas dos estudantes no estudo sobre identificação de erros no código.

		Avaliação da correção das soluções			
		Incorreta	**Não respondida**	**Correta**	**Total**
Avaliação da identificação dos trechos de código	**Incorreta**	8	0	3	11
	Não respondida	0	74	0	74
	Correta	21	0	84	105
	Total	29	74	87	190

Das 116 respostas, em 72% delas os alunos foram capazes tanto de encontrar o trecho de código quanto de corrigi-lo. Em 7% dos casos os alunos não foram capazes de identificar o trecho de código e em 21% dos casos os alunos identificaram o trecho de código, mas não foram capazes de corrigi-lo. Dos 19 alunos participantes, apenas 3 conseguiram responder todas as questões corretamente, ou seja, para todas as questões eles encontraram o trecho de código problemático e o corrigiram.

Considerando a Tabela 5.2, é possível afirmar que em 50% dos casos, o mecanismo de *feedback* criado poderia ser útil aos estudantes. Em 21 casos o uso do sistema de *feedback* aqui proposto seria útil aos alunos, pois eles foram capazes de indicar o trecho de código, mas não de corrigi-lo. Em tais situações, eles forneceriam ao sistema o trecho de código que imaginavam estar errado e, caso houvesse algum *feedback* relacionado cadastrado, o mecanismo apresentaria a mensagem correspondente. Houve 74 casos nos quais os estudantes não responderam as questões. Nessas situações, o mecanismo também seria útil, dado que o sistema também é capaz de fornecer *feedback* mesmo sem que nenhum trecho seja marcado ou mesmo que o aluno não tenha iniciado a solução (nesses casos, o sistema fornece dicas sobre como iniciar a solução).

Vale ressaltar que nenhum dos alunos respondeu as questões em sequência, ou seja, todos os alunos que responderam o questionário de forma incompleta "pularam" algumas questões e deliberadamente escolheram as questões para responder. Isto pode indicar que eles procuraram por questões mais fáceis ou que eles acreditavam que seriam capazes de responder.

Como já foi mencionado, alunos em tais situações também podem se beneficiar com o uso do mecanismo criado.

5.5 Aplicação de Quase-Experimento com o Mecanismo de *Feedback*

Nesta seção é apresentada a validação do mecanismo de *feedback* criado por meio da aplicação de um quase-experimento (LEVY; ELLIS, 2011). A decisão da realização de um quase-experimento ocorreu devido ao fato de não ser possível realizar um controle experimental completo no que se refere à seleção randomizada dos sujeitos envolvidos, e de não ser possível a criação de um grupo de controle. Esta impossibilidade ocorre devido os experimentos serem aplicados em uma turma em andamento da disciplina de introdução a programação. O experimento não poderia afetar o andamento da disciplina, e separar os alunos em grupo de controle e experimental poderia gerar conflitos entre eles, visto que não seria possível controlar sua comunicação, pois acabariam sabendo a qual grupo pertencem e quais intervenções estavam sendo realizadas em cada um.

As próximas seções irão tratar sobre a questão de pesquisa que norteia a realização do quase-experimento, a descrição, o design e aplicação e, por fim, os resultados e discussões.

5.5.1 Questão de Pesquisa

Um quase-experimento foi conduzido com o intuito de avaliar o impacto da utilização do mecanismo de *feedback*, que provê informações de auxílio ao estudante durante o processo de elaboração de soluções para exercícios de programação.

O objetivo do quase-experimento foi verificar se ocorreria uma melhoria significativa dos estudantes com relação ao seu conhecimento de programação após a utilização do mecanismo. Sendo assim, foi definida a seguinte questão de pesquisa:

Questão de Pesquisa: *Os alunos apresentam uma melhoria significativa em seu desempenho em programação após a utilização do mecanismo de feedback?*

Com base na questão proposta, foram definidas as seguintes hipóteses:

- $H0$ - Não existe diferença significativa entre as médias do pré-teste e do pós-teste

($\mu_{pre} = \mu_{pos}$);

- $H1$ - A média do pós-teste é significativamente maior que a média do pré-teste ($\mu_{pre} < \mu_{pos}$).

Na próxima seção serão descritos os participantes do quase-experimento.

5.5.2 Participantes do Quase-Experimento

Para responder a questão de pesquisa foi realizado um quase-experimento com um grupo de 34 estudantes da disciplina de introdução a programação de um curso técnico de informática para Internet. Tais estudantes pertencem à mesma turma na qual foram aplicados os estudos descritos nas seções 5.4.1 e 5.4.2. Os participantes do estudo da seção 5.4.1 não participaram do quase-experimento, pois poderiam enviesar os resultados, visto que as mensagens de *feedback* usadas foram criadas baseadas nas soluções elaboradas por eles nesse estudo.

5.5.3 Design e Aplicação do Quase-Experimento

Esta seção descreve o planejamento do quase-experimento e seu procedimento de aplicação.

O quase-experimento foi dividido em três etapas, totalizando 4 horas de duração, são elas: Aplicação de pré-teste (1 hora de duração); Resolução de questões de programação usando o mecanismo de *feedback* desenvolvido (2 horas de duração); Aplicação de pós-teste (1 hora de duração);

Primeiramente, foi realizado um pré-teste para verificar o desempenho dos alunos em programação. Após o uso do mecanismo de *feedback* foi aplicado um pós-teste para verificar novamente o desempenho dos alunos. O pré-teste era composto por uma única questão de programação denominada Móbile (ver Apêndice D) e o pós-teste também era composto por uma única questão denominada Sedex (ver Apêndice D).

Para avaliar o desempenho dos alunos no pré-teste, foram formulados oito casos de testes de programação. Os casos de testes constituem um par de entrada e saída, onde a solução do aluno é executada com a entrada do caso de teste e a saída da fornecida pela solução do aluno é comparada com a saída do caso de teste. Para cada caso que a solução do estudante retornasse a saída esperada, eram atribuídos 1,25 ponto. Caso a solução retornasse correta-

mente todas as saídas esperadas, o estudante obteria nota 10,0. A mesma forma de avaliação foi aplicada no pós-teste.

Na etapa de resolução de questões de programação usando o mecanismo de *feedback*, os alunos utilizaram o sistema Marvin para solucionar três exercícios de programação com a ajuda do mecanismo de *feedback*. A lista de questões que os estudantes deveriam solucionar está disponível no Apêndice C.

Todas as questões de programação aplicadas durante o quase-experimento exigiam que o aluno praticasse os seguintes conhecimentos de programação: declaração de variáveis, estruturas de seleção, expressões lógicas e aritméticas.

5.5.4 Resultados e Discussões

Nesta seção são apresentados os resultados da aplicação do quase-experimento acompanhados de uma discussão sobre o impacto observado do uso do mecanismo de *feedback* no desempenho dos alunos.

Para avaliar o impacto o uso de mecanismo, primeiramente foi realizada uma verificação se os resultados dos pré-teste e do pós-teste possuíam distribuições normais. Para isto, foi utilizado o teste Shapiro-Wilk. Tanto as notas do pré-teste ($p - valor$ = 0,08 > 0,05) quanto as do pós-teste ($p - valor$ = 0,079 > 0,05) apresentam distribuição normal. Sendo assim, a hipótese nula do teste de Shapiro-Wilk não foi rejeitada, ou seja, possuem distribuição normal.

Dado que as médias do pré-teste e do pós-teste foram comparadas, e que ambas possuem distribuição normal, foi selecionada a aplicação do teste *t student* pareado para a realização do teste de hipótese. O resultado do teste indicou um $p - valor$ = 6,674e-06, o que rejeita a hipótese nula, indicando que os resultados dos estudantes no pós-teste são significativamente maiores que os obtidos pelos mesmos na realização do pré-teste.

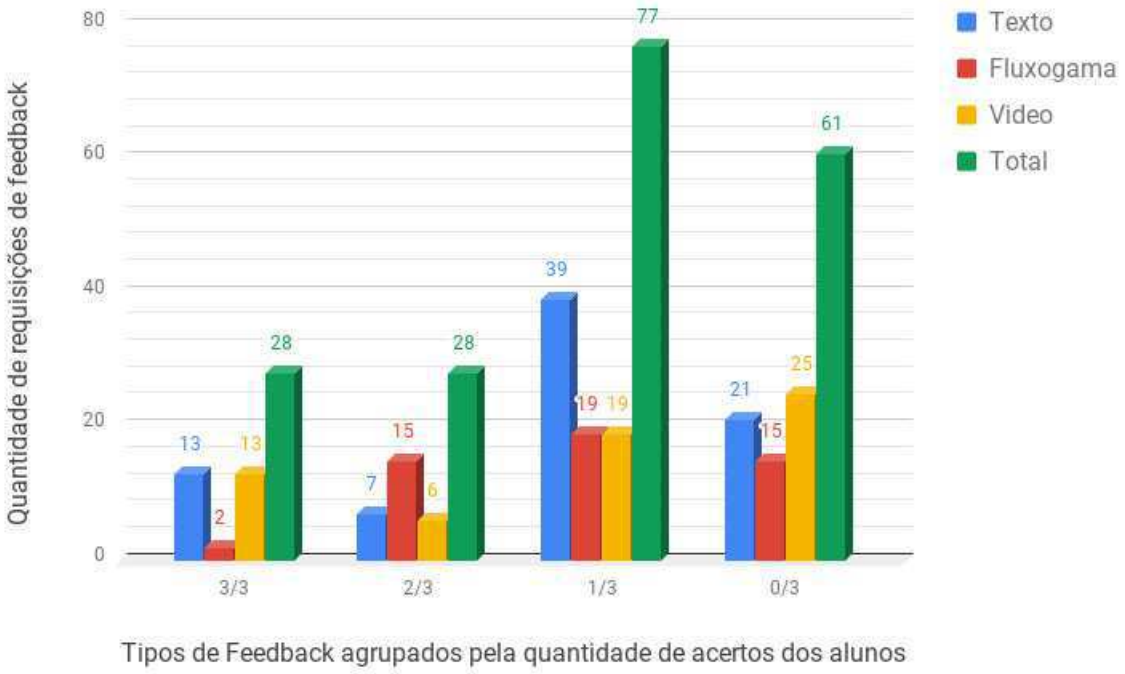
O resultado do experimento aponta que o uso do mecanismo teve um impacto positivo na aprendizagem dos estudantes. A média das avaliações do pré-teste foi de 4,6 com mediana 5,0, comparando com a média do pós-teste, 5,8, e a mediana, 6,2, é perceptível a melhoria de desempenho.

Para obter uma melhor compreensão do resultado encontrado, foi realizada uma análise da utilização do mecanismo de *feedback* durante a resolução das questões de programação

no quase-experimento. Na Figura 5.12, é exibido o gráfico que informa a quantidade de requisições de mensagens de *feedback* por parte dos alunos discriminadas por seu tipo (texto, fluxograma e vídeo). As requisições foram agrupadas de acordo com o desempenho dos estudantes na resolução dos exercícios.

Na Figura 5.12, da esquerda para a direita, o primeiro grupo contém os dados das requisições de *feedback* dos alunos que acertaram todas as três questões propostas durante a utilização do mecanismo de *feedback*. O segundo grupo contém as requisições dos alunos que acertaram duas das três questões. O terceiro grupo contém as requisições dos alunos que acertaram apenas uma das três questões. E, por fim, o último grupo contém as requisições dos alunos que não conseguiram acertar nenhuma das questões propostas. Para esse estudo, foi considerado que o estudante acertou a questão apenas se ele conseguiu submeter uma solução que passou em todos os casos de testes criados para ela.

Figura 5.12. Gráfico de requisições de *feedback* durante o experimento.



Fonte: autoria própria.

No grupo de estudantes que acertaram todas as questões da lista, o *feedback* menos requisitado foi o fluxograma, estando o *feedback* de texto e vídeo empatados. O inverso ocorre com o grupo de estudantes que acertaram duas das três questões, que preferiram o *feedback* de fluxograma, estando os outros dois também tecnicamente empatados no número de requi-

sições.

O grupo de estudantes que não acertou nenhuma questão foi o único cujas requisições por *feedback* de vídeo superaram as demais. Na Tabela 5.3 é apresentado um resumo dos dados de requisições de *feedback* levando em consideração todos os alunos.

Tabela 5.3. Dados estatísticos das requisições de *feedback* dos estudantes.

Feedback	**Quant.**	%	**Média/aluno**	**Mediana/aluno**	**Desvio Padrão/aluno**
Texto	80	41,2%	2,35	1	3,0
Fluxograma	51	26,3%	1,5	0,5	2,2
Vídeo	63	32,5%	1,85	1	2,5
Total	194	100%	5,70	1	2,6

É perceptível na Figura 5.12 que os estudantes com melhor desempenho (acertaram mais questões) requisitaram menos *feedback*, enquanto os alunos com desempenho mais baixo pediram um número significativo de ajuda. Tal comportamento já era esperado, dado que os alunos que possuem mais dificuldades são os que mais necessitam de apoio. Entretanto, faz-se necessário uma interpretação cuidadosa com relação ao impacto do *feedback* recebido por estes diferentes grupos, uma vez que outros fatores podem influenciar a resolução correta de uma questão, como, por exemplo, o estudante pode receber uma mensagem de *feedback* que o ajudaria a avançar o estado da sua solução, porém, não é capaz de interpretá-la e aplicá-la.

O grupo dos participantes que não conseguiram solucionar corretamente nenhuma questão da lista é formado por 11 alunos, dos quais fazem parte os que obtiveram menores resultados no pré-teste.

O total de submissões de solução ao sistema, discriminadas por questão, e o total de dicas requisitadas por este grupo podem ser vistas na Tabela 5.4.

Conforme Tabela 5.4, a maioria dos alunos desse grupo não fez muitas requisições de ajuda; na realidade, o quantitativo de ajuda desse grupo é alto devido a 3 alunos específicos (identificados como 48, 40 e 32). Juntos eles realizaram 65% do total de requisições de *feedback*. Eliminando os 3 *outliers*, obtem-se o gráfico exibido na Figura 5.13.

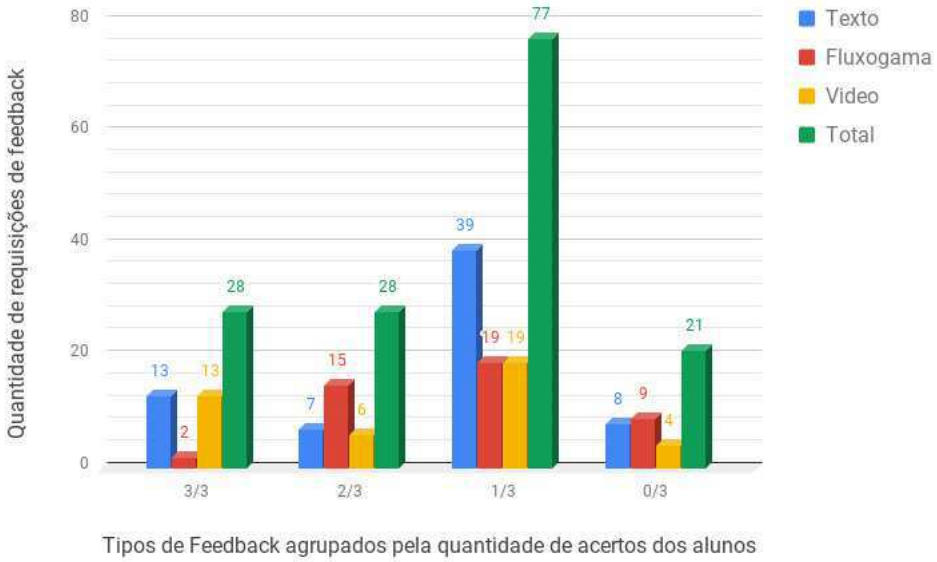
O mesmo procedicmento de eliminação dos alunos considerados *outliers* foi aplicado ao grupo de estudantes que acertou apenas uma das três questões, o resultado da inclusão deste procedimento é apresentado no gráfico da Figura 5.14.

Ainda com relação ao grupo de participantes do quase-experimento que não acertou ne-

Tabela 5.4. Dados estatísticos das submissões de soluções no sistema e requisição de dicas dos alunos que não conseguiram acertar nenhuma questão da lista.

Estudantes	**Submissões**				**Dicas**
#	**Q1**	**Q2**	**Q3**	**Total**	**Total**
26	1	1	1	3	0
45	2	2	1	5	0
59	1	1	0	2	0
16	2	0	0	2	3
29	2	1	1	4	3
47	2	0	0	2	3
60	1	0	0	1	4
17	1	0	0	1	8
48	3	1	0	4	10
40	1	1	0	2	12
32	2	1	0	3	18

Figura 5.13. Gráfico de requisições de *feedback* sem os *outliers* do grupo de estudantes que acertou 0/3 das questões.

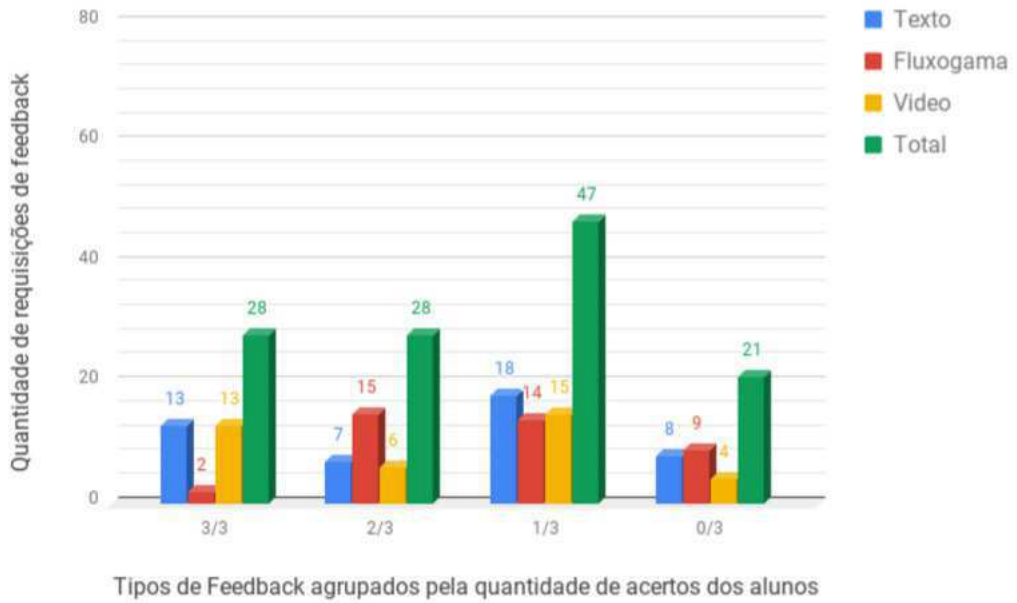


Fonte: autoria própria.

nhuma das questões propostas, após análise manual das soluções e mensagens de *feedback* fornecidas a este grupo, pode-se constatar que em 85% dos casos as mensagens de *feedback* eram úteis para o aluno progredir em sua solução. Isso significa que, mesmo que as mensagens recebidas sejam úteis e aplicáveis, em muitos casos, os estudantes aplicaram corretamente aquilo que a mensagem sugeria. Entretanto, o tempo decorrido entre o recebimento da

mensagem de *feedback* e a submissão de uma nova submissão era mais alto que os demais. Possivelmente, se fosse fornecido mais tempo na segunda etapa do quase-experimento (re-solução de questões com o mecanismo), eles poderiam ter finalizado corretamente a questão.

Figura 5.14. Gráfico de requisições de *feedback* sem os *outliers* dos grupos de estudantes que acertaram 0/3 e 1/3 das questões.



Fonte: autoria própria.

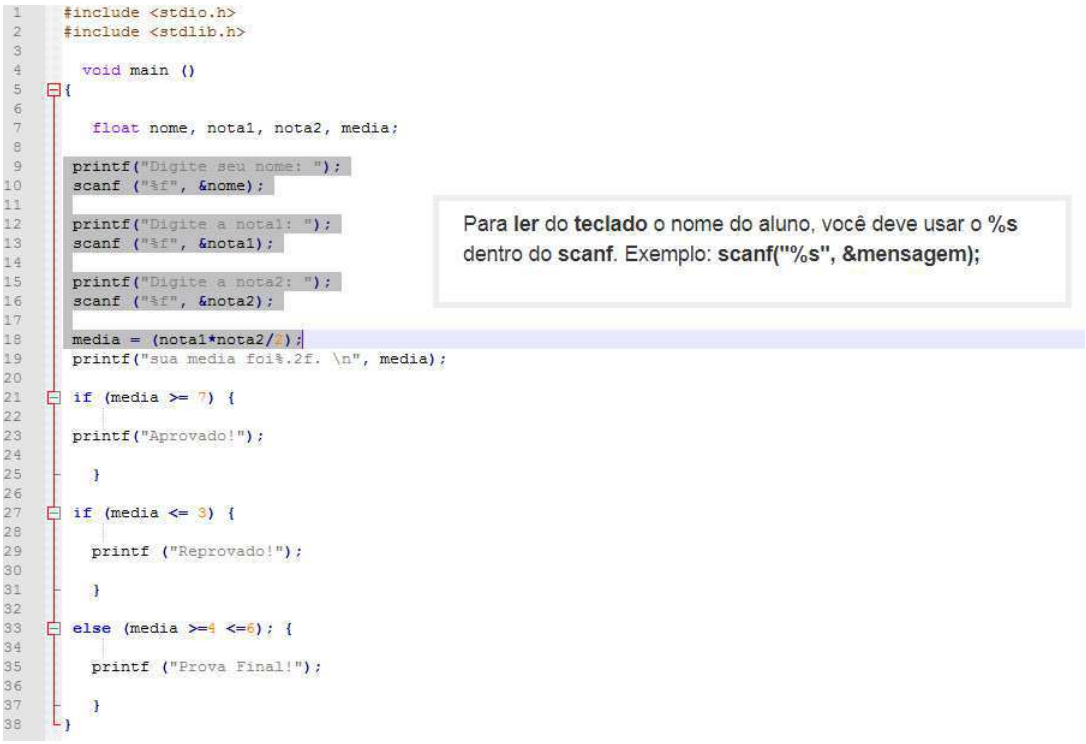
Para exemplificar a situação citada, pode-se utilizar como exemplo o estudante #16 da Tabela 5.4. Este aluno tentou solucionar apenas a questão 1 durante todo o tempo de uso do sistema.

Analisando o *log* do sistema que registra as interações dos estudantes foi possível verificar que, em sua primeira tentativa, este aluno #16 submeteu uma solução completa ao sistema, porém a mesma estava errada. Antes de submeter sua segunda tentativa, ele requisitou 3 *feedbacks*. Para as duas primeiras mensagens o aluno demonstrou ter compreendido o *feedback* alterando corretamente sua solução. Na terceira e última mensagem, o aluno não conseguiu realizar a alteração necessária para chegar à solução correta. O tempo determinado para esta etapa do experimento acabou sem que ele submetesse uma nova solução.

Na Figura 5.15, é exibido o estado da solução do aluno #16 quando o mesmo requisitou a primeira dica, com destaque para o trecho de código selecionado pelo aluno para requisição de *feedback*. A direita da solução é exibida a mensagem de *feedback* selecionada e exibida

pelo mecanismo de *feedback*. Neste caso, mesmo não tendo solucionado corretamente a questão, pode-se considerar que o mecanismo ajudou o aluno a progredir e corrigir erros.

Figura 5.15. Exemplo de *feedback* apresentado para o aluno, dado o trecho da solução que foi selecionado.



Fonte: autoria própria.

Por fim, é possível concluir que o mecanismo de *feedback* proposto nesta pesquisa causou um impacto positivo na aprendizagem dos estudantes. Foi constatado que mesmo os participantes do quase-experimento que não conseguiram solucionar corretamente questões de programação durante o uso do mecanismo receberam mensagens adequadas e conseguiram, em sua maioria, aplicar as dicas e sugestões recebidas.

5.5.5 Ameaças à Validade do Quase-Experimento

No que diz respeito às ameaças à validade do quase-experimento aplicado para validação do mecanismo de *feedback*, foram identificadas ameaças interna e externa. Quanto a ameaças à validade interna, a maturação constitui uma ameaça, pois o tempo de aplicação foi de 4 horas ininterruptas; com o passar do tempo, os participantes podem ter se cansado e se

sentido desmotivados a continuar.

Referente às ameaças à validade externa, duas foram identificadas. A primeira quanto à seleção dos participantes, os grupos eram formados por 34 estudantes de uma turma de introdução a programação, todos os alunos da turma participaram, então não houve seleção e nem aplicação de randomização. A segunda diz respeito a história, dado que no momento da aplicação do experimento os estudantes estavam em pleno período letivo do curso técnico de informática, de forma que os conteúdos que eles estavam aprendendo ou o período de avaliações das disciplinas que eles estavam cursando podem ter interferido no desempenho deles no experimento.

Capítulo 6

Considerações Finais e Sugestões para Pesquisas Futuras

Neste capítulo final, será apresentada uma síntese da pesquisa realizada com uma exposição de suas principais contribuições. Além disto, serão discutidas algumas das limitações encontradas na pesquisa, para as quais são sugeridas pesquisas que possam ser realizadas no futuro para dar continuidade ao estudo.

6.1 Considerações Finais

Esta pesquisa teve como objetivo apresentar um mecanismo de *feedback* para ser utilizado em ambientes educacionais de apoio ao ensino de programação. Inicialmente, a pesquisa proporcionou a apresentação do problema de aprendizagem de programação e da necessidade de os alunos receberem ajuda durante a atividade de resolução de exercícios de programação. Após a descrição do problema, foram apresentadas as principais pesquisas encontradas na literatura que visam auxiliar o estudante durante o processo de resolução de problemas de programação.

Durante o processo de planejamento do mecanismo de *feedback* proposto, surgiu a necessidade de especificar as características que as mensagens de *feedback* teriam. Foi realizado um levantamento da literatura de pesquisas que visavam propor uma caracterização de *feedback*. Após esse levantamento, constatou-se dificuldades no uso dos modelos encontrados, além de não contemplarem todas as características que seriam determinadas no meca-

nismo que estava sendo proposto. Sendo assim, propôs-se como uma das contribuições deste trabalho um modelo conceitual para especificação de *feedback* pedagógico, que foi utilizado para especificar o mecanismo de *feedback* proposto nesta pesquisa e pode ser reutilizado para realizar especificação e análises comparativas de outros mecanismos de *feedback*.

A validação do mecanismo foi realizada por meio de um quase-experimento, que teve como objetivo avaliar o impacto do uso do mecanismo na aprendizagem de programação do estudante. Para isso, o mecanismo foi integrado a um sistema de aprendizagem de programação já existente. Os resultados do quase-experimento mostraram que o uso do mecanismo teve um impacto positivo na aprendizagem dos estudantes. Além disso, na maioria dos casos, os estudantes conseguiram aplicar corretamente as mensagens de *feedback* recebidas, ou seja, conseguiram corretamente refletir sobre as informações das mensagens em modificações em suas soluções para progredir na resolução dos exercícios de programação.

A principal contribuição desta pesquisa é a proposta de mecanismo de *feedback* para auxiliar estudantes durante a resolução de exercícios de programação. Outras contribuições consistem em: a) um modelo conceitual para especificação de mecanismos de *feedback* em ambientes educacionais; b) possibilidade de integração do mecanismo de *feedback* com outros ambientes educacionais de programação; c) análise do impacto do provimento de *feedback* durante o processo de resolução de problemas de programação.

A pesquisa descrita possui algumas limitações que precisam ser destacadas. O uso em larga escala do mecanismo, em diferentes contextos, é necessário para obter maior confiança no seu impacto positivo sobre o aprendizado do estudante. Podem ser utilizadas outras configurações e técnicas existentes para seleção das mensagens de *feedback*, sendo necessárias análises comparativas para identificar quais produzem os melhores resultados. A aplicação de um quase-experimento, diante da impossibilidade de realização de um experimento controlado, também constitui uma limitação à pesquisa.

6.2 Sugestões para Pesquisas Futuras

Nesta seção, são apresentadas sugestões de trabalhos futuros derivados desta dissertação.

- **Aplicação de um número maior de experimentos em contextos distintos**. Utilização do mecanismo de *feedback* em um número maior de turmas, em diferentes contex-

tos, para obter maiores informações sobre a eficácia do *feedback* durante a resolução de exercícios de programação. Melhores resultados podem ser alcançados com a realização de um experimento durante todo o semestre de um curso de programação.

- **Comparar diferentes técnicas de seleção de mensagens de *feedback***. Na pesquisa foram utilizadas técnicas de análise estática de algoritmo por meio de comparação de *string*. Outras técnicas podem ser empregadas, como comparação de grafos.

- **Construção de um serviço de *software* para provimento de *feedback***. A partir do modelo conceitual e do mecanismo de *feedback* criados, é possível construir um serviço de *software* que forneça a ambientes de aprendizagem de programação as mensagens de *feedback* para auxiliar o estudante. Tal solução facilitaria o uso do mecanismo por outros ambientes de aprendizagem.

Referências Bibliográficas

AGUIAR, J. et al. Um estudo sobre a influência das dimensões do modelo felder-silverman na recomendação de recursos educacionais baseada nos estilos de aprendizagem dos alunos. In: *Anais do XXVIII Simpósio Brasileiro de Informática na Educação*. Recife, PE: [s.n.], 2017. p. 1277–1286.

AGUIAR, J.; FECHINE, J.; COSTA, E. Utilização da ferramenta five labs para identificação de traços de personalidade dos estudantes. In: *Anais do XXI Workshop de Informática na Escola*. Maceió, AL: [s.n.], 2015. p. 157–166.

ALA-MUTKA, K. A survey of automated assessment approaches for programming assignments. *Computer Science Education*, Routledge, v. 15, n. 2, p. 83–102, 2005.

ALA-MUTKA, K. M. A survey of automated assessment approaches for programming assignments. *Computer Science Education*, Routledge, v. 15, n. 2, p. 83–102, 2005.

ALVES, F.; JAQUES, P. Um ambiente virtual com feedback personalizado para apoio a disciplinas de programaçã. In: *Anais do XXV Simpósio Brasileiro de Informática na Educação*. Dourados, MS: [s.n.], 2014. p. 1078–1082.

AMORIM, M. C. M. d. S. et al. Aprendizagem e jogos: diálogo com alunos do ensino médio-técnico. *Educação & Realidade*, scielo, v. 41, p. 91–115, 03 2016. ISSN 2175-6236.

ANTRL. 2018. <http://www.antlr.org/>. Acesso: 01-08-2018.

ANZAI, Y.; SIMON, H. A. The theory of learning by doing. *Psychological Review*, US: American Psychological Association, v. 86, n. 2, p. 124–140, mar. 1979. ISSN 1939-1471.

ANZAI, Y.; SIMON, H. A. The theory of learning by doing. v. 86, p. 124–40, 04 1979.

BEN-ARI, M. et al. Perspectives on program animation with jeliot. In: *Revised Lectures on Software Visualization, International Seminar*. London, UK, UK: Springer-Verlag, 2002. p. 31–45. ISBN 3-540-43323-6.

BERGIN, S.; REILLY, R. G. Programming: factors that influence success. In: *SIGCSE*. [S.l.: s.n.], 2005.

BLIKSTEIN, P. Using learning analytics to assess students' behavior in open-ended programming tasks. In: *Proceedings of the 1st International Conference on Learning Analytics and Knowledge*. New York, NY, USA: ACM, 2011. (LAK '11), p. 110–116. ISBN 978-1-4503-0944-8.

CAMPOS, C. P. d.; E., F. C. Boca: um sistema de apoio a competições de programação. In: *Anais dos Workshop sobre Educação do Computação*. Salvador, BA: [s.n.], 2004.

CARDOSO, A. C. S. Feedback em contextos de ensino-aprendizagem on-line. *Linguagens e Diálogos*, v. 2, n. 2, p. 17 – 34, 2011.

CHAVES, J. et al. Mojo: Uma ferramenta para integrar juízes online ao moodle no apoio ao ensino e aprendizagem de programação. v. 5, p. 246, 01 2014.

COOK, T. D.; CAMPBELL, D. T.; SHADISH, W. *Experimental and quasi-experimental designs for generalized causal inference*. [S.l.]: Houghton Mifflin Boston, 2002.

CORBETT, A. T.; ANDERSON, J. R. Locus of feedback control in computer-based tutoring: Impact on learning rate, achievement and attitudes. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. New York, NY, USA: ACM, 2001. (CHI '01), p. 245–252. ISBN 1-58113-327-8.

COSTA, E. et al. Modelos de Feedback para Estudantes em Ambientes Virtuais de Aprendizagem. In: ____. *Jornada de Atualização em Informática na Educação (JAIE 2016)*. Uberlândia: Sociedade Brasileira de Computação, 2016. cap. 1, p. 1 – 38. ISBN 978-85-7669-336-9.

DENNIS, M.; MASTHOFF, J.; MELLISH, C. Adapting Progress Feedback and Emotional Support to Learner Personality. *International Journal of Artificial Intelligence in Education*, v. 26, n. 3, p. 877 – 931, 2016. ISSN 1560-4306.

DOUCE, C.; LIVINGSTONE, D.; ORWELL, J. Automatic test-based assessment of programming: A review. *J. Educ. Resour. Comput.*, ACM, New York, NY, USA, v. 5, n. 3, set. 2005. ISSN 1531-4278. Disponível em: <http://doi.acm.org/10.1145/1163405.1163409>.

FLAVEL, J. H. Metacognition and Cognitive Monitoring: A New Area of Cognitive-Development Inquiry. *American Psychologist*, n. 34, p. 906 – 911, 1979.

GILLELAND, M. *Levenshtein Distance, in Three Flavors*. 2009. Disponível em: <http://people.cs.pitt.edu/~kirk/cs1501/Pruhs/Spring2006/assignments/editdistance/Levenshtein\\%20Distance.htm>.

HATTIE, J.; TIMPERLEY, H. The power of feedback. *Review of Educational Research*, v. 77, n. 1, p. 81–112, 2007.

HEFFERNAN, N. T.; HEFFERNAN, C. L. The assistments ecosystem: Building a platform that brings scientists and teachers together for minimally invasive research on human learning and teaching. *International Journal of Artificial Intelligence in Education*, v. 24, n. 4, p. 470–497, Dec 2014. ISSN 1560-4306.

HONG, J. Guided programming and automated error analysis in an intelligent prolog tutor. *Int. J. Hum.-Comput. Stud.*, Academic Press, Inc., Duluth, MN, USA, v. 61, n. 4, p. 505–534, out. 2004. ISSN 1071-5819.

HYYRYNEN, V. et al. Mypeerreview: An online peer-reviewing system for programming courses. In: *Proceedings of the 10th Koli Calling International Conference on Computing Education Research*. New York, NY, USA: ACM, 2010. (Koli Calling '10), p. 94–99. ISBN 978-1-4503-0520-4.

INVENTADO, P. S.; SCUPELLI, P. A data-driven methodology for producing producing online learning system design patterns. In: *Proceedings of the 22Nd Conference on Pattern Languages of Programs*. USA: The Hillside Group, 2015. (PLoP '15), p. 26:1–26:17. ISBN 978-1-941652-03-9.

INVENTADO, P. S.; SCUPELLI, P. Design patterns for helping students to learn to represent math problems in online learning systems. In: *Proceedings of the 21st European Conference on Pattern Languages of Programs*. New York, NY, USA: ACM, 2016. (EuroPlop '16), p. 28:1–28:19. ISBN 978-1-4503-4074-8.

INVENTADO, P. S.; SCUPELLI, P. Design patterns for math problems and learning support in online learning systems. In: *Proceedings of the 10th Travelling Conference on Pattern Languages of Programs*. New York, NY, USA: ACM, 2016. (VikingPLoP '16), p. 8:1–8:16. ISBN 978-1-4503-4200-1.

INVENTADO, P. S.; SCUPELLI, P. Patterns for learning-support design in math online learning systems. In: *Proceedings of the 23rd Conference on Pattern Languages of Programs*. USA: The Hillside Group, 2016. (PLoP '16), p. 18:1–18:13.

INVENTADO, P. S. et al. Feedback design patterns for math online learning systems. In: *Proceedings of the 22Nd European Conference on Pattern Languages of Programs*. New York, NY, USA: ACM, 2017. (EuroPLoP '17), p. 31:1–31:15. ISBN 978-1-4503-4848-5.

JAIN, D. et al. Selection and ranking of e-learning websites using weighted distance-based approximation. *Journal of Computers in Education*, v. 3, n. 2, p. 193–207, Jun 2016. ISSN 2197-9995.

KEUNING, H.; HEEREN, B.; JEURING, J. Strategy-based feedback in a programming tutor. In: *Proceedings of the Computer Science Education Research Conference*. New York, NY, USA: ACM, 2014. (CSERC '14), p. 43–54. ISBN 978-1-4503-3347-4.

KEUNING, H.; JEURING, J.; HEEREN, B. A systematic literature review of automated feedback generation for programming exercises. *ACM Trans. Comput. Educ.*, ACM, New York, NY, USA, v. 19, n. 1, p. 3:1–3:43, set. 2018. ISSN 1946-6226. Disponível em: <http://doi.acm.org/10.1145/3231711>.

KÖLLING, M. et al. The bluej system and its pedagogy. *Computer Science Education*, Routledge, v. 13, n. 4, p. 249–268, 2003.

KYRILOV, A.; NOELLE, D. C. Do students need detailed feedback on programming exercises and can automated assessment systems provide it? *J. Comput. Sci. Coll.*, Consortium for Computing Sciences in Colleges, USA, v. 31, n. 4, p. 115–121, abr. 2016. ISSN 1937-4771. Disponível em: <http://dl.acm.org/citation.cfm?id=2904127.2904147>.

LAHTINEN, E.; ALA-MUTKA, K.; JäRVINEN, H.-M. A study of the difficulties of novice programmers. In: *Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*. New York, NY, USA: ACM, 2005. (ITiCSE '05), p. 14–18. ISBN 1-59593-024-8.

LEVY, Y.; ELLIS, T. A guide for novice researchers on experimental and quasi-experimental studies in information systems research. v. 6, p. 151–161, 01 2011.

LOUDEN, K. *Compiladores - Princípios e Práticas*. [S.l.]: Pioneira Thomson Learning, 2004. ISBN 9788522104222.

MAAS, A. et al. Offering verified credentials in massive open online courses: Moocs and technology to advance learning and learning research (ubiquity symposium). *Ubiquity*, ACM, New York, NY, USA, v. 2014, n. May, p. 2:1–2:11, may 2014. ISSN 1530-2180.

MALONEY, J. et al. The scratch programming language and environment. *Trans. Comput. Educ.*, ACM, New York, NY, USA, v. 10, n. 4, p. 16:1–16:15, nov. 2010. ISSN 1946-6226.

MANSO, A.; MARQUES, C. G.; DIAS, P. Portugol ide v3.x: A new environment to teach and learn computer programming. In: *IEEE EDUCON 2010 Conference*. [S.l.: s.n.], 2010. p. 1007–1010. ISSN 2165-9559.

MARCELINO, M.; MIHAYLOV, T.; MENDES, A. H-sicas, a handheld algorithm animation and simulation tool to support initial programming learning. In: *2008 38th Annual Frontiers in Education Conference*. [S.l.: s.n.], 2008. p. T4A–7–T4A–12. ISSN 0190-5848.

MARINI, J. A. da S. Metacognição e leitura. *Psicologia Escolar e Educacional*, scielo, v. 10, p. 343 – 345, 12 2006. ISSN 1413-8557.

MORY, E. H. Feedback research revisited. In: *Handbook of research for educational communications and technology*. [S.l.: s.n.], 2003.

NARCISS, S. Feedback strategies for interactive learning tasks. In: ____. *Handbook of Research on Educational Communications and Technology*. [S.l.]: Routledge, 2007. cap. Chapter 11, p. 125–144.

NARCISS, S. Designing and Evaluating Tutoring Feedback Strategies for digital learning environments on the basis of the Interactive Tutoring Feedback Model. *Digital Education Review*, n. 23, p. 7 – 26, 2013.

NOSCHANG, L. F. et al. Portugol studio: Uma ide para iniciantes em programação. In: *Anais do Congresso Anual da Sociedade Brasileira de Computação*. Brasília: [s.n.], 2014. p. 535–545.

OSTROW, K. S.; HEFFERNAN, N. T. Testing the multimedia principle in the real world: A comparison of video vs. text feedback in authentic middle school math assignments. In: *EDM*. [S.l.: s.n.], 2014.

PAES, R. et al. Ferramenta para a avaliação de aprendizado de alunos em programação de computadores. In: *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*. Campinas, SP: [s.n.], 2013. p. 203–212.

PANTALEÃO, E.; AMARAL, L.; SILVA, G. B. e. Uma abordagem baseada no ambiente robocode para ensino de programação no ensino médio. *Revista Brasileira de Informática na Educação*, v. 25, n. 03, p. 95, 2017. ISSN 2317-6121. Disponível em: <http://br-ie.org/pub/index.php/rbie/article/view/6452>.

PETERS, O.; KÖRNDLE, H.; NARCISS, S. Effects of a formative assessment script on how vocational students generate formative feedback to a peer's or their own performance. *European Journal of Psychology of Education*, v. 33, n. 1, p. 117–143, Jan 2018. ISSN 1878-5174.

PETIT, J.; GIMÉNEZ, O.; ROURA, S. Jutge.org: An educational programming judge. In: *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*. New York, NY, USA: ACM, 2012. (SIGCSE '12), p. 445–450. ISBN 978-1-4503-1098-7.

POLITZ, J. G.; KRISHNAMURTHI, S.; FISLER, K. Captainteach: A platform for in-flow peer review of programming assignments. In: *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education*. New York, NY, USA: ACM, 2014. (ITiCSE '14), p. 332–332. ISBN 978-1-4503-2833-3.

PRICE, T. W.; DONG, Y.; LIPOVAC, D. isnap: Towards intelligent tutoring in novice programming environments. In: *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. New York, NY, USA: ACM, 2017. (SIGCSE '17), p. 483–488. ISBN 978-1-4503-4698-6.

RAZZAQ, L.; HEFFERNAN, N. T. Hints: Is it better to give or wait to be asked? In: ALEVEN, V.; KAY, J.; MOSTOW, J. (Ed.). *Intelligent Tutoring Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. p. 349–358.

RIVERS, K.; KOEDINGER, K. R. Data-driven hint generation in vast solution spaces: a self-improving python programming tutor. *International Journal of Artificial Intelligence in Education*, v. 27, n. 1, p. 37–64, Mar 2017. ISSN 1560-4306.

ROBERTS, E. An overview of minijava. In: *Proceedings of the Thirty-second SIGCSE Technical Symposium on Computer Science Education*. New York, NY, USA: ACM, 2001. (SIGCSE '01), p. 1–5. ISBN 1-58113-329-4. Disponível em: <http://doi.acm.org/10.1145/364447.364525>.

SANTOS, C. P. et al. Desafio de programação para meninas do ensino médio: Um relato de experiência. In: *Anais do Workshop de Informática na Escola*. Recife, PE: [s.n.], 2017. p. 137–144.

SANTOS, J.; RIBEIRO, A. Jonline: proposta preliminar de juiz online didático para ensino de programaçã. In: *Anais do XXII Simpósio Brasileiro de Informática na Educação*. Aracaju, SE: [s.n.], 2011.

SEFFRIN, H. et al. Dicas inteligentes no sistema tutor inteligente pat2math. In: *Anais dos Simpósio Brasileiro de Informática na Educação*. Rio de Janeiro, RJ: [s.n.], 2012.

SHUTE, V. J. Focus on Formative Feedback. *Review of Educational Research*, v. 78, n. 1, p. 153 – 189, 2008.

SILVA, P. et al. Um mapeamento sistemático sobre iniciativas brasileiras em ambientes de ensino de programação. In: *Anais do XXVI Simpósio Brasileiro de Informática na Educação*. Maceió, AL: [s.n.], 2015. p. 367–375.

SILVA, T. R. Desenvolvendo a programação de jogos digitais no ensino médio: um relato de experiência utilizando a ferramenta construct2. In: *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*. Recife, PE: [s.n.], 2017. p. 1142–1151.

SINGH, R.; GULWANI, S.; SOLAR-LEZAMA, A. Automated feedback generation for introductory programming assignments. In: *Proceedings of the 34th ACM SIGPLAN Conference on Programming Language Design and Implementation*. New York, NY, USA: ACM, 2013. (PLDI '13), p. 15–26. ISBN 978-1-4503-2014-6.

SMITH, B.; ENG, M. Moocs: A learning journey. In: CHEUNG, S. K. S. et al. (Ed.). *Hybrid Learning and Continuing Education*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. p. 244–255. ISBN 978-3-642-39750-9.

SOBRINHA, V. dos S. et al. Plataforma para auxílio ao ensino de programação e robótica pedagógica. *Revista Principia - Divulgação Científica e Tecnológica do IFPB*, v. 1, n. 31, p. 104–112, 2016. ISSN 2447-9187.

STRIJBOS, J.-W.; PAT-EL, R. J.; NARCISS, S. Structural validation of a feedback perceptions questionnaire. In: *Proceedings of the 9th International Conference of the Learning Sciences - Volume 2*. [S.l.]: International Society of the Learning Sciences, 2010. (ICLS '10), p. 334–335.

TENNYSON, M. F.; BECK, M. A study of knowledge retention in introductory programming courses. *J. Comput. Sci. Coll.*, Consortium for Computing Sciences in Colleges, USA, v. 33, n. 4, p. 13–20, abr. 2018. ISSN 1937-4771.

WENGER, E. *Artificial Intelligence and Tutoring Systems: Computational and Cognitive Approaches to the Communication of Knowledge*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1987. ISBN 0-934613-26-5.

XINOGALOS, S.; SATRATZEMI, M.; DAGDILELIS, V. An introduction to object-oriented programming with a didactic microworld: objectkarel. *Computers & Education*, v. 47, n. 2, p. 148 – 171, 2006. ISSN 0360-1315.

Apêndice A

Questões de Programação Usadas para Criação do *Feedback*

A seguir, são apresentados os enunciados das questões de programação solucionadas pelos estudantes, citadas na Seção 5.4.2.

1. **Avaliação Escolar**

   Enunciado: Elabore um algoritmo que leia o nome, a nota da avaliação 1 e a nota da avaliação 2 de um aluno. Ao final, imprima o nome do aluno, suas notas, a média aritmética e uma das mensagens: Aprovado, Reprovado ou em Prova Final (a média é 7,0 para aprovação, menor que 3,0 para reprovação e as demais em prova final).

2. **Distribuição de turmas**

   Enunciado:Sabe-se que a direção de uma determinada escolinha faz a distribuição de seus alunos de acordo com as idades dos mesmos. Dessa forma, os alunos são distribuídos nas seguintes turmas de acordo com a classificação a seguir: i) TURMA A com alunos de 4 a 5 anos; ii) TURMA B com alunos de 6 a 8 anos; iii) TURMA C com alunos de de 9 a 10 anos; iv) SEM TURMAS abaixo de 4 anos e acima de 10 anos. Desenvolva um algoritmo que leia a idade de uma única criança e informe em qual turma a mesma irá ter aulas. O algoritmo deve se preocupar em responder para o usuário que a escolinha não possui turmas para a criança caso a mesma tenha menos que 4 anos ou mais que 10 anos.

3. **Aumento Salarial**

Enunciado: Uma empresa decidiu conceder um aumento de salários a seus funcionários de acordo com a seguinte distribuição: quem possui o salário atual abaixo 400,00 reais receberá 15% de aumento; quem possui o salário atual de 400,00 até 700,00 reais receberá 10% de aumento; aqueles que possuem o salário atual de 700,00 até 1000,00 reais receberão 7% de aumento; quem possui o salário atual de 1000,00 até 1600,00 reais receberá 3% de aumento; aqueles que possuem o salário acima de 1600,00 reais não receberão aumento. Escreva um algoritmo que leia o salário atual de um funcionário e escreva o percentual de seu aumento e o valor do salário corrigido a partir desse aumento.

4. **Prêmio do Milhão**

   Enunciado: Alice e Bia criaram uma página na Internet com informações sobre o Macaco-prego-de-peito-amarelo, uma espécie em extinção. A página mostra como todos podem ajudar a manter o habitat natural para evitar que a espécie seja extinta. Uma empresa gostou tanto da iniciativa de Alice e Bia que prometeu doar um prêmio para que as duas amigas possam realizar outras iniciativas semelhantes. A empresa decidiu que o prêmio seria dado quando a soma do número de acessos à página chegasse a 1 milhão. Dada a lista de acessos diários que ocorreram à página de Alice e Bia, escreva um programa para determinar quantos dias foram necessários para a soma dos acessos chegar a 1 milhão e as amigas ganharem o prêmio.

5. **Móbile**

   Enunciado: O móbile na sala da Maria é composto de três hastes exatamente como na figura abaixo. Para que ele esteja completamente equilibrado, com todas as hastes na horizontal, os pesos das quatro bolas A, B, C e D têm que satisfazer todas as seguintes três condições: A = B+C+D; e D = B+C; e B = C. Nesta tarefa, dados os pesos das quatro bolas, seu programa deve decidir se o móbile está ou não completamente equilibrado.

Apêndice B

Lista de Exercícios para Verificar Habilidade de Correção de Erros

A seguir, é apresentada a lista de exercícios realizada pelos estudantes citada na Seção 5.4.3.

TODAS as questões a seguir possuem a descrição de um problema e um algoritmo com a sua solução, porém o algoritmo possui erros ou está incompleto. Você deve:

- Identificar quais linhas possuem erros;
- Explicar o motivo do erro ou o motivo de estar incompleto;
- Corrigir o algoritmo.

1) Dado o problema: Faça um algoritmo que leia um número inteiro e mostre na tela seu antecessor e seu sucessor.

	Solução com erro	Insira a solução com a correção
1	Algoritmo "semnome"	
2		
3	Var	
4	// Seção de Declarações das variáveis	
5	n, antecessor, sucessor: inteiro	
6		
7	Inicio	
8		
9	escreva("Digite um número: ")	
10	leia(n)	
11	antecessor <- n-1	
12	sucessor <- n-1	
13	escreval("Antecessor: ", antecessor)	
14	escreval("Sucessor: ", n)	
15		
16	Fimalgoritmo	

Qual as linhas do algoritmo que possuem erros?

Explique qual o motivo do algoritmo estar errado ou incompleto.

2) Dado o problema: Elabora um programa que leia uma temperatura em graus Celsius e apresente-a convertida em graus Fahrenheit. A fórmula de conversão é:

$$F = \frac{9}{5}.C + 32$$

Onde F é a temperatura em Fahrenheit e C a temperatura em graus Celsius.

	Solução com erro	Insira a solução com a correção
1	Algoritmo "semnome"	
2		
3	Var	
4	c,f: real	
5		
6	Inicio	
7		
8	escreva("Digite uma temperatura em graus	
9	Celsius: ")	
10	leia(c)	
11	f <- 9/(5*c)+32	
12	escreval("Temperatura em Fahrenheit: ", f)	
13		
14	Fimalgoritmo	
15		
16		

Qual as linhas do algoritmo que possuem erros?

Explique qual o motivo do algoritmo estar errado ou incompleto.

3) Dado o problema: Faça um algoritmo que leia dois números inteiros e mostra na tela esses números em ordem crescente.

	Solução com erro	Insira a solução com a correção
1	Algoritmo "semnome"	
2		
3	Var	
4	n1,n2: inteiro	
5		
6	Inicio	
7	escreva("Digite primeiro número: ")	
8	leia(n1)	
9	escreva("Digite segundo número: ")	
10	leia(n2)	
11	se n1>n2 entao	
12	escreva("Ordem crescente: ", n2, n1)	
13	fimse	
14	se n2<n1 entao	
15	escreva("Ordem crescente: ", n1, n2)	
16	fimse	
17		
18	Fimalgoritmo	

Qual as linhas do algoritmo que possuem erros?

Explique qual o motivo do algoritmo estar errado ou incompleto.

4) Dado o problema: Construir um algoritmo que leia três valores e os imprima em ordem crescente.

	Solução com erro	Insira a solução com a correção
1	Algoritmo "semnome"	
2		
3	Var	
4	n1,n2,n3: inteiro	
5		
6	Inicio	
7	escreva("Digite primeiro número: ")	
8	leia(n1)	
9	escreva("Digite segundo número: ")	
10	leia(n2)	
11	escreva("Digite terceiro número: ")	
12	leia(n3)	
13	se (n1<n2) e (n1<n3) entao	
14	escreva(n1)	
15	se n2<n3 entao	
16	escreva(n3, n2)	
17	senao	
18	escreva(n2, n3)	
19	fimse	
20	fimse	
21	se (n2<n1) e (n2<n3) entao	
22	escreva(n2)	
23	se n1<n3 entao	
24	escreva(n1, n3)	
25	senao	
26	escreva(n3, n1)	
27	fimse	
28	fimse	
29	se (n3<n1) e (n3<n2) entao	
30	escreva(n3)	
31	se n1<n2 entao	
32	escreva(n1, n2)	
33	senao	
34	escreva(n2, n1)	
35	fimse	
36	fimse	
37		
38	Fimalgoritmo	

Qual as linhas do algoritmo que possuem erros?

Explique qual o motivo do algoritmo estar errado ou incompleto.

5) Dado o problema: Faça um algoritmo que leia um número N e mostre todos os números múltiplos de 5, no intervalo de 1 a N.

	Solução com erro	Insira a solução com a correção
1	Algoritmo "semnome"	
2		
3	Var	
4	n: inteiro	
5	num: inteiro	
6	Inicio	
7		
8	escreva("Digite um número: ")	
9	leia(n)	
10	num <- 0	
11	repita	
12	num <- num + 1	
13	escreval(num)	
14	ate num=n	
15		
16	Fimalgoritmo	

Qual as linhas do algoritmo que possuem erros?

Explique qual o motivo do algoritmo estar errado ou incompleto.

6) Dado o problema: Elabore um algoritmo que mostre todos os números de 100 a 150, e ao final a soma de todos os números nesse intervalo.

	Solução com erro	Insira a solução com a correção
1	Algoritmo "semnome"	
2		
3	Var	
4	n: inteiro	
5	soma: inteiro	
6	Inicio	
7	soma <- 0	
8	para n de 100 ate 150 faca	
9	escreval(n)	
10	fimpara	
11	escreva("Soma dos números: ", soma)	
12	Fimalgoritmo	
13		
14		
15		
16		

Qual as linhas do algoritmo que possuem erros?

Explique qual o motivo do algoritmo estar errado ou incompleto.

7) Dado o problema: Elabore um algoritmo que leia 4 nomes e depois mostre-os na ordem inversa em que foram digitados.

	Solução com erro	Insira a solução com a correção
1	Algoritmo "semnome"	
2		
3	Var	
4	nomes: vetor [1..4] de caracter	
5	n: inteiro	
6		
7	Inicio	
8	escreval("Digite os nomes dos alunos: ")	
9	para n de 1 ate 4 faca	
10	leia(nomes[n])	
11	fimpara	
12	escreval("Nomes digitados na ordem	
13	inversa: ")	
14	para n de 4 ate 1 faca	
15	escreval(nomes[n])	
16	fimpara	
17		
18	Fimalgoritmo	

Qual as linhas do algoritmo que possuem erros?

Explique qual o motivo do algoritmo estar errado ou incompleto.

8) Dado o problema: Elabore um algoritmo que preencha uma matriz 5 x 3 com números inteiros aleatórios, use randi(10), mostre a matriz na tela e em seguida calcule e mostre a soma dos elementos da matriz.

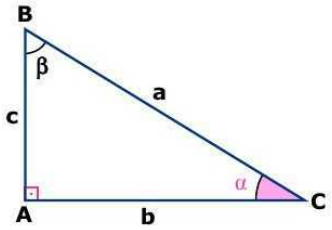
	Solução com erro	Insira a solução com a correção
1	Algoritmo "semnome"	
2		
3	Var	
4	matriz: vetor [1..5, 1..3] de inteiro	
5	i,j: inteiro	
6	soma: inteiro	
7		
8	Inicio	
9		
10	soma <- 0	
11	escreval("Digite os nomes dos alunos: ")	
12	para i de 1 ate 5 faca	
13	para j de 1 ate 3 faca	
14	matriz[i,j] <- randi(10)	
15	fimpara	
16	fimpara	
17		
18	escreval("Matriz gerada: ")	
19		

20	para i de 1 ate 5 faca	
21	para j de 1 ate 3 faca	
22	escreva(" ", matriz[i,j])	
23	fimpara	
24	escreval("")	
25	fimpara	
26		
27	escreva("Soma dos elementos da matriz: ",	
28	soma)	
29		
30	Fimalgoritmo	

Qual as linhas do algoritmo que possuem erros?

Explique qual o motivo do algoritmo estar errado ou incompleto.

9) Dado o problema: Faça um algoritmo que receba como entradas as medidas dos catetos de um triângulo retângulo, e calcule dentro de uma <u>função</u> a medida da hipotenusa retornando esse valor para o algoritmo principal para que ele seja mostrado na tela. Use passagem de parâmetro por valor.



$a^2 = b^2 + c^2$

	Solução com erro	Insira a solução com a correção
1	Algoritmo "semnome"	
2	Var	
3	b,c: inteiro	
4	a: real	
5	Inicio	
6	escreval("Digite o valor do 1º cateto: ")	

7	leia(b)	
8	escreval("Digite o valor do 2º cateto: ")	
9	leia(c)	
10		
11	a <- raizq(b*b + c*c)	
12		
13	escreval("Hipotenusa: ", a)	
14		
15	Fimalgoritmo	

Qual as linhas do algoritmo que possuem erros?

Explique qual o motivo do algoritmo estar errado ou incompleto.

10) Dado o problema: Elabore um algoritmo para informar se um número digitado pelo usuário é primo ou não. Use uma <u>função</u> para fazer a verificação e retornar um valor do tipo caracter que pode ser "é primo" ou "não é primo". Use passagem de parâmetro por valor.

	Solução com erro	Insira a solução com a correção
1	Algoritmo "semnome"	
2		
3	Var	
4	num: inteiro	
5	msg: caracter	
6		
7	funcao eh_primo(n: inteiro): caracter	
8	var	
9	i: inteiro	
10	inicio	
11	para i de 2 ate (n-1) faca	
12	se (n mod i = 0) entao	
13	escreva(n, " ", i)	
14	RETORNE "não é primo"	
15	fimse	
16	fimpara	
17	RETORNE "é primo"	

18	fimfuncao	
19		
20		
21	Inicio	
22		
23	escreval("Digite um número: ")	
24	leia(num)	
25	msg <- eh_primo(num)	
26	escreval(msg)	
27		
28	Fimalgoritmo	
29		

Qual as linhas do algoritmo que possuem erros?

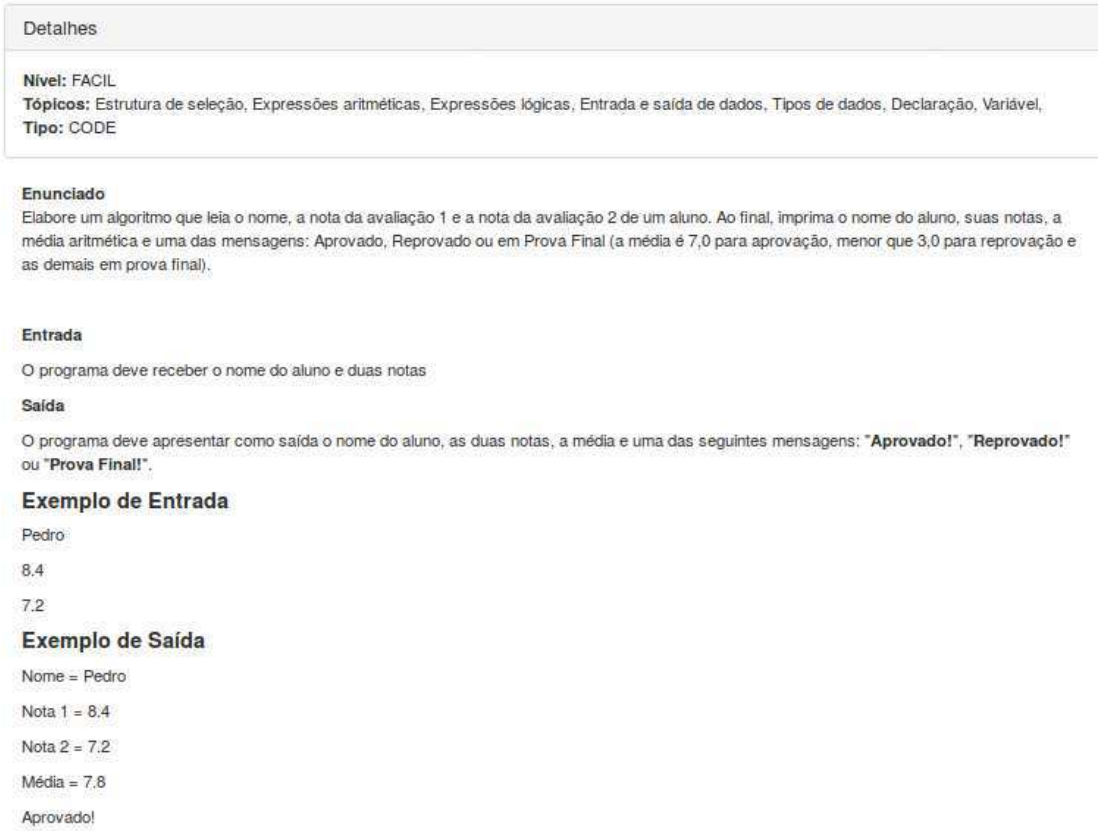
Explique qual o motivo do algoritmo estar errado ou incompleto.

Apêndice C

Questões usadas no Experimento

Questões utilizadas no experimento citado na Seção 5.5.

Figura C.1. **Problema: Aprovação Escolar**.



Detalhes

Nível: FACIL
Tópicos: Estrutura de seleção, Expressões aritméticas, Expressões lógicas, Entrada e saída de dados, Tipos de dados, Declaração, Variável,
Tipo: CODE

Enunciado
Elabore um algoritmo que leia o nome, a nota da avaliação 1 e a nota da avaliação 2 de um aluno. Ao final, imprima o nome do aluno, suas notas, a média aritmética e uma das mensagens: Aprovado, Reprovado ou em Prova Final (a média é 7,0 para aprovação, menor que 3,0 para reprovação e as demais em prova final).

Entrada

O programa deve receber o nome do aluno e duas notas

Saída

O programa deve apresentar como saída o nome do aluno, as duas notas, a média e uma das seguintes mensagens: "**Aprovado!**", "**Reprovado!**" ou "**Prova Final!**".

Exemplo de Entrada

Pedro

8.4

7.2

Exemplo de Saída

Nome = Pedro

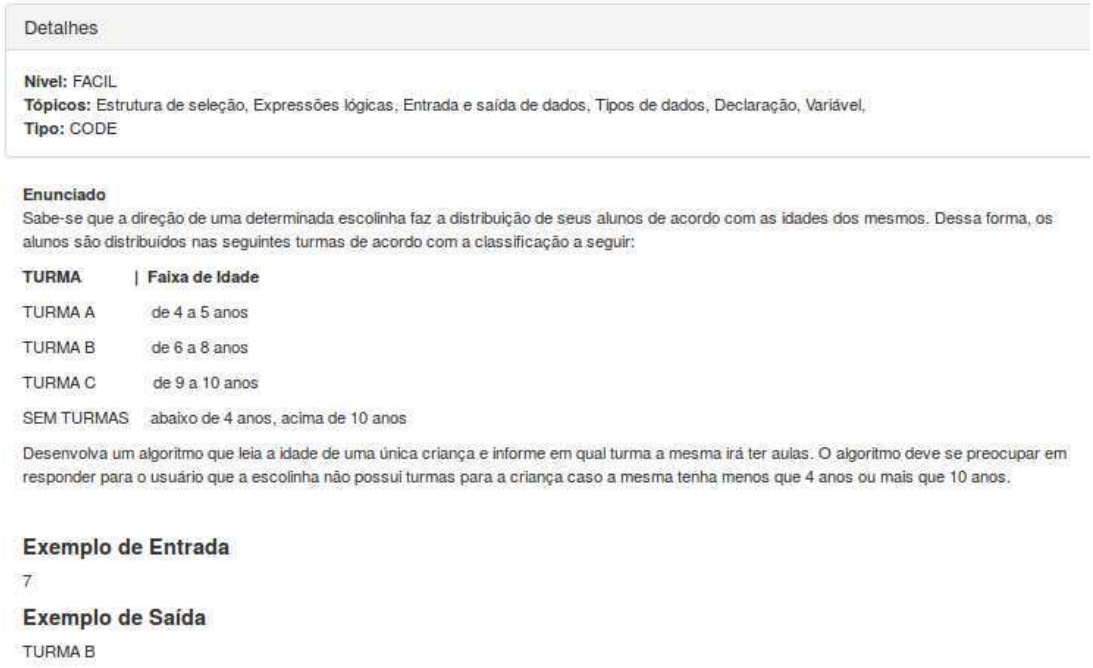
Nota 1 = 8.4

Nota 2 = 7.2

Média = 7.8

Aprovado!

Figura C.2. **Problema: Turmas da Escolinha**.



Detalhes

Nível: FACIL
Tópicos: Estrutura de seleção, Expressões lógicas, Entrada e saída de dados, Tipos de dados, Declaração, Variável,
Tipo: CODE

Enunciado
Sabe-se que a direção de uma determinada escolinha faz a distribuição de seus alunos de acordo com as idades dos mesmos. Dessa forma, os alunos são distribuídos nas seguintes turmas de acordo com a classificação a seguir:

TURMA	Faixa de Idade
TURMA A	de 4 a 5 anos
TURMA B	de 6 a 8 anos
TURMA C	de 9 a 10 anos
SEM TURMAS	abaixo de 4 anos, acima de 10 anos

Desenvolva um algoritmo que leia a idade de uma única criança e informe em qual turma a mesma irá ter aulas. O algoritmo deve se preocupar em responder para o usuário que a escolinha não possui turmas para a criança caso a mesma tenha menos que 4 anos ou mais que 10 anos.

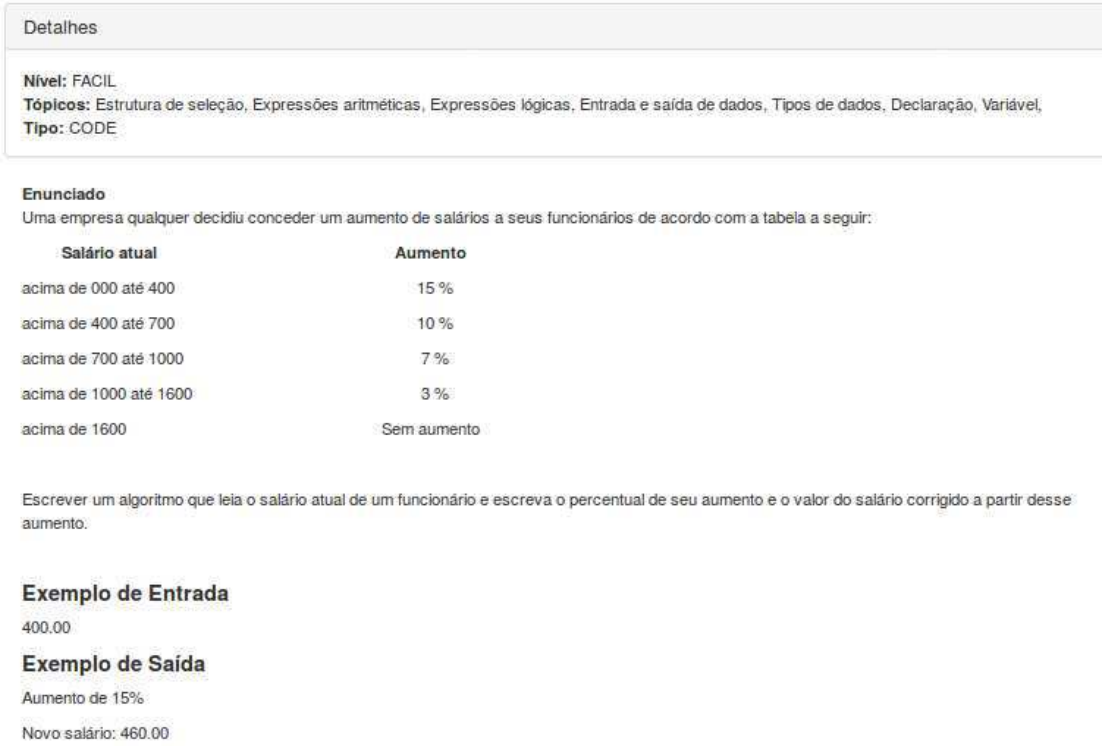
Exemplo de Entrada

7

Exemplo de Saída

TURMA B

Figura C.3. **Problema: Aumento de Salário**.



Detalhes

Nível: FACIL
Tópicos: Estrutura de seleção, Expressões aritméticas, Expressões lógicas, Entrada e saída de dados, Tipos de dados, Declaração, Variável,
Tipo: CODE

Enunciado
Uma empresa qualquer decidiu conceder um aumento de salários a seus funcionários de acordo com a tabela a seguir:

Salário atual	Aumento
acima de 000 até 400	15 %
acima de 400 até 700	10 %
acima de 700 até 1000	7 %
acima de 1000 até 1600	3 %
acima de 1600	Sem aumento

Escrever um algoritmo que leia o salário atual de um funcionário e escreva o percentual de seu aumento e o valor do salário corrigido a partir desse aumento.

Exemplo de Entrada

400.00

Exemplo de Saída

Aumento de 15%

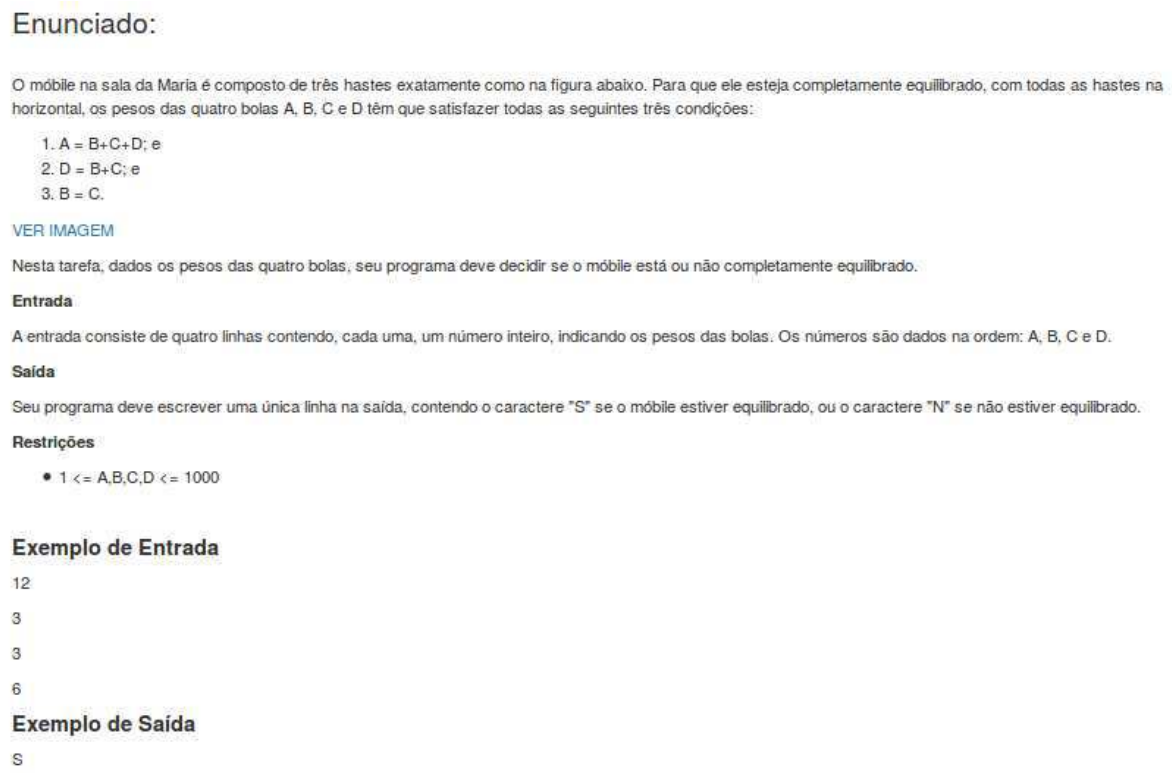
Novo salário: 460.00

Apêndice D

Questões usadas no Pré-teste e Pós-teste do Experimento

Questões utilizadas no experimento citado na Seção 5.5.

Figura D.1. **Problema:Móbile**.



Enunciado:

O móbile na sala da Maria é composto de três hastes exatamente como na figura abaixo. Para que ele esteja completamente equilibrado, com todas as hastes na horizontal, os pesos das quatro bolas A, B, C e D têm que satisfazer todas as seguintes três condições:

1. A = B+C+D; e
2. D = B+C; e
3. B = C.

VER IMAGEM

Nesta tarefa, dados os pesos das quatro bolas, seu programa deve decidir se o móbile está ou não completamente equilibrado.

Entrada

A entrada consiste de quatro linhas contendo, cada uma, um número inteiro, indicando os pesos das bolas. Os números são dados na ordem: A, B, C e D.

Saída

Seu programa deve escrever uma única linha na saída, contendo o caractere "S" se o móbile estiver equilibrado, ou o caractere "N" se não estiver equilibrado.

Restrições

- 1 <= A,B,C,D <= 1000

Exemplo de Entrada

12

3

3

6

Exemplo de Saída

S

Figura D.2. **Problema:Sedex**.

Enunciado:

A Copa do Mundo de 2010 será realizada na Africa do Sul. Bolas de futebol são muito fáceis de transportar, já que elas saem das fábricas vazias e só são enchidas somente pelas lojas ou pelos consumidores finais. Infelizmente o mesmo não pode ser dito das bolas de boliche. Como elas são completamente sólidas, elas só podem ser transportadas embaladas uma a uma, em caixas separadas.

A SBC -- Só Boliche Cascavel -- é uma fábrica de bolas de boliche que trabalha somente através de encomendas e envia todas as bolas por SEDEX. Como as bolas têm tamanhos diferentes, a SBC tem vários tamanhos de caixas diferentes para transportá-las.

Escreva um programa que, dado o diâmetro de uma bola e as 3 dimensões de uma caixa (altura, largura e profundidade), diz se a bola de boliche cabe dentro da caixa ou não.

Entrada

A primeira linha da entrada contém um inteiro N que indica o diâmetro da bola de boliche. A segunda linha da entrada contém 3 números inteiros separados por um espaço cada: a altura A, seguida da largura L e da profundidade P.

Saída

Seu programa deve imprimir uma única linha, contendo a letra 'S' caso a bola de boliche caiba dentro da caixa ou 'N' caso contrário.

Restrições

- 1 <= N <= 10^4
- 1 <= A <= 10^4
- 1 <= L <= 10^4
- 1 <= P <= 10^4

Exemplo de Entrada

3

2 3 5

--

5

5 5 5