Estudo sobre o Sequenciamento Inteligente e Adaptativo de Enunciados em Programação de Computadores

Carolina Moreira[1], **Andrey Ricardo Pimentel**[1], **Eleandro Maschio**[2]

[1]Programa de Pós-Graduação em Informática
Universidade Federal do Paraná (UFPR)
Curitiba – PR – Brasil

[2]Coordenação do Curso de Tecnologia em Sistemas para Internet
Universidade Tecnológica Federal do Paraná (UTFPR)
Guarapuava – PR – Brasil

{cmoliveira,andrey}@inf.ufpr.br, eleandrom@utfpr.edu.br

Abstract. *This article details an approach to deal with intelligent and adaptive exercise ordering, considering the student dynamic modeling in Computer Programming domain. The methodology uses genetic graphs as a basis for the internal representation. Thus, overlaying aspects highlight the progress of the learner compared to the domain knowledge, and to the individual contribution of each exercise. Then is proposed a heuristic search process that indicates capabilities to be developed by the student, as also an adaptive exercise ordering that contributes in this evolution. The model update and the extraction of useful information about learning, lastly, are analyzed.*

Resumo. *O presente artigo detalha uma abordagem para tratar do sequenciamento inteligente e adaptativo de enunciados, considerando a modelagem dinâmica do aprendiz, no domínio de Programação de Computadores. A metodologia utiliza grafos genéticos como base para a representação interna. Com isso, aspectos de sobreposição destacam o progresso do aprendiz frente ao conhecimento do domínio, além da contribuição individual de cada enunciado. Propõe-se, então, um processo de busca heurística que indica capacidades a serem desenvolvidas pelo aprendiz, bem como o sequenciamento adaptativo de enunciados que contribuam nessa evolução. A atualização do modelo e a extração de informações úteis sobre a aprendizagem, por fim, são analisadas.*

1. Introdução

O ensino em Programação de Computadores tem sido um desafio por décadas [Norvig 2001]. Embora se trate de uma disciplina fundamental para qualquer curso da área de Ciência da Computação, o aprendizado dela mostra-se bastante difícil para muitos iniciantes. Isso se corrobora pelo alto índice de desistência dessas disciplinas e, até mesmo, de evasão dos cursos [Miliszewska and Tan 2007]. Assim, há a necessidade de encontrar meios que tornem o assunto menos impactante para aqueles que iniciam na área.

Além disso, as pessoas que mostram interesse em aprender a programar são bastante diversas. Elas se distinguem em aspectos como idade, sexo, nível de escolaridade,

DOI: 10.5753/cbie.wcbie.2015.1320

maneira como aprendem e aptidão para resolver problemas lógicos. Como consequência, é extremamente difícil criar um material único que atenda a toda essa diversidade de perfis.

Dentre os vários métodos sugeridos para melhorar o processo de ensino, a adoção de aulas individuais tem se provado eficaz [Weragama 2013]. No entanto, aulas individuais com instrutores humanos exigem proporcionalmente mais recursos e, logo, tornam-se impraticáveis de serem aplicadas em contextos com muitos aprendizes, como no ensino superior.

A criação e difusão dos Sistemas Tutores Inteligentes (STIs) se deu como alternativa viável em resposta a essas necessidades [Nwana 1990, Turine et al. 1994]. Tal abordagem tenta se aproximar das aulas individuais, considerando as características particulares do aprendiz para alterar a interação. Nesse sentido, a modelagem do aprendiz responsabiliza-se por armazenar a representação atual do conhecimento e do desempenho do indivíduo sobre os tópicos tutorados. A partir dela, o sistema consegue definir o nível de dificuldade relativo de um conteúdo frente ao conhecimento do aprendiz, como também propor sugestões personalizadas. Em mais alto nível, a inteligência de um sistema pode conseguir reconhecer as dificuldades de um aprendiz e prover diagnóstico para um tutoramento efetivo [Desmarais and d. Baker 2012].

Paralelamente, qualquer ambiente de ensino que utilize enunciados precisa, de alguma maneira, determinar a sequência em que eles serão oferecidos ao aprendiz. Alguns sistemas apresentam os enunciados em ordem predefinida [Weber and Brusilovsky 2001]. Outros, permitem que os aprendizes selecionem o próximo enunciado a partir de uma lista estática de alternativas [Weber and Möllenberg 1995]. Em ambas as situações, a escolha do enunciado não considera as capacidades atuais do aprendiz.

Levando em consideração a modelagem do aprendiz, um STI tem a capacidade de sugerir enunciados próprios ao nível de conhecimento do indivíduo. Isso se mostra positivo, pois enunciados com dificuldade muito abaixo desse nível podem causar entediamento, e do contrário, sendo muito acima, provocar desmotivação [Pimentel and Direne 1998, Soldado and du Boulay 1995]. Ambos os casos são passíveis do abandono da atividade proposta.

Dessa forma, o sequenciamento adaptativo de enunciados busca auxiliar diferentes perfis de aprendizes, independentemente do nível de conhecimento. Pode-se diversificar a ordem e mesmo a quantidade de enunciados com a intenção de potencializar o processo de ensino-aprendizagem para cada aprendiz em particular.

Diante do exposto, a pesquisa tem o objetivo de atuar no sequenciamento adaptativo de enunciados frente a uma modelagem dinâmica do aprendiz, no contexto de Programação de Computadores. Para isso, propõe-se uma continuidade imediata aos estudos de [Maschio and Direne 2013, Maschio 2013], que modela o processo de aquisição de conhecimento em ambientes inteligentes por meio de grafos genéticos.

2. Resenha Literária

O referencial teórico da pesquisa alicerça-se em três temas principais, detalhados nas seções seguintes: modelo de sobreposição e grafos genéticos (2.1), aquisição de conhecimento em Programação de Computadores (2.2) e sequenciamento de enunciados (2.3).

2.1. Modelo de Sobreposição e Grafos Genéticos

O Modelo do Aprendiz, no contexto de um STI, remete à competência do aprendiz no conteúdo tutorado, podendo incluir suas preferências relativas ao processo de aquisição de conhecimento. Esse modelo, portanto, contém uma representação do estado atual de conhecimento e do desempenho do aprendiz sobre o domínio ensinado [Giraffa 2003]. Ele é atualizado de forma dinâmica pelo sistema e representa informações de aprendizes em particular. Todavia, o modelo pode ser adaptado a necessidades específicas, representando informações coletivas ao invés de individuais, ou ainda por tempo fixado em detrimento de um histórico completo.

Além disso, o Modelo do Aprendiz pode variar na maneira com que representa internamente as informações obtidas, sendo o Modelo de Sobreposição (*Overlay*) uma delas. Ele representa o conhecimento do aprendiz como um subconjunto do conhecimento do domínio. A sobreposição desses conjuntos evidencia os conhecimentos que devem ser apropriados pelo aprendiz. Essa abordagem implica que a representação de ambos os modelos (do domínio e do aprendiz) sejam comparáveis.

A pesquisa de [Maschio and Direne 2013] buscou uma técnica de representação que pudesse descrever as capacidades do domínio de Programação de Computadores e que, frente ao Modelo de Sobreposição, denotasse a incidência do conhecimento do aprendiz sobre o conhecimento do domínio. O citado estudo optou pelo Grafo Genético [Goldstein 1979], que se trata de um tipo específico de grafo cujo conhecimento do domínio (ou de um experto) é descrito por um conjunto de capacidades (vértices) interconectados por relações (arestas) que evoluem: da simplificação à elaboração, do desvio à correção, da abstração ao refinamento, da especialização à generalização e do pré-requisito ao pós-requisito.

2.2. Aquisição de Conhecimento em Programação de Computadores

O processo de aprendizagem em domínios de natureza prática e complexa fundamenta-se na aquisição de conhecimentos sobre **princípios** e consequente desenvolvimento deles até que constituam **perícias** na área [Lesgold et al. 1988]. A **aquisição de princípios** corresponde à assimilação, pelo aprendiz, dos fundamentos de um domínio específico, ou seja, do conhecimento formal inicialmente repassado. O **desenvolvimento de perícias** acaba sendo o processo de construção de conhecimento por meio da experiência, levando ao crescimento da aptidão pela prática. Esse processo envolve a integração dos princípios aprendidos com a experiência que se alcançou até o momento.

Diante da mesma perspectiva de [Lesgold et al. 1988], conforme [Maschio 2013], a aprendizagem em Programação de Computadores pode ser entendida como a aquisição de conhecimentos sobre os princípios de lógica de programação e consequente desenvolvimento deles até consolidarem perícias na área. Os princípios envolvem a compreensão das instruções isoladas, considerando a rigidez léxica, sintática e semântica da linguagem utilizada. As perícias se tratam do encadeamento e aninhamento dessas instruções na construção de um algoritmo. Portanto, a aplicação de ambas as categorias de conhecimento são exigidas no ato de programar.

Nesse sentido, [Pimentel and Direne 1998] identificou o seguinte conjunto de capacidades presentes no estereótipo de um programador perito: **(1)** precisão sintática; **(2)** precisão semântica; **(3)** identificação de estruturas principais no programa fonte (busca

por palavra chave); **(4)** simulação mental dos estados do computador durante a execução; **(5)** catálogo de erros; **(6)** mapeamento mental das estruturas do programa; **(7)** checagem de pré-condições; **(8)** análise do problema; **(9)** integração dos subproblemas; **(10)** generalização da solução; **(11)** reutilização de soluções já conhecidas; e **(12)** catálogo de soluções. Em continuidade, a pesquisa de [Maschio 2013] atuou na revisão desse conjunto de capacidades, com o objetivo de utilizá-lo para catalogar enunciados de programação.

O estudo constatou a necessidade de definir perícias de mais baixo nível de abstração que, por sua vez, correspondessem à prática em Programação de Computadores. Isso foi justificado pela dificuldade de relacionar enunciados com as perícias isoladas do conjunto definido por [Pimentel and Direne 1998]. Consequentemente, [Maschio 2013] reconheceu que esse conjunto de capacidades se caracterizava pelo alto nível de abstração e pela sobrejacência a outras perícias mais próximas dos elementos instrucionais da programação. Entretanto, eram justamente essas últimas perícias que se faziam relevantes à modelagem e evolução do aprendiz frente ao conhecimento do domínio, bem como à catalogação e ao sequenciamento de enunciados.

A abordagem para identificar esse segundo conjunto de perícias foi reconhecer frações mais restritas de habilidades em correspondência às instruções e princípios de programação. Assim, uma primeira tentativa identificou 335 perícias, hierarquizadas em cinco níveis, considerando apenas um subconjunto de conhecimento do domínio que chegava até o conteúdo de Estruturas de Repetição. Com o objetivo de facilitar o processo de catalogação dos enunciados, o conjunto de perícias foi sintetizado em termos daquelas pertencentes aos primeiros níveis hierárquicos. Esse conjunto foi modelado por meio de um grafo genético composto por 41 perícias e 60 relações, sendo apontado pelo pesquisador como uma versão experimental e sujeita a alterações pela comunidade científica.

2.3. Sequenciamento de Enunciados

Contribuições anteriores evidenciaram a complexidade dos enunciados propostos como um dos principais componentes de motivação do aprendiz [Soldado and du Boulay 1995]. Equivalentemente, [Pimentel and Direne 1998] sustenta que "a repetição sistemática de enunciados completamente diferentes, porém de graus de complexidade aproximadamente iguais pode elevar consideravelmente a autoconfiança do aprendiz, mantendo-o motivado e produtivo". Em contrapartida, [Maschio 2013] apontou que a extrema rigidez na ordenação das galerias de enunciados é um fator adverso em ambientes de ensino.

No ensino de Programação de Computadores, [Pimentel and Direne 1998] descreve **medidas cognitivas**, diante da perspectiva do desenvolvimento de perícias, que se mostram úteis no sequenciamento automático e adaptativo de enunciados. Elas possuem a função de quantificar cognitivamente um enunciado, sendo responsáveis por estimar quanto ele exige de um aprendiz em termos de conhecimentos adquiridos e de capacidades desenvolvidas na progressão do aprendizado. As medidas cognitivas consideram subcomponentes relacionados à complexidade do software (e.g. linhas de código, complexidade estrutural e detalhes de implementação), somados ao conjunto de capacidades identificadas no estereótipo de um programador perito (Seção 2.2).

Nesse mesmo sentido, a **carga cognitiva** de um enunciado é definida como a capacidade que ele tem de exercitar o aprendiz na construção de um programa de computador, contribuindo para o desenvolvimento de perícia na área. Ela consiste na ponderação das

medidas cognitivas presentes em um enunciado. Essa ponderação é necessária porque diferentes medidas possuem níveis distintos de contribuição para cada enunciado. Os pesos são determinados por meio de informações obtidas de especialistas no domínio.

Assim, as medidas cognitivas conseguem amparar a escolha do próximo enunciado em uma sessão de ensino. Elas auxiliam na classificação de enunciados por grau de exigência/contribuição, podendo adaptar a sequência deles ao nível atual do aprendiz. A mesma pesquisa implementou isso por meio da ferramenta **Sequence**, que avalia os enunciados quanto à complexidade do software e considera as perícias de precisão sintática e semântica, análise do problema, reutilização de soluções e simulação mental.

Conforme antedito, os estudos de [Pimentel and Direne 1998] foram retomados por [Maschio 2013]. A pesquisa revisou as capacidades do estereótipo de um programador experto e modelou essas perícias em um grafo genético (seções 2.2 e 2.1). Na abordagem, tanto o conhecimento do aprendiz quanto a contribuição de cada enunciado constituem subgrafos do conhecimento do domínio. A referida pesquisa implementou duas ferramentas, uma destinada à **descrição do conhecimento do domínio** e outra que se concentra na **catalogação e elicitação de enunciados**. Aspectos de modelagem do aprendiz não atingiram o estágio de implementação.

Por fim, em outros sistemas recentemente divulgados, se observa que são escassas as publicações que forneçam detalhes dos mecanismos que apoiam a escolha do próximo enunciado. Abaixo são relacionados outros três sistemas tutores que promovem o sequenciamento de enunciados e suas respectivas contribuições:

- **JV^{2}M:** destina-se ao ensino do processo de compilação em linguagens orientadas a objetos por meio da metáfora de uma aventura em 3D. Ele determina que o melhor momento para intervir é quando o aprendiz comete um erro, pois se consegue utilizar a própria falha como oportunidade focada de explicação. O próximo exercício é apresentado quando se detecta que o aprendiz tem conhecimento suficiente para avançar ao próximo nível [Gómez-Martín et al. 2005];
- **SQL-Tutor:** aborda o ensino da linguagem de pesquisa SQL. Ele comparou duas estratégias para a seleção de enunciados, sendo que a primeira utiliza a complexidade estática especificada no momento da autoria, enquanto a outra considera medidas dinâmicas de dificuldade que são calculadas individualmente para cada aprendiz. O estudo indicou que a segunda estratégia trouxe maiores benefícios ao desempenho dos aprendizes no processo de ensino [Mitrovic and Martin 2004];
- **PHP ITS:** promove o ensino de desenvolvimento web básico para iniciantes. Ele considera que o exercício mais adequado ao indivíduo, em um determinado momento, é aquele que abrange o menor número de conteúdos ainda não aprendidos [Weragama 2013].

3. Fundamentos da Solução Proposta

Diante do exposto, ainda não foram voltados esforços de pesquisa à continuidade dos estudos de [Maschio and Direne 2013, Maschio 2013]. Uma possível continuidade imediata, objeto do presente estudo, reside na implementação da modelagem dinâmica do aprendiz. Com isso, busca-se prover um processo de busca heurística que oriente conhecimentos a serem explorados e, igualmente, seja utilizado no sequenciamento adaptativo de enunciados ao aprendiz. Logo, é consequente incidir sobre a alimentação do Modelo do Aprendiz, conforme seu desempenho na resolução dos enunciados propostos.

3.1. Modelagem do Aprendiz

Admitindo a modelagem em grafo genético das perícias no domínio de Programação de Computadores, o conhecimento de um aprendiz é considerado como o subconjunto (ou subgrafo) do conhecimento de um experto. Assim, as perícias faltantes ao aprendiz são destacadas pela sobreposição do segundo grafo no primeiro. A Figura 1 mostra um exemplo simplificado, de grafo genético, que representa o estágio atual do aprendiz frente ao conhecimento do domínio. A sobreposição propriamente dita é evidenciada.

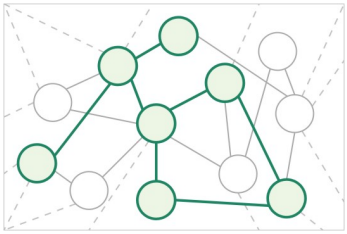

Figura 1. Sobreposição do Modelo do Aprendiz frente ao conhecimento do domínio

Assume-se a modelagem não apenas das perícias desenvolvidas pelo aprendiz (vértices do grafo), mas também da estrutura complementar que fornece indícios de como ele aprendeu (as relações ou arestas do grafo). Apesar do armazenamento dessas informações (representação interna) ser trivial, existe dificuldade em inferir tais elementos por meio da interação com aprendiz. A Seção 3.4 propõe alternativas.

3.2. Sugestão do Tópico Tutorado/Exercitado

Partindo dos modelos do aprendiz e do domínio, provê-se um processo de busca heurística que orienta conteúdos a serem explorados pelo aprendiz. Ele indica perícias que deveriam ser prioritariamente desenvolvidas (compreendidas e exercitadas) pelo indivíduo, conforme o seu nível atual de conhecimento (Figura 2).

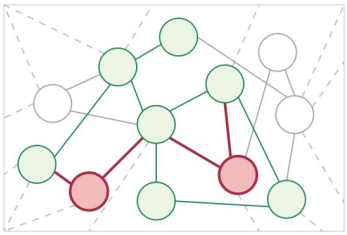

Figura 2. Indicação de regiões do grafo a serem exploradas

A abordagem considera como critérios:

(a) priorização de perícias insuficientemente desenvolvidas;
(b) perícias localizadas na fronteira entre o subgrafo do Modelo do Aprendiz e o restante do conhecimento do domínio;
(c) distância das perícias desse subconjunto até aquela definida como inicial para o conhecimento do domínio;
(d) estratégias de tutoria bem-sucedidas para aquele aprendiz (tipos de relações genéticas, como generalização, analogia, simplificação e desvio).

3.3. Sequenciamento de Enunciados

Escolhida a perícia (ou conjunto delas) como tópico a ser exercitado, pode-se encontrar enunciados que correspondam a essa demanda. Isso se torna possível porque o mesmo grafo genético (que representa o conhecimento do domínio) é empregado como gabarito para os enunciados a serem propostos. Nessa perspectiva, cada enunciado contribui para que se desenvolvam perícias específicas do aprendiz, sendo também um subgrafo do conhecimento do domínio (Figura 3). Trata-se de uma extensão que [Maschio and Direne 2013] propôs para os estudos de [Goldstein 1979].

Assumindo essa possibilidade, o conjunto de enunciados selecionados consegue ser catalogado de acordo com o referido gabarito (Figura 3). O sequenciamento, então, pode ocorrer de maneira automática, considerando, por exemplo, a quantidade total de perícias envolvidas na resolução.

Contudo, esta pesquisa investe na adaptação inteligente desses enunciados de acordo com o perfil atual do aprendiz. Com isso, diferentes exercícios podem ser sugeridos para a tutoria de uma mesma perícia, considerando aprendizes distintos. A abordagem utilizada, mencionada na Seção 2.3, é privilegiar os enunciados que abranjam o menor número de perícias não desenvolvidas, isto é, que sejam mais próximos do conhecimento do aprendiz naquela ocasião.

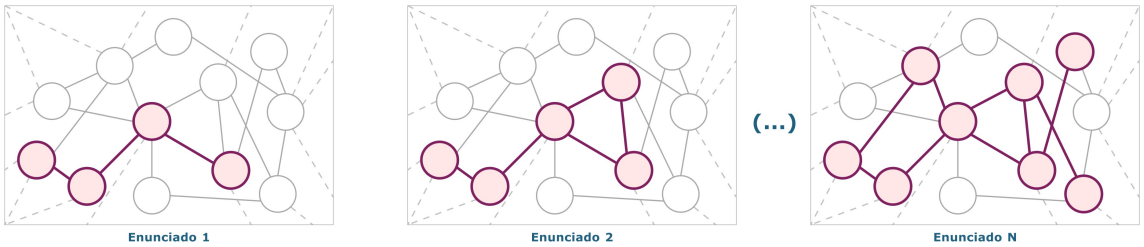

Figura 3. Catálogo de enunciados em sobreposição ao conhecimento do domínio

3.4. Atualização do Modelo do Aprendiz

Após o cumprimento do enunciado, faz-se necessária uma avaliação para verificar se ocorreu o desenvolvimento das perícias abordadas. A alimentação do Modelo do Aprendiz é indispensável para que se passe a refletir o nível atual de conhecimento do indivíduo, considerando o possível progresso. Entretanto, apenas a sugestão do enunciado para o aprendiz não é condição suficiente para se inferir que as perícias foram desenvolvidas.

Esta pesquisa não se concentra no mérito de implementar um corretor automático para as soluções. Ela precisa de uma alternativa que forneça parâmetros para que o Modelo do Aprendiz seja atualizado, permitindo que a próxima sugestão de enunciado já considere o avanço que se possa ter conquistado.

Uma das alternativas para essa finalidade é supor a avaliação das resoluções por um especialista externo (humano), levando em conta o gabarito de perícias fornecido pelo enunciado. A Figura 4 expõe tal possibilidade, assumindo que a avaliação externa assinala quais foram as perícias desenvolvidas daquelas exercitadas. O procedimento registra o progresso do aprendiz e consequente evolução na estrutura do grafo genético.

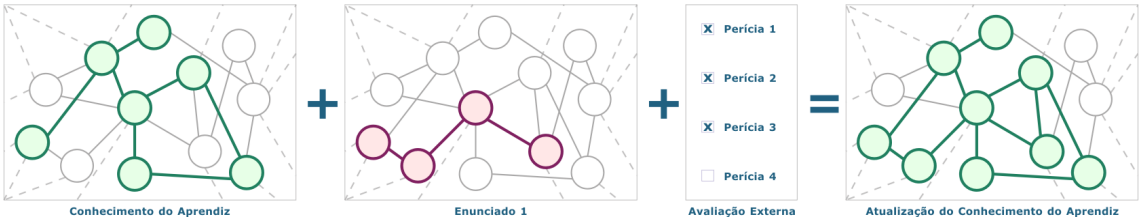

Figura 4. Atualização do Modelo do Aprendiz

Outra alternativa resume-se em substituir o especialista externo por um diagnóstico automático simples, a exemplo do que ocorre nas maratonas de programação. A abordagem essencialmente consiste em comparar a saída esperada com aquela fornecida pela resolução do aprendiz, tendo o mesmo conjunto de dados como entrada. Isso depende que, internamente, se realize a chamada de um compilador. Contudo, ocorre que essa segunda alternativa é bastante sensível à natureza dos enunciados. No momento da autoria, tais enunciados deverão ser compostos de maneira que a resolução não seja possível sem o emprego das perícias que se pretende desenvolver.

Idealmente, para a atualização do Modelo do Aprendiz, avaliar uma solução vai muito além de observar se ela fornece saídas corretas. Interessa saber quais foram as capacidades desenvolvidas, ou seja, o aprendizado que se obteve com o enunciado sugerido. Mesmo uma solução com comportamento incorreto pode ter trazido benefícios ao aprendizado.

3.5. Acompanhamento do Processo de Aprendizagem

A representação em grafo abre oportunidades de extrair dos modelos informações úteis à didática do instrutor. A pesquisa tem analisado algumas dessas possibilidades e considera:

(a) A sintetização do Modelo do Aprendiz de um conjunto, como uma sala de aula, para que se observe como as capacidades são ensinadas pelo instrutor. Com isso, pode-se notar que um instrutor se sobressai no ensino de alguns conteúdos, enquanto um segundo acaba negligenciando outros;
(b) A apresentação de um histórico do desenvolvimento de perícias do aprendiz durante um intervalo de tempo específico (Figura 5). Consegue-se, assim, exibir o grafo sendo conquistado (coloração dos vértices) à proporção do tempo;
(c) A aplicação desse mesmo recurso para sintetizar o Modelo do Aprendiz de um conjunto, auxiliando a visualizar a progressão do aprendizado de uma sala inteira, ou de um curso inteiro.

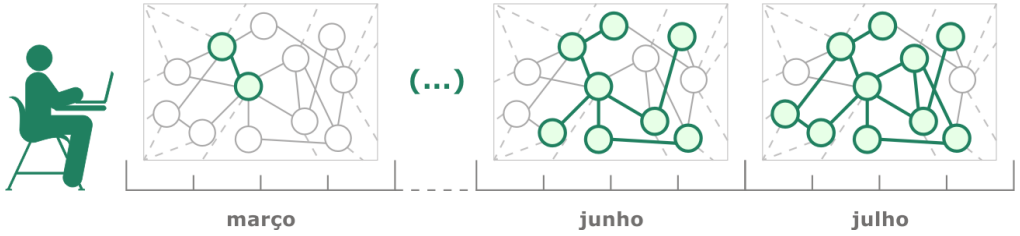

Figura 5. Progresso do aprendiz ao longo do tempo

3.6. Prototipação da Ferramenta

As funcionalidades descritas estão sendo instanciadas em um protótipo de ferramenta que objetiva o seguinte ciclo de uso:

(a) Indicar perícias a serem prioritariamente desenvolvidas pelo aprendiz, considerando o seu nível atual de conhecimento;
(b) Sequenciar e sugerir enunciados que contemplem o desenvolvimento das perícias sugeridas; e
(c) Atualizar o Modelo do Aprendiz conforme a avaliação de desempenho nas soluções dos enunciados propostos;
(d) Extrair e apresentar, por meio de representações externas gráficas, estáticas e dinâmicas, informações úteis ao processo de ensino-aprendizagem.

4. Considerações Finais e Trabalhos Futuros

O presente artigo delineou uma abordagem para tratar do sequenciamento adaptativo de enunciados, considerando a modelagem dinâmica do aprendiz, no domínio de Programação de Computadores. A pesquisa decorreu da observação de uma potencialidade pouco explorada dos estudos de [Maschio 2013], favorecendo um possível espaço de contribuição. Diante do levantamento realizado, percebeu-se uma quantidade reduzida de pesquisas que abrangessem o sequenciamento automático de enunciados em ambientes de ensino. Mais especificamente, foi notado que tais estudos se mostram ainda mais dispersos em se tratando do sequenciamento adaptativo em tutoria inteligente.

A metodologia apresentada considera que o modelo do domínio, do aprendiz e o catálogo de enunciados sejam representados por grafos genéticos. Com isso, aspectos de sobreposição evidenciam o progresso do aprendiz frente ao conhecimento do domínio, além da contribuição individual de cada enunciado para que isso ocorra. Propõe-se, então, um processo de busca heurística que indique capacidades a serem prioritariamente desenvolvidas pelo aprendiz, bem como o sequenciamento adaptativo de enunciados que auxiliem nessa evolução. A atualização do Modelo do Aprendiz, conforme seu desempenho ao resolver os enunciados propostos, e a extração de informações úteis sobre a aprendizagem, por fim, são analisadas.

Esforços imediatos se concentram na implementação desse formalismo em um protótipo de ferramenta. Subsequentemente, objetiva-se proceder com a coleta de dados a partir do uso inicial dessa ferramenta, que indicará tanto a possível eficiência da abordagem quanto a demanda de alterações no protótipo. Considera-se também a extensão das funcionalidades doa ferramenta baseada na coleta de dados que se mencionou. Prevê-se, em paralelo, a avaliação formal com aprendizes no intuito de constatar os benefícios de utilização da abordagem e do protótipo.

Como principal resultado, espera-se contribuir para a dinamização do aprendizado na área, por meio da adaptação de sequências de enunciados às particularidades dos aprendizes. Por fim, acredita-se que parte dos resultados alcançados podem se estender à aquisição de conhecimento em outros domínios de natureza prática e complexa.

Referências

Desmarais, M. C. and d. Baker, R. S. J. (2012). A review of recent advances in learner and skill modeling in intelligent learning environments. In *User Modeling and User-Adapted Interaction*, volume 22, pages 9–38. Springer Netherlands.

Giraffa, L. M. M. (2003). Fundamentos de sistemas tutores inteligentes. Relatório técnico, Pontifícia Universidade Católica do Rio Grande do Sul.

Goldstein, I. P. (1979). The genetic graph: A representation for the evolution of procedural knowledge. *International Journal of Man-Machine Studies*, 11(1):51–77.

Gómez-Martín, M. A., Gómez-Martín, P. P., and González-Calero, P. A. (2005). Game-based learning as a new domain for case-based reasoning. *1st Workshop on Computer Gaming and Simulation Environments*.

Lesgold, A., Rubinson, H., Feltovich, P., Glaser, R., Klopfer, D., and Wang., Y. (1988). *The nature of expertise*, chapter Expertise in a Complex Skill: Diagnosing X-ray Pictures, pages 311–342. Lawrence Erlbaum Associates, Inc, Hillsdale, NJ, England.

Maschio, E. (2013). *Modelagem do Processo de Aquisição de Conhecimento Apoiado por Ambientes Inteligentes*. Tese de doutorado, Universidade Federal do Paraná, Curitiba.

Maschio, E. and Direne, A. I. (2013). Processo de aquisição de conhecimento em Programação de Computadores apoiado por ambientes inteligentes. In *Anais do XXIV Simpósio Brasileiro de Informática na Educação. SBIE 2013*, Campinas, São Paulo. Sociedade Brasileira de Computação.

Miliszewska, I. and Tan, G. (2007). *Befriending Computer Programming: A Proposed Approach to Teaching Introductory Programming*, volume 4, pages 277–289. Issues in Informing Science and Information Technology.

Mitrovic, A. and Martin, B. (2004). Evaluating adaptive problem selection. In *Adaptive Hypermedia and Adaptive Web-Based Systems: Third International Conference*, volume 3137, pages 185–194, Eindhoven, The Netherlands. Springer Berlin Heidelberg.

Norvig, P. (2001). Teach yourself programming in ten years. http://norvig.com/21-days.html.

Nwana, H. S. (1990). Intelligent tutoring systems: an overview. *Artificial Intelligence Review*, (4):251–277.

Pimentel, A. R. and Direne, A. I. (1998). Medidas cognitivas no ensino de programação de computadores com sistemas tutores inteligentes. *Revista Brasileira de Informática na Educação (RBIE)*, 3:17–24.

Soldado, T. D. and du Boulay, B. (1995). Implementation of motivational tactics in tutoring systems. *Journal of Artificial Intelligence in Education*, 6(4):337–378.

Turine, M. A. S., Maltempi, M. V., and Hasegawa, R. (1994). Sistemas tutores inteligentes: uma revisão descritiva. Relatório técnico, Universidade de São Paulo.

Weber, G. and Brusilovsky, P. (2001). ELM-ART: An adaptive versatile system for web-based instruction. In *International Journal of Artificial Intelligence in Education*, volume 12, pages 351–384.

Weber, G. and Möllenberg, A. (1995). *Cognition and Computer Programming*, chapter ELM programming environment: A tutoring system for LISP beginners, pages 373–408. Ablex Publishing Corporation, Norwood, New Jersey, USA.

Weragama, D. S. (2013). *Intelligent Tutoring System for Learning PHP*. Tese de doutorado, Queensland University Of Technology, Brisbane, Austrália.