Uma arquitetura para Sistemas Tutores Inteligentes para o ensino de programação baseados na web

Luiz Augusto de Macedo Morais[1], Rivanilson da Silva Rodrigues[1], Sormany Silva Dantas[1], José Lopes Gomes Filho[1], José Wilker de Lima Silva[2]

[1]Graduandos em Licenciatura em Computação - Universidade Estadual da Paraíba. e-mails: {luizaugustomm, rivanilson, sormanysd}@gmail.com, zefilho@msn.com

[2]Coordenador do curso de Matemática - Universidade Estadual da Paraíba. e-mail: wilker@uepb.edu.br

Resumo: O presente artigo visa propor uma arquitetura para Sistemas Tutores Inteligentes para o ensino de programação baseados na *web*. Para isso, foram analisadas arquiteturas presentes em tutores relacionados ao ensino de programação e banco de dados e, a partir daí, criou-se uma arquitetura que possui as principais características presentes nas demais. Ao final deste trabalho, será exposta a nova arquitetura e cada um de seus componentes será descrito.

Palavras–chave: sistemas tutores inteligentes, arquiteturas de STIs, inteligência artificial na educação

1. INTRODUÇÃO

De acordo com Kranch (2012), "o ensino de programação para iniciantes tem sido um problema nos últimos 40 anos e é um dos sete maiores desafios da computação". O autor ainda expõe que a literatura possui muitas citações sobre a fraca performance dos alunos após concluírem cursos introdutórios de programação. Além disso, outro fator muito importante é a grande taxa de desistência por parte dos alunos em cursos desta natureza.

Para amenizar/solucionar as dificuldades encontradas pelos alunos, pesquisadores tentaram desenvolver durante as últimas décadas sistemas que auxiliassem o processo de aprendizagem de programação. Dentre estes sistemas, pode-se destacar o Proust (JOHNSON; SOLOWAY, 1984) que é um diagnosticador de programas em Pascal, o LispTutor (ANDERSON; REISER, 1985) que se baseia na Teoria do Controle Adaptativo do Pensamento (ACT), e o SQL-Tutor (MITROVIC; OHLSSON, 1999) que é um sistema para o ensino de banco de dados e utiliza a Modelagem Baseada em Restrições (*Constraint Based Modelling*). Todos estes sistemas são chamados de Sistemas Tutores Inteligentes (STI).

Um Sistema Tutor Inteligente é um ambiente de aprendizagem que permite que um estudante aprenda um determinado assunto com o auxílio do computador. Tal sistema requer uma grande compreensão das várias dimensões envolvidas no processo de ensino. "Para o desenvolvimento de Sistemas Tutores Inteligentes é necessário integrar diferentes técnicas de Inteligência Artificial" (POZZEBON, 2008).

Segundo Gavidia e Andrade (2003), um STI é formado basicamente de quatro partes fundamentais: o módulo de interface ou comunicação, que serve de elo entre o aluno e o sistema; o módulo do estudante, que avalia o aluno e cria um modelo de acordo com seu perfil; o módulo do domínio, que possui todo o "conhecimento" presente no tutor e; o módulo do tutor ou pedagógico, que

ISBN 978-85-62830-10-5

recebe as informações criadas pelo módulo do estudante e estabelece uma estratégia pedagógica para o aluno segundo seu perfil.

Segundo França et al (2010) os STIs ainda não conseguem atender a todas as dificuldades encontradas no ensino de programação. No entanto, pode-se perceber que a eficácia destes sistemas vem melhorando ao longo dos anos, de acordo com os resultado obtidos de Johnson e Soloway (1984), Anderson e Reiser (1985) e Mitrovic e Ohlsson (1999).

Com a ascensão da Internet no fim da década de 1990, viu-se a possibilidade da implementação de STIs baseados na *web*. Muitos dos sistemas criados anteriormente, foram migrados para uma versão online, como é o caso do LispTutor, com o LispTutor Jr. (HASTINGS, 2012). Além disso, outros STIs foram criados especialmente para a *web* como, por exemplo, o BITS de Butz et al. (2006).

Um dos principais motivos para a adoção da Internet como módulo de comunicação dos STIs é o acesso fácil e amplo a este meio. Butz et al. (2006) apoiam esta ideia e acrescentam que "os sistemas de aprendizagem baseados na *web* estão aumentando sua popularidade devido às suas vantagens em relação aos livros didáticos tradicionais de papel". Outra característica que os STIs baseados na *web* possuem é que "eles são capazes de incluir formas ricas de multimídia e elementos interativos, como áudio, vídeo e animação" (BUTZ et al., 2006).

De acordo com o que foi supracitado, o presente trabalho visa analisar as arquiteturas de STIs existentes e, a partir disso, propor uma arquitetura para sistemas tutores inteligentes para o ensino de programação baseados na *web*.

2. MATERIAL E MÉTODOS

A pesquisa do presente trabalho foi de caráter exploratório, pois envolveu um levantamento bibliográfico sobre o conteúdo a ser estudado (SILVA ; MENEZES , 2001).

O trabalho aborda, em sua essência, a arquitetura de um STI para o ensino de programação baseado na *web*. Para alcançar este objetivo, foram feitas análises comparativas entre arquiteturas existentes por meio do método SAAM (*Software Architecture Analysis Method*), descrito por Kazaman et al. (1994). Ao final da análise, foi proposta uma arquitetura para suprir as deficiências encontradas nas arquiteturas estudadas.

2.1. Método SAAM

Segundo Kazaman et al. (1994), o SAAM é um método de análise de arquiteturas de software serve para comparar soluções arquiteturais. De acordo com Silva Filho (2009), esse método possui basicamente cinco atividades:

Desenvolvimento de cenários: Desenvolvimento de cenários de tarefas que ilustram os tipos de atividades que o sistema de software deve prover suporte. No nosso caso, ao invés de cenários, selecionaremos características importantes ao sistema.

Descrição de arquitetura: Para cada análise, deve-se possuir uma representação arquitetural do sistema. No caso deste trabalho, a representação arquitetural das arquiteturas analisadas foi feita pelos autores estudados (GAVIDIA e ANDRADE, 2003; MITROVIC, 2001; HOLLAND, 2009).

Avaliação de cada cenário: Para cada cenário, deve-se determinar se a arquitetura candidata oferece suporte diretamente às tarefas associadas ao cenário (pontuação positiva, +) ou se alguma modificação é necessária (pontuação negativa, -). Para isso, será construída uma tabela que conterá as arquiteturas e as características exigidas e será dada uma pontuação a cada arquitetura.

Determinação da interação de cenários: A interação de cenários mostra as tarefas às quais os componentes da arquitetura estão associados.

ISBN 978-85-62830-10-5

Avaliação de cenários e da interação de cenários: Nesta etapa, realiza-se uma avaliação global atribuindo uma pontuação a cada cenário e interação de cenário.

Para nosso estudo, apenas as três primeiras atividades desse método serão realizadas, pois se mostram mais compatíveis com a realidade da análise de arquiteturas de STIs.

2.2. Desenvolvimento de características

A arquitetura proposta foi desenvolvida com base em três características: (a) possuir o núcleo comum da arquitetura clássica, (b) ser adaptada para o ensino de programação e (c) ser baseada na *web*.

A primeira característica se deve ao fato de que a arquitetura clássica já está consolidada, sendo que a maioria dos STIs desenvolvidos a utilizam como base, a exemplo dos trabalhos citados neste texto.

Já a decisão de uma arquitetura adaptada ao ensino de programação se difere entre os autores. Contudo, para este STI foi utilizada a modelagem baseada em restrições (*constraint-based modelling*) ao invés da arquitetura *model-tracing*[1]. No entanto, como este assunto é bastante amplo, não será abordado neste trabalho e poderá ser discutido em trabalhos futuros.

O terceiro fator se deve ao fato de que a maioria dos STIs atuais são baseados na *web*, pois assim, eles conseguem atingir um vasto número de usuários.

2.3. Descrição de arquiteturas

A seguir, serão descritas as três arquiteturas candidatas, propostas por Gavidia e Andrade (2003), Mitrovic (2001) e Holland (2009), respectivamente.

2.3.1. Arquitetura clássica

A primeira arquitetura a ser abordada é a clássica. De acordo com Gavidia e Andrade (2003), esta arquitetura possui basicamente quatro componentes, sendo eles: modelo do aluno, modelo do domínio, modelo do tutor e interface.

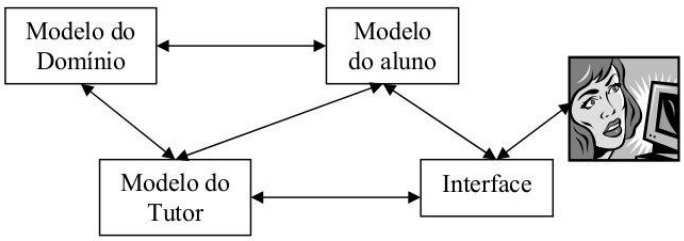

Figura 1- Arquitetura clássica de um STI, segundo Gavidia e Andrade (2003)

Esta é uma das primeiras arquiteturas que foram concebidas para STIs. Por ser de caráter genérico, ela pode, em tese, ser utilizada para o desenvolvimento de STIs de qualquer natureza. No entanto, na prática, ela serve como base para a criação de arquiteturas mais específicas, como é o caso da arquitetura proposta neste trabalho.

1 Se quiser saber mais sobre as vantagens e desvantagens de cada técnica, sugerimos ler a discussão entre Kodaganallur et al. (2005) e Mitrovic e Ohlsson (2006).

ISBN 978-85-62830-10-5

Por causa de sua generalidade, esta arquitetura não é apropriada para a modelagem baseada em restrições, pois não possui os modelos adequados para seu funcionamento. Além disso, ela não foi concebida para ser baseada na *web*, o que poderia causar eventuais transtornos em sua implementação.

2.3.2. Arquitetura do SQL-Tutor

A segunda arquitetura analisada foi desenvolvida por Mitrovic (2001) para o SQL-Tutor, um STI para o ensino de banco de dados. Esta arquitetura possui componentes adicionais em relação à arquitetura clássica, entretanto, eles são apenas extensões dos componentes já existentes.

O módulo do aluno foi renomeado para CBM (*Constraint-based modeller*) ou modelador baseado em restrições. Este módulo ainda é responsável por avaliar os estudantes, mas, ele recebe esse nome pois serve como modelador para o modelo do estudante, que são os dados obtidos pela avaliação dos alunos.

Já o módulo do domínio foi separado em dois novos modelos: *constraints* e banco de dados de problemas e soluções. Estes *constraints* (ou restrições) são regras que formam todo o conhecimento do tutor. Por meio delas, o tutor irá testar se as soluções submetidas pelos alunos estão corretas e, caso contrário, dar o *feedback* apropriado.

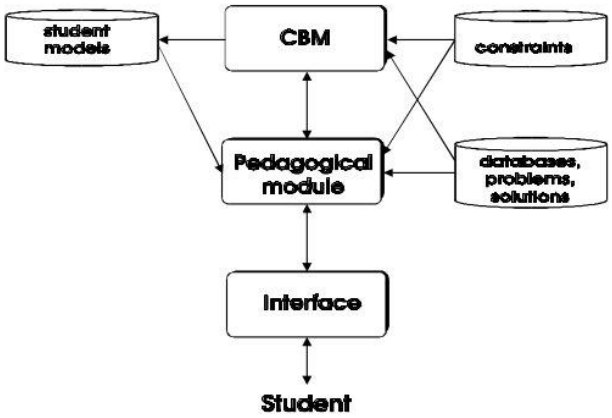

Figura 2- Arquitetura do SQL-Tutor, segundo Mitrovic (2001)

A única característica que esta arquitetura não possui são os componentes baseados na *web*.

2.3.3. Arquitetura do J-LATTE

A terceira arquitetura a ser analisada foi proposta por Holland (2009) para o J-LATTE, um STI para o ensino de Java que utiliza a modelagem baseada em restrições em seus modelos do domínio e estudante.

O SQL-Tutor e o J-LATTE possuem arquiteturas bem semelhantes, pois utilizam a mesma técnica de modelagem em seus modelos do domínio e estudante. A principal diferença entre esses dois tutores é que o segundo é baseado na *web* e, por isso, seu módulo de interface é dividido em submódulos.

ISBN 978-85-62830-10-5

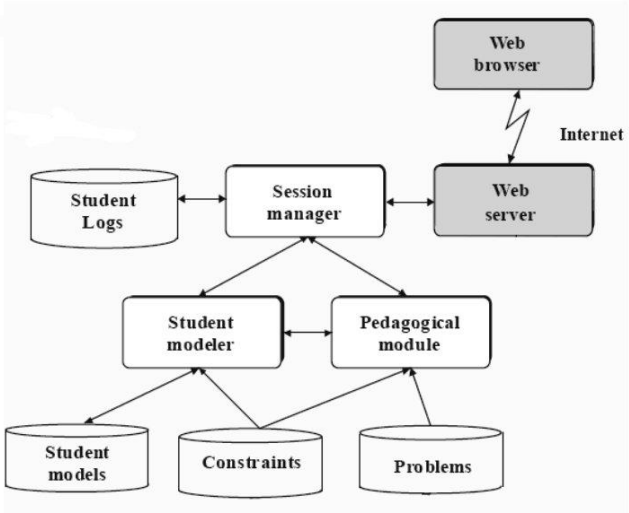

Figura 3- Arquitetura do J-LATTE, segundo Holland (2009)

O gerenciador de sessão (*session manager*) é responsável por receber quaisquer requisições do servidor *web* (*web server*) e identificar cada usuário. Por meio da identificação dos usuários, o gerenciador de sessão cria *logs* sobre o que o usuário fez durante a sessão e salva-os num banco de dados chamado *Logs* do estudante (*Student logs*).

A única desvantagem encontrada nesta arquitetura é que ela separa as informações sobre os estudantes em dois componentes (*Student logs* e *Student models*), o que a difere da arquitetura clássica. Além disso, esta divisão de componentes poderia causar inconsistência nos dados dos estudantes. Entretanto, para esta hipótese ser comprovada, seria necessário um estudo mais aprofundado do J-LATTE.

2.4. Avaliação de cada característica

Nesta seção, será exposta uma tabela que contém as características expostas na seção 2.2 e as arquiteturas descritas na seção 2.3.

As arquiteturas que possuírem os componentes responsáveis pelas características, receberão uma pontuação positiva (+) na respectiva característica, caso contrário, receberão uma pontuação negativa (-).

Tabela 1 - Características presentes em cada arquitetura analisada

	NUC	PROG	WEB
Clássica	+	-	-
SQL-Tutor	+	+	-
J-LATTE	-	+	+

NUC = núcleo comum da arquitetura clássica; PROG = adaptada ao ensino de programação; WEB = baseada na web

A partir da análise feita por meio da tabela, foi desenvolvida uma arquitetura que possui as vantagens presentes nas três arquiteturas estudadas. A seguir, serão explicados os componentes desta nova arquitetura.

ISBN 978-85-62830-10-5

3. RESULTADOS E DISCUSSÃO

A partir das arquiteturas analisadas, foi desenvolvida uma arquitetura que propõe satisfazer os três fatores mostrados neste trabalho: ser baseada na arquitetura clássica, adaptada ao ensino de programação e baseada na *web*.

A figura a seguir mostra a arquitetura desenvolvida e seus módulos integrantes.

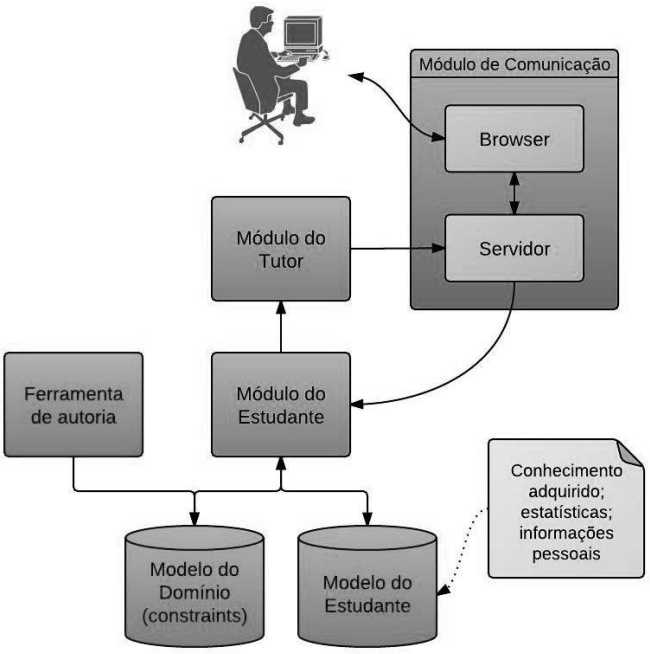

Figura 4 - Arquitetura proposta para um STI para o ensino de programação baseado na *web*

A arquitetura é formada por cinco entidades fundamentais: o módulo de comunicação, módulo do tutor, módulo do estudante, modelo do domínio e modelo do estudante. Sendo que o módulo e modelo do estudante se diferem.

3.1. Módulo de comunicação

Este módulo, também chamado de interface, é responsável por proporcionar a comunicação entre o usuário e o STI.

É neste módulo que são incorporados os componentes baseados na *web* presentes no J-LATTE. Entretanto, eles foram organizados de uma forma mais geral para não limitar a arquitetura. Com isso, o gerenciador de sessões (*session manager*) foi incorporado ao componente Servidor.

3.2. Módulo do tutor

O módulo do tutor, como na maioria dos STIs, possui um conjunto de estratégias pedagógicas e utiliza os dados obtidos do módulo do estudante para guiar os aprendizes de acordo com seu perfil.

3.3. Módulo do estudante

ISBN 978-85-62830-10-5

Este módulo serve para avaliar os alunos de acordo com o "conhecimento" presente no modelo do domínio na forma de restrições (*constraints*). Todos os dados obtidos por meio da avaliação são processados e armazenados no modelo do estudante.

3.4. Modelo do domínio

O modelo do domínio é um conjunto de todas as restrições criadas por uma ferramenta de autoria. Estas restrições são entidades propostas por Ohlsson (1992) que, quando são violadas, produzem um *feedback* apropriado ao aluno de acordo com o erro cometido por ele.

3.5. Modelo do estudante

O modelo do estudante é um banco de dados que possui o perfil de todos os usuários do STI. Este perfil é modelado de acordo com dados que são criados pelo módulo do estudante, bem como informações disponibilizadas pelo próprio estudante.

4. CONCLUSÕES

A arquitetura aqui proposta foi desenvolvida com o objetivo de combinar as principais características encontradas nas arquiteturas pesquisadas para construir um sistema tutor inteligente para o ensino de programação baseado na *web* que utilizasse a modelagem baseada em restrições. Contudo, esta arquitetura também pode ser utilizada para o desenvolvimento de qualquer STI baseado em restrições.

Como esta pesquisa foi de caráter bibliográfico, ainda não se pôde obter resultados práticos da implementação dessa arquitetura no desenvolvimento de um STI. Por isso, o próximo passo desta pesquisa será implementar e acompanhar a eficácia deste modelo na prática.

REFERÊNCIAS

ANDERSON, J. R.; REISER, B. J. The LISP tutor: it approaches the effectiveness of a human tutor. **BYTE**, v.10 n.4, p.159-175, 1985.

BUTZ, C.J.; HUA, S.; MAGUIRE, R.B. A Web-based Bayesian Intelligent Tutoring System for Computer Programming. **International Journal of Web Intelligence and Agent Systems**. [S.l.], v. 4, p. 77-97, 2006.

FRANÇA, E. L. et al. Utilização de Objetos de Aprendizagem em Sistemas Tutores Inteligentes para o ensino da Programação. In: SIMPÓSIO DE EXCELÊNCIA EM GESTÃO E TECNOLOGIA, 7., 2010, Resende, Rio de Janeiro: [s.n.]. **Anais...** Resende, Rio de Janeiro: [s.n.], 2010. p. 1-8.

GAVIDIA, J. J. Z.; ANDRADE, L. C. V. **Sistemas Tutores Inteligentes**. 2003. Trabalho de conclusão da disciplina Inteligência Artificial, Programa de Pós-Graduação da COPPE, Universidade Federal do Rio de Janeiro. Rio de Janeiro, 2003.

HASTINGS, P. **LispTutor Jr**. Disponível em: <http://reed.cs.depaul.edu/peterh/Tools/lisptutor.html>. Acesso em: 03 jun 2012.

HOLLAND, J. **A Constraint-based ITS for the Java Programming Language**. 2009. 150f. Dissertação (Mestrado em Ciência da Computação). University of Canterbury, New Zealand, 2009.

ISBN 978-85-62830-10-5

JOHNSON, W. L.; SOLOWAY, E. Proust: knowledge-based program understanding. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGENEERING. 7., 1984, Orlando, Florida: [s.n.]. **Proceedings...** Orlando, Florida: [s.n.], 1984. p. 369-380.

KAZAMAN, R. et al. SAAM: A Method for Analysing the Properties of Software Architectures. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 16., 1994, Sorrento, Italy. **Proceedings**... Sorrento, Italy: [s.n.], 1994. p. 81-90.

KODAGANALLUR, V.; WEITZ, R. R.; ROSENTHAL, D. A Comparison of Model-Tracing and Constraint-Based Intelligent Tutoring Paradigms. **International Journal of Artificial Intelligence in Education**. **v. 15, n. 2, p. 117-144, 2005.**

KRANCH, D. A. Teaching the novice programmer: A study of instructional sequences and perception. **Education and Information Technologies**, Springer, v. 17, n. 3, p. 291-313, 2012.

MITROVIC, A. et al. **Constraint-based Tutors**: a success story. Engineering of Intelligent Systems. Lecture Notes in Computer Science. v. 2070, p. 931-940, 2001.

MITROVIC, A. OHLSSON, S. A Critique of Kodaganallur, Weitz and Rosenthal, "A Comparison of Model-Tracing and Constraint-Based Intelligent Tutoring Paradigms". **International Journal of Artificial Intelligence in Education**. [S.l], v. 16, n. 3, p. 277-289, 2006.

MITROVIC, A.; OHLSSON, S. Evaluation of a Constraint-Based Tutor for a Database Language. **International Journal of Artificial Intelligence in Education**. [S.l.], v. 10, p. 238-256, 1999.

OHLSSON, S. Constraint-based Student Modelling. **International Journal of Artificial Intelligence in Education**. [S.l.], v. 3, p. 429-447, 1992.

POZZEBON, E. **Um modelo para Suporte ao Aprendizado em Grupo em Sistemas Tutores Inteligentes**. 2008. 157 f. Tese (Doutorado em Engenharia Elétrica) - Universidade Federal do Rio Grande do Sul, Florianópolis, 2008.

SILVA , E. L.; MENEZES , E. M. **Metodologia da Pesquisa e Elaboração de Dissertação**. 3. ed. Florianópolis: Laboratório de Ensino a Distância da UFSC, 2001.

SILVA FILHO, A. M. Análise da Arquitetura de Software: Essencial para Arquitetura de Software. **Revista Engenharia de Software Magazine**. [S.l.], v. 1, n. 1, p. 50-56, jul. 2009.

ISBN 978-85-62830-10-5