Sistema de Apoio ao Ensino de Modelagem de Software. Uma extensão para o editor de programação BlueJ
Autor: Luciano Gaspar - E-mail: tigaspar@hotmail.com
Orientador: Prof. Dr. Ítalo Santiago Vega
Linha de pesquisa: Modelagem de sistemas de software

Sistema de Apoio ao Ensino de Modelagem de Software Uma extensão para o editor de programação BlueJ

Autor: Luciano Gaspar / E-mail: tigaspar@hotmail.com
Orientador: Prof. Dr. Ítalo Santiago Vega
Linha de pesquisa: Modelagem de sistemas de software

INTRODUÇÃO

Muitos trabalhos tem sido publicados referente ao ensino de lógica de programação, ambientes de desenvolvimento de software, linguagens de programação e ensino de modelagem de software(Tabela 1).

Utilizando os conceitos apresentados por Brooks[5], esses trabalhos podem ser classificados de duas formas: (i) trabalhos de pesquisas que abordam aspectos acidentais no processo de desenvolvimento de software; (ii) trabalhos de pesquisas que apresentam conceitos novos na concepção das estruturas abstratas do software (aspectos essenciais).

Na Tabela 1, estão listados os trabalhos pesquisados. Eles foram classificados quanto ao objetivo pedagógico que se deseja atingir e se o enfoque dado atacam aspectos acidentais, essenciais ou ambos, na produção de software.

	Objetivo Pedagógico				Aspectos de Brooks	
Referência	Lógica de Programação	Ambiente de Aprend./Desenv.	Linguagem de Programação	Modelagem de Software	Acidentais	Essenciais
Baker[1]				•		•
Beck [2]				•		•
Dann [7]	•	•	•		•	
Goldwasser [9]	•	•	•		•	
Kölling [12]		•	•		•	
Kölling [11]		•	•		•	
Kölling [10]		•	•		•	
Levy [13]	•	•	•		•	
Marcelino [14]	•	•	•		•	
Ostiling [15]		•	•	•	•	
Paterson [16]		•	•	•		•
Polok [17]		•		•	•	•
Wiener [19]	•	•	•		•	

Tabela 1: Classificação das referências de acordo com Brooks
Fonte: Elaborado pelo autor

Observa-se na Tabela 1 que os principais trabalhos atuam sobre o ensino e aprendizagem de linguagens de programação e seus ambientes de desenvolvimento.

Atualmente os cursos de graduação na área de computação adotam linguagens de programação orientadas a objetos(OO) nas diferentes disciplinas.

O paradigma OO provocou mudanças no ensino de linguagens de programação e consequentemente resultou em vários métodos de ensino das linguagens de programação orientadas a objetos [7], [9], [10], [11], [13], [19].

Para o ensino dos conceitos fundamentais de orientação a objetos, sintaxe e semântica das linguagens de programação existem ambientes de desenvolvimento de uso profissional e acadêmico.

No âmbito acadêmico destaca-se o BlueJ[1] que favorece o processo de ensino-aprendizagem para iniciantes na linguagem Java, bem como a aplicação dos conceitos de orientação a objetos, limitando-se aos aspectos da programação e oferecendo como visão de projeto e interação apenas o Diagrama de Classes[4].

Essas ferramentas tem obtido sucesso, porém trata-se de um enfoque classificado por Brooks[5] como dificuldades(aspectos) acidentais.

Os aspectos acidentais criam barreiras que podem ser transpostas facilmente, basta dedicação do aluno e um conjunto de manuais ou livros de referência.

Já os aspectos essenciais lidam com a complexidade crescente do software. Tal complexidade está relacionada à intangibilidade, as mudanças de requisitos e conformidade[5].

Uma forma de lidar com esta complexidade é investir na criação de modelos. Os modelos tornam-se indispensáveis para compreensão do que está sendo desenvolvido.

Blaha e Rumbaugh[3] definem modelo como uma abstração de algo que facilita seu entendimento antes de construí-lo. Abstração é uma capacidade humana fundamental que permite lidar com a complexidade, omitindo detalhes não essenciais.

No processo de produção do software, cria-se modelos para visualizar como ele(software) é ou como quer que ele seja, ou ainda descrever sua estrutura e

[1] BlueJ é um ambiente de desenvolvido de software para o ensino de programação orientada a objetos na linguagem Java. Foi proposto por Michael Kölling[10]

comportamento, guiando sua construção e mantendo registros das decisões tomadas na sua concepção[4].

Para Booch, Rumbaugh e Jacobson[4] nenhum modelo único é suficiente. Qualquer sistema não trivial será melhor investigado por meio de um pequeno conjunto de modelos quase independentes com vários pontos de vista.

Cada área de conhecimento adota tipos específicos de modelos. Na engenharia de software contemporânea, segundo Booch, Rumbaugh e Jacobson[4] adota-se uma perspectiva orientada a objetos, onde a UML é empregada para visualização, especificação, construção e documentação de artefatos que orientam o desenvolvimento do software.

No contexto do ensino, o trabalho desenvolvido apoia-se na abordagem dos quadrantes de modelagem apresentado por Vega[18]. Nela quatro diagramas de UML são utilizados para descrever um modelo de software, dentre eles estão os diagramas de classes e de máquina de estados, que é o foco deste trabalho.

Esses modelos podem servir de inspiração para dar origem a uma arquitetura que sustente as necessidades do que está sendo modelado. De acordo com Zachman[20] arquitetura é o conjunto de artefatos ou um descritivo relevante de um objeto, de forma que este possa ser construído de acordo com os requisitos(de qualidade), bem como mantido ao longo da sua vida útil.

Neste sentido modelar um sistema significa determinar e representar um conjunto de informações arquitetadas sob diferentes vistas que servirão de guia de referência para a produção de algo concreto(código fonte).

Brooks[6] afirma que modelar é importante; ensinar a modelar é importante, pois existem grandes lacunas em cada disciplina de modelagem. Tais lacunas tornam-se evidentes quando se tenta ensinar aos estudantes como modelar bem.

Neste sentido a principal justificativa deste trabalho para o desenvolvimento de um software de apoio ao ensino de modelagem é a qualidade do código gerado pelo aluno, sob o ponto vista arquitetural.

Uma solução de software arquitetada favorece sua a escalabilidade, manutenção, divisão do trabalho em equipes de desenvolvimento, reuso de código e evita problemas como espalhamento de código.

Muitas vezes os alunos não conseguem vivenciar em sala de aula os impactos de um sistema não arquitetado, visto que os problemas oriundos da complexidade do software surgem na medida em que o número de linhas aumentam e os requisitos mudam.

OBJETIVOS

O objetivo deste trabalho é apresentar uma abordagem do uso do Diagrama de Máquina de Estados(DME) aplicado no ensino de modelagem de software, com foco nos tópicos classificados por Brooks[5] como dificuldades essenciais, destacando a importância e o reflexo da modelagem dinâmica nas linhas de código produzidas por alunos e futuros programadores.

Complementar a abordagem proposta, é realizado uma análise das contribuições e limitações da ferramenta BlueJ no ensino de modelagem de software e desenvolvido uma extensão de suas funcionalidades.

RESULTADOS

Inspirado em [8], o exemplo de Jogo de Dados foi utilizado para ilustrar problemas de uma arquitetura monolítica, que embora a solução proposta por Deitel[8] atenda as necessidades do jogador de dados, ela estabelece como parâmetro para o aluno uma solução de código procedimental e portanto organizada de maneira que limita a percepção do aprendiz na melhoria da arquitetura que visa minimizar os problemas enfrentados no ciclo de vida de um sistema.

O uso do diagrama de máquina de estados pode contribuir na percepção do aluno dos aspectos comportamentais do software, favorecendo uma arquitetura modularizada e apontando necessidades de refinamentos.

A partir desta pesquisa constatou-se a ausência de uma ferramenta didática que permita o aluno construir modelos estáticos e dinâmicos de software e perceber a relação entre eles contribuindo na produção sistemas modularizados.

Neste sentido foi desenvolvido um protótipo denominado SAEMOS (Sistema de Apoio ao Ensino de Modelagem de Software) que amplia as funcionalidades do BlueJ permitindo a construção de diagramas de máquina de

estados(DME), que composto pelo diagrama de classe(DCL), nativo do BlueJ, permite ao aluno desenvolver e testar modelos de software com vistas a estrutura e comportamento do código.

Para a implementação deste protótipo foi realizado um estudo aprofundado na arquitetura do BlueJ que poderá auxiliar no desenvolvimento de novas funcionalidades.

Referências Bibliográficas

[1] Baker, A., Navarro, E. O., and Hoek, A. An experimental card game for teaching software engineering processes. Journal of Systems and Software 75, 1-2 (2005), 3–16.

[2] Beck, K., and Cunningham, W. A laboratory for teaching object-oriented thinking. In Anais do International Conference on Object-Oriented Programming, Systems, Languages and Applications OOPLSA89 (New Orleans, EUA, 1989).

[3] Blaha, M. R., and Rumbaugh, J. R. Object-Oriented Modeling and Design with UML (2nd Edition). Prentice Hall, Upper Saddle River, New Jersey, EUA, 2004.

[4] Booch, G., Rumbaugh, J., and Jacobson, I. Unified Modeling Language Reference Manual, The (2nd Edition). Addison-Wesley Professional, Redwood City, CA, USA, 2005.

[5] Brooks, F. P. No Silver Bullet: essence and accidents of Software Engineering. IEEE Computer, April 1987.

[6] Brooks, F. P. The Design of Design: Essays from a Computer Scientist. Addison-Wesley Professional, New York, USA, 2010.

[7] Dann, W. P., Cooper, S., and Pausch, R. Learning To Program With Alice. Prentice Hall, Upper Saddle River, New Jersey, EUA, 2005.

[8] Deitel, H. M. e Deitel, P. J. Java como programar. 6ª ed. São Paulo: Pearson Prentice Hall, 2005

[9] Goldwasser, M. H., and Letscher, D. Using python to teach object-oriented programming in cs1. PyCon 2008(2008).

[10] Kölling, M. The design of an object-oriented environment and language for teaching. PhD thesis, University of Sydney, Sydney, 1999.

[11] Kölling, M., and Rosenberg, J. Guidelines for teaching object orientation with java. ITiCSE 2002, (2001).

[12] Kölling, M. The problem of teaching object-oriented programming, part 1: Languages. Journal of Object-Oriented Programming 11, 8 (1999), 8–15.

[13] Levy, R. B., Ari, M. B., and Uronen, P. A. The jeliot 2000 program animation system. Computers & Education 40, (2003), 1–15.

[14] Marcelino, E. R. Ensino de programação em um ambiente colaborativo. Master's thesis, Universidade Católica de Santos, Santos, Brasil, 2007.

[15] Ostling, M. A sequence diagram editor for bluej. Master's thesis, Umea University, Umea, Sweden, 2004.

[16] Paterson, J. H., Haddow, J., and Nairn, M. A design patterns extension for the Bluej ide. In ITiCSE'06 (2006), pp. 280–284.

[17] Polok, J. Quality measurement of programming examples. extension for programming editor bluej. Master's thesis, Umea University, Umea, Sweden, 2008.

[18] Vega, I. S. Conceitos Fundamentais de Modelagem de Software. Notas de Aula, São Paulo, SP, Brasil, 2011.

[19] Wiener, R. An Object-Oriented Introduction to Computer Science Using Eiffel. Prentice Hall, Upper Saddle River, New Jersey, EUA, 1996.

[20] Zachman, J. Enterprise Architecture: The Issue of the Century. Database Programming and Design Magazine, San Francisco, CA, USA, 1997.