Universidade de São Paulo
Mestrado em Ciência da Computação

Instituto de Matemática e Estatística
Disciplina MAC5701

Diagnóstico de programas em Tutores de Programação Inteligentes Baseado em Padrões Pedagógicos Plano de Estudo para Monografia - Tópicos em Computação

Karina Valdivia Delgado
kvd@ime.usp.br

Orientadora: Prof. D. Leliane Nunes de Barros

9 de Abril de 2004

1 Introdução

Este trabalho tem por objetivo apresentar um plano de estudo para o levantamento bibliográfico e o estudo de técnicas usadas para diagnóstico de programas que está diretamente ligado com meu tema de dissertação.

2 A área de estudo

Um Sistema Tutor Inteligente (Intelligent Tutoring System ITS) pode ser usado como ferramenta complementar para a educação, o qual tenta modelar alguns aspectos envolvidos em ensino [MM97]. O ITS oferece considerável flexibilidade na apresentação do material e uma grande habilidade para responder a idiossincrasia das necessidades do aluno [JEBH96]. Os principais componentes do ITS descritos na literatura são: (1) o modelo do estudante, (2) o modulo pedagógico, (3) o

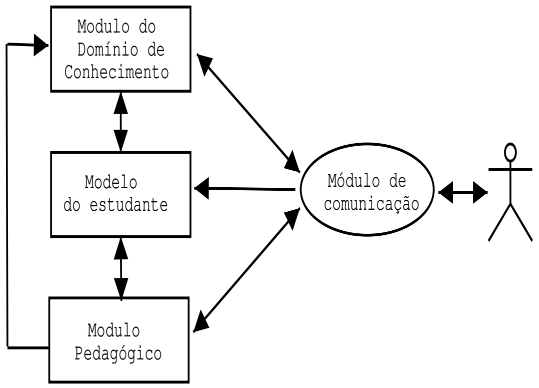

Figura 1: Interação entre os componentes de um Sistema Tutor Inteligente

modulo do domínio de conhecimento e (4) o modulo de comunicação. A figura 1 mostra a interação entre esses módulos. O modelo do estudante guarda informação sobre os aprendizes representando o estado do conhecimento do aluno. O modelo do estudante é usado como uma entrada do modulo pedagógico, que especifica o processo de ensino, por exemplo quando e como apresentar um novo tópico ao estudante. O modulo do domínio de conhecimento contem a representação do conhecimento que o ITS deseja que o estudante aprenda. Interações entre o estudante e o sistema são controlados por o modulo de comunicação.

Para [MM97] o modelo do estudante pode ser considerado o mais importante componente de um ITS, se o sistema não detecta de forma precisa o estado do conhecimento do estudante então a efetividade dos outros módulos pode ser comprometida. A identificação do atual estado de conhecimento é responsabilidade das técnicas de diagnóstico. Diagnóstico cognitivo é o processo de inferir o estado cognitivo da persona de seu desempenho. Para Sistemas Tutores inteligentes o termino *diagnóstico cognitivo* é com freqüência usado como sinônimo de *modelado do estudante* para denotar o processo de construção da representação do conhecimento do estudante a partir de evidencias proveidas pelas entradas da resolução de problemas [Sel93]. A palavra diagnóstico em educação tem sido usada para referir-se a atividades pedagógicas que ajudam a recolher e inferir informação acerca do estudante e suas ações e não meramente como a detecção de erros [Wen87].

Em um Tutor de Programação Inteligente o diagnóstico automático dos programas do estudante é essencial e é freqüentemente considerada uma tarefa difícil. Por um lado muitas variações podem existir no programa que satisfaz os requerimentos especificados no problema de programação. Por outro lado não existem ainda ferramentas eficientes para detectar erros semânticos e lógicos de um programa. Além é esperado que o estudante tenha oportunidades de aprendizagem durante o processo de diagnóstico de seus programas.

Nos artigos de *debugging* automático tem duas categorias de sistemas: os tutores de programação inteligente e os sistemas de diagnóstico de programas de propósito general. Existem técnica desenvolvidas em ambas categorias que gostaria de pesquisar [Duc93] [SW98] [AS86] [JS84] [XC03]; em especial diagnóstico baseado em modelos que é uma técnica de filtragem que tem sucesso para o diagnóstico de sistemas físicos e também tem sido usada para *debugging* de programas [Wie01] [CMW00] [MSWW] [CW03] [Rei87].

Outro tema relacionado com meu tema de dissertação é os Padrões pedagógicos de programação, que são descrições de problemas comuns e sua solução que está escrita em um formato adequado que facilita sua reutilização [Wal01] [Ber99] [AW98] [Bri02] [PC03] [Pro00].

3 Escopo do estudo

O estudo envolverá a leitura de artigos relacionados a:

- Tutores de programação inteligentes.
- Sistemas de diagnóstico de programas de propósito geral.
- Diagnóstico de programas baseado em modelos.
- Padrões pedagógicos de programação.

4 Plano de Estudo

- Abril: estudo das técnicas de diagnóstico usadas em Tutores de programação inteligentes e em sistemas de diagnóstico de programas de propósito geral.
- Maio: estudo de Diagnóstico de programas baseado em modelos e padrões pedagógicos de programação.
- Junho: elaboração da monografia tendo como base os resumos anteriormente concluídos.

Referências

[AS86] J. R. Anderson and E. Skwarecki. The automated tutoring of introductory computer programming. In *16 Communications of the ACM*, pages 842–849, 1986. http://act-r.psy.cmu.edu/publications/pubinfo?id=114.

[AW98] Owen Astrachan and Eugene Wallingford. Loop patterns, 1998. http://www.cs.duke.edu/~ola/patterns/plopd/loops.html.

[Ber99] Joseph Bergin. Patterns for selection version 4, 1999. http://csis.pace.edu/~bergin/patterns/Patternsv4.html.

[Bri02] Stina Bridgeman. Intro to computing i, 2002. http://cs.colgate.edu/faculty/stina/courses/cosc/101/f02/syllabus.html.

[CMW00] Markus Stumptner Cristinel Mateis and Franz Wotaw. A value-based diagnosis model for java programs. In *Eleventh International Workshop on Principles of Diagnosis (DX)*, 2000. http://www.dbai.tuwien.ac.at/staff/wotawa/dx2000c.ps.gz.

[CW03] Rong Chen and Franz Wotawa. Debugging with an enriched dependency-based model or how to distinguish between aliasing and value assignment. In *Seventeenth International Workshop on Qualitative Reasoning*, 2003.

[Duc93] M. Ducasse. A Pragmatic Survey of Automated Debugging. In *Proc. 1st Workshop on Automated and Algorithmic Debugging*, volume 749 of *LNCS*, 1993. http://citeseer.ist.psu.edu/367030.html.

[JEBH96] Mia K. Stern Joseph E. Beck and Erik Haugsjaa. Applications of AI in Education. In *ACM Crossroads*, 1996. http://www.acm.org/crossroads/xrds3-1/aied.html.

[JS84] W. Lewis Johnson and Elliot Soloway. Proust: Knowledge-based program understanding. In *In Proceedings of the 7th international conference on Software engineering, Florida, United States*, pages 369 – 380, 1984.

[MM97] Fernandes C. Marietto M., Omar N. Tendências nas Áreas de Sistemas de Tutoria Inteligente e Modelagem do Aprendiz. In *VIII Simpósio Brasileiro de Informática na Educação, São José dos Campos*, 1997. http://www.lti.pcs.usp.br/~graca/Docs/Pub/SBIE97.ps.

[MSWW] Cristinel Mateis, Markus Stumptner, Dominik Wieland, and Franz Wotawa. Java ai support for debugging java programs. `http://citeseer.ist.psu.edu/383111.html`.

[PC03] Ron Porter and Paul Calder. A pattern-based problem-solving process for novice programmers. In *Proceedings of the fifth Australasian conference on Computing education*, pages 231–238. Australian Computer Society, Inc., 2003.

[Pro00] Viera K. Proulx. Programming patterns and design patterns in the introductory computer science course. In *Proceedings of the thirty-first SIGCSE technical symposium on Computer science education*, pages 80–84. ACM Press, 2000.

[Rei87] R Reiter. A theory of diagnosis from first principles. *Artif. Intell.*, 32(1):57–95, 1987.

[Sel93] Jonh Self. Model-based cognitive diagnosis. In *AAI, AI-ED Technical Report No 82*, 1993. `http://citeseer.nj.nec.com/self93modelbased.html`.

[SW98] Markus Stumptner and Franz Wotawa. A survey of intelligent debugging. *AI Communications 11*, pages 35–51, 1998. `http://citeseer.nj.nec.com/stumptner98survey.html`.

[Wal01] Eugene Wallingford. The elementary patterns home page, 2001. `http://www.cs.uni.edu/~wallingf/patterns/elementary/`.

[Wen87] Etienne Wenger. *Artificial intelligence and tutoring systems: computational and cognitive approaches to the communication of knowledge*. Morgan Kaufmann Publishers Inc., 1987.

[Wie01] Dominik Wieland. *Model-Based Debugging of Java Programs Using Dependencies*. PhD thesis, PhD Thesis,Technische Universität Wien, 2001.

[XC03] Songwen Xu and Yam San Chee. Transformation-based diagnosis of student programs for programming tutoring systems. *IEEE Trans. Softw. Eng.*, 29(4):360–384, 2003.