Universidade Federal de Pernambuco
Centro de Informática
Graduação em Ciência da Computação

Proposta de Trabalho de Graduação

Transformação de programas Java seqüenciais em concorrentes

Aluno: Rafael Machado Duarte (rmd@cin.ufpe.br)
Orientador: Alexandre Cabral Mota (acm@cin.ufpe.br)

Recife, 10 de Maio de 2005.

1 Contexto

A programação de sistemas concorrentes já vem sendo realizada há bastante tempo, sendo sua utilização motivada pelos benefícios obtidos, especialmente em tipos específicos de sistemas [1]. Mesmo assim, seu uso ainda é bastante restrito a profissionais especializados, pois a programação de sistemas concorrentes é conceitualmente mais difícil de ser realizada e entendida do que a programação seqüencial; isto porque o programador tem que gerenciar a coexistência e coordenação de múltiplas atividades concorrentes. Além disso, o uso de concorrência pode trazer problemas com os quais a maioria dos programadores não está acostumada a lidar: *deadlock*, *livelock*, não-determinismo, etc. Diversas linguagens de programação possuem recursos voltados para a programação concorrente (dentre as quais poderíamos citar C# e C++), dentre elas, Java [6] é uma das que mais se destaca.

Java se tornou uma das linguagens mais populares na comunidade de desenvolvimento [5], poucos porém sabem usar os recursos de concorrência providos pela linguagem. Java provê diversos mecanismos para a inserção e controle da concorrência em seus programas [8]. Para que se possa criar sistemas concorrentes complexos em Java, é necessário dominar bem esses mecanismos, pois o uso de concorrência aumenta bastante a complexidade do sistema e a probabilidade de erros acontecerem; este fato afugenta bastante os desenvolvedores, fazendo com que eles não tirem proveito das vantagens que a concorrência pode trazer para o sistema.

Nesse contexto, foi pensado um modo de facilitar a programação de sistemas concorrentes com o uso de leis de transformação [7]. A intenção é inserir concorrência em um sistema seqüencial garantindo que nenhuma propriedade indesejada será acrescentada. Este ponto é essencial para que se possa efetivamente implementar esse tipo de refactoring em ambientes de desenvolvimento integrados (IDE), o que automatizaria o uso das transformações por desenvolvedores.

2 Objetivos

O objetivo deste trabalho consiste em definir leis de transformação para a inserção de concorrência em programas Java seqüenciais. Para tanto, serão estudados os mecanismos de concorrência em Java e como eles podem ser inseridos em um programa sem que nenhuma propriedade indesejada seja adicionada ao sistema. A intenção inicial é propor leis puramente sintáticas que serão provadas usando argumentos informais; o uso de provas formais não está previsto no presente escopo, pois não há uma semântica formal completa definida para Java, e defini-

la não faz parte do presente trabalho.

Após a a definição de tais leis, suas aplicabilidades serão estudadas utilizando-se um estudo de caso. Também há a intenção de se estudar o ganho de desempenho e a melhoria na organização do sistema obtida após as transformações. O trabalho pretende se basear outras pesquisas que envolvem leis de transformações de programas, só que em contextos diferentes [3, 4, 2].

3 Cronograma

Atividade	*Maio*				*Junho*				*Julho*				*Agosto*			
Estudar concorrência em Java	*	*	*	*	*											
Propor leis de transformação					*	*	*	*	*	*						
Realizar estudo de caso									*	*	*	*	*			
Confeccionar o relatório					*	*	*	*	*	*	*	*	*	*	*	
Preparar a apresentação															*	*

Referências

[1] Greg R Andrews. *Foundations of Multithreaded, Parallel and Distributed Programming*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2000.

[2] P. Borba and A. Sampaio. Basic Laws of ROOL: an Object-Oriented Language. In *Revista de INFORMÁTICA TEÓRICA e APLICADA, Instituto de Informática-UFRGS*, 2000.

[3] Leonardo Cole and Paulo Borba. Deriving refactorings for aspectj. In *OOPSLA '04: Companion to the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications*, pages 202–203, New York, NY, USA, 2004. ACM Press.

[4] ML Cornelio, ALC Cavalcanti, and ACA Sampaio. Refactoring by Transformation. In *Proceedings of REFINE'2002*, Electronic Notes in Theoretical Computer Science, unknown 2002. Invited Paper.

[5] Developer.com. Programming languages popularity, 2004.

[6] James Gosling, Bill Joy, Guy Steele, and Gilad Bracha. *The Java Language Specification Second Edition*. Addison-Wesley, Boston, Mass., 2000.

[7] C. A. R. Hoare, I. J. Hayes, H. Jifeng, C. C. Morgan, A. W. Roscoe, J. W. Sanders, I. H. Sorenson, J. M. Spivey, and B. A. Sufrin. Laws of programming. In M. Broy, editor, *Programming and Mathematical Method*, pages 95–122. Springer, Berlin, Heidelberg, 1992.

[8] Doug Lea. *Concurrent Programming in Java: Design Principles and Patterns*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1996.

Datas e Assinaturas

10 de Maio de 2005

Alexandre Cabral Mota
(Orientador)

Rafael Machado Duarte
(Proponente)