Temas para projecto
Disciplina de Projecto/Seminário do Curso de Especialização

João Miguel Fernandes

27 de Fevereiro de 2004

Este documento apresenta um conjunto de projectos, propostos por João Miguel Fernandes, professor do Dep. Informática da Universidade do Minho, na área dos sistemas embebidos, metodologias de desenvolvimento, engenharia de software e modelação de sistemas.

1 Consistência de modelos UML para software embebido

UML é uma linguagem de modelação de sistemas, composta por vários diagramas, que tem tido um uso cada vez alargado. A existência de vários diagramas introduz um problema que se relaciona com a coerência entre os diversos modelos. Para o desenvolvimento de software embebido, existem algumas propostas que sugerem quais os diagramas a usar e em que sequência.

Este projecto deverá caracterizar quais os mecanismos que podem ser usados para garantir que os vários modelos UML, criados durante o desenvolvimento dum sistema, são coerentes entre si.

1. Küster J., Stroop J., *Consistent Design of Embedded Real-Time Systems with UML-RT*, 4th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC 2001), Magdeburg, Germany, IEEE CS Press, pp. 31–40, Apr/2001. http://www.computer.org/proceedings/isorc/1089/10890031abs.htm.

2. Fernandes J.M., Machado R.J., Santos H.D., *Modeling Industrial Embedded Systems with UML*, 8th ACM/IEEE/IFIP Int. Workshop on Hardware/Software Codesign (CODES 2000), San Diego, California, United States, ACM Press, pp. 18–22, May/2000. http://doi.acm.org/10.1145/334012.334016.

2 Modelação de sistemas de controlo distribuído com profiles UML

A abordagem "Model-Driven Development" (MDD) já mostrou que pode aumentar significativamente a qualidade e a fiabilidade dos sistemas de software. Não existe aparentemente nenhuma razão para que esta abordagem não traga benefícios similares ao desenvolvimento de sistemas embebidos.

Com este projecto deve resultar um método MDD especialmente concebido para sistemas de controlo distribuído, especialmente aqueles baseados em redes de controlo industrial (fieldbus-based systems). O método deve recorrer à notação UML, nomeadamente ao "Profile for Schedulability, Performance and Time Specification".

1. Barbosa M.B., Fernandes J.M., *A Model-Based Approach to the Design of Fieldbus Systems*, IEEE

Real-Time and Embedded Technology and Applications Symposium (RTAS 2004), Toronto, Canadá, IEEE Computer Society Press, May/2004. [em apreciação].

2. OMG. *UML Profile for Schedulability, Performance, and Time Specification*, 2002. http://www.omg.org/uml

3 Transformação de código Java em código AspectJ, via modelos UML

Decorre, neste momento, uma tese de doutoramento, em que o objectivo é construir um catálogo de técnicas de refabricação, com o intuito de transformar programas Java (em que o conceito de classe é nuclear) em programas AspectJ (em que além das classes, também se podem usar aspectos). A finalidade última desse catálogo é, mantendo a funcionalidade do programa original, aumentar a sua legibilidade e manutibilidade.

Este projecto irá definir como a transformação de código Java em código AspectJ pode beneficiar duma modelação intermédia com a linguagem UML. Basicamente é preciso definir mecanismos de transformação de código Java em modelos UML, com o intuito de posteriormente transformar ou usar esses modelos para obter código AspectJ.

1. Pessoa Monteiro M., Fernandes J.M., *Some Thoughts On Refactoring Objects To Aspects*, Taller de Trabajo en Desarrollo de Software Orientado a Aspectos (DSOA'2003), no âmbito das VIII Jornadas de Ingeniería del Software y Bases de Datos (JISBD'2003), Alicante, Espanha, Ed. L. Fuentes, J. Hernandéz e A. Moreira, pp. 55–64, Informe Técnico TR20/2003, Dep. Informática, Universidad de Extremadura, Cáceres, Espanha, Nov/2003.

2. Pessoa Monteiro M., *Refactoring Legacy Objects to Aspects*, PhD. Thesis Proposal, DI, U.Minho, Jan/2003. Disponível em: http://gec.di.uminho.pt/mpm.

4 Métricas para comparação de programas Java com programas AspectJ

Decorre, neste momento, uma tese de doutoramento, em que o objectivo é construir um catálogo de técnicas de refabricação, com o intuito de transformar programas Java (em que o conceito de classe é nuclear) em programas AspectJ (em que além das classes, também se podem usar aspectos). A finalidade última desse catálogo é, mantendo a funcionalidade do programa original, aumentar a sua legibilidade e manutibilidade.

Este projecto irá definir um sistema de métricas de software que permita aferir se os programas refabricados são de facto mais legíveis e mais fáceis de manter. Refira-se que um programa Java pode ser visto como um programa AspectJ sem aspectos (i.e. o compilador de AspectJ aceita programas Java), o que facilita a comparação.

1. Pessoa Monteiro M., Fernandes J.M., *Some Thoughts On Refactoring Objects To Aspects*, Taller de Trabajo en Desarrollo de Software Orientado a Aspectos (DSOA'2003), no âmbito das VIII Jornadas de Ingeniería del Software y Bases de Datos (JISBD'2003), Alicante, Espanha, Ed. L. Fuentes, J. Hernandéz e A. Moreira, pp. 55–64, Informe Técnico TR20/2003, Dep. Informática, Universidad de Extremadura, Cáceres, Espanha, Nov/2003.

2. Pessoa Monteiro M., *Refactoring Legacy Objects to Aspects*, PhD. Thesis Proposal, DI, U.Minho, Jan/2003. Disponível em: http://gec.di.uminho.pt/mpm.

5 Definição dum paradigma de computação paralela baseado em objectos e aspectos

Este projecto corresponde a uma tarefa do projecto "PPC-VM: Portable Parallel Computing based on Virtual Machines", financiado pela FCT. A descrição deste tema segue em inglês.

This task is to design or extend an object-oriented parallel programming paradigm and to implement it on a prototype. Current paradigms are assessed and limitations to develop scalable parallel applications identified. A new paradigm is developed that follows a declarative parallelism approach, where the programmer specifies a large number of fine-grained parallel activities that can be coalesced during execution and mapped to a wide range of computing platforms, including reconfigurable hardware. This kind of approach does not rely on compile time/programmer grain-size decisions: at compile-time, fine-grained tasks are specified and, at run-time, the adequate grain-size will be selected, based on the running conditions. The programmer is responsible to specify all the parallelism opportunities. Support for distributed execution, including transparent remote object creation and invocations are also provided. The new paradigm is compared to the object + thread class and RMI approach provided in Java. This task includes a full implementation of the proposed paradigm on top of an existing virtual machine and its evaluation with a medium complexity case study.

Several sub-task are identified:

T1.1. Comparative assessment of relevant object oriented parallel paradigms, namely their support for fine-grained objects that can be dynamically mapped to a wide range of computing platforms.

T1.2. Design and/or extend an object oriented parallel programming paradigm and its implementation on top of a VM.

T1.3. Development of a medium complexity case study and asses its performance evaluation.

These subtasks are executed by the sequence listed, with a small overlap among them. Subtask 1.2 will require the most of person*month power. Subtasks 1.1 and 1.2 can begin independently from other tasks, but the implementation part of T1.2 strongly requires some of the services implemented on task 3.