1 Introduction

This was the first workshop to be held on this topic at ECOOP and was attended by approximately 15 participants from around the world.

The workshop was organised into two sections – the morning session contained paper presentations followed by in-depth discussion in the afternoon session. This report first presents some background to the workshop and explores its terms of reference and then briefly summarises the afternoon’s discussion. The remainder of this chapter then includes the papers presented at the workshop.

1.1 Reflection and Real-Time

Reflection [1, 2] within a computing system can be defined as the process of performing computation on an internal representation of the system and, by doing so, changing the underlying system in a controlled manner. Conversely, changes in the system are reflected in changes in the internal representation with a causal link between the two ensuring they remain consistent.

A reflective system can thus divide its computation into two separate parts – computation about the system (non-functional computation) and that about the problem itself (functional computation). This separation has been termed a separation of concerns.

The real-time properties of a system are an example of a non-functional system requirement. Many other non-functional properties are possible, for example fault-tolerance or security, and a system may include any combination. Non-functional properties are orthogonal to functional properties they may be, at least in theory, ‘mixed-in’ to a system without requiring modification to functional code.

Real-time presents a number of unique challenges due to the introduction of another axis of measurement – time. The concerns of a real-time programmer require temporal guarantees of behaviour (in addition, and separate from, existing guarantees of functional behaviour). These temporal guarantees have traditionally been provided through static analysis of the system to arrive at a priori guarantees that the system is schedulable even in the event of worst case behaviour [3]. This can give rise to severe problems, for example, it may not be possible to check schedulability until the final stage of development at which point any required changes are expensive.
Consequently, there is considerable demand for flexible systems that can adapt to changing environments, unexpected faults, etc. The ability to mix-and-match requirements and the potential for compile and run-time adaptation offered by a reflective system is very appealing and has lead to a number of reflective real-time systems and also prompted the motivation for this workshop.

2 Presentations

There were six presentations at the workshop, five of which are included in this report:

- *Weak Protection for Reflective Operating Systems*, Shigeru Chiba, Takeshi Nishimura, Kenichi Kourai, Atsushi Ohnoki & Takashi Masuada, The University of Tokyo, Japan.
- *Reflective ORBs: Supporting Robust, Time-critical Distribution*, Ashish Singhai, Aamod Sane and Roy Campbell, University of Illinois at Urbana-Champaign, USA.

The sixth paper, which is not included in this report, was on the workshop organiser’s project, *Design for Validation* [4], and was used as an introduction to the workshop and also to present the organiser’s research background.

3 A Summary of Workshop Discussions

As a result of the discussion session, participants at the workshop created a list of eleven points for consideration. This section lists each of these points in turn and outlines the motivation for inclusion on the list and also some results of discussion undertaken at the workshop. Some of the points were discussed in depth and a consensus reached, others remain more open research questions.

3.1 What do we need to reflect upon for real-time?

There are many properties upon which one may wish to reflect (monitor and possibly influence) in a real-time system. Apart from the obvious such as CPU time used, a flexible real-time system will also require such properties as the current “value” of a task. The consensus reached by the workshop participants was that a complete list would be impossible to compile since there are an infinite possible properties. However, this was not seen as a problem since using a combination of compile and run-time reflection one can open-up the run-time system to obtain the required information.