THE ADVENTURES OF SELFMAN

Project no.	034084
Project acronym:	SELFMAN
Project title:	*Self Management for Large-Scale Distributed Systems based on Structured Overlay Networks and Components*

European Sixth Framework Programme
Priority 2, Information Society Technologies

Deliverable reference number and title:	D.6.5a First progress and assessment report with lessons learned.
Due date of deliverable:	July 15, 2007
Actual submission date:	July 15, 2007
Start date of project:	June 1, 2006
Duration:	36 months
Organisation name of lead contractor for this deliverable:	UCL
Revision:	1.0
Dissemination level:	CO

Contents

1 Executive summary

The first year of the SELFMAN project has started to realize the goal of combining structured overlay networks and component models to build self-managing applications. We have built robust structured overlay networks that we intend to use in the rest of the project. We have designed a transaction protocol that we intend to use in the next year of the project. We have learned several design rules for building self-managing systems. We have started to reformulate the design of structured overlay networks in the terminology of self-managing systems. We have designed a rich component model with a formal semantics that can express the self-management operations. The first year of the project has made progress on many fronts, all of them apparently related to self management. In the second year of the project we expect to be able to focus this work more narrowly as we understand better exactly what is needed for self management.

2 Contractors contributing to the Deliverable

All contractors have contributed to this deliverable: **UCL**, **KTH**, **ZIB**, **INRIA**, **FT R&D**, and **NUS**.

3 Results

This deliverable gives some of the initial lessons learned from the first year of the project. Since much of the work is still in progress, we cannot give many conclusive lessons. However, some things can be said. The first year of the project has achieved its four goals:

- To build robust structured overlay networks with the right mechanisms for self management. See:
  - Deliverable D1.1: Report on low-level self-management primitives for structured overlay networks.
  - Deliverable D1.3a: First report on security in structured overlay networks.
- To design a programming model and component architecture that can express all the self management operations. See:
  - Deliverable D2.1a: Report on basic computation model.
  - Deliverable D2.2a: Report on architectural framework specification.
  - Deliverable D2.3a: Report on formal operational semantics (components and reflection).
- To define a transaction protocol that works over a structured overlay network. See:
  - Deliverable D3.1: First report on formal models for transactions over structured overlay networks).
- To collect relevant application scenarios. See:
  - Deliverable D5.1: User Requirements.

In addition to these deliverables, the Periodic Activity Report summarizes the activities done by the partners to achieve these goals. It gives a global overview of the work in the project and explains how this work fits together to achieve the project's overall goals. We give here some of the important lessons learned that are scattered throughout these documents:

- **Design rules**. The most important lessons learned regarding self management are summarized in Deliverable D2.1a. There we give a first set of design rules for self-managing systems and a first reformulation of a structured overlay network (the relaxed ring) in terms of a feedback loop architecture.
- **Event-driven architecture**. More lessons are given in Deliverable D2.2a, which proposes an asynchronous event-driven component-based middleware architecture that is a good substrate for building self-managing applications. Experience of several partners on different fronts, starting from the work in Subproject 3 of the EVERGROW project in 2004, has led to this architecture. The book of Guerraoui and Rodrigues on reliable distributed programming

is another indication that this architecture is a useful default. Finally, the experience we have with asynchronous computation using the Oz language also points in the same direction.

- **Asynchronous versus synchronous**. The SELFMAN project will build two self-managing architectures that have quite a different foundation: a Java-based synchronous architecture and an Oz-based asynchronous architecture. An example using the first is the DKS architecture, which uses locking for its join algorithm. An example using the second is the relaxed ring architecture of P2PS, which uses no locking but only asynchronous message passing for its join algorithm. We expect that further progress in these two directions will give us a deeper understanding of how to build synchrony in large-scale distributed systems.