Dependable On-line Upgrading of Distributed Systems

Alexander Romanovsky
University of Newcastle upon Tyne, UK
alexander.romanovsky@newcastle.ac.uk

Iain Smith
Dependable Systems Ltd., UK
iain.smith@dependable-systems.com

1. Introduction

Systems are upgraded to improve their functionality, provide new services, correct faults and accommodate changes in the system environment. The number of application areas in which systems should be able to deliver continuous reliable service is growing. Today's systems still experience considerable downtime (in range of 30%-60%) on upgrading and patching the operating system, middleware, and applications. On-line system upgrading is nowadays quickly becoming an issue affecting the success of many an enterprise. It can, for example, reduce the downtime of telecommunication services, allow for on-the-fly bug corrections of space mission software and make it possible to dynamically change a complex Internet application built as an integration of existing independent web services.

The complexity of modern applications is very high and on-line upgrading should not be performed in an ad hoc fashion. Several international organisations (including OMG and Java Community Process) are working on proposals for specifying models and APIs supporting on-line system upgrades. There is a need to design general systematic and practical methods and techniques supporting development and deployment of upgradable systems.

System dependability is a crucial property that must not be undermined during upgrading or as a result of it. Besides, it should be possible to use the natural redundancy of having both the upgraded and old software to allow for a smooth reversion to the old version if necessary.

2. Aims and topics

The main theme of the workshop is to develop approaches to dependable systematic on-line system upgrading. The workshop aims at

- bringing together practitioners, researchers and system developers working on the issues related to on-line upgrading of distributed systems
- developing a better understanding of the problems that developers face while dealing with on-line system upgrading
- defining a research agenda for developing distributed upgradable systems.

We sought submissions from both industry and academia on all topics related to on-line upgrading of distributed systems. These include, but were not limited to:

- software architectures for upgradable systems
- tools and platforms supporting dependable on-line upgrading
- error detection and recovery during system upgrading
- software engineering issues of developing systems that are easy to modify on-line
- structuring techniques to make systems upgradable and on-line upgrading reliable
- formal approaches to reasoning about properties of systems during and after their upgrading
- exception handling techniques for system upgrade
- fault tolerant techniques guaranteeing consistency of upgrading
- approaches addressing issues of interoperability/compatibility of different versions
- employing redundancy and diversity during and after system upgrading
- issues of on-line upgrading of complex systems built of components and legacy code
- techniques to support on-line dealing with interface changes
- dynamic upgrading of database systems.

3. Workshop program

The workshop program includes the following invited talks:

- M. E. Segal (Telcordia Technologies, USA). Online Software Upgrading: New Research Directions and Practical Considerations
- L. Moser (Eternal Systems, Inc., USA). Online Upgrades Become Standard.

Regular papers accepted for publication in the workshop proceedings are:

- C. Dislis (Motorola Ireland Ltd., Ireland). Improving Service Availability via Low-outage Upgrades.
- H. Evans (Glasgow University, Scotland). Dynamic On-line Object Update in the Grumps System.
- C. Jones, A. Romanovsky, I. Welch (University of Newcastle upon Tyne, UK). A Structured Approach to Handling On-Line Interface Upgrades.

Proceedings of the 26 th Annual International Computer Software and Applications Conference (COMPSAC'02)
0730-3157/02 $17.00 © 2002 **IEEE**

- C. Liu, D. J. Richardson (University of California, Irvine, USA). Using RAIC for Dependable On-line Upgrading of Distributed Systems.
- R. P. Bialek (University of Copenhagen, Denmark). The Architecture of a Dynamically Updatable, Component-based System.
- P. Brada (University of West Bohemia in Pilsen, Czech Republic). Metadata Support for Safe Component Upgrades.
- R. Pandey, S. Malabarba, T. Stapko, B. Hashii (University of California, Davis, USA). Dynamically Evolvable Distributed Systems.
- X. Shan, J. J. Li (Avaya Labs, USA). A Case Study of Dependable Software Upgrade with Distributed Components.
- M. R. V. Chaudron (Eindhoven University of Technology, The Netherlands), F. van de Laar. (Philips Research Laboratories, The Netherlands). An Upgrade Mechanism Based on Publish/Subscribe Interaction.
- M. Solarski (FOKUS, Germany), H. Meling. (Norwegian University of Science and Technology, Norway). Towards Upgrading Actively Replicated Servers on-the-fly.

4. Problem scope

This *Dependable On-line Upgrading of Distributed Systems* workshop broadens the discussion from the 'simple' upgrading of objects in a closed environment to the asynchronous integration of upgraded components (objects, tasks, interfaces, etc.) in a distributed, independently managed, heterogeneous environment in which dependability is key. Inevitable in this environment is the possibility of upgrade errors and mismatches.

The scope of the problem is daunting. A complete solution will have to address many issues from the installation of unanticipated changes (bug fixes) to the installation of significant changes in interfaces and function in large distributed systems, all while minimising the impact on the running system and maintaining its dependability and security.

Dependability demands (of reliability and performability) lead to the distinction between anticipated (planned) and unanticipated change, and sizing the scope of the change (no external impact, functional, interface). Unanticipated changes, typically bug fixes, often have no external impact and can be applied independently, while planned functional changes usually require synchronisation or managed late binding to handle the changed interfaces/functions.

Large, complex systems also impose the practical requirement for asynchronous upgrades, and the concurrent support of multiple versions of any component. Handling interface differences may involve techniques from self-describing messages with sophisticated version identifiers to dynamic rebuilding of communication links. Upgrading of complex systems requires employing structured techniques that clearly identify the scope of upgrading (this can be defined either off-line or on-the-fly). Some upgrades replace simple elements of system structure, while others may replace a number of interconnected components.

The challenges of scaling to large systems may require automated upgrade distribution and runtime replacement mechanisms. Simply guaranteeing that all impacted components have been upgraded is non-trivial, while recognising the components that may not be upgraded (because of interdependencies etc.) further complicates the process. Some components may be readily quiesced in a saved state prior to replacement, while long running components will have to have their state transferred to the upgrade.

The systems must remain dependable both during and after upgrade. Some level of transactional ACID (Atomicity, Consistency, Isolation and Durability) properties is required with the ability to commit or rollback the changes. The fault handling process must recognise and resolve faults resulting from the introduction of upgrades.

Finally, the upgrade process must always be secure against intrusion and unauthorised use.

The papers accepted for the workshop address many of the issues outlined above and we believe that we have put together a strong workshop program. In particular, paper by Dislis supplies a practical example of the need to reduce upgrade outages, Evans uses containers each combining a number of JVMs to control the upgrades, Liu and Richardson take the redundant array (RAIC) approach to upgrading, Jones et al provide an overview of the problem of on-line interface changes with an approach to handling the resulting faults, Bialek proposes a layered architecture to handle state transfers, Brada puts forward an upgrade scheme based on revision ID metadata, Pandey et al address evolution of Java programs with an emphasis on introducing type-safe dynamic program modification and an adaptive security model, Shan and Li give an example of using modelling to validate the architecture prior to upgrade, Chaudron and van de Laar describe publish/subscribe upgrading, Solarski and Meling introduce a technique for runtime upgrading of distributed replicated software.

Workshop web page: www.cs.ncl.ac.uk/people/alexander.romanovsky/home.formal/doluds.html

Acknowledgments. We are grateful to Mark E. Segal and Louise Moser for accepting our invitations to deliver invited talks, and to the authors of all submitted papers for their interest in the workshop theme. Our sincere thanks go to the members of the workshop Steering and Program Committee: Cliff Jones, Jenny Li and Francis Tam.