Lecture Notes in Computer Science 2596
Edited by G. Goos, J. Hartmanis, and J. van Leeuwen

Springer
Berlin
Heidelberg
New York
Hong Kong
London
Milan
Paris
Tokyo

Alberto Coen-Porisini
André van der Hoek (Eds.)

Software Engineering and Middleware

Third International Workshop, SEM 2002
Orlando, FL, USA, May 20-21, 2002
Revised Papers

Springer

Volume Editors

Alberto Coen-Porisini
Università degli Studi dell'Insubria
Dipartimento di Informazione e Comunicazione
via Ravasi, 2, 21100 Varese, Italy
E-mail: alberto.coenporisini@uninsubria.it

André van der Hoek
University of California, Irvine
School of Information and Computer Science
444 Computer Science Building
Irvine, CA 92697-3425, USA
E-mail: andre@ics.uci.edu

Cataloging-in-Publication Data applied for

A catalog record for this book is available from the Library of Congress

Bibliographic information published by Die Deutsche Bibliothek
Die Deutsche Bibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data is available in the Internet at <http://dnb.ddb.de>.

CR Subject Classification (1998): D.2, C.2.4, D.3.4

ISSN 0302-9743
ISBN 3-540-07549-6 Springer-Verlag Berlin Heidelberg New York

Springer-Verlag Berlin Heidelberg New York
a member of BertelsmannSpringer Science+Business Media GmbH

http://www.springer.de

Printed in Germany

Typesetting: Camera-ready by author, data conversion by DA-TeX Gerd Blumenstein
Printed on acid-free paper SPIN 10927625 06/3142 5 4 3 2 1 0

Preface

The 3rd International Workshop on Software Engineering and Middleware (SEM 2002) was held May 20–21, 2002, in Orlando, Florida, as a co-located event of the 2002 International Conference on Software Engineering. The workshop attracted 30 participants from academic and industrial institutions in many countries.

Twenty-seven papers were submitted, of which 15 were accepted to create a broad program covering the topics of architectures, specification, components and adaptations, technologies, and services.

The focus of the workshop was on short presentations, with substantial discussions afterwards. Thus, we decided to include in this proceedings also a short summary of every technical session, which was written by some of the participants at the workshop.

The workshop invited one keynote speaker, Bobby Jadhav of CalKey, who presented a talk on the design and use of model-driven architecture and middleware in industry.

We would like to thank all the people who helped organize and run the workshop. In particular, we would like to thank the program committee for their careful reviews of the submitted papers, Wolfgang Emmerich for being an excellent General Chair, and the participants for a lively and interesting workshop.

December 2002

Alberto Coen-Porisini
André van der Hoek

Summary of Technical Sessions

Summary of Session I – Architectures

The first session of the workshop was devoted to architectural aspects of middleware and three papers were presented. The first two papers focused on integration of components while the third paper focused on Quality of Service in the context of embedded systems.

In the first paper the author claimed that current middleware provides an excellent support for building two-tier and three-tier architectures but many large scale applications require different kind of architectures in order to meet their specific requirements. The approach presented consists of a Flexible Process Topology (FPT) architecture that is a set of concepts for custom and flexible process topologies. The main idea, introduced in the first paper, is to decouple application code, process topology and data distribution. The way in which such decoupling is carried out is by introducing generic object managers in each process of the topology chosen for building the application. During the "after presentation" discussion it was pointed out that there are relationships between the FPT approach and Architecture Design Languages (ADL).

The second paper proposed an architecture allowing interoperation of heterogeneous distributed components, by means of the Uniframe Resource Discovery Service (URDS) that provides services for automated discovery and selection of components. The work presented has been carried out in the context of the Uniframe project aiming at providing a framework for interoperation based on a meta-component model, namely the Unified Meta Model, and on the ability of defining and validating Quality of Services requirements both at the component and system levels. The presentation focused on the architecture of URDS that allows client components to issue queries for locating components providing the desired QoS and functional requirements.

Finally, the third paper focused on Consumer Electronics Embedded Systems (CEEMS) where QoS resource management is a very important issue. The work presented consists of an architecture addressing such issue in order to maximize the output quality of applications . The architecture, named HOLA-QoS, is composed of a number of layers handling the main system entities so that it becomes possible to select and set system configurations and to monitor and adapt system behavior in case of faults or when interfacing with external agents.

Summary of Session II – Specification

In this session two papers were presented addressing the issue of specification in the context of middleware-based systems. The papers, however, presented two different viewpoints when dealing with specification: the first one presented a way for specifying the behavior of components, while the second one provided

an overview of the work done by the authors in specifying the interfaces for a standard driven software architecture in a specific domain, namely E-learning.

In the first paper the focus was on the current limitations of standards for developing distributed systems, which are limited to the specification of interfaces and do not address the problem of specifying the behavioral aspects behind the interfaces. The authors claimed that the usual approach consisting of an external behavioral specification has some drawbacks. For example, one has to keep the consistency between the specification and the implementation, changes cannot be done dynamically, since any modification in the specification leads to recompiling (part of) the system. Thus, the approach proposed is to provide the behavioral specification inside components rather that outside. Thus, the paper presents a set of abstraction provided by means an Event/Rule Framework.

The second paper focused on a methodology for specifying interfaces for a specific domain, namely E-learning. The methodology is based on the application of the Unified Software Development Process together with other methodologies found in literature. From their perspective, the authors viewed middleware systems as the technology they chose to use for building an application and thus the concentrated on the standardization of the E-learning domain.

Summary of Session III – Components and Adaptation

One of the motivations for middleware was to allow multiple, possibly distributed components to be connected to interoperate. One of the nice things about middleware was that it was supposed to handle components written in multiple languages, essentially proving the glue among implementations.

One of the most discussion-provoking themes in this session was the notion that we are now moving from building middleware to connect components to building middleware which connects middleware. This is more than simply a mapping of languages and interfaces, but also requires an adaptation of the support structure, such as events mechanisms. The ideal adaptation mechanism of choice is a meta-model which is shared by all middleware. But wait! If this is a step in which middleware connects middleware, then the next step is middleware which connects middleware which connects middleware, and we have landed in a recursive function with no base case.

The fundamental question may be how we ended up with so many middleware systems in the first place.

It is a given that the industry will always come up with incompatible models, and, because of competition, it is in their interest to do so, so there will never be THE component model, or THE middleware. Oddly (or perhaps not), this is not specific to software. Even in hardware, vendors compete for advantage. Some standards do survive, but even still, there will be four or five different options for each kind of component the industry standardizes.

New things are always different from the old, but eventually the competitors will consolidate — once the competition is over. Does anyone really believe this? Assuming this claim is false, then we will be in this game for a long time, constantly building new middleware, each different from the last. Therefore,

a critical research issue is whether we can automatically generate mediation between middleware. Unfortunately, given the history of automatic generation versus hand tuning, automatic versions are likely to suffer poor performance, and will likely simply lose features where it is not known how to translate.

The good news is that a simple translation mechanism may be all that is needed in order to enable coordination among components of multiple middleware. The further good news is that people usually agree on simple standards, it is when things start to become complicated that people (researchers) usually start to disagree. Disagreement leads to competition and competition leads to the persistence of multiple technologies.

Generally, middleware is still in the *maturing* phase, but in the long run, the best the software engineering world can expect of middleware may be a reference model which describes the common characteristics of middleware. One positive gain from the study of adaptation systems is the close examination of multiple middleware systems, yielding a better understanding of what middleware is actually doing and what it should be doing (if anything).

As a brief summary, in the papers in this section you will see multiple types of adaptation. 1. adaptation of different services implemented for the same component model. 2 adaptation between different component models, but ignoring the semantics of the components. 3. adaptation of components to provide services for the application, providing services in a different way, gluing services together, tweaking services so that they work in a new environment, and 4. reconfiguration of the architectural models. Conceivably all four papers in this section could be combined into a single system with the ability to adapt components to provide the necessary functionally, provide the adaptation of components from different frameworks to a unified framework, and use type adaptation to adapt to the new components.

Summary of Session IV – Technology

The papers presented in this session deal with system performance from different perspectives. "An Evaluation of the Use of Enterprise Java Beans 2.0 Local Interfaces" investigates the tradeoffs of using EJB local interfaces to improve system performance in case of co-located method calls and "Dynamic Instrumentation for Jini Applications" presents an approach that supports the instrumentation of Jini services.

For nicely clustered EJB applications where cluster internal beans are completely concealed from the outside local interfaces are well suited to improve system performance. In many scenarios as found in real world applications, however, the use of local interfaces complicate the design and program complexity because facade beans with dual interfaces need to be provided or otherwise cluster sizes increase exponentially.

A problem of local interfaces is that they do not provide access transparency. Parameters are passed to local interfaces using a call by reference invocations semantics instead of a call by copy invocation semantics. Additionally, clients

need either to request the local or remote interface explicitly. Hence, intra-node and inter-node calls are inherently different unlike the design taken by Emerald.

Although EJB2.0 does not meet the above requirements it should be possible to build a system on top of EJB2.0 that finally generates the necessary local and remote interfaces and provides the necessary access transparency while preserving the performance of co-located invocations.

In Jini, a service is registered at a lookup service together with a proxy. The proxy provides a Java interface for the client to be able to interact with the service, therefore, shielding the client from the wire protocol. When a client needs to interact with the service it retrieves the proxy from the lookup service and interacts with the service. The second paper introduces instrumentation at Jini's infrastructural level by wrapping Jini proxies with a monitor that can record various resource consumption data. The proxies use the dynamic proxy API of jdk-1.3 for wrapping the proxies. Using this approach Jini services can be instrumented dynamically at run-time. Modification of Jini's lookup service is not required since proxies are introduced by changing the service's lease time.

Summary of Session V – Services

In this session, three papers related to services that use or are embedded into middleware, are presented. In the first paper, "Message Queuing Patterns for Middleware-Mediated Transactions", the authors present a set of design patterns for distributed transaction management supported at the level of message-oriented middleware (MOM). In the second paper, "Towards Dynamic Reconfiguration of Distributed Publish-Suscribe Middleware", a novel algorithm that improves the performance of dynamic reconfiguration of distributed publish-suscribe systems is presented. The third paper, "Active Replication of Software Components", presents active replication as an alternative to CORBA's replication, to overcome its drawbacks, such as multicast and logging overheads, and lack of tolerance of non-deterministic behavior.

There are some common elements among the papers. The first and the third paper deal with the issue of reliability in distributed systems; the first is about transaction reliability, while the third deals with the issue of reliability in virtual synchrony. The first and second papers have in common the issue of messaging, but with different assumptions: reliable vs. unreliable communication channel.

Although the works presented have common elements, it unlikely that any single middleware could incorporate all of them in an integrated fashion, due to conflicting requirements such as reliable vs. unreliable communication channels.

Committee Listings

General Chair

Wolfgang Emmerich (University College London, UK)

Program Committee Chairs

Alberto Coen-Porisini (Università dell'Insubria, Italy)
André van der Hoek (University of California, Irvine, USA)

Program Committee

Gordon Blair (Lancaster University, UK)
Alfred Bröckers (Adesso, Germany)
Peter Croll (University of Wollongong, Australia)
Flavio De Paoli (Università di Milano Bicocca, Italy)
Premkumar Devanbu (University of California, Davis, USA)
Naranker Dulay (Imperial College, London, UK)
Wolfgang Emmerich (University College London, UK)
Rachid Guerraoui (EPFL, Switzerland)
Arno Jacobson (University of Toronto, Canada)
Mehdi Jazayeri (TU Vienna, Austria)
Wojtek Kozacynski (Rational, USA)
Peter Löhr (Free University of Berlin, Germany)
Dino Mandrioli (Politecnico di Milano, Italy)
Julien Maisonneuve (Alcatel, France)
Koichiro Ochimizu (Japan Institute of Technology, Japan)
David Rosenblum (University of California, Irvine, USA)
Gustavo Rossi (Universidad Nacional de La Plata, Argentina)
Isabelle Rouvellou (IBM Watson Research Center, USA)
Dirk Slama (Shinka Technologies, Germany)
Stefan Tai (IBM Watson Research Center, USA)
André van der Hoek (University of California Irvine, USA)

Table of Contents

Technologies

Services

Invited Talk