HELSINKI UNIVERSITY OF TECHNOLOGY
DIGITAL SYSTEMS LABORATORY
Series **A:** Research Reports ISSN 0783–5396
No. **50**; March 1998 ISBN 951–22–3996–5

PROCEEDINGS OF THE HECSE WORKSHOP ON EMERGING TECHNOLOGIES IN DISTRIBUTED SYSTEMS

Ilkka Niemelä (Ed.)

Digital Systems Laboratory
Department of Computer Science and Engineering
Helsinki University of Technology
Otaniemi, FINLAND

Helsinki University of Technology
Department of Computer Science and Engineering
Digital Systems Laboratory
P.O.Box 1100
FIN–02015 HUT, FINLAND

HELSINKI UNIVERSITY OF TECHNOLOGY
DIGITAL SYSTEMS LABORATORY
Series **A**: Research Reports
No. **50**; March 1998

ISSN 0783–5396
ISBN 951–22–3996–5

Proceedings of the HeCSE Workshop on Emerging Technologies in Distributed Systems

Ilkka Niemelä (Ed.)

Digital Systems Laboratory
Department of Computer Science and Engineering
Helsinki University of Technology
Otaniemi, FINLAND

Printing: Libella Painopalvelut Oy 1998

Helsinki University of Technology
Department of Computer Science and Engineering
Digital Systems Laboratory
P.O.Box 1100
FIN–02015 HUT, FINLAND

Phone: $\frac{09}{+358\text{–}9}$ 4511
Telex: 125 161 htkk fi
Telefax: +358–9–451 3369
E-mail: lab@saturn.hut.fi

Preface

The term "Information and Communications Technologies (ITC)" is presently commonly used to characterise the family of theories, methods, and technologies emerging from the marriage of the previously distinct information and telecommunications technologies.

At the present, many traditional areas of telecommunications are revolutionalised by novel approaches from computing technologies, from Internet telephony to digital television. The qualitative difference between the brave new world and the older technologies is the relative openness of the underlying communications platforms and methods. In particular, the Internet is open to all who wish and are able to develop and promote new technologies, middleware, or applications. The fierce competition makes progress very fast: months in the ordinary world correspond years on the Internet. From the viewpoint of computing, the Internet and related technologies have equally important qualitative effects. The computer on our desk is no longer just a thing that computes; it is a window to an ocean of data.

The HeCSE Workshop on Emerging Technologies in Distributed Systems, held in Lammi, Jan 12–13, 1998, was motivated by these developments. Its goal was to offer the postgraduate students of HeCSE, the Helsinki Graduate School of Computer Science and Engineering, and their mentors and professors an opportunity to present and discuss work related to open distributed systems. A mixture of invited talks, paper presentations, and short poster presentations were given in a lively and relaxed atmosphere during two days. This document records the formal part of the meeting; equally important, of course, is its informal side – the interaction, discussions, and ideas arising from these.

I'd like to express my warm thanks to other members of the organising committee for their help during the preparation of the workshop:

Kari Alho (Helsinki University of Technology)
Lea Kutvonen (University of Helsinki)
Ilkka Niemelä (Helsinki University of Technology)
Martti Tienari (University of Helsinki)

Special thanks are due to Inkeri Verkamo for arranging the venue and other practicalities of the workshop, and to Ilkka Niemelä for editing these proceedings. I also thank all participants for creating the workshop with us.

February 1998

Prof. Martti Mäntylä
Director of HeCSE

Table of Contents

Invited Talks

A Comparison of CORBA, DCOM and RMI 1
Kari Alho

Analysis of Distributed Processes: Can Java Programs be Verified? 3
Nisse Husberg

Internet Agents for Engineering Process Modelling and Support 5
Martti Mäntylä

Introduction to Wireless Application Protocol (WAP) 9
Jari Ojala

Submitted Papers

Architectures for Distributed Systems: Open Distributed Processing Reference Model 10
Lea Kutvonen

A Survey of Techniques and Guidelines for Improving the Performance of CORBA-based Distributed Systems 23
Pekka Kähkipuro

Fast Address Lookup for Internet Routers 33
Stefan Nilsson and Gunnar Karlsson

Posters

Framework for Distributed Special Needs Education Using an Agent-Based Learning Environment 43
Jaakko Kurhila

A Client-Server Tool for Image Retrieval Applications 45
Pyry Kuusi, Kjell Lemström, Petri Kanerva, Sami Koponen, Sari Männynsalo, Juha Riihimäki, Janne Teinilä, and Timo Virtanen

A Comparison of CORBA, DCOM and RMI

Kari Alho
Helsinki University of Technology
Department of Computer Science and Engineering
Laboratory of Information Processing Science
P.O. Box 9555, FIN-02015 HUT
Kari.Alho@hut.fi

Abstract

Distributed object-oriented programming frameworks have recently received significant attention. The enormous expansion of the WWW has created a vast demand for modern distributed programming technologies for the Internet. The three major contestants are CORBA, DCOM and Java RMI. This talk will briefly introduce the three technologies, and compare their architectures and programming layers.

CORBA is a middleware specification for heterogeneous distributed systems. It is governed by OMG, an industry consortium of over 700 companies. CORBA defines a standard architecture for distributed programming, APIs for using basic services through an ORB (Object Request Broker), and standard ways to define and implement application-specific interfaces. A large number of commercial and research implementations of CORBA realize the specification on different hardware and programming languages. However, the implementations differ significantly in the extent that they realize CORBA Services, a set of useful add-on APIs (e.g., transactions, security) for all application domains.

DCOM is the distributed extension to COM (Component Object Model), Microsoft's object framework. COM is in extremely widespread use on Windows desktops. DCOM emerged with NT 4.0, it is built on top of DCE RPC, and Software AG has been chartered to port it to non-Window platforms. Since DCOM is an add-on to COM, which is a binary standard, it is more tedious to the programmer, who must, e.g., handle memory management. A future version of COM, called COM+, should make the programming interface simpler.

Java RMI defines and implements Remote Method Invocation of Java objects. The programming model is simple, one extends predefined classes, and almost all new methods can thereafter work remotely. RMI is also well integrated into the Java language (since it is not supported in other languages). RMI is standard in the Java language since JDK 1.1 release. One problem is that many current web browsers are not yet based on JDK 1.1.

One can conclude that the three technologies are architecturally quite similar – all offer similar basic services. Programming is different, but COM+ will make DCOM very similar to CORBA. The semantics of inheritance differ: only CORBA support multiple inheritance of interfaces, but DCOM and RMI allow one object to implement multiple interfaces. There seems to exist a market for all three: CORBA is strong in enterprise-wide solutions and legacy applications, DCOM on PC desktops, and RMI in Java applications that can be deployed in Internet.

Analysis of Distributed Processes: Can Java Programs Be Verified?

Nisse Husberg
Helsinki University of Technology
Department of Computer Science and Engineering
Digital Systems Laboratory
P.O. Box 1100, FIN-02015 HUT
Nisse.Husberg@hut.fi

Abstract

The analysis of distributed processes is reviewed based on 20 years experience with theoretical research in concurrent formalisms, especially high level Petri nets and distributed transition systems, and reachability analyzer design. The approach is mainly generator based, i.e. the behaviour is derived from a formal model of the processes. A *directed graph* can be used as the common basis for the formalisms and the category which is generated from it will contain the different behaviours of the system, both interleaved and truly concurrent.

The experiences from the *Emma* TNSDL analyzer shows that it is possible to model the features of a real concurrent programming language but the more complex data types will make the reachability analysis difficult because large data value ranges will cause a combinatorial explosion in the size of the reachability graph. The analysis must therefore make extensive use of reduction methods and abstraction.

There has, however, been a considerable improvement in practical analysis methods during the last years including efficient storage techniques, smart state coding and on-the-fly analysis. No method will be useful for all kinds of systems and it will be necessary to offer a *multi-method* analyzer. Also incomplete verification methods will be needed as a complement in some cases where the reachability graph is infinite or extremely large. The aim must not be to generate as many states as possible but rather as few as possible provided that all possible error states are included.

Very general analyzers are not so efficient and it will be better to have a number of specialized analyzer which makes it possible to use the specific features of the input language to restrict the necessary analysis. This will be the principle of the *Modular Reachability Analyzer (Maria)* which is under development at the Digital Systems Laboratory.

The inclusion of time into the model usually adds another degree of complexity. It has, however, been possible to model the behaviour of timers in SDL using an *expiration window* concept. A similar technique can be used to remove unnecessary concurrency and obtain a smaller reachability graph.

The new modular analyzer will make it easy to add new methods and algorithms. It will have interfaces to other tools and several task-specific frontends e.g. for SDL making it possible to work in a well known environment.

Thus it will be possible to verify concurrent programs e.g. in Java, but especially the inclusion of complex data types will be problematic. A lot of both theoretical and practical work is still needed before a useful industrial analyzer can be designed.

Internet Agents for Engineering Process Modelling and Support

Martti Mäntylä
Helsinki University of Technology
Department of Computer Science and Engineering
Laboratory of Information Processing Science
P.O. Box 1100, FIN-02015 HUT
Martti.Mantyla@hut.fi

Extended Abstract

A long-standing research area in engineering applications of computing is the integration of heterogeneous systems such CAD, CAM, production management, product data management, and workflow management. Various methods, tools, and techniques have been identified in this work, notably standards for product data representation (STEP and related ISO standards) and protocols for accessing such data (SDAI). The area has also proved attractive for methods and tools developed elsewhere, such as in distributed object systems (CORBA, Java), computer-supported co-operative work (CSCW), and artificial intelligence. Yet integration of engineering systems has proved a tough nut to break. Although this difficulty must be attributed to several sources, one major culprit for the limited success is that the actual engineering processes that underlie the systems are unknown, badly understood, or complex – or most often, all three of these.

In this work, I have found it useful to systematically avoid making "simplifying assumptions" on the integration potential of engineering processes. That is, I do not assume that the methods, systems, and tools used by engineers are compatible, or that they can directly exchange information. Nor do I assume that shared databases are available or feasible. Virtual enterprises give one case that seems to justify this stance.

In its purest form, a virtual enterprise is a dynamically created entity composed of a unique collection of collaborating partners for the purpose of satisfying a unique customer need, and dissolving itself after its purpose has been

fulfilled. Such an organisation can be competitive if it is capable of tapping fully into the competence of its partners, and if it can apply best practices within the limits of these competencies.

To allow this, best practice processes must be made visible and available to the dynamic consortium that forms a virtual team. In addition, the processes must be augmented with applicable information and communications technologies (ICT) to reduce the interaction costs to an acceptable level and to facilitate communication of experts from different disciplines and partner companies.

At the present, the Internet and its technologies are rapidly changing the basic ICT platform that forms the basis of process integration. With the rapid emergence of Internet-savvy CAD, PDM, document management, workflow management, and other engineering systems, the technological prerequisites for engineering process integration are rapidly becoming a reality.

The research described in this presentation is based on the belief that even with these tools, engineering integration will remain half-hearted at best unless the underlying, shared knowledge of engineering processes can be made explicit, visible and sharable by its participants, and applicable to tailor and control engineering systems and system integration frameworks. In other words, integration must be based on an explicit and shared domain ontologies for engineering – the shared nomenclatures describing products, their structure in terms of subsystems and components, the engineering activities that build them, the resources and actors that are needed to implement the activities, and the various relationships between such ontology entities.

Tom Gruber, the father of Ontolingua, defines simply:

> "An ontology is a specification of a conceptualization. That is, an ontology is a description of the concepts and relationships that can exist for an agent or a community of agents for the purpose of enabling knowledge sharing and reuse."
> [http://www-ksl.stanford.edu/kst/what-is-an-ontology.html]

Shared ontology forms the basis of communication by giving a meaning to the terms and structures used. It also represents a commitment made by the communicating partners to obey the defined constraints, relationships, and semantics in their communication.

A shared ontology for engineering, therefore, should codify concepts and connotations for describing products, engineering processes, organisations responsible for carrying out engineering tasks, and all types of relations, constraints, and axioms that may be applicable.

To be useful, an ontology need not be all-encompassing or globally applicable. Nor does it need to be perfectly defined before it can be used as a basis of communication. On the contrary, we believe that during their life-cycle useful ontologies for engineering are born to cover specialised domains, grown in scope, mutated to cover further activities, split in parts to reuse their best practices in novel areas, and merged again – in short, they are adapted to varying business conditions and needs. It is precisely in this area of ontology development and life-cycle where we aim to make a significant contribution over the state of the art.

In the present work, a concept system for modelling shared engineering process ontologies and agent protocols for engineering process integration is being developed. Process and product ontologies of the system are based on the following base entities:

Artefact: Artefacts are the things created as the final or intermediate product during an engineering process. Available artefacts may also be used as a basis of further redesign activities. Artefacts form a taxonomy of different types of things, such as specifications, test plans, user manuals, software modules, and CAD models (and their fine-grained constituents, such as geometric features).

Activity: Activities represent the actual work being performed during engineering. An activity is a temporal thing: it starts and ends somewhere in (known or unknown) time. It is also a process in that it consumes some input artefacts and resources and produces some output artefacts. Similarly to artefacts, activities also form a taxonomy.

Actors: Actors are the things that make activities happen; they are intended to model generically the people and organisations that really do the engineering work. They might also be used to model other useful things to have such as money, physical space, or machine. Unlike artefacts, which are "short-lived" things, actors are thought of as fairly rigid things that evolve only relatively slowly (as compared to the typical duration of activities). As before, they form a taxonomy.

In addition to the basic entities (and their taxonomic siblings), the base ontology has relations, modelling various sorts of dependencies between the entities. Like the other entities, relations form taxonomies.

A specific objective of the base ontology is to grasp the time evolution of engineering processes. Therefore, all entities of the model have a life-cycle model associated to them. A life-cycle model can be thought of as a finite state machine which is associated to a model entity. State transitions are controlled by events either from the user of the system, or propagated by other entities as a side effect of other state transitions.

Distributed coordination of agents is based on shared model entities created through a matchmaking and dialogue control protocol. In addition to shared entities, proxies are used to diminish network traffic due to event propagation.

The system makes use of methods for knowledge sharing via Internet agents resulting from various USA based projects such as SHADE and SHARE. In particular, the system is based on a combination of a C++-based knowledge representation system implementing a frame storage, a Lisp-based definition language providing access to frames and tools for their manipulation, and a KQML-based agent communication facility. Sample ontologies and protocols developed so far cover integrated product and process modelling, life-cycle management, and change propagation.

Introduction to Wireless Application Protocol (WAP)

Jari Ojala
Nokia Research Center
Software Technology Laboratory
P.O. Box 422
(Itälahdenkatu 22B, 00210 Helsinki)
FIN-00045 NOKIA GROUP
Jari.Ojala@research.nokia.com

Abstract

Wireless Application Protocol (WAP) is a proposal made by the Wireless Application Protocol Forum Ltd. (WAP Forum) to create an industry-wide specification for developing applications and services over wireless communications networks. WAP will provide access to Internet, Intranet and operator services and defines a general purpose application programming model to deliver services to wireless devices. WAP architecture supports multi-vendor interoperability by defining optional and mandatory components. Founding members of the WAP Forum are Ericsson, Motorola, Nokia and Unwired Planet but the WAP Forum has established a procedure to accept new members. The complete WAP specification proposal was made public in February 1998 and is available on the Internet: www.wapforum.org.