Network of Excellence in Distributed and Dependable Computing Systems
IST Contract No. IST-2000-25088

CaberNet Vision of Research and Technology Development in Distributed and Dependable Systems

Edited by Alexander Romanovsky
and
Richard Snow

List of Contributors

Chapter 2: David Powell (LAAS), Denis Besnard (Newcastle University)

Chapter 3: Sandro Etalle (Twente University), Pieter Hartel (Twente University), Heiko Krumm (Dortmund University), Paulo Verissimo (University of Lisbon), Peter Ryan (Newcastle University)

Chapter 4: Francois Armand (Jaluna), Gerhard Fohler (Maalardalen University), Peter Puschner (TU Vienna), Manuel Rodriguez (Critical Software), Andy Wellings (York University)

Chapter 5: Roberto Beraldi (Roma University), Barbara Hughes (Trinity College), Rene Meier (Trinity College), Hugo Miranda (University of Lisbon), Luis Rodrigues (University of Lisbon), Paulo Verissimo (University of Lisbon)

Chapter 6: Sacha Krakowiak (Joseph Fourier University), Francois Armand (Jaluna)

Chapter 7: Geoffrey Coulson (Lancaster University), Maarten van Steen (Vrije University)

Chapter 8: Markus Hillenbrand (Kaiserslautern University), Valerie Issarny (INRIA), Savas Parastatidis (Newcastle University), Ferda Tartanoglu (INRIA), George Vasilakis (FORTH), Marcus Venzke (Hamburg-Harburg TU), Friedrich Vogt (Hamburg-Harburg TU)

Chapter 9: Holger Peine (Fraunhofer IESE)

Chapter 10: Alberto Montresor (Bologna University), Matthias Wiesmann (EPFL), Dietrich Fahrenholtz (Hamburg-Harburg TU), Ricardo Jimenez-Peris (TU Madrid), Marta Patino-Martinez (TU Madrid)

Chapter 11: Dirk Henrici (Kaiserslautern University), Michael Kleis (GMD FOKUS), Paul Mueller (Kaiserslautern University), Bernd Reuther (Kaiserslautern University), Detlef Bosau (Stuttgart University)

Chapter 12: Miguel Castro (MS Research), Elisa Turrini (Bologna University)

Chapter 13: Emil Lupu (Imperial College), Mike Fisher (BT), Morris Sloman (Imperial College)

Chapter 14: Santosh Shrivastava (Newcastle University)

Chapter 15: Latella Diego (ISTI CNR), Mieke Massink (ISTI CNR), Gethin Norman (Birmingham University), Dave Parker (Birmingham University), Cliff Jones (Newcastle University)

Chapter 16: Rogerio de Lemos (Kent University), Valerie Issarny (INRIA)

Foreword

CaberNet[1] is the Network of Excellence (NoE) in distributed and dependable systems. It is funded by the European Commission's ESPRIT Programme. Its mission is to co-ordinate top-ranking European research in distributed and dependable systems, to make it accessible to governments and industries and to enhance the quality of professional training with regards to such systems. CaberNet addresses all aspects of networked computer systems design. These systems range from embedded systems used to control an aircraft in flight to globe-spanning applications searching for information on the World-Wide Web.

This document presents a CaberNet vision of Research and Technology Development (RTD) in Distributed and Dependable systems. It takes as a basis the state-of-the-art (SOTA) Report[2] prepared by John Bates in 1998: this document was commissioned by CaberNet as a first step towards the definition of a roadmap for European research in distributed and dependable systems. This report overviewed the developments in the main areas to which the CaberNet members made outstanding contributions, which were the most important at the time of its preparation, and analysed the most important trends in R&D in those areas.

The NoE is the collective author of this new document, which was put together by integrating contributions coming from many CaberNet partners. A dedicated CaberNet workshop (November 2003, Porto Santo, Portugal) and a one-day meeting of the CaberNet Links-to-Industry Forum (December 2003, London, UK) were organised to consolidate the Network understanding of the RTD Vision.

The Vision document is intended to serve as a policy-directing document. But it is equally valuable as a high level overview of recent and current activities in the selected areas, emphasising directions in which R&D in distributed and dependable systems are likely to be moving in the future.

Since CaberNet started, several very extensive roadmapping activities have been initiated by the Commission on topics of direct relevance to CaberNet's interests. We have used our involvement in some of these activities and their results (in particular, those of the AMSD and ARTIST roadmapping projects) and this, clearly, adds value to our work. Moreover, we would not have been able to prepare the vision document as it is now without using and referring to these roadmaps. But we would like to emphasise that the aim of this document is to present a vision, something which is distinctly different from a roadmapping activity: it has a broader focus and contains a more general, visionary view of the future and of the main current trends; it focuses on a number of selected topics of CaberNet excellence without attempting to give a complete coverage of the very wide total area of distributed and dependable systems; and, lastly, the document looks into a longer-term future for which we believe it would not be reasonable to define concrete milestones. On a practical note, this activity had available to it far less in the way of resources than any of the recent roadmapping activities sponsored by the Commission, this document being just one of a number of CaberNet's outputs.

To ensure the continuity of this work with respect to the SOTA document we took the structure of the SOTA report as a starting point but a number of changes have been made to reflect recent trends.

My responsibility as the editor of the Vision document was mainly focusing on combining the contributions from a large number of individual CaberNet members, resolving minor mismatches, letting people express their views and integrating their opinions and visions into the document. I am grateful to all the contributors for their willingness to work on the document, to all members of the CaberNet Links-to-Industry Forum for sharing the industry views and concerns with us, to all participants of the 5th CaberNet Plenary workshop for helping me in shaping the document structure and for their many suggestions on how to improve it, to Brian Randell, Valerie Issarny and Rogerio de Lemos for their invaluable advice on how to approach and conduct this work, to Valerie Issarny and Brian Randell for their useful suggestions on how to improve the document structure, and, finally, to all members of the CaberNet Executive Board for their support and assistance.

Alexander Romanovsky

Newcastle upon Tyne, UK
January 2004

[1] http://www.newcastle.research.ec.org/cabernet/

[2] J. Bates. The State of the Art in Distributed and Dependable Computing. A CaberNet-Sponsored Report. October 1998. http://www.newcastle.research.ec.org/cabernet/sota/index.html

Foreword to the Revised Edition

Following the Final Review meeting of CaberNet held in March 2004, the reviewers asked for a number of modifications to be made to the CaberNet Vision document. We are grateful to the reviewers for pointing out ways in which the document could be improved.

Unfortunately, Dr Romanovsky was unable to undertake the task of editing the revised version, so this responsibility fell to me, as Director of CaberNet. I am very pleased, however, to acknowledge the informal advice and assistance that Dr Romanovsky has so willingly given.

I would also like to thank all those CaberNet colleagues who have provided additional material, and who have willingly given of their valuable time to read and comment on this new version. Especially, may I record my gratitude to Geoffrey Coulson, Cliff Jones, Paul Müller, Peter Puschner, Brian Randell, Peter Ryan and Paulo Verissimo for their individual contributions.

Richard Snow

Newcastle upon Tyne, UK
July 2004

Table of Contents

Part I. Introduction

1 Introduction

1.1 The Aims

This document presents a CaberNet vision of Research and Technological Development (RTD) in Distributed and Dependable systems. It is intended to assist policy-makers in business, government and academia, by providing a high level overview of recent and current activities in selected areas, outlining directions in which R&D in distributed and dependable systems is likely to be moving in the future.

Since CaberNet started, several extensive roadmapping activities have been initiated by the Commission. We have used our involvement in some of these activities and their results and this, clearly, adds value to our work. In particular the CaberNet members have been very much involved in two roadmapping projects: AMSD and ARTIST.

ARTIST (Advanced Real-Time Systems Information Society Technologies)[3] is a 3-year IST project started in April 2002 with the objective of coordinating the R&D effort in the area of advanced real-time systems by

- improving academics' and the industry's awareness of the area, especially of existing innovative results and technologies, standards and regulations
- defining innovative and relevant work directions, identifying obstacles to scientific and technological progress and propose adequate strategies for circumventing them.

AMSD (Accompanying Measures on System Dependability; 2002-2003)[4] was a 1-year IST project aiming to:

- achieve a higher level synthesis of the results of the many on-going road-mapping activities related to various aspects of system dependability, identify gaps and overlaps, and produce an overall road-map for the subject of system dependability
- develop a detailed road-map, covering the various aspects of dependability for one particular class of system, dependable embedded systems, as a contribution both to the planning of research on embedded systems and, alongside road-maps being produced by other projects, to the development of the overall dependability road-map
- identify the constituents and balance of a possible wide-ranging Information Society Dependability Initiative within FP6 that will gain the widespread support of industry, governments and academia
- undertake constituency and consensus-building activities that will help to mobilise this support and to maximise the likelihood of a successful outcome from such an Initiative.

We would not have been able to prepare the document as it is now without using and referring to this work. But we would like to emphasise that the aim of this document is to present a vision, which is distinctly different from these roadmapping activities: (i) it has a broader focus and contains a more general, visionary view of the future and of the main current trends; (ii) nevertheless, it focuses on a number of selected topics of CaberNet excellence without attempting to give a complete coverage of the entire wide area of distributed and dependable systems; and, (iii), as a Vision document it looks into a longer-term future for which we believe it would not be reasonable to define concrete milestones.

It is worth noting that while there has been focused roadmapping on dependability (AMSD), no clearly comparable work on distributed systems exists (which is understandable considering the wide focus of the topic). The present report is unique in covering these two together and their synergy. However, on a practical note, readers should be aware that this activity had available to it far less in the way of resources than any of the recent roadmapping projects sponsored by the Commission, it being just one of the activities, and this document just one of the outputs, of CaberNet.

[3] http://www.systemes-critiques.org/ARTIST/Overview/

[4] http://www.am-sd.org

1.2 CaberNet SOTA Report

Our document takes as a basis the state-of-the-art (SOTA) Report[5] on the state of the art in distributed and dependable computing that was prepared by John Bates in 1998. This report was commissioned by CaberNet in 1997 as a first step towards the definition of a roadmap for European research in distributed and dependable systems. It was generally accepted that the SOTA report was a success in mapping out the worldwide State-Of-The-Art and in presenting a balanced analysis and individual perception of Europe's position in 1998 with reference to the worldwide State of the Art. In addition to analysing technological developments, the report assessed the areas in which Europe was ahead of the rest of the world, and those in which it lagged behind, the relationship between the state of the art in industry and academia and the likely future development and consequences of State of the Art projects that must be taken into consideration.

The 1998 SOTA report was structured into a number of sub-topics each of which was described in an individual chapter. These are the technical chapters that it included:

2 Network Architecture

3 Operating Systems

4 Real-time Operating Systems

5 Dependable Systems

6 Distributed Object Technology

7 Distributed Shared Memory

8 Mobile Agents

9 The World Wide Web

10 Distributed Event-based Systems

11 Mobile Computing

12 Distributed Multimedia Platforms

13 Distributed Groupware

14 Support for (Re)configuration in Distributed Systems

15 Supporting Persistent Data for Distributed Applications

16 Network Storage Services

17 Distributed Systems Management

18 Distributed Systems Security

19 Languages for Distributed Programming

20 Distributed Systems Theory

We would like to stress, however, that although this present document took the SOTA report as its starting point, its purpose is significantly different in that it aims to present a vision of future developments and is therefore less specific than either a detailed survey of the current State of the Art, or the multi-volume roadmap documents referred to earlier.

1.3 Structure of the Present Document

[5] J. Bates. The State of the Art in Distributed and Dependable Computing. A CaberNet-Sponsored Report. October 1998. http://www.newcastle.research.ec.org/cabernet/sota/index.html

This present report comprises four technical sections (System Properties, System Architecture, Advanced Features and Software Engineering), each of which consists of a number of chapters. The System Properties part covers R&D related to ensuring the essential properties of distributed systems: dependability, security, real-timeliness and mobility. The System Architecture part overviews the basic technology stack: operating systems, component technology, service-oriented architecture and mobile agents. The topics analysed in the Advanced Features part are related to providing specific features required for developing novel distributed and dependable systems. The Software Engineering part consists of two chapters covering important phases of the life cycle: rigorous design and software architectures. CaberNet does not regard traditional Software Engineering topics as part of its major research activity, except in these two areas. This was reflected to some extent in the fact that the 1998 SOTA report contained no discussion of traditional Software Engineering topics at all. In total the present document has just fifteen technical chapters.

It was our original intention to reproduce the structure of the 1998 SOTA report as closely as possible, in order to clarify the continuity of the CaberNet position over a number of years. But since that report was written, the area of distributed and dependable systems has been going through a period of revolutionary change, caused by the unprecedented growth of the Internet, by the ever-increasing reliance our society puts on information systems and by various pioneering developments in hardware and communication technologies. To reflect these changes in the RTD of distributed and dependable systems and to better consolidate the current position of the CaberNet members, it was decided to use a somewhat modified structure, compared with that of the SOTA document. Below is the list of technical chapters included in the present document. (The document starts with this introductory chapter and finishes with a conclusion chapter.)

Part II: System Properties

2 Dependable Systems

3 Distributed System Security

4 Real-time Systems

5 Mobile Systems

Part III: System Architecture

6 Operating Systems

7 Distributed Object and Component Technologies

8 Service-oriented Computing: Web Services

9 Mobile Agents

Part IV: Advanced Features

10 Group Communication

11 Distributed Multimedia Platforms

12 Network Storage Services

13 Network and Distributed System Management

14 Control and Coordination in Dynamic Virtual Organisations

Part V: Software Engineering

15 Rigorous Design

16 Software Architectures for Distributed and Dependable Systems

Three chapters now cover much wider topics than the corresponding chapters in the SOTA report: (i) Real-time Systems, (ii) Distributed Object and Component Technologies and (iii) Service-oriented Computing: Web Services. This directly reflects the growing R&D activity in these areas and is clearly indicative of current trends. A new chapter on Software Architectures has been introduced to present the CaberNet vision in this area, one in which a number of the project partners have gained an international reputation. The introduction of a new chapter on Dynamic Virtual Organisations is symptomatic of another important trend - the need to provide high level services and features to support development of current and future distributed systems. This Vision document does not have a chapter on Distributed Shared Memory: we believe that this is an indication of the decreasing interest in the topic. It includes a chapter on Operating Systems (OS), although the main focus of CaberNet is clearly not on OS research. We decided to include this chapter for the following two reasons: to

illustrate the SOTA work in the area by outlining relevant research performed by several CaberNet partners and to provide a brief overview in this area in order to help the readers understand other topics which directly rely on research in operating systems.

Apart from the work focusing on Java and Ada, two languages that are now widely accepted and used in industry, languages for programming distributed systems is not a very active area of research (at least within the CaberNet community) For this reason the Vision document has no separate chapter on this topic, but a subsection dedicated to it is included in the chapter on Real-time Systems.

Unlike the 1998 SOTA report, this Vision document has no chapter on Network Architectures. This should not be taken to suggest a lack of interest in this area: all CaberNet RTD activities clearly rely on and use state-of-the art results in the area. Moreover, some chapters (e.g., that on mobile computing) discuss relevant RTD in Network Architecture in their specific contexts.

Reconfiguration in distributed systems is another topic which has not been given a separate chapter in this Vision report. The decision to drop it was taken, notwithstanding the central role that reconfiguration will play in future distributed and dependable systems, because it was felt that topics related to reconfiguration are better discussed in the contexts of other chapters: service-oriented computing, dependability, software architectures, virtual organisations, etc.

Due to the fact that CaberNet partners have not been active in the area of persistent data, the topic of the relevant chapter in the SOTA document was narrowed down to replicated databases and a special subsection was included in the chapter on Group Communication.

In the light of the growing importance of safety and security we have decided to include in this document dedicated (sub-) sections discussing them, although we fully realise that they are attributes of dependability discussed in Chapter 2, and of distributed systems security, which is the topic of Chapter 3.

Formal methods – now commonly referred to as Rigorous Design – appears as a separate chapter within the section on Software Engineering. As noted earlier, “traditional” software engineering is not a topic of great interest within the CaberNet community, save in the two areas of Rigorous Design and Software Architectures for Distributed and Dependable Systems. It might have seemed appropriate for Rigorous Design to include a discussion of verification and validation. The chapter does include a section on Model Checking, but on the whole, the general topic has been discussed within the individual chapters as it applies to the topic of that chapter; see, for example, Chapter 4 on Real-Time Systems.

Each chapter in this Vision document follows the same pattern: it starts with a brief description of the general theme, followed by a discussion of the current research (focusing mainly on the work by the CaberNet partners) and finishes with an overview of the likely directions of future work and the vision. Since the report, which covers a variety of topics, is a very extensive one, it has been our intention to prepare self-contained and largely independent chapters to make it easier for the reader to use; this is why there is, unavoidably, some repetition. Moreover, some overlaps in the material included in the individual chapters are inevitable due to the multidimensional structure of current research in distributed and dependable systems. For the same reason a few topic areas (such as peer-2-peer, ad hoc networking, aspect oriented programming) do not have dedicated chapter/subsections; instead each of them is discussed in several relevant chapters.

Current R&D in industry is discussed in nearly every chapter, and a number of chapters present the industry views on future trends in the areas. In addition to this, we have included a dedicated subsection in Chapter 17 on the industry vision: it summarises the positions of the members of the CaberNet Link-to-Industry Forum expressed during two meetings (held in 2002 and 2003), which focused on the CaberNet vision of the future RTD of distributed and dependable systems in Europe.

Appendix A contains a list of all the projects in which CaberNet partners have recently been or are now involved. For each of these projects a brief description is provided consisting of the project acronym, the full project title, start and finish date, URL, CaberNet members and other partners involved, the project abstract, the list of relevant chapters of the Vision document and a list of three representative publications reporting outcomes of the project. The intention is to make the main document readable without having to consult the Appendix. All projects mentioned in the main text without references are CaberNet projects; their brief descriptions may be found in the Appendix.

1.4 New Trends

Europe's leading role. The 1998 SOTA report identified a number of possible directions of work that would ensure the leading role of Europe in the RTD in distributed and dependable systems. These include a better marketing of the work by European academics and industrialists in the worldwide community and developing a more confident strategy of feeding European research ideas into commercial products and of the research funding bodies recouping their investments. There have been a number of positive developments in these directions, such as

- the introduction of new instruments in FP6 with a clear emphasis on involving SMEs (small and medium enterprises) to make faster technology transfer possible
- extensive roadmapping activities sponsored by the Commission in a number of areas related to distributed and dependable systems
- national and European support by the relevant funding agencies for large efforts aimed at consolidating work when a large number of partners work in related areas
- direct support of interdisciplinary and multidisciplinary R&D.

CaberNet has clearly contributed to better marketing of the work done by leading European sites active in the areas of distributed and dependable system development.

Academia and Industry. Many contributors to this document are from academia, which we believe is a very important source of the vision into the future RTD of distributed and dependable systems. Many significant achievements in the area of dependable and distributed systems are coming from academia. Several spin-off companies and SMEs have been created in the recent years by academics to ensure direct transfer of their knowledge into industrial developments. These are based around the world-leading research conducted by a number of CaberNet academic partners (for example, TU Vienna on time-triggered architecture, Newcastle University on transactions, etc.). Unfortunately, there are relatively few examples like these. There is still a need for a more focused support to facilitate technology transfer from leading European universities to European industry and to ensure that European academia research is oriented towards the practical requirements of IT systems.

A number of chapters in this report are focused on the work conducted by industry: analysis of real time OS mainly deals with Linux, service-oriented computing is now driven by organisations in which industrial partners play a dominant role (such as W3C[6]), a very similar situation is found regarding the development of component technologies. In these areas it is important to ensure that academia can make a considerable contribution to and have a significant effect on making the important decisions.

New application areas. This document reflects the fact that a number of new application areas are emerging. They include ambient intelligence[7] environments, complex large-scale networked control systems, virtual organisations, advanced Web-based applications (including various "e-s": e-voting, e-government, e-commerce, etc.), large-scale systems supporting complex scientific computations (e.g. based on Grids), novel telecommunication systems, applications based on world-wide distributed active data bases, systems monitoring complex environments, etc. Further progress in these areas will only be made possible by innovative RTD in distributed and dependable systems.

[6] http://www.w3.org/

[7] ISTAG. Scenarios for Ambient Intelligence in 2010. EC. Community Research. IST. 2001.

Part II. System Properties

2 Dependable Systems

Dependability is defined as the property of a computer system that enables its users to place justified reliance on the service it delivers. Dependability is a generic concept, generalizing the notions of availability, reliability, integrity, confidentiality, maintainability, safety and security. The aim of dependability research is thus to define methods to support trust and confidence in computer-based systems. This requires techniques for protecting and assessing systems with respect to a wide spectrum of faults that can be broadly classified in five classes [Avizienis 2001]: physical faults, non-malicious design faults, malicious design faults, non-malicious interaction faults, and malicious interaction faults (intrusions).

In this chapter we first review recent and current work on dependability by CaberNet and then give our vision for research in the next five to ten years.

2.1 Current and Recent Work

Current research in dependability covers a wide spectrum of critical systems, going from embedded real-time systems, to large open networked architectures. A trend in recent years has been an emphasis on commercial off-the-shelf (COTS) components [Arlat 2000][Popov 2002a] and open source software (OSS) [Lawrie 2002][David 2003]. Furthermore, there are signs of the growing maturity of this field of research, as illustrated by the emergence of dependability-specific development methods [Kaâniche 2002].

Dependability methods can be categorized as *fault prevention*, *fault tolerance*, *fault removal* and *fault forecasting*. Fault prevention and fault removal are sometimes considered together as constituting *fault avoidance*, as opposed to fault tolerance and fault forecasting, which together constitute *fault acceptance*.

2.1.1 Fault Prevention

Fault prevention aims to prevent the occurrence or the introduction of faults. It consists in developing systems in such a way as to prevent the introduction of design and implementation faults, and to prevent faults from occurring during operation (see Chapter 15). In this context, any general engineering technique aimed at introducing rigor into the design process can be considered as constituting fault prevention. However, some areas currently being researched are more specific to the dependable computing community. One such area is the formal definition of security policies in order to prevent the introduction of vulnerabilities. The definition of a security policy consists in identifying the properties that must be satisfied and the rules that applications and organizations must obey in order to satisfy them. For example, work being carried out in the MP6 project in France is specifically aimed at defining role-based access control policies applicable to information systems in the health and social sectors [Abou El Kalam 2003a][Abou El Kalam 2003b].

Another area of active research into fault prevention concerns the human factors issues in critical “socio-technical” systems. For example, research initiated at York University on the allocation of functions between humans and machines [Dearden 2000] has served as the basis of a prototype database tool to assist communication between system designers and human factors experts [Mersiol 2002]. In the UK, the Interdisciplinary Research Collaboration in Dependability of Computer-Based Systems (DIRC) has a strong human factors component. The areas covered in this project include human-machine interaction in healthcare systems [Baxter 2003][Tan 2003] and aviation [Besnard et al 2003], sociological aspects of situated work [Clarke et al 2003] as well as human impairments to security [Besnard and Arief 2003].

2.1.2 Fault Tolerance

Fault-tolerance techniques aim to ensure that a system fulfils its function despite faults [Arlat 1999]. Current research is centred on *distributed* fault-tolerance techniques (including fault-tolerance techniques for embedded systems), *wrapping* and *reflection* technologies for facilitating the implementation of fault-tolerance, and the generalization of the tolerance paradigm to include deliberately malicious faults, i.e., *intrusion-tolerance*.

Distributed fault-tolerance techniques aim to implement redundancy techniques using software, usually through a message-passing paradigm. As such, much of the research in the area is concerned with the definition of distributed algorithms for fault-tolerance. Replication is an essential paradigm for implementing distributed services that must remain available despite site inaccessibility or failure [Pinho 2002][Leeman 2003]. Research in this area goes back to the 1980s (see, e.g., [Powell 1988]). The management of replicated groups in distributed systems requires an underlying facility for group communication [Montresor 2001][Kemma 2003][Mena 2003] (Chapter 10) and consensus [Mostéfaoui 2001]. Recent contributions in this area include the notion of External Group Method Invocation (EGMI) developed in the Jgroup/ARM project. EGMI allows in

clients to request services from a replicated group without them having to be members of the group themselves. Another important issue in fault-tolerant replicated groups is that of automatically recovering and re-integrating failed replicas. This topic has been addressed, for example, in the GUARDS project [Bondavalli 1998][Powell 1999] and in the Jgroup/ARM project [Meling 2002].

In closed embedded systems the design of fault-tolerance algorithms may be simplified if it is possible to substantiate the strong assumptions underlying the *synchronous* system model. Therefore, most current research on fault-tolerance in such systems follows this approach, often using the time-triggered paradigm [Powell 2001][Elmenreich 2002][Steiner 2002]. Note also that, especially in embedded systems, state-of-the-art fault tolerance techniques cannot ignore that most faults experienced in real systems are *transient* faults [Bondavalli 2000a].

In many distributed systems, however, especially large-scale systems, it is difficult to substantiate the strong assumptions underlying the synchronous system model, so several teams are defining paradigms able to deal with *asynchrony* (see, for example, [Mostéfaoui 2001]). One approach being followed at the University of Lisbon is to consider a reliable timing channel for control signals that is separate from the asynchronous channel used to carry payload traffic [Casimiro 2002][Veríssimo 2002]. This work has inspired the Trusted Timely Computing Base (TTCB) paradigm developed in the MAFTIA project [Lung 2003].

An alternative approach is to consider a timed asynchronous model, which involves making assumptions regarding the maximum drift of hardware clocks accessible from non-faulty processes. Using this model, a European-designed fail-safe redundancy management protocol [Essamé 1999] is currently being implemented in the context of the automation of the Canarsie subway line in New York [Powell 2002].

Other areas of active research concern fault-tolerance in large, complex distributed applications. Of special note in this area are techniques aimed at the coordinated handling of multiple exceptions in environments where multiple concurrent threads of execution act on persistent data [Beder 2001][Tartanoglu 2003]; fault-tolerance in peer-to-peer systems [Montresor 2002]; recursive structuring of complex cooperative applications to provide for systematic error confinement and recovery [Xu et al 2002] and mechanisms for dealing with errors that arise from architectural mismatches [de Lemos et al 2003].

The implementation of distributed fault-tolerance techniques is notoriously difficult and error-prone, especially when using off-the-shelf components (COTS) that typically (a) have ill-defined failure modes and (b) offer opaque interfaces that do not allow access to internal data without which fault-tolerance cannot be implemented. There is thus considerable interest in addressing these difficulties using wrapping technologies to improve robustness [Rodriguez 2000][Anderson et al 2003] (e.g. within DSoS and DOTS projects) and reflective technologies to allow introspection and intercession [Killijian 2000].

In the mid 1980's, the European dependability community had the (then) outrageous idea that the fault tolerance paradigm could also be extended to address security issues through the notion of *intrusion-tolerance* [Fraga 1985][Dobson 1986][Deswarte 1991]. Such techniques are now receiving a justified revival in interest as it is realized that intrusion prevention (though authentication, authorization, firewalls, etc.), like any other prevention technique, cannot offer absolute guarantees of security. Intrusion-tolerance has been addressed by the European MAFTIA project (see, e.g. [Deswarte 2001][Correria 2002]) and, in the USA, is now the subject of a complete DARPA program (called OASIS, for Organically Assured & Survivable Information Systems), in which European researchers are also taking an active part through the DIT project [Valdes 2002][Deswarte 2003][Saidane 2003].

2.1.3 Fault Removal

Fault removal, through verification and validation techniques such as inspection, model-checking, theorem proving (e.g., see [Nestmann 2003]), simulation (e.g., see [Urbán 2001]) and testing, aims to reduce the number or the severity of faults.

An interesting research area in fault removal, with strong links to fault forecasting, is that of *probabilistic verification*, an approach that aims to provide stochastic guarantees of correctness by neglecting systems states whose probability of occupation is considered negligible. For instance, the VOSS project is addressing the modelling and verification of stochastic aspects of computer systems such as distributed systems, networks and communication protocols [Baier 2002a].

Much of the fault removal research carried out by CaberNet members is focused on software testing, especially with respect to faults (sometimes referred to as robustness testing), and on testing of fault-tolerance mechanisms (via fault injection).

One approach to *software testing* that has been investigated in depth at LAAS-CNRS is *statistical testing*, which is based on the notion of a test quality measured in terms of the coverage of structural or functional criteria. This notion of test quality enables the test criterion to be used to define a statistical test set, i.e., a probability distribution over the input domain, called a test profile, and the number of executions necessary to satisfy the quality objective. Recent research has focused on developing statistical test-sets from UML state diagram descriptions of real-time object-oriented software [Chevalley 2001a] and the assessment of test-sets for object-oriented programs using mutation analysis [Chevalley 2001b].

Recent research at the University of Milan has been directed at the assessment of the correlation between software complexity, as measured by object-oriented metrics, and fault proneness. If such a correlation were to exist then these metrics could be used to make the software testing process more cost effective by testing modules by decreasing order of complexity, as measure by these metrics. A case study on three different versions of a very large industrial object-oriented system (more than 2 million lines of code) shows, however, that object-oriented metrics do not provide a better predictor of fault proneness than the traditional lines of code (LOC) metric [Denaro 2003].

Robustness testing aims to assess how well a (software) component protects itself against erroneous inputs. This work, which is the focus of the AS23 project, aims to assess how well a (software) component protects itself against erroneous inputs. One approach for robustness testing, which was studied within the DSoS project, is called "property-oriented testing". Here, the determination of test profiles is specifically aimed at verifying safety properties, typically of an embedded control system — heuristic search techniques are used to explore the input space (including both functional and non-functional inputs), attempting to push the system towards a violation of its required safety properties [Abdellatif 2001].

Fault injection can either be used for robustness testing, in the sense defined above, or as a means for testing fault-tolerance mechanisms with respect to the specific inputs of such mechanisms, i.e., the faults they are supposed to tolerate [Buchacker 2001][Höxer 2002]. In fact, due to the *measures* of robustness or coverage that this technique allows, fault injection is also often a means of experimental evaluation (see fault forecasting below).

Finally, it is worth mentioning that some current research is focused on testing the use of the reflective technologies considered earlier as a means for simplifying the implementation of fault-tolerance [Ruiz-Garcia 2001].

2.1.4 Fault Forecasting

Fault forecasting is concerned with the estimation of the presence, the creation and the consequences of faults. This is a very active and prolific field of research within the dependability community. Analytical and experimental evaluation techniques are considered, as well as simulation [Baier 2002b].

Analytical evaluation of system dependability is based on a stochastic model of the system's behaviour in the presence of fault and (possibly) repair events [Haverkort 2001]. For realistic systems, two major issues are that of: (a) establishing a faithful and tractable model of the system's behaviour [Fota 1999][Kanoun 1999][Zanos 2001][Betous-Almeida 2002][Haverkort 2002], and (b) analysis procedures that allow the (possibly very large) model to be processed [Bell 2001][Popov 2002b]. The Netherlands-Germany bilateral project VOSS is researching two complementary aspects: how to facilitate the construction of very large models through the integration of probabilistic automata and stochastic automata, and how to apply model checking to very large stochastic models [Baier 2002a].

Ideally, the analytical evaluation process should start as early as possible during development in order to make motivated design decisions between alternative approaches (see [Bondavalli 2001] for an example of some recent research in this direction). Specific areas of research in analytical evaluation include: systems with multiple phases of operation [Bondavalli 2000b][Mura 2001]; and, in the DSoS project, large Internet-based applications requiring a hierarchical modelling approach [Kaâniche 2001][Kaâniche 2003].

Evaluation of system dependability can sometimes be advantageously linked with performance evaluation through joint performance-dependability — or *performability*— measures. For example, [Bohnenkamp 2003] investigates the trade-off between reliability and effectiveness for a recently defined protocol for configuring Ipv4 addresses in ad-hoc and domestic networks.

In the area of software-fault tolerance, specific attention must be paid to modelling dependencies when assessing the dependability achieved by diversification techniques for tolerating design faults [Littlewood 2001a][Littlewood 2001b].

Experimental evaluation of system dependability relies on the collection of dependability data on real systems. The data of relevance concerns the times of or between dependability-relevant events such as failures and repairs. Data may be collected either during the test phase (see, e.g., [Littlewood 2000]) or during normal operation (see, e.g., [Simache 2001][Simache 2002]). The observation of a system in the presence of faults can be accelerated by means of fault-injection techniques (see also fault removal above), which constitute a very popular subject for recent and ongoing research. Most of this work on fault injection is based on software-implemented fault injection (SWIFI), for which several tools have been developed (see, e.g., [Carreira 1998][Fabre 1999][Höxer 2002][Rodriguez 2002]). The data obtained from fault injection experiments can be processed statistically to characterize the target system's failure behaviour in terms of its failure modes [Marsden 2001][Marsden 2002] or to assess the effectiveness of fault-tolerance mechanisms in terms of coverage [Cukier 1999][Aidemark 2002]. Currently, especially in the context of the DBench project, there has been research into using fault injection techniques to build dependability *benchmarks* for comparing competing systems/solutions on an equitable basis [Kanoun 2002][Buchacker 2003][Vieira 2003].

2.2 Future Trends

Our vision for future directions in dependable systems is guided by the Ambient Intelligence (AmI) landscape portrayed in the ISTAG scenarios [ISTAG 2001], which depict an Information Society that will "transform the way we live, learn, work and play" [PITAC 1999][AMSD 2003]. That vision of the Information Society will never be realised unless governments, companies and citizens can confidently place their trust in the correct, continuous, safe and secure delivery of AmI. Dependability research is thus essential to support the AmI landscape. Future directions can be discussed in terms of four key aspects of AmI: *openness*, *mobility*, *adaptability*, and *workability*:

- *Openness.* Systems in the AmI space must have the potential to interact with one another. This introduces the potential for a wide range of new threats to the infrastructure and also introduces the potential for significant new complexity and the concomitant increase in the density of faults in systems.

- *Mobility*. The AmI infrastructure must be capable of supporting both real and virtual mobility of system components (both hardware and software). This again raises the possibility of many new potential faults in the system that need to be addressed.

- *Adaptability*. The AmI infrastructure needs to adapt to rapidly changing environments in order to maintain the required level of service. The extensive deployment of this kind of technology can have both positive and negative impacts on the dependability of systems.

- *Workability*. This addresses issues of both how manageable the system is for the human system administrators and how systems synergize (or fail to synergize) with the work context they are intended to support [Besnard 2003]. This is particularly important when we consider the need for management to be scalable and for the systems to seamlessly blend into the operating context.

These key aspects of AmI are now discussed in terms of research directions in each of the four dependability methods identified earlier.

Fault prevention

The key to fault prevention is *rigorous design*, i.e., the adoption of design practices that eliminate particular classes of error. For example, by adopting appropriate design practices one can avoid deadlock in distributed systems. The term "formal methods" applies to the use of specifications that are completely mathematical and justifications of design and code whose proofs (that they satisfy their specifications) are checked by machine. Although there is a long history of such research, such complete formality is more appropriate for safety-critical applications; given today's tools, complete formality is not cost-effective for most applications. *Rigorous* methods are those "informed by formality". Often the formality informing rigorous methods takes the form of a mathematical model that is adequate to justify the soundness of the methods.

Areas that can be identified as future directions of research into rigorous design methods include:

- *Model Driven Development:* There is a need for formal models that better capture features of system that are key to dependability. For example, *openness* demands an approach to modelling trust, confidence and knowledge in the presence of malicious parties; *mobility* demands models of resource consumption, real-time and real space; *adaptability* demands models of evolving operating

environments and emergent properties of massively distributed systems; *workability* is concerned with modelling human activity, system configuration and the ability of an operator to observe the system. All of these, more or less formal, models inspire approaches to fault prevention (and fault removal) that will target particular classes of fault.

- *Influencing Current Design Practice:* Rigorous design is a pragmatic activity; its primary aim is to package formal insights in a way that makes them applicable in a real-world design context. A particularly important aspect of these methods is their scalability and the search for *compositional* methods that allow scalability through separation of concerns. Most CaberNet teams involved in rigorous design technologies are concerned to influence industrial design practice. Approaches here include influencing standardisation bodies and providing better formal underpinnings to existing design notations and practices. Here, many teams have current activities to influence design in UML that could serve as a route to disseminate rigorous approaches.

- *Tool Definition and Development:* The more rigorous methods can be supported by formal tools, the more likely they are to be adopted and the more likely they are to be correctly applied in practice. There is a plethora of formal methods tools in Europe and the US. We need to define core toolsets for particular classes of application and to enable inter-working across tools. In particular, we need to develop tools that aid the development of dependable information infrastructures and of approaches to moving many more rigorous approaches towards practical use.

Fault tolerance

Fault-tolerance through self-healing and self-protection constitutes a major technology for building systems that can be depended on, for both life-critical and business-critical applications. With respect to the *openness*, *mobility*, *adaptability*, and *workability* aspects of AmI, we can identify the following research challenges.

Openness. Fault-tolerance should be able to provide solutions to the dependability challenges of open networked environments, which include: risks to availability and even safety due to complexity, hidden interdependencies and malicious attacks; compromises to confidentiality and privacy due to intrusions and insider malfeasance. In particular, we need to investigate:

- Innovative protocols for solving basic distributed system problems (e.g., information dissemination, consensus, election, group membership…) in the presence of uncertain delays, accidental faults and malicious attacks.

- Scalable fault-tolerance algorithms that can ensure availability and integrity of computation and data in highly asynchronous and uncertain environments (e.g., for peer-to-peer systems).

- New distributed architectures capable of detecting and/or tolerating malicious intrusions with a view to attaining acceptable availability and confidentiality under attack.

Mobility. Several distinctive features of mobile systems offer significant fault-tolerance challenges:

- Range limitation, obstacles to signal propagation, power source depletion, voluntary disconnections… all mean that network partitioning is common rather than exceptional, so there is a need to re-visit existing fault-tolerance paradigms and consider not only asynchrony, but also extreme models with permanent partitioning in which liveness properties may be difficult or even impossible to define. The migration from connection-centric to data-centric interaction should favour data replication and peer-to-peer tolerance paradigms.

- Compared to fixed networks, there are limited power sources, higher error rates and lower bandwidths, which mean that new fault-tolerance paradigms need to be investigated with delicate trade-offs between computation and communication.

- The susceptibility of wireless communication to eavesdropping, the absence of guaranteed connectivity to trusted third parties for authentication and authorization, the susceptibility of unattended ubiquitous devices to tampering all mean that security issues are of utmost importance. The intrusion tolerance paradigm in such a setting offers an interesting research direction.

- Location-awareness, while opening interesting application perspectives, also raises frightening privacy issues that again might find solutions in tolerance paradigms based on ephemeral pseudonyms.

- New cooperative peer-to-peer models need to be defined and studied from a tolerance perspective, including for example, distributed trust models based on reputation and mutual confidence, micro-economy models for service-sharing, etc.

Adaptability. Self-healing and self-protection are of course forms of adaptation, in that they aim to modify the structure or the control of a system in order to react to the discovery of faults. However, challenging topics linked specifically with adaptation can be identified:

- Adaptation of fault-tolerance techniques to the current level of threat (e.g., adaptive replication, adaptive encryption…) and the means to carry out such adaptation (e.g., reflective systems, aspect-oriented programming, exception-handling…)

- Use of adaptation techniques for self-optimisation and self-management of complex systems in the face of a wide variety of accidental and malicious threats.

- Tolerance of deficiencies in adaptation algorithms, especially those based on heuristics or on learning.

Workability. There is also a need to consider techniques for tolerating classes of human-related faults that have hitherto received little attention:

- Configuration faults: it is difficult to configure complex systems, especially in the face of changing environments and requirements. Such faults can lead to catastrophic failures [METT 1993] and are a major source of vulnerabilities that could be exploited for malicious attack [Whitten and Tikkar 1999]. Techniques are needed for discovering and correcting such faults on-line, and adapting other configuration elements accordingly.

- Data faults: the quality and the consistency of data are fundamental in critical data-intensive systems. We need techniques that can tolerate poor data through identification and correction procedures, or possibly by masking techniques that exploit data redundancy. Human agents are seen as a privileged approach, since humans are often better than machines at identifying semantic discrepancies, possibly through associated metadata.

- Observation faults: inadequate system observability is a major source of operator mistakes (which can in turn lead to configuration and data faults). Hence there is a need to explore the tolerance paradigm as a complement to traditional prevention techniques, exploiting the capacity of humans to adapt to degraded levels of observability and to aggregate multiple sources of disparate information. These latter issues have been identified in the literature [Hollnagel 1987] and now constitute a strong basis for future research.

Fault removal

To build dependable information infrastructures for AmI, a rigorous modelling and design process must be followed including verification and validation (V&V) activities aimed at fault removal. V&V activities address the analysis and testing of developed artefacts to ascertain that the resulting (intermediate and final) products form:

- A *consistent construction*: when going from more abstract to more concrete models (refinement); when considering differing views at a same level of a system (projection); and when combining interacting pieces that were developed separately (composition);

- A *useful infrastructure***:** with respect to the services delivered and the ambient requirements, considering functional as well as dependability aspects, namely reliability, timeliness, robustness, and similar.

The primary challenges posed to existing V&V methods and techniques in the context of the *openness*, *mobility*, *adaptability*, and *workability* aspects of AmI, result from:

- *Scale:* the *pervasiveness* and *interdependency* of the information infrastructures needed to realize a trustworthy AmI lead to an incessant growth in the size and complexity of the systems that have to be analyzed. This factor affects all four aspects.

- *Dynamicity:* the common denominator of *adaptability*, *openness* and *mobility* is that they make *a priori*, off-line V&V activities inadequate, because we can no longer rely on a stable model or design of the system being built. To address this, one possible approach is to seek methods for reasoning about

infrastructures undergoing dynamic changes, and meta-models (e.g., component-based models of the architecture) that can characterize evolving infrastructures. Another approach is to entrust basic dependability properties (e.g., safety, security) to kernel subsystems that are simple enough to respond to established V&V methods.

- *Data intensiveness:* this is an issue especially for *workability* and *mobility*. While methods and techniques have mostly focused in the past on checking the functional behaviour of a system, in data intensive systems the consistency of the data also needs to be verified, and the links between data faults and threats to dependability identified.

- *Malicious faults*: in the context of security protocols for open systems, faults could be injected during the communication by malicious agents in order to forge unjust authentication or to steal secret information.

The primary means by which the above challenges might be tackled are:

- *Abstraction:* effective abstraction mechanisms need to be adopted in methods and techniques used for analyzing and testing a large complex information infrastructure, so that dependability checks can be carried on models describing the system at the relevant level of abstraction.

- *Composition:* methods and techniques are required to verify the dependable evolution/adaptation of the infrastructure, resulting from the intentional/accidental change of the interconnected component structure. Hence, we need to investigate verification techniques aimed at checking properties of the dynamic composition of system components, and test approaches allowing for component-based testing as well as test asset reuse when systems are reconfigured by component insertion/removal/substitution.

- *Complementarity of approaches:* we need to consider mutual influence and integration between approaches addressing dependability from different backgrounds and perspectives, e.g., formal verification with model-based testing, or conformance testing with fault injection techniques. We also need to extend testing and formal verification to deal with non-functional aspects, most notably hard and soft real-time properties.

Fault forecasting

System evaluation is a key activity of fault forecasting, aimed at providing statistically well-founded quantitative measures of how much we can rely on a system. In particular, system evaluation achieved through modelling supports the prediction of how much we will be able to rely on a system before incurring the costs of building it, while system evaluation achieved through measurement and statistical inference supports the assessment of how much we can rely on an operational system.

These two basic techniques are by no means mutually exclusive, although they have traditionally been applied at different stages of the system life cycle. Our opinion is that research into both techniques is needed as well as into synergistic collaboration between the two throughout the system life cycle.

A number of new issues are raised by the *openness*, *mobility*, *adaptability*, and *workability* aspects of AmI:

- Ambient Intelligence, in particular the *openness* aspect, raises the need for dependability evaluation to encompass not only unexpected accidental faults but also intentional faults perpetrated by malicious entities. Clearly, there is a need to develop modelling paradigms that will enable the evaluation of comprehensive dependability measures taking into account both types of faults in an integrated way. Such modelling paradigms must be based on sound assumptions concerning the vulnerabilities of the target systems and the attackers' behaviour. Such assumptions should be derived from direct measurements and observations.

- In many systems of the future, humans will form a vital link in the complete dependability chain. In this respect, the adaptive nature of human behaviour is an important phenomenon [Besnard and Greathead, 2003]. Hence, it is of crucial importance to model and evaluate *the impact of human/operator operational behaviour* on overall system dependability. Current (technically-focused) system evaluation approaches to a large extent neglect this issue.

- Future systems will adapt themselves to new and evolving circumstances. To assess the dependability of such systems, system evaluation techniques may have to be enhanced so as to *cope with increasingly dynamic systems*.

- Future systems will heavily rely on *mobile technologies*. To provide an adequate level of service, fault-tolerance techniques will be devised to provide a certain level of service even in the presence of disconnections. System evaluation needs to provide adequate support for the development and evaluation of these techniques.

As a consequence, the following research themes can be identified:

- An *expanded role for system evaluation activities* to further integrate them into the system life cycle (there is a need for a stronger interaction with the other dependability methods) and to better understand the limits of system evaluation (how much we can rely on modelling and measurement given the uncertainties in the real system being modelled or measured).

- Better assessment – and better integration of this assessment into system evaluation – of the *dependability effects of human activities* (development, maintenance, use of, and attacks against, computer-based systems), currently often limited to generic assessments of "quality" and/or conformance checks about the application of prescribed precautions.

- *Attacking complexity of system descriptions.* There is a definite need for composition, supported by property-driven abstractions in the definition of models and measurements, and also a need to provide macro structure to models, for example using layered approaches, to reflect the layered structure of real systems. Since different formalisms are suited to the description of different system characteristics, forcing the use of a single formalism may result in an inadequate and non-faithful representation of some relevant aspects. We, therefore, advocate a *multiformalism system evaluation environment* in which formalisms can work together in a synergistic manner, allowing different modelling styles (with different models of time, e.g., continuous, discrete, deterministic or stochastic) to be applied at different stages of the evaluation.

- *Attacking complexity of solutions.* By solution, we mean computation of both classical dependability properties and temporal logic properties, possibly in a multiformalism and heterogeneous context. We envision three main sources of complexity: *large state spaces*, the presence of *rare events*, and *new classes of stochastic models* and their analysis.

- *Measurements* in complex and open systems. Measurement-based assessment of large scale deployed networked systems raises several challenging problems, encompassing controllability and observability issues, as well as the definition of suitable dependability measures. Such problems are exacerbated when mobility is taken into account. Also, several research issues are raised when considering the collection and processing of real data that is needed to understand and characterize malicious attacks, and to model the impact of these attacks on computer system security. We need to develop a methodology and suitable means for collecting and analysing such data. Particular attention needs be put on the characterization of attacker behaviour based on the collected data and the definition of stochastic models that will enable the system administrators and designers to assess the ability of their systems to resist potential attacks.

- *Benchmarking for dependability*, to provide a uniform, repeatable, and cost-effective way of performing the evaluation of dependability and security attributes, either as stand-alone assessment or, more often, for comparative evaluation across systems and components. The shift from system evaluation techniques based on measurements to the standardized approaches required by benchmarking touches all the fundamental problems of current measurement approaches (representativeness, portability, intrusion, scalability, cost, etc.) and needs a comprehensive research approach, with a special attention to COTS and middleware aspects.

3 Distributed System Security

This chapter focuses on several challenges that we consider strategic for the coming years in the area. The first challenge concerns bridging the gap existing between the formal and the computational view of cryptography. The second with identifying and modelling side-channel leakages due to timing, power consumption, EM radiation etc. properties of the system. The third challenge is guaranteeing the secure operation of the IT systems. The fourth challenge is concerned with intrusion tolerance.

3.1 Current and Recent Work

The field of security is undergoing a major change. Designing secure systems is currently an art [Anderson 2001], but there are clear signs that it is gradually becoming more of an engineering discipline. Formal methods are transforming this field radically; in the last few years they have been applied with great success in the analysis of security protocols. In fact, thanks to the use of techniques such as model checking, many security protocols, previously considered secure (in some cases, for many years), were eventually found to harbour subtle flaws, see for example Lowe's uncovering of a flaw in the Needham-Schroeder Public Key protocol, [Lowe 1996]. Nowadays, it is clear that a formal verification phase will (eventually) become commonplace in the development of security-critical protocols. This can rightly be regarded as a first step on the path of the renovation of security engineering: a renovation that will lead to new formal tools and engineering methodologies for the development of secure critical software. This step is a significant one, yet it is very small in comparison to what still needs to be done. Even within the topic of verification of security protocols there are more unsolved problems than solved ones; for instance while the verification of unicast (peer-to-peer) secrecy and authentication protocols is currently a simple matter of employing the right tools, in order to handle the more complex scenarios that are emerging there are problems that need to be addressed. For instance, clear challenges for the short and medium period are:

- Handling multicast protocols. In many real-life situations, for instance in wireless networks, an agent is asked to participate in a protocol together with a number of partners that is not known in advance. For this, a number of so-called multicast protocols have been devised, ranging from multicast authentication to multicast nonrepudiation, often using restricted proxies. Standard techniques for the verification of security protocols cannot deal with the multicast case: for this we have to develop and implement new abstraction techniques, for example using interdependence concepts [Lazic 1999].

- Handling negotiation, payment, abuse-freeness and fairness. Preliminary work in this direction has been done using tools (based on game semantics) such as the Mocha model-checker [Alur et al 1998]. However, it cannot deal with (symbolic) communication, which is crucial for verifying protocols admitting malicious participants. See also work of Schunter [Schunter 2000].

Comment [p y a1]: Not sure that I understand this last sentence.

Under a systemic perspective, another revolution is taking place, in terms of the confluence of the fields of fault tolerance and security. Whilst they have taken separate paths until recently, the problems to be solved are similar in nature: to keep systems working correctly, despite the occurrence of mishaps, which we could commonly call faults (accidental or malicious); to ensure that, when systems do fail (again, on account of accidental or malicious faults), they do so in a non harmful/catastrophic way. In classical dependability, and mainly in distributed settings, fault tolerance has been the workhorse of the many solutions published over the years. Classical security-related work has on the other hand, with few exceptions, concentrated on intrusion prevention, or intrusion detection without systematic forms of processing the intrusion symptoms. A new approach has slowly emerged during the past decade, and gained impressive momentum recently: intrusion tolerance (IT)[8].

In the rest of this chapter we focus on a few long-term challenges that we believe will be of cardinal importance in the deployment of advanced methods and architectures in the field of security.

- Integration of the formal and computational views of cryptography. These two views have been regarded as unreconcilable for a long time; a first bridge between them is the seminal work of Abadi and Rogaway [Abadi and Rogaway 2002] and Pfitzmann et al [Pfitzmann et al 2000]. However, this is only the first step in a direction that requires much more investigation.

[8] for example, MAFTIA: http://www.maftia.org and OASIS: http://www.tolerantsystems.org

- Side-channel attacks. Side channels can be defined as any sources of information available to the adversary that are not typically captured in the usual system models used in information flow analysis. Most formal models abstract away from details of execution timings, power consumption, and electromagnetic radiation etc. All of these features of a system's behaviour could potentially be observed by an adversary and used to attack the system. Such attacks can be highly effective, as they are often able to recover secrets that are – for security reasons – not communicated, such as private keys. Below, we discuss a simplified version of a timing attack to illustrate a connection between security and real-time properties of distributed systems. We suggest several avenues for further research on this and similar connections.

- Technical management. It is a widely accepted fact that a considerable proportion of security incidents results from the inadequate technical management. System administrators and application engineers are often overstrained by the complexity of the system structure, by the frequency of dynamic changes, and by the high number of existing service interdependencies. Considering the current transition from coarse-grained distributed systems to fine-grained, component-structured, service-based, mobility-enhanced, and dynamically adaptable systems, the importance of secure system management will increase drastically.

- Intrusion tolerance. This is the notion of handling (i.e. reacting to, counteracting, recovering from, masking) a wide set of faults encompassing intentional and malicious faults (we may collectively call them intrusions), which may lead to failure of the system security properties if nothing is done to counter their effect on the system state. In short, instead of trying to prevent every single intrusion, these are allowed, but tolerated: the system has the means to trigger mechanisms that prevent the intrusion from generating a system failure.

3.2 Reconciling Two Views of Cryptography

In the last few years we have witnessed the development of two different, but still related, abstract views of cryptographic operations: the formal and the computational one. In the former, the messages are modelled as formal expressions of a term algebra. The (cryptographic) operations, such as message pairing and encryption, are modelled as term constructors. In this setting, an adversary and its abilities can be modelled in terms of the messages the adversary knows; see for e.g. [Dolev and Yao 1983]. Furthermore, the security properties a protocol is supposed to achieve are also modelled formally [Paulson 1998][Burrows et al 1990][Ryan et al 2000]. This view adopts the so-called black-box cryptography, which assumes perfect encryption: an encrypted message never leaks any information regarding the keys and the plaintext, regardless of the encryption algorithm used and of the length of the keys. These assumptions yield a model that is viable for formal verification. This has given a great impetus to research in verification of security protocols.

However, the perfect encryption assumption is a strong one and runs the risk of overlooking subtle interactions between features of the protocol design and of the underlying cryptographic algorithms. A classic example of such an interaction can be found in [Ryan and Schneider 1998].

In the computational model, messages are considered to be (more realistically) bit-strings, while cryptographic operations are seen as functions over these bit-strings. Here, an adversary is modelled as any efficient algorithm, while the security properties of a cryptographic protocol are defined on terms of the probability of the adversary to perform a successful attack [Goldreich 1997][Bellare 1997].

Both of the two models above have advantages and disadvantages. On the one hand, the formal model allows one to reason about cryptographic protocols more easily and generally. However, such benefits arise from the adoption of fairly strong assumptions (such as freeness of the term algebra, and fixing the adversary model.) On the other hand, the computational model, by considering messages as bit-strings and modelling the adversary as any efficient algorithm, provides a more realistic model and thus offers more convincing security guarantees. However, proving protocols correct in the computational model is more difficult and less general than in the formal model.

In the work of Abadi and Rogaway [Abadi and Rogaway 2002], it is shown that if two formal expressions are similar to a formal adversary, then their corresponding computational interpretations, represented as bit-strings in the computational model, are also indistinguishable to any computational adversary. This result comprises a very important step towards relating the formal and computational model.

The work of Abadi and Rogaway, first published in 2000 [Abadi and Rogaway 2000], was later extended by

Abadi and Jurjens [Abadi and Jurjens 2001] and Laud [Laud 2001]. In these works, similar soundness results were obtained for richer formal languages, where instead of considering values of formal expressions, outputs of programs are analysed. However, both of these extended languages still treat the encryption operation as using atomic keys. Micciancio and Warinschi [Micciancio and Warinschi 2003] considered the converse of the soundness result (i.e., completeness of the formal language of [Abadi and Rogaway 2002]). In their work, it is shown that a completeness result can be obtained by considering a stronger encryption scheme, namely an authenticated encryption scheme.

Comment [p y a2]: The chronology here doesn't make sense. Who wrote this bit? I'm not familiar with all these refs so it's not immediately clear to me how to fix this.

Further extensions of the seminal work [Abadi and Rogaway 2002] deal with encryption cycles in expressions. In the computational model, the security of a traditional encryption scheme can be compromised if an adversary gets hold of a message containing an encryption cycle. Thus, in the original work of Abadi and Rogaway, formal expressions were restricted to be cycle free. However, further work of Black et al. [Black et al 2002] and Laud [Laud 2002] has shown that, in fact, this discrepancy can be addressed in two different ways: either by considering a new, stronger security definition of the encryption scheme [Black et al 2002], or by strengthening the adversary model of the formal model, such that it can be able to "break" encryption cycles [Laud 2002].

Recently, Bellare and Kohno [Bellare and Kohno 2003] have studied the security of cryptosystems against related-key attacks and also provided a construction of a secure cryptosystem against a certain kind of such attacks. Related keys are different from composed keys — a related key is something that is constructed from an already existing good key and some non-key data, whereas a composed key is constructed from non-key data only.

Finally, Laud and Corin [Laud and Corin 2003] have considered an extension of the formal model presented by Abadi and Rogaway, in which it is allowed to use composed keys in formal encryption. See also the work from IBM Zurich [Pfitzmann et al 2000].

3.3 Side-channel Attacks

A functionally correct system must satisfy a range of systemic (i.e. non-functional) requirements to be fit for purpose. Systemic requirements include energy efficiency, low noise emission, low EMF radiation, (real)-timeliness, security, cost effectiveness, etc. The main difficulty is that while functional correctness can often be modularised, systemic 'correctness' cannot, because systemic properties are not compositional, i.e. the systemic properties of individual components are rarely preserved when the components are integrated because the components tend to interfere with each other in many unexpected ways. Engineers usually over-dimension designs to cater for the worst-case scenario, which makes systems more costly than strictly necessary.

Comment [p y a3]: This description of side channel attacks seems quite bizarre to me, but I don't really have the time to completely re-write it.

We offer insight into the interaction of two systemic properties: security and real-timeliness, where we use the timing attack [English and Hamilton 1996] [Kocher 1996] as the prime example of an undesirable interaction. Mounting a timing attack requires the ability to measure time with predictable accuracy, which is of course exactly what real-time systems are all about. Therefore we claim that the security of a system can be weakened in principle by making the system suitable for real-time applications. It is precisely this kind of interaction that is horribly difficult to predict, and which ultimately determines whether a system is fit for purpose.

Comment [p y a4]: Surely Kocher was the first to identify timing channel attacks.

Timing attacks can be used to discover a secret key by measuring the time taken by cryptographic operations. Other systemic properties can be exploited in the same way, for instance to mount power attacks [Chari et al 1999], and attacks based on fault induction [Biham and Shamir 1997], etc. The basic assumptions of timing analysis are:

- The run time of a cryptographic operation depends to some extent on the key. With present hardware this is likely to be the case, but note that there are various efficient hardware based proposals to make the timing attack less feasible through 'noise injection' [May et al 2001]. Software approaches to make the timing attack infeasible are based on the idea that the computations in two branches of a conditional should take the same amount of time ('branch equalisation'). This is costly [Agat 2000].

- A sufficiently large number of encryptions can be carried out, during which time the key does not change. A challenge response protocol is ideal for Timing Attacks.

- Time can be measured with known error. The smaller the error, the fewer time measurements are required.

Different versions of timing attack have been reported to be effective with RSAREF (the publicly available version of RSA) [Kocher 1996], DES [Hevia and Kiwi 1998], and RC5 [Handschuh and Heys 1998]. Timing attacks usually take place in a controlled environment, where the system under attack (often a smart card) is connected to measurement equipment, particularly to control the errors in the timing measurements. A recent report [Canvel et al 2003] describes a timing attack on OpenSSL. However, the report acknowledges that mounting this attack over the network is unlikely to be successful because time measurement is too inaccurate. We believe that in a distributed system with real-time properties, timing attacks on the security protocols of such system may become a threat.

To predict the kinds of attacks that could be mounted on the security of a distributed system would appear to be difficult in its full generality. However, by singling out specific systemic properties we believe that progress can be made by modelling the cause and effects of the attacks. Timing aspects of encryption schemes are typically analysed using statistical methods and tools such as Matlab, where attacks are modelled as signal detection problems. The signal is in some way correlated with the secret keys. The noise originates from timing inaccuracy, unknown key bits, etc. [English and Hamilton 1996]. The level of abstraction is generally too low to take protocol aspects into account. Security protocols are typically analysed using formal methods and tools such as the FDR model-checker for CSP [Ryan et al 2000], the Casper tool that provides a security protocol analysis front end to FDR [Kocher 1996][Lowe 1997], and CoProVe [Canvel et al 2003], where attacks are modelled by an attacker who learns secrets by inspecting, deleting and repeating messages. The level of abstraction is generally too high to take timing information into account.

3.4 Secure Operation of IT-systems

The security of IT-systems does not only depend on their suitable design, on the trustworthiness of their components or on the employment of suitable security services. In practice, a very high portion of security incidents results from inadequate technical management. System administrators and application engineers are often overstrained by the complexity of the system structure, by the frequency of dynamic changes, and by the high number of existing service interdependencies. In the light of the current transition from coarse-grained distributed systems to fine-grained, component-structured, service-based, mobility-enhanced, and dynamically adaptable systems, the importance of secure system management will increase drastically.

Administrators will have to accomplish the secure management of highly interactive service systems, where fine-grained and dynamically changing structures demand scalable and adaptable security services. Since these distributed applications are faced with open user-groups as well as with competing component vendors, service providers, and application hosts, multi-party scenarios emerge where several relevant principals exist each having its own special security requirement profile. While “old” applications made only a binary distinction between friendly insiders and the hostile environment, now subtle trust relations exist between many principals, relations which moreover may change in the course of time. The trust relations influence the functionality of an application and are affected by the experiences the principals have with the running application. The partial reachability and the restricted resources of small and mobile devices produce tight and changing conditions for the security services. There is a considerable trade-off between the overhead of protection functions and the achievable security levels. To deal explicitly with such a trade-off we need highly adaptable security services. The configuration, monitoring, control, adaptation, and repair of an application security subsystem is exacting task, which demands automated administration functions supported by management meta-services and tools.

Three projects at the University of Dortmund are addressing these issues:

- Current research in the project “Model-based security management” focuses on the automated derivation of detailed system configurations from abstract security policy descriptions by a combination of the approach of policy-based management with the approach of model-based management [Lück et al 2002][Lück et al 2001][Lück et al 1999]. The combined approach uses a hierarchically layered model, which represents the existing abstract policy directly in its top layer, and provides a detailed model of the computer network and its components in its bottom layer. The model is object-oriented, and the modelling is performed interactively. It is supported by a graphical modelling tool and by a library of predefined system component classes. The designer of a model considers the real system, identifies its logical components and relations, and defines a corresponding object model by tool-assisted creation of an object instance diagram. The modelling tool automatically performs the necessary policy refinement steps and verifies the correctness of the derived policies. It checks the consistency of the policies and proves that the derived lower level policy completely enforces the given high-level policy. If the system and in particular the contained security components are not sufficient

for the enforcement of the abstract policy, the verification fails and the tool highlights possible reasons.

- In addition, the project “Formal analysis of security properties” has started to concentrate on the development of a method which achieves the formal modelling of computer networks, their security mechanisms and services as well as behaviours of attackers and administration processes in order to investigate effects of attacks and erroneous administration activities as well as undesired interferences of attack and administration processes.

- Current research in the project “Trust and security aspects of distributed component-structured software” considers the special properties of component-structured software resulting from the independent development, combination, and deployment of components and component services [Herrmann 2003a][Herrmann 2003b][Herrmann et al 2002]. In particular, here, the high number of principals is a reason for more subtle security risks than in monolithic programs and special protection mechanisms are needed, which, however, introduce overhead and influence the application performance. Therefore a trust management system is applied which keeps track of the current risks, experiences, and trust levels of components and principals. The protection mechanisms employed in the applications are equipped with adaptation functions which adjust the monitoring efforts to the current trust levels and requirements.

3.5 The Case for Intrusion Tolerance

There is a significant body of research on distributed computing architectures, methodologies and algorithms, both in the fields of dependability and fault tolerance, and in security and information assurance. These are commonly used in a wide spectrum of situations: information infrastructures; commercial web-based sites; embedded systems. Their operation has always been a concern, due in large part to the use of COTS, compressed design cycles, and openness. Whilst dependability and security have taken separate paths until recently, the problems to be solved are of similar nature: keeping systems working correctly, despite the occurrence of mishaps, which we could commonly call faults (accidental or malicious); ensure that, when systems do fail (again, on account of accidental or malicious faults), they do so in a non harmful/catastrophic way. In classical dependability, and mainly in distributed settings, fault tolerance has been the workhorse of the many solutions published over the years. Classical security-related work has on the other hand addressed, with few exceptions, intrusion prevention, or intrusion detection without systematic forms of processing the intrusion symptoms.

It is known that distribution and fault tolerance go hand in hand: one distributes to achieve resilience to common mode faults, and/or one embeds fault tolerance in a distributed system to resist the higher fault probabilities coming from distribution. Contrary to some vanishing misconceptions, security and distribution also go hand in hand: one splits and separates information and processing geographically, making life harder to an attacker. This suggests that (distributed) malicious fault tolerance, a.k.a. (distributed) intrusion tolerance is an obvious approach to achieve secure processing. If this is so obvious, why has it not happened earlier?

In fact, the term “intrusion tolerance” has been used for the first time in [Fraga and Powell 1985], and a sequel of that work led to a specific system developed in the DELTA-4 project [Deswarte et al. 1991]. In the following years, a number of isolated works, mainly on protocols, took place that can be put under the IT umbrella [Castro and Liskov 1999] [Reiter 1995] [Kihlstrom et al. 2001] [Alvisi et al. 2000] [Malkhi et al. 2001] [Ateniese et al. 2000] [Hiltunenet al. 2001], but only recently did the area develop explosively, with two main projects on both sides of the Atlantic, the OASIS and the MAFTIA projects, doing structured work on concepts, mechanisms and architectures. One major reason is that distributed systems present fundamental problems in the presence of malicious faults. On the other hand, classical fault tolerance follows a framework that does not completely fit with the universe of intentional and/or malicious faults. These issues are discussed below.

Dependability has been defined as that property of a computer system such that reliance can justifiably be placed on the service it delivers. The service delivered by a system is its behaviour as it is perceived by its user(s); a user is another system (human or physical) which interacts with the former [Avizienis et al. 2001]. Dependability is a body of research that hosts a set of paradigms, amongst which is fault tolerance, and it has grown under the mental framework of accidental faults, with few exceptions [Fraga and Powell 1985] [Dobson and Randell 1986], but we will show that the essential concepts can be applied to malicious faults in a coherent manner.

Malicious failures make the problem of reliability of a distributed system harder: failures can no longer be considered independent, as with accidental faults, since human attackers are likely to produce ``common-mode" symptoms; components may perform collusion through distributed protocols; failures themselves become more severe, since the occurrence of inconsistent outputs, at wrong times, with forged identity or content, can no

longer be considered of “low probability”; furthermore, they may occur at specially inconvenient times or places within the system, driven by an intelligent adversary’s mind.

Traditionally, security has evolved as a combination of: preventing certain attacks from occurring; removing vulnerabilities from initially fragile software; preventing attacks from leading to intrusions. For example, in order to preserve confidentiality, it would be unthinkable to let an intruder read any confidential data at all. Likewise, integrity would assume not letting an intruder modify data at all. That is, with few exceptions, security has long been based on the prevention paradigm. Intrusion tolerance proposes that it be also based on the tolerance paradigm.

What is Intrusion Tolerance? As mentioned earlier, the tolerance paradigm in security: assumes that systems remain to a certain extent vulnerable; assumes that attacks on components or sub-systems can happen and some will be successful; ensures that the overall system nevertheless remains secure and operational, with a quantifiable probability. In other words: faults - malicious or otherwise – do occur; they generate errors, i.e. component-level security compromises; error-processing mechanisms make sure that security failures are nevertheless prevented. Obviously, a complete approach combines tolerance with prevention, removal, forecasting - namely the classic dependability fields of action!

In consequence, achieving dependability implies the use of combinations of: fault prevention, or how to prevent the occurrence or introduction of faults; fault removal, or how to reduce the presence (number, severity) of faults; fault forecasting, or how to estimate the presence, creation and consequences of faults; and last but not least, fault tolerance, or how to ensure continued correct service provision despite faults. Thus, achieving dependability vis-a-vis malicious faults (e.g. attacks and vulnerabilities) will mean the combined use of classical prevention and removal techniques with tolerance techniques.

An intrusion has two underlying causes: *vulnerability* - fault in a computing or communication system that can be exploited with malicious intent; *attack* – a malicious intentional act aimed at a computing or communication system, with the intent of exploiting a vulnerability in that system. The combination of these causes leads to *intrusions*: malicious operational faults resulting from a successful attack on a vulnerability [Verissimo et al 2003].

In order to understand better the relationship between security and classical dependability, observe the adjectives “trusted” and “trustworthy”; they have been often used inconsistently and up to now, exclusively in a security context [Adelsbach et al 2002]. However, the notions of “trust” and “trustworthiness” can be generalized to point to generic properties and not just security; and there is a well-defined relationship between them - in that sense, they relate strongly to the words “dependence” and “dependability”.

Consider *trust* as the *accepted dependence* of one component (human or technical) on a set of properties (functional and/or non-functional) of another component, subsystem or system. In consequence, a trusted component has a set of properties that are relied upon by another component (or components). Conversely, *trustworthiness* would be the measure in which a component, subsystem or system, meets a set of properties (functional and/or non-functional) and is essentially synonymous with dependability. The trustworthiness of a component is, not surprisingly, defined by how well it achieves a set of functional and non-functional properties, deriving from its architecture, construction, and environment, and evaluated as appropriate.

The definitions above have obvious (and desirable) consequences for the design of systems aiming at providing security and security-related properties: the latter can be achieved by following principles of malicious fault tolerance, that is intrusion tolerance. In consequence, one can line-up several intrusion-tolerance strategies, establishing guidelines for the construction of modular and complex secure systems.

There are a few main frameworks and strategies along which intrusion tolerant systems will emerge and be built. The main frameworks for developing intrusion-tolerance mechanisms are: secure and fault-tolerant communication; software-based intrusion tolerance; hardware-based intrusion tolerance; auditing and intrusion detection. Likewise, systems architects will follow diverse strategies according to the end system's needs in terms of combined security and fault tolerance. Strategies will be conditioned by several factors, such as: type of operation, classes of failures (i.e., power of intruder); cost of failure (i.e., limits to the accepted risk); performance; cost; available technology. In essence, as detailed in [Verissimo et al 2003], the main strategic decisions are: fault avoidance vs. fault tolerance; confidential operation; reconfigurable operation; recoverable operation; fail-safe.

3.6 Future Trends

We have illustrated a few challenges lying ahead in the field of security.

The first one concerns bridging the gap between the formal and the computational view of cryptography. Only when this gap is closed will we have formal verification tools that achieve what they promise: viz. verifying that a given protocols is totally free from weaknesses that could be exploited to extract some information from encrypted messages. The development of the theory underpinning this has already been initiated [Abadi and Rogaway 2002], but work on modelling methods and tools is yet to start. The second one concerns modelling and verifying security protocols by taking into account their real-time properties. The notional separation of security protocols from encryption schemes has made it possible to make significant progress in security modelling and analysis in each separate domain. However, timing affects the protocols and protocols affect the timing, so that eventually security protocols and encryption schemes will have to be studied simultaneously. A simplified version of the timing attack has been studied; future work will take the statistics of the attack into account.

We believe that both points are essential for the development of sound engineering methods for security in distributed systems. One possible avenue of research is to use existing tools for the verification of both encryption schemes and security protocols. This would have the advantage that we can leverage the power of the tools, which, at the cost of many person years, have been engineered to be able to cope with sizeable models. It may also be possible to make a connection between different tools (for example Uppaal, MoDeST and CoProVe) so that results from one tool can be fed into the other and vice versa. Another, equally important avenue of research is to develop modelling methods for protocols that are a little less abstract, and modelling methods for security schemes that are a little more abstract. Each represents a bridgehead, which, we believe, will eventually support a strong bridge between the two domains. The problems that we have discussed relating to timing will appear also in relation to the other systemic parameters, thus requiring the study of interaction between a multitude of systemic parameters. One might hope that eventually a general theory and associated methods and tools might emerge that will support the security engineer.

The third challenge is ensuring the secure operation of the IT systems, focusing on building secure highly-structured, mobility-enhanced, and dynamically adaptable systems, can be met by developing and employing approaches which aim at the outlined objectives of a better understanding of management processes, of automation support for technical security management, and of the development of suitable trust-based adaptation functions. The vision is that suitable tool-assistance support will enable non-expert users to define their security requirements consistently and on a very abstract level. The users moreover can resort to run-time support which performs the policy enforcement in a reliable, trustworthy, cost-efficient, and in particular fully automated manner.

Finally, Intrusion Tolerance as a body of knowledge will play two very important roles in the near future: it will be the main catalyst of the evolution of the area of dependability (see Chapter 2); and it will offer workable avenues to deal with the problem of achieving security in complex and open systems. The challenges put by looking at faults under the perspective of “malicious intelligence” have brought to the agenda hard issues such as uncertainty, adaptivity, incomplete knowledge, interference, and so forth. Under this thrust, researchers have sought solutions, sometimes under new names or slight nuances of dependability, such as trustworthiness or survivability. We believe that fault tolerance will witness an extraordinary evolution, which will have applicability in all fields and not only security-related ones. We will know that this has been achieved when the security community no longer talks about accidental faults, attacks and intrusions, but just faults.

4 Real-time Systems

The focus of this chapter is on building real-time systems, which are the systems that must meet their temporal specifications. The first section discusses the real-time operating systems with the main emphasis on RTD related to Linux; the next section outlines current and future work on languages for programming real-time systems (mainly Java and Ada), the following section is about real-time scheduling. Section 4.4 addressed the issue of Time-Triggered Architectures, and is followed by a section that overviews approaches to worst case execution time analysis. The final section discusses verification and validation of real-time systems (focusing mainly on safety critical applications).

Development of safety critical systems is an important area of work within the scope of real-time systems. Today, safety critical real time systems play an important role in our society. Their application extends to a large variety of domains, including the automobile, railway, avionics and space domains. Computer systems in such domains are submitted to very stringent dependability requirements, in particular in terms of safety (non-occurrence of catastrophic consequences after a failure). Besides, they must meet very stringent timing requirements, typically referred to as deadlines. The missing of a deadline can entail the failure of the system (hard deadlines), which in certain cases can become catastrophic in terms of human lives or economic losses (critical real time systems).

4.1 Real-time Operating Systems

4.1.1 Linux

Linux is rapidly becoming the dominating operating system for developing real-time applications. More and more real-time operating system (RTOS) users, specially large companies looking for cost savings, are considering migrating to Linux, avoiding royalties on commercial operating systems (OSs), and leveraging on open source bases without being locked-in with a single vendor or on a single kind of hardware platform.

In addition, real-time systems are more and more demanding in terms of other features. Most systems get connected to the larger world through IP. Therefore, such systems require a validated IP stack. Given the pace of evolution in such a domain, it is cheaper and safer to rely on systems such as Linux rather than on proprietary OSs which have to run behind the latest evolution in terms of network features. Windowing systems, data bases, management and control software are also “general purpose” software, which may be leveraged on when joining the Linux world.

There is however a possible limitation to this move: the Gnu Public License! RTOS users and developers who have invested in their proprietary environment, may not want to give to the community their drivers if they carry some major intellectual property providing a competitive advantage. This is however not an issue for companies whose main added value resides above the operating system in the application's world. Another issue related to licensing in this context is liability.

One more factor slowing or hindering the move to Linux is the existence of legacy applications and drivers that users do not want to rewrite or adapt to a Linux environment, mostly for time and economic constraints.

There is a high level of interest in enhancing Linux (as a commodity OS) with real-time features. Expression of such interest may be found in the specifications edited by the Carrier Grade Linux work group of OSDL[9]. The Embedded Linux Consortium,[10] although more interested in small configurations, has also some focus on real-time APIs. The OCERA IST project[11] is working on extending Linux with the new real-time functionalities.

Depending upon the needs of the targeted applications, different means are used for providing different real-time capabilities:

- Soft real-time is provided by means of specific modifications to the Linux kernel to improve the preemptability of the system, or to provide O(1) scheduler. This approach is used by MontaVista.

[9] http://www.osdl.org/lab_activities/carrier_grade_linux/

[10] http://www.embedded-linux.org/

[11] http://www.ocera.org/

- Hard real-time requires more drastic control on the resources. A number of systems address this by running real-time systems alongside the Linux kernel. Such systems are loaded as modules of the Linux kernel and are therefore dependent on Linux itself. This mechanism is used by RT-Linux and RTAI.

Some other systems use a cleaner approach in which the RT environment is not dependent upon the Linux system but may run either alongside the Linux kernel or without the Linux kernel. Adeos and Jaluna-2 use this mechanism. They rely on a small layer, similar to a Virtual Machine Monitor (see Chapter 6), but much thinner for better performance. This approach enables them to run on the same physical box, an RT/OS as well as any selected commodity OS, whether Linux, BSD or even Windows. In addition, it may appear to be a way to solve the migration issue for systems based on a proprietary OS and willing to benefit from the richness of Linux. Of course such dual-personality systems are more expensive in terms of memory consumption.

Another indicative move is for non-Linux OS developers to provide a Linux interface: QNX is claiming they have a Linux API, which enables them to run Linux applications on top of the QNX systems.

Here is a (non-complete) list of Real-Time Linux vendors: MontaVista, Jaluna-2, LynuxWorks, TimeSys, RTLinux (FSM Labs), RTAI/Lineo.

We will consider Jaluna work in detail to demonstrate how industry approaches this. Jaluna offers different OSs: Jaluna-1 and the Jaluna-2 family.

Jaluna-1 is a RTOS which is the latest version of the ChorusOS product line previously developed by Chorus systems and Sun Microsystems. Jaluna-1 provides a quite complete POSIX API. It is based on the C5 micro-kernel (formerly Chorus microkernel) and has an OS layer derived from FreeBSD 4.1. Jaluna-1 comes with a Host-Target development environment. It addresses the carrier-grade requirements of real-time, embedded elements of network equipment, such as PBXs, switches, bridges, routers, etc. It provides a number of high-availability features including error confinement, hot restart, service reconfiguration, as well as simple and efficient means to securely manage system resources.

Jaluna-2/RT OS brings together Linux and real-time capabilities for embedded systems. A hardware resource dispatcher enables the C5 microkernel and Linux kernel to run side by side. The overhead imposed by the dispatcher is really small permitting the real-time properties of the C5 environment to be maintained. Thus Linux performances are really close to those of a native Linux implementation.

This approach is further extended within the J2/VL system, which enables multiple Linux instances to run alongside the C5 microkernel. In this configuration, each system is almost isolated from the others. This work bears some similarities with Virtual Machine Monitors, but avoids most of the overhead usually incurred by such approaches, thus allowing a real-time environment to be supported.

4.1.2 Academia Research

There are many good reasons why nowadays universities are not developing new RTOSs. But there is one interesting development we would like to mention: it is the FIASCO[12] system developed by Dresden University in Germany within the DROPS project (the Dresden Real-Time Operating System project). This project develops design techniques for constructing distributed RTOSs that support applications with Quality of Service requirements. To achieve this every component of the OS guarantees a certain level of service to applications. The key component is L4Linux, - the Linux server on top of the L4 microkernel. In addition, separate real-time components - designed from scratch - provide deterministic service to real-time applications.

4.1.3 COTS Real-Time Operating Systems and Dependability

An important trend today is to build safety critical real-time systems from COTS software components, particularly from COTS real-time operating systems. The use of COTS software can enable the time-to-market and development costs of real-time applications to be reduced. However, the development process of the COTS components does not usually match the stringent requirements imposed by safety critical systems. A conceptual and experimental framework for dependability benchmarking of COTS-based systems (the DBench project) can provide system developers with means for identifying malfunctioning or weakest parts of the system, and comparing the dependability of competing solutions based on COTS real-time operating systems (e.g., RTEMS).

4.1.4 Market Perspectives

[12] http://os.inf.tu-dresden.de/drops/

The RTOS market is highly fragmented with many commercial providers and many more home-grown systems. One of the reasons for this fragmentation lies in the need to customize the system to the dedicated application area. These specific needs might be related either to foot-print and memory cost constraints (mobile phones, car equipments, etc.) or to requirements for proof-based certifications (for aerospace embedded systems, automotive systems, etc.). Here is a partial list of the main non-Linux based RTOSs: Windriver (VxWorks and pSoS), QNX, OSE, Microsoft WinCE, Symbian, Jaluna-1, LynxOS, ThreadX, RTEMS.

As may be feared, there are a lot of systems and many APIs for very similar services. However, there have been several attempts to make these systems converge at least from the API point of view. Among the standardization efforts, one may quote:

- POSIX Real-time extensions and profiles[13]
- ITRON[14]: a Japanese standardization effort
- OSEk[15]: an industry standard for an open-ended architecture for distributed control units in vehicles
- Embedded Linux Consortium[16]; this is defining Linux profiles to have it work in the embedded space. Although the Consortium is more interested by small configurations it has also some focus on real-time APIs.
- Carrier Grade Linux[17]: Commercial companies having interest in Linux for the telecommunication industry are working together to define the evolution of Linux to make it cope with telecom requirements. This includes many aspects among which real-time features.

4.2 Languages for Real-Time and Embedded Systems

Embedded systems are mainly small (for example, mobile phones) but can sometimes be extremely large (for example air traffic control systems). For small embedded applications, sequential languages like C and C++ reign supreme. For the larger real-time high integrity systems, Ada still dominates. In the telecommunications market, CHILL is popular. In Germany, Pearl is widely used for process control and other industrial automation applications.

Although there is little doubt that the Java language has been immensely successful in a wide range of application areas, it has yet to establish itself completely in the real-time and embedded markets. The introduction of a Real-Time Specification for Java could dramatically alter the status quo. In the future, if MicroSoft's C# programming language starts to gain momentum, extensions will inevitably be considered to make it more appropriate for real-time systems.

Rather than consider all possibly real-time programming languages, this section focuses on two representative of the landscape: Ada and Real-Time Java (in particular the Real-Time Specification for Java). Ada represents the class of concurrent real-time programming languages which were developed in late 1970s and early 1980s (including Pearl and CHILL). These have been extended over the years to embrace object-oriented programming and to give better support for real-time and distributed systems. Real-time Java represents the current trend of supporting architecture-neutral real-time systems potentially in an open environment (and points to the direction that languages such as C# might take in future). Synchronous languages (such as Esterel, Signal or Lustre) and functional languages (such as Erlang) are not considered, as their use is either confined to a particular company (e.g. Ericsson and Erlang) or targeted at supporting only reactive systems (e.g. Esterel). Sequential languages such as C and C++ are also not reviewed as their advantages and disadvantages are well known. Research-oriented languages are not covered as they are not in widespread use.

[13] http://www.opengroup.org/rtforum

[14] http://www.itron.com

[15] http://www.osek-vdx.org/

[16] http://www.embedded-linux.org/

[17] http://www.osdl.org/cgl/

4.2.1 Current and Recent Work

The Ada Language

The development of the Ada programming language forms a unique and, at times intriguing, contribution to the history of computer languages. As all users of Ada will know, the original language design was a result of competition between a number of organisations, each of which attempted to give a complete language definition in response to a series of requirements documents. This gave rise to Ada 83. Following ten years of use, Ada was subject to a complete overhaul. Object-oriented programming features were added (through type extensibility rather than via the usual class model), better support for programming in the large was provided (via child packages) and the support for real-time and distributed programming was enhanced. The resulting language, Ada 95, is defined by an international ISO standard.

An important aspect of the new Ada language is the model it presents for concurrent programming. This model is both complex and wide ranging. It builds upon the original Ada 83 language features for tasking but provides many additional facilities required to meet the challenges of modern systems development, particularly in the areas of real-time and embedded systems.

The Ada 95 definition has a core language design plus a number of domain-specific annexes. A compiler need not support all the annexes but it must support the core language. Most of the tasking features are contained in the core definition. But many of the important features for real-time programming are to be found in Annex D.

A listing of the key language features of Ada (for real-time programming) would contain: protected objects for efficient encapsulation of shared data and interrupt handling, fixed priority scheduling integrated with a priority ceiling protocol, the requeue statement that eliminates many sources of race conditions in application code, a monotonic clock with associated abstract data types for time with absolute and relative delay primitives, dynamic priorities and an asynchronous transfer of control capability that is integrated into the syntax of the core language. These features provide an expressive environment for programming flexible schedulers [Bernat and Burns 2001].

The Ada 95 programming language addresses most of the issues associated with building fixed-priority real-time systems. Although Ada has not had the same commercial success as Java, it is widely accepted (even by its opponents) that Ada has a good technical solution for a wide range of real-time problems (for example, see the Preface to [Bollella et al 2000]).

The definition of Ada benefits from being an international standard. Currently the definition is being reviewed (as ISO requires every ten years). Major changes will not be made to Ada but a number of amendments will update the definition. Many of these will be in the real-time domain and are likely to include the incorporation of timing events [Burns and Wellings 2002], support for a wider range of scheduling paradigms [Aldea Rivas and Gonzalez Harbour 2002a], and execution-time budgeting.

One important new addition to the language is support for high-integrity concurrent real-time programs. Ada 95, via the use of restrictions on the use of the language, allows different execution profiles to be defined. However, in the past the language has shied away from profiles. In recent years, the real-time Ada community has developed its own profile called Ravenscar [Burns et al 2003]. This profile defines a subset of the tasking and real-time features, and is aimed at the high integrity real-time domain where predictability, efficiency and reliability are all crucial. Ravenscar is now being used in this domain with a number of vendors offering Ravenscar-specific run-time kernels (including one open-source version). As a consequence of its success, it has been decided to incorporate the profile into the language definition.

Clearly Ada is not as popular as it was in the 90s. Although it does support object-oriented programming, languages such as C++ and Java have been more successful in propagating this form of programming. This is partly a result of choosing the type extensibility model, but also because Ada has never been able to regain the momentum it lost during the early years when compilers were expensive and efficiency of code execution was perceived to be poor.

Ada remains the language of choice in many high integrity domains and often these are also real-time. Arguably Ada provides greater support than any other mainstream engineering language for producing real-time and embedded code. By taking a language focus (rather than an API approach) Ada enables significant static analysis to be undertaken by the compiler and associated tools. This leads to high quality code and cost-effective code production.

Real-Time Java

Since its inception in the early 1990s, there is little doubt that Java has been a great success. The Java environment provides a number of attributes that make it a powerful platform to develop embedded real-time applications. Since embedded systems normally have limited memory, an advantage that some versions of Java (for instance J2ME) present is the small size of both the Java runtime environment and the Java application programs. Dynamic loading of classes facilitates the dynamic evolution of the applications embedded in the system. Additionally, the Java platform provides classes for building multithreaded applications and automatic garbage collection. The main problem of Java garbage collection, however, is that it introduces random pauses in the execution of applications. Moreover, Java does not guarantee determinism nor bounded resource usage, which are needed in this type of system. For these reasons, Java was initially treated with disdain by much of the real-time community. Although the language was interesting from a number of perspectives the whole notion of Java as a real-time programming language was laughable. "Java and Real-time" was considered by many as an oxymoron.

In spite of the real-time community's misgivings, Java overall popularity led to several attempts to extend the language so that it is more appropriate for a wide range of real-time systems. Much of the early work in this area was fragmented and lacked clear direction. In the late 1990s, under the auspices of the US National Institute of Standards and Technology (NIST), approximately 50 companies and organizations pooled their resources and generated several guiding principles and a set of requirements for real-time extensions to the Java platform [Carnahan and Ruark 1999]. Among the guiding principles was that Real-Time Java (RTJ) should take into account current real-time practices and facilitate advances in the state of the art of real-time systems implementation technology. The following facilities were deemed necessary to support the current state of real-time practice [Carnahan and Ruark 1999]: fixed priority and round robin scheduling; mutual exclusion locking (avoiding priority inversion); inter-thread communication (e.g. semaphores); user-defined interrupt handlers and device drivers (including the ability to manage interrupts and timeouts and aborts on running threads).

The NIST group recognized that profiles (subsets) of RTJ were necessary in order to cope with the wide variety of possible applications, these included: safety critical, no dynamic loading, and distributed real-time profiles. There was also an agreement that any implementation of RTJ should provide: a framework for finding available profiles, bounded preemption latency on any garbage collection, a well-defined model for real-time Java threads, communication and synchronization between real-time and non real-time threads, mechanisms for handling internal and external asynchronous events, asynchronous thread termination, mutual exclusion without blocking, the ability to determine whether the running thread is real-time or non real-time, a well-defined relationship between real-time and non real-time threads.

Solutions that comply with the NIST requirements are API-based extensions. There are three proposed solutions.

- The Real-Time Specification for Java [Bollella et al 2000] gives a new and very different specification to meet the requirements.
- The Real-Time Core Extension for the Java Platform (RT Core) [J-Consortium 2000] consists of two separate APIs: the Baseline Java API for non real-time Java threads, and the Real-Time Core API for real-time tasks.
- The Basic RT JavaO specification [Krause and Hartmann 1999] is very simple extension, is presented as an alternative or a complement to this last solution.

Perhaps the most high-profile attempt is the one backed by Sun and produced by The Real-Time for Java Expert Group [Bollella et al 2000]. This effort now has considerable momentum. In contrast, progress with the J-consortium's CORE-Java has been hampered by the lack of a reference implementation and poor support from potential users. The approach taken by the Real-Time Specification for Java (RTSJ) has been to extend the concurrency model so that it support real-time programming abstractions, and to provide a complementary approach to memory management that removes the temporal uncertainties of garbage collection. In particular, the RTSJ enhances Java in the following areas: memory management, time values and clocks, schedulable objects and scheduling, real-time threads, asynchronous event handlers and timers, asynchronous transfer of control, synchronisation and resource sharing, physical memory access. It should be stressed that the RTSJ only really addresses the execution of real-time Java programs on single processor systems. It attempts not to preclude execution on shared-memory multiprocessor systems but it, for example, has no facilities directly to control the allocation of threads to processors. A reference implementation is now available.

The NIST core requirements for real-time Java extensions were identified above. [Wellings 2004] analyses the extent to which the RTSJ has met them and argues that it meets the vast majority of them, with the one exception: the RTSJ does not explicitly address the issues of profiles other than by allowing an implementation to provide alternative scheduling algorithms (e.g., EDF) and allowing the application to locate the scheduling

algorithms. There is no identification of, say, a safety critical systems profile or a profile that prohibits dynamic loading of classes. Distributed real-time systems are not addressed but there is another Java Expert Group which is considering this issue [JCR 2000].

4.2.2 Future Trends

The future for Ada is unclear as it is perceived to be an "old" language in many areas of computing. However, the Ada real-time community in Europe is very active and topics currently being addressed include: subsets for high integrity applications, kernels, and hardware implementations. Arguably there has never been a better time to do real-time research using Ada technology. Over the next year, Ada will be updated to Ada 2005. The likely changes to the language include adding Java-like interfaces. Furthermore, support for real-time will be enhanced to allow more dynamic systems and to provide CPU monitoring and budgeting facilities.

In contrast with Ada, the future for Java augmented by the RTSJ is more positive. However, there are still obstacles to be overcome before Java can replace its main competitors in the embedded and real-time systems application areas. The main issues are in the following areas [Wellings 2004]:

- **Specification problems and inconsistencies** — A preliminary version was released in June 2000 in parallel with the development of a "Reference Implementation" (RI). Inevitably, the completion of the RI showed up errors and inconsistencies in the specification.

- **Profiles** — There is a need to consider RTSJ in the context of J2ME and, in particular, to produce a profile for use in high-integrity (and possibly safety-critical) systems. The Ravenscar-Java [Kwon et al 2003] profile is perhaps the first step along this road.

- **Implementation** — to generate efficient implementations of real-time virtual machines (both open source and propriety ones) for the full specification and the profiles;

- **Maintaining Momentum** — to stimulate evolution of the specification in a controlled and sustained manner to add new functionality (new schedulers, multiple schedulers, multiple criticalities, alternative interrupt handling models, real-time concurrency utilities) and to address new architectures (multiprocessor and distributed systems).

Although there will continue to be research-oriented real-time and embedded systems programming languages, most initial research ideas are explored using APIs into middleware, modified kernels or virtual machines. Hence general techniques for flexible scheduling, quality of service specification and negotiation etc. will probably be explored in those areas first. However, there is no doubt that once programming techniques materialised, it is necessary to find the appropriate language abstraction that will enable these techniques to be used in a reliable and secure manner.

Another important area is RTD of modelling languages. In the context of safety-critical systems, a promising approach is to account for safety properties early in the software development process of a real-time application. UML is becoming very popular today and important research is being undertaken to use it in the architectural design of embedded real-time applications of the railway field (the PRIDE project). The basics of the UML language are also subjected to extensive research, aimed at providing precise semantics to allow the specification of high-integrity real-time systems (the PURTA project). More generally, in the aerospace domain, research is being carried out by the University of York (the DARP project) to validate frameworks for safety-critical aerospace systems, especially during the software development process.

4.3 Real-time Scheduling

Real-time scheduling deals with algorithms for the access of activities to resources in order to meet temporal demands. These schemes include decisions of which tasks to execute when, i.e., the assignment of tasks to the CPU, as well as other resources, including resource contention protocols and consideration of network parameters.

Real-time scheduling algorithm typically consist of an online and an offline part: the *online part* steers a simple dispatcher, executing scheduling decisions based on a set of rules, which are devised in the *offline phase*. A *schedulability test* determines whether a set of tasks will meet their deadlines, if scheduled online by the specific set of rules.

4.3.1 Current and Recent Work

Most scheduling algorithms have been developed around one of three basic schemes: *table driven*, *fixed priority*, or *dynamic priority*. Depending on whether a majority of scheduling issues are resolved before or during system runtime, they are classified as *offline*, or *online.*

Offline scheduling - Table driven scheduling (TDS – see, for example, [Kopetz 1997][Ramamamritham 1990]) constructs a table determining which tasks to execute at which points in time at runtime. Thus, feasibility is proven constructively, i.e., in the table, and the runtime rules are very simple, i.e., table lookup. TDS methods are capable of managing distributed applications with complex constraints, such as precedence, jitter, and end-to-end deadlines. As only a table lookup is necessary to execute the schedule, process dispatching is very simple and does not introduce large runtime overhead. On the other hand, the a priori knowledge about all system activities and events may be hard or impossible to obtain. Its rigidity enables deterministic behaviour, but limits flexibility drastically.

Online scheduling methods overcome these shortcomings and provide flexibility for partially or non-specified activities. A large number of schemes have been described in the literature. These scheduling methods can efficiently reclaim any spare time coming from early completions and allow handling overload situations according to actual workload conditions. Online scheduling algorithms for real-time systems can be distinguished into two main classes: fixed-priority and dynamic-priority algorithms.

- Fixed priority scheduling (FPS) [Liu and Layland 1973] [Tindell et al 1994] is similar to many standard operating systems, assigning before runtime of the system priorities to tasks, and executing the task with the highest priority to execute from the set of ready tasks at runtime. Fixed priority scheduling is at the heart of commercial operating systems such as VxWorks or OSE.

- Dynamic priority scheduling, as applied by earliest deadline first (EDF - [Liu and Layland 1973][Spuri and Buttazzo 1996]), selects that task from the set of ready tasks, which has the closest deadline at runtime; priorities do not follow a fixed patterns, but change dynamically at runtime. To keep feasibility analysis computationally tractable and runtime overhead to execute rules small, however, tasks cannot have arbitrary constraints.

4.3.2 Future Trends

In the following part of this subsection we present the vision in a number of important areas.

Flexible Scheduling

Flexible scheduling is an underlying theme to most novel scheduling trends which go beyond the standard model of completely known tasks with timing constraints expressed as periods and deadline, providing “yes or no” answers as to whether all tasks meet their deadlines.

Issues addressed include handling of applications with only partially known properties (e.g., aperiodic tasks for FPS [Sprunt et al 1989], EDF [Spuri and Buttazzo 1996], and TDS [Fohler 1995]), relaxed constraints and such that cannot be expressed solely by periods and deadlines, coexistence of activities with diverse properties and demands in a system, combinations of scheduling schemes, and adaptation or changes to scheduling schemes used at run-time (see, for example, [Regehr and Stankovic 2001][Aldea Rivas and Gonzalez Harbour 2002b][Fohler 1995]).

Adaptive Systems

In adaptive real-time systems, resource needs of applications are usually highly data dependent and vary over time. In this context, it is more important to obtain systems that can very well adapt their execution to the changing environment than to apply the too pessimistic hard real-time techniques.

Reservation-Based Resource Management

With reservation-based scheduling [Lipari and Baruah 2000][Mok et al 2001], a task or subsystem receives a real-time share of the system resources according to a (pre-negotiated) contract. Thus, the contract contains timing requirements. In general, such a contract boils down to some approximation of having a private processor that runs at reduced speed.

The reservation-based scheduling can be extended to include more system resources. Typically, all bandwidth-providing resources are amenable to a similar approach.

Reservation Based Resource Management provides for acceptable response for soft real-time tasks, while bounding their interference with hard-real-time tasks, incorporation of real-time legacy code in a larger, often safety-critical, system, and composition of pre-integrated and pre-validated subsystems that can quickly be integrated to form new systems.

Scheduler Composition

Hierarchical scheduling [Regehr and Stankovic 2001] means that there is not just one scheduling algorithm for a given resource, but a hierarchy of schedulers. The tasks in the system are hierarchically grouped. The root of the hierarchy is the complete system; the leaves of the hierarchy are the individual tasks. At each node in the hierarchy, a scheduling algorithm schedules the children of that node.

The practical value of a 2 level-hierarchy is immediately obvious: intermediate nodes are applications that are independently developed and/or independently loaded. If the root scheduler provides guaranteed resource reservation and temporal isolation, the applications can (with some additional precautions) be viewed to be running on private processors. In most proposals, the root scheduler provides some form of guaranteed bandwidth allocation

Scheduling with Energy Considerations

The majority of scheduling algorithms focuses on temporal aspects and constraints. With the current trends towards higher integration and embedding processors in battery-powered devices, energy consumption becomes an increasingly important issue. From the scheduling perspectives, algorithms look at tradeoffs between processor speed and energy consumption [Aydin et al 2001].

4.4 The Time-Triggered Architecture

Distributed and dependable hard real-time systems require an adequate communication architecture for interconnecting cooperating nodes. Such an architecture has to provide the basic services required to meet the timing and dependability requirements of the respective applications. The time-triggered architecture (TTA) [Kopetz 2003] provides an architecture and computing infrastructure for the design and implementation of dependable distributed embedded real-time systems.

4.4.1 Current and Recent Work

The TTA infrastructure comprises both a number of concepts and run-time services provided by time-triggered communication protocols and a methodology for application design and development. A large real-time application is decomposed into nearly autonomous clusters and nodes, and a fault-tolerant global time base of known precision is generated at every node. In the TTA, this global time base is used to precisely specify the interfaces among the nodes, to simplify the communication and agreement protocols, to perform prompt error detection, and to guarantee the timelines of real-time applications.

The TTA supports a two-phased design methodology, consisting of the architecture design and component design phases. During the architecture design phase, the interactions among the distributed components and the interfaces of the components are fully specified both in the value domain and in the temporal domain. In the succeeding component implementation phase, the components are built, taking these interface specifications as constraints. This two-phased design methodology is a pre-requisite for the composability of applications implemented in the TTA and for the reuse of pre-validated components within the TTA.

About 25 years of research on the TTA yielded solid foundations, implementation experiences, and experimental evidence about the properties and strengths of the TTA. The corner stones upon which the TTA is built can be grouped into (a) the conceptual basis of modeling time, state, and observations, (b) architecture design principles, (c) the communication infrastructure, (d) fault-tolerance support, and (e) the TTA design methodology.

Conceptual Basis for Modelling Time, State, and Observations

The *Sparse-Time Model* [Kopetz 1997, Kopetz 1998] allows an application to establish and maintain a consistent temporal order of all events occurring within the TTA. This is achieved by partitioning time into an infinite sequence of alternating durations of activity and silence, and by ensuring that all significant events (e.g., message sending) only occur during time intervals of activity.

The sparse time base defines a system-wide notion of time, which is a prerequisite for drawing a borderline between the past and the future, and thus the definition of a *system-wide distributed state*. The intervals of silence on the sparse time base form a consistent dividing line between the past and the future and the interval when the state of the distributed system is defined. Such a consistent view of time and state is very important for arguing about possible system behaviors and implementing fault tolerance by replication.

The dynamics of a real-time application are modeled by a set of relevant state variables, the *real-time entities* (*RT entities*) that change their state as time progresses. The information about the state of a RT entity at a particular instant is captured by a so-called *observation*, an atomic data structure consisting of the name of the RT entity, the observed value, and the instant when the observation was made. As application software cannot keep its data consistent with RT entities at all instants (in practice there is a limit to the rate at which observations can be made), it uses real-time images to reflect the values of the entities. A *real-time image* (*RT image*) is defined as a temporally accurate picture of an RT entity at an instant, if the duration between the time of observation and that instant is smaller than an accuracy interval allowed by the application.

Structure of the TTA: The basic building block of the TTA is a node. A node comprises a processor with memory, an I/O subsystem, a TT communication controller, an operating system, and the relevant application software in a self contained unit. Two replicated communication channels connect the nodes, thus forming a cluster. The cluster communication system comprises the physical interconnection network and the communication controllers of all nodes of the cluster. In the TTA, the communication system is autonomous and periodically executes an a priori specified time-division multiple access (TDMA) schedule. The communication systems reads state messages from the communication network interfaces (CNI) of the sending nodes at the a priori known fetch instants and delivers these messages to the CNIs of all other nodes of the cluster at the a priori known delivery instants, thereby replacing the previous versions of the respective state messages. The times of the periodic fetch and delivery actions are coded in the message scheduling table (message description list - MEDL) of each communication controller. In larger systems, clusters can be connected by gateways.

The TTA uses bus guardians to shield the network from babbling idiot failures of nodes (nodes disturbing the network traffic by sending at the wrong time). In a bus network topology each of the redundant links of a node is connected to the bus via a guardian. In the TTA star configuration, the guardians are integrated into two replicated central star couplers. The star configuration has a number of advantages over the bus, including resilience to arbitrary node faults, e.g., slightly-off-specification (SOS) faults [Bauer 2001].

Architecture Design Principles

The TTA provides a *consistent distributed computing base* to all correct nodes. It continuously executes a distributed agreement (membership) algorithm to determine if any node participating in the joint action has been affected by a failure. Upon detection of a node failure (or even a violation of the fault hypothesis) the TTA notifies the application, which takes appropriate action based on this information.

The communication controllers and the host computers of the TTA communicate via the CNI, a *temporal firewall* interface. The temporal firewall is a pure data-flow interface that does not allow the passage of control signals. This way it prevents the propagation of control errors. The integrity of the data in the temporal firewall is assured by the non-blocking write protocol which guarantees mutual exclusion [Kopetz 1993].

In order to keep interface complexity manageable, the TTA implements three different types of node interfaces: the *real-time service (RS) interface* for the exchange of time-critical data with other nodes and the environment, the *diagnostic and management (DM) interface*, for setting node parameters and retrieving information about the internals of a node, and the *configuration and planning (CP) interface*, for connecting nodes to other nodes of the system and configuring TTA systems [Kopetz 2000].

The TTA is *composable* in the temporal domain. This has been achieved by an RS interface design that supports the four central principles for composability. These four principles are: (a) the independent development of nodes, (b) stability of prior services, (c) the constructive integration of the nodes to generate the emerging services, and (d) replica determinism [Kopetz 2003].

The *fault-tolerance layer* of the TTA implements a number of mechanisms to support active redundancy by replication and voting. Mechanisms like replica coordination, voting, membership fusion, internal state alignment, and re-integration after transient failures are hidden in this layer and therefore do not have to be dealt with in the application software.

Communication Infrastructure

According to the principle of time-triggered communication the instants at which information is delivered at or fetched from the CNIs of a node are determined a priori and are common knowledge to all nodes of the TTA (a priori common knowledge means that the instants for sending and receiving of information are defined and distributed to all nodes before the TTA system starts its real-time operation). The knowledge about these instants translates into deadlines for tasks that execute on the hosts and into times when new data from the communication system is available for the tasks. It is the responsibility of the TT communication service to transport the information from the sending CNI to the receiving CNI within the intervals delimited by the a priori known fetch and delivery instants.

The two communication protocols of the TTA that follow the time-triggered communication paradigm are TTP/C and TTP/A. TTP/C provides a dependable real-time communication service with fault-tolerant clock synchronization and a membership service for safety critical real-time applications. TTP/A is time-triggered field bus targeted at connecting smart transducers in all types of distributed real-time control systems. Event channels can be constructed on top of the basic TT communication service by assigning an a priori specified number of bytes of selected TT messages to the event-triggered message transport service.

Fault-tolerance Support

In the TTA, it is assumed that a chip is a single fault-containment region. A properly configured TTA star cluster will tolerate an arbitrary failure mode of a single TTA node (chip). A faulty unit will be consistently detected by the membership protocol and isolated within two communication rounds of the TDMA protocol. The TTA masks an asymmetric (Byzantine) fault of any node by architectural means.

For internal, physical faults the preferred fault-tolerance strategy is active replication of independent nodes. The effective partitioning of the nodes and the masking of Byzantine faults by the TTA provides the prerequisites to logically group a set of nodes or software subsystems that compute the same function into a fault-tolerant unit (FTU). Such an FTU will tolerate the failure of any of its constituents without a degradation of service. FTUs can consist of either three TMR nodes or pairs of self-checking nodes.

In the case of TMR systems the FTU layers of the nodes receiving a message vote on the messages received from the replicas of the sending FTU before delivering the majority result to their host at the delivery instant. Periodically, every host must output its internal state for the same voting procedure by the other nodes of the FTU.

The fault-tolerant service can only be maintained if the environment complies with the fault hypothesis. If the environment violates the fault hypothesis – which must be a rare event in a properly designed application – then the TTA activates a never-give-up (NGU) strategy. The NGU strategy is highly application specific. It is activated by the TTP/C protocol together with the application if an exhaustion of resources prevents the application from providing its minimum required service.

TTA Design Methodology

Composability and the associated reuse of nodes and software can only be realized if the architecture supports a two-level design methodology.

In the *architecture design phase*, an application is decomposed into clusters and nodes. In a top-down design, after the decomposition has been completed, the CNIs of the nodes are specified in the temporal domain and in the value domain. In a bottom-up design, a TTP/C schedule is constructed that satisfies the constraints of already available nodes with their given temporal specifications. In either case, the architectural design phase yields the precise interface specifications of all nodes and the communication schedule for the bus.

In the *application design phase*, the software for the host computers is developed. The implementation must obey the constraints imposed by the delivery and fetch instants established during the architecture design phase. The software for each node is developed in separation where the delivery and fetch instants are the pre- and postconditions for the temporal validation of the software. The composability properties of the TTA ensure that a property that has been established at the node level will also hold at the system level.

The two design phases of the TTA are supported by a comprehensive set of integrated design tools. [TTTech].

4.4.2 Future Trends

A major obstacle in the development of more advanced TT and other embedded systems is electronic hardware cost, which is decisive for the economic success in a mass market. In today's systems hardware costs escalate significantly with every additional electronic function, since a new electronic control unit (ECU) is required for every major new function ("1 Function – 1 ECU" problem). Intellectual property issues prevent the multiplexing of ECUs among multiple functions. Suppliers in the automotive and other supply chains are reluctant to open their intellectual property (IP), which is contained in compiled software modules to OEMs or other suppliers in order to maintain their competitive edge. Hence, what is needed is an *integrated distributed architecture* for embedded systems where the number of ECUs is significantly reduced by providing *multiple encapsulated execution environments* for different functions that are *integrated within a single physical component* and protected from each other. The research agenda in that area has to cover the topics of intellectual property protection on an ECU, resource management for encapsulated task execution environments, memory protection schemes that support error containment between application tasks (from multiple vendors) on single ECUs, and diagnostic services for ECUs executing software from multiple different vendors.

By analogy with the desire to reduce the number of ECUs, there is also the potential of cost benefits by reducing the number of connectors and physical wires by providing *encapsulated virtual networks* on a single wire. By reducing the number of physical contact points, virtual networks can also improve dependability. The major research topics include the construction of virtual event-triggered subnetworks on top of the TTA, error containment mechanisms for event-triggered subnetworks, the support for migration of functionality of event-triggered architectures between different components, concepts for constructing an infrastructure of dedicated subnetworks to transfer diagnostic information to diagnostic units without affecting the correct temporal behaviour of the RS interfaces of TTA applications, and virtual subnetworks for transporting multimedia traffic on top of the TTA.

Another major aspect in the development of more advanced embedded systems is the necessity of *diagnostic concepts* designed for maintenance. Today the replacement of correctly operating components by maintenance engineers represents misguided efforts. A solution to this problem will involve architectures with thorough diagnostic support. While basic diagnostic mechanisms have been investigated in previous projects (e.g. NextTTA), maintenance-oriented systemic application level diagnosis remains an open issue. Important research topics concerning diagnosis include interface specification approaches designed for diagnosis, the use of out-of-norm assertions to detect abnormal (and possibly erroneous) behaviour, maintenance-oriented failure models that can help in deciding on the necessity of replacing components exhibiting recurring transient failures, advanced maintenance strategies, diagnostic support for components multiplexed among different services, and diagnostic support for legacy applications that will have to be integrated into TTA systems.

4.5 Worst-Case Execution-Time Analysis

Worst-Case Execution-Time Analysis (WCET Analysis) is that step of the timing analysis of real-time software that complements scheduling and schedulability analysis. While scheduling/schedulability analysis assesses the timely operation of task sets on a given system under the assumption of given needs for computing resources (CPU time), WCET analysis yields exactly this task-level timing information. WCET analysis does so by computing (upper) bounds for the execution times of tasks of a given application on a given computer system.

4.5.1 Current and Recent Work

At a very abstract level, WCET analysis consists of two parts, the characterization of execution paths that describes the possible action (instruction) sequences in the analyzed code, and the execution-time modelling that models the timing of each action on an execution path and uses these time values to compute execution-time bounds for larger sections of code and entire tasks.

Describing the possible execution paths cannot be fully automated in the general case (due to the Halting Problem), and thus the user has, in general, to be asked to provide meta-information about the possible program behaviours in the form of source-code annotations that characterize the possible execution paths. The execution

time modelling, on the other hand, is best performed at the machine level. Thus, the translation path information and timing information between these two levels of code representation is also essential for the WCET-analysis user. Work on WCET analysis is therefore separated into three problem domains:

- Characterization of execution paths
- Translation of path information from the source-language level to the machine-language representation, and
- Hardware-level execution time analysis to compute the execution-time details.

To date, research on WCET analysis has mainly focused on the third of the three domains, hardware modelling. For simple processor and memory architectures, high-quality modelling techniques have been developed and implemented in prototype tools. The WCET bounds computed by these tools typically over-estimate real worst-case times by less than 5 percent. Modelling of more complex architectures is of much higher complexity and results are more pessimistic [Colin and Puaut 2000]. In particular, so-called timing anomalies make WCET analysis very difficult [Lundqvist and Stenström 1999]. As a consequence of timing anomalies, timing analysis has to be done very defensively, which leads to more pessimistic results.

Work reported on characterizing execution paths or translating path information from the source-level code representation to the machine-language level is rare. Work on characterizing execution path can be found in [Engblom and Ermedahl 2000]. As for translating path information, existing work seems to suggest that this can only be meaningfully accomplished if the transformation of path information is done by the compiler. Attempts to use compiler-external tools for this transformation did not prove very successful.

4.5.2 Future Trends

In the following part of this subsection we present the vision in a number of important areas.

Probabilistic WCET Analysis

Deriving exact analytical timing models for the complex hardware architectures that are deployed in high-performance computer systems for embedded applications is very difficult, if not infeasible. Thus a number of research groups are seeking solutions that avoid the manual construction of complex timing models. Instead, models for processor timing are built from timing measurements [Petters and Färber 1999]. In a series of measurement runs the software under evaluation is executed with different input-data scenarios, during which the execution times of parts of the code are measured and logged. The data from the experiments are then used to build an execution-time model that is subsequently applied to compute a WCET bound for the software under test. As the experiments can in general only assess a small part of the entire space of possible executions (and execution-time scenarios), the so-derived WCET bounds can only be guaranteed to hold with some probability. Thus the term probabilistic WCET analysis is used [Bernat et al 2002]. Probabilistic WCET analysis targets applications for which WCET bounds must hold with high probability.

WCET Analysis for Model-Based Application Development

One problem traditional WCET analysis suffers is poor uptake of the technology by industry. This seems to be mainly true to the fact that industry asks for fully automatic tools. On the other hand, the WCET-tool prototypes developed in various research projects strongly rely on meta-information about possible program behaviours to compute their results. WCET Analysis for model-based application development aims at eliminating this gap between industrial needs and existing WCET-tool technology.

Model-based application development tool chains use code generators to produce the application code. Based on the observation that this auto-generated code is restrictive enough in its structure that the path information needed for WCET analysis can be computed automatically (i.e., without the need for user annotations), the analysis is restricted and adapted to the code structures produced by the code generator. The idea is that using this restrictive model, a fully automatic WCET analysis should become possible. First research results and prototype tools are reported in [Erpenbach and Altenbernd 1999][Kirner et al 2002].

Architectures Supporting WCET Analysis

An alternate way towards a fully automatic WCET analysis is via using problem-specific software and hardware architectures [Puschner 2002a]. This line of research is motivated by the following three main arguments:

- The fundamental problems of WCET analysis (a general solution for a fully automatic path analysis does not exist, documentation about the details of processor architectures is kept secret and unavailable, and the complexity of WCET analysis of modern architectures makes an exact WCET analysis infeasible) [Puschner and Burns 2002],

- The absence of WCET-oriented algorithms and software architectures, i.e., program structures that aim at an optimization for the worst case rather than average performance,

- The demand formulated by engineers of safety-critical systems that calls for simple software structures whose properties can be easily assessed [Hecht et al 1997].

As for software architectures that support temporal predictability, the most prominent proposal is the WCET-oriented programming model [Puschner 2003]. By using this programming model the programmer tries to avoid input-data dependent branching in the code as far as possible. In the ideal case, the resulting code is completely free from input-data dependent control decisions. If input-data dependent control decisions cannot be completely avoided, it is advised to restrict to a minimum operations that are only executed for a subset of the input-data space. Following the WCET-oriented programming paradigm, one can expect to obtain execution times that are either constant or almost invariable. Also, as the coding keeps the number of input-data dependent control decisions low, WCET analysis has to consider a smaller number of execution paths than with traditional algorithms and less pessimistic WCET bounds can be expected.

WCET-oriented programming cannot guarantee to eliminate all input-dependent branches. Therefore, the single-path conversion was conceived [Puschner and Burns 2002][Puschner 2002b]. The single-path conversion transforms input-data dependent branching code into code that is executed unconditionally. Loops with input-dependent termination conditions are transformed into loops with constant iterations counts and branching statements are translated into sequential code using if conversion [Allen et al 1983]. The resulting code only has a single possible execution path.

A different solution for providing predictable timing is proposed in the form of the VISA architecture [Anantaraman et al 2003]. The VISA architecture uses a pipeline that can be switched between two modes of operation, a high performance mode and a simple mode of operation for which timing can be easily predicted. Normally, the architecture executes in the high-performance mode, but if a piece of software is detected to run the risk of violating its deadline, then the system is switched to the simple mode, for which it can be guaranteed that the timing holds.

4.6 Verification and Validation of Real-time Systems

4.6.1 Current and Recent Work

Significant work is being undertaken by European industry concerning the verification and validation of safety critical systems from the space segment domain. It concerns all the on-board software of a spacecraft. Besides being safety-critical and subjected to stringent timing requirements, on-board spacecraft software is hardware-constrained, optimised, very expensive, spacecraft- and mission-dependent, and submitted to a rigorous development process.

On-board spacecraft software can be broken into the following categories:

- *On-board Basic/Low Level software*. It is the software dealing with the spacecraft hardware and interfacing with the higher level software described below. Various types of software components can be included in this category, like Real-Time Operating Systems (RTOS), Attitude and Orbit Control Systems (AOCS), Data Handling (DH) systems, or Failure Detection, Isolation and Recovery (FDIR) software mechanisms. As long as the on-top software is safety-critical, these essential software components are safety critical as well.

- *On-board Mission software*. It is the higher-level spacecraft software that manages and monitors the execution of the mission. This is specific to the spacecraft mission operation and phases, and is typically mission critical (although it can obviously be safety critical). In certain spacecrafts it is merged with the software type below.

- *On-board Payload software*. It is the software that controls and monitors specific payload hardware. Depending on the payloads, dependability aspects might be considered less important than for the

previous two sub-categories, but anyhow more project dependent. Payload software generally executes on separate processors from the one(s) running the Basic and Mission software.

Most of the on-board software is of criticality class A or B, which means that it supports functions that can potentially lead to *catastrophic* or *critical* consequences, such as the loss of life, the loss of the mission, or major damage to the overall system. Some examples of class A and B software are the collision-avoidance software of a man-rated transfer vehicle, and the AOCS of a launcher vehicle.

Critical Software SA[18] is extensively involved in the verification and validation of space segment software in the framework of several industrial and research projects, especially with the European Space Agency (ESA). In the framework of the CryoSat project, ESA is undertaking one of its first serious attempts to apply static code analysis techniques to the calculation of the Worst-Case Execution Time (WCET, see [Puschner and Koza 1989]) of a complex on-board satellite software application [Rodríguez et al 2003]. The STADY project [Moreira et al 2003] is concerned with the development of a new methodology combining static and dynamic verification techniques for the safety evaluation of space software. This new methodology has been particularly applied to analysis of the robustness of the Open Ravenscar Kernel (ORK) [de la Puente and Ruiz 2001]. Also, the validation of several safety and dependability techniques (e.g., fault injection) is the subject of research of the RAMS project, whose results are illustrated by means of the robustness and stress testing of the RTEMS operating system. The CARES project [Critical Software 2003] is concerned with the certification of space software from a safety viewpoint. This project will have an important impact in ESA standards adopted by most of the European space industry (e.g., [ECSS 2002]).

4.6.2 Future Trends

A set of well-known problems impair today's Verification and Validation process of the on-board spacecraft software:

- The inadequate sizing of on-board resources to accommodate required functionality
- The exponential increase of on-board software functionality
- The systematic underestimation of the verification and validation resources
- The manual testing and validation implementation, without easy-to-use means of determining test coverage and completeness.

The major European customers and companies concerned with the development of space software, such as the European Space Agency (ESA), the Centre National d'Etudes Spatiales (CNES), and the Defence Evaluation and Research Agency (DERA[19]) have recently initiated important actions (e.g., through R&D programs) for so-called "technology harmonisation". This encompasses not only the analysis of common failures in large space projects, but also the elaboration of appropriate recommendations. A summary of these recommendations regarding V&V is provided hereafter:

- The functionalities of the current test and validation facilities should be extended. Today, the available facilities are mainly meant to software integration and do not really support the design and development of software test benches.
- A clear picture of the software test coverage and completeness should be provided when the validation process is declared finished. This especially requires having full observability and control of the software under test.
- Automatic means and tools should be developed to support regression testing after software changes.
- More emphasis should be put on robustness testing (i.e., ability to isolate and minimise the effect of errors), at both sub-system and system level.
- Safety and mission critical software should be submitted for an Independent Software Verification & Validation (ISVV) process.

[18] http://www.criticalsoftware.com/

[19] from 2001 DERA is separated into two organizations: an independent science and technology company QinetiQ and MOD's Defence Science and Technology Lab

- The suitability of all pre-developed software (e.g., COTS) proposed for reuse should be evaluated for its intended use as required by the new ECSS standards.

Undoubtedly, these recommendations will be the main drivers for the future trends and directions of the European space industry concerning software verification and validation of on-board spacecraft software.

5 Mobile Computing

Recent technological advances (such as GPRS, IEEE 802.11, Bluetooth and UMTS [UMTS]) have provided portable computers with wireless interfaces that allow remote communication even while a user is mobile. Mobile computing - the use of a portable computer capable of wireless networking – has revolutionised the way computers are used by organizations and individuals.

The interest in new services can be observed by the degree of success and wide adoption of the new generation of mobile phones, also known as smart phones. Smart phones extend basic mobile phone functionalities with a myriad of services such as email and Internet access, digital cameras, MPEG players, personal organizer facilities, and games. Smart phones are one step toward the integration of Personal Digital Assistants (PDAs), portable music players, gaming consoles, and mobile phones. Using these devices, users will have the opportunity to connect to the Internet at any time. These new devices were made possible thanks to the significant improvements in the underlying communication infrastructure and technology. In terms of bandwidth it is worth noticing that it is very likely that the cellular networks will be released from current bandwidth constraints imposed by GSM, with the deployment of the UMTS networks. Wireless Internet Service Providers are progressively increasing their coverage over populated and business areas such as airports, convention centers, etc. For example, a testbed for large-scale ad hoc wireless networks for metropolitan areas is being constructed in Dublin in the scope of the WAND project.

Wireless networks of the future will connect microprocessors embedded in all the devices surrounding us. The millions of nodes per city that such a future scenario implies point to new types of applications where the instantaneous delivery of data is not necessarily important and where structureless networks will rely on ad hoc connections between nearby nodes to establish multi-hop dynamic routes in order to propagate data and messages between out-of-range nodes. This is a vision of ubiquitous computing.

Ubiquitous computing is becoming a reality and requires a pervasive but not intrusive computing environment. The basic components of such a smart environment will be small nodes with sensing and wireless communication capabilities and with the ability to organize themselves flexibly into a network for data collection and delivery. Realising the capabilities of mobile and ubiquitous computing has significant challenges especially at the architectural and protocol/software level. Mobile and pervasive computing systems have different requirements to those of applications built for traditional computing systems [Belaramani et al 2003]. Mobile applications cannot assume that they have the quality and stability of resources that traditional applications can. Extensive research is being carried out in the fields of communication protocols, data processing and application support. Some of the current challenges and leading edge projects in mobile computing will be outlined in this chapter.

Some interesting applications in the area are already emerging and, among the research projects from CaberNet project members one can find several illustrative examples, including: traffic monitoring and driving support (JETS, ITransIT, and UTC-NG), provision of multimedia information to museum visitors in the CARMEN project, support of search and rescue operations (project Emergency Multimedia), Smart-spaces (in which projects CAMS, AIMS, GRASS, FMCAT, WhATT and GLOSS are contributing), goods handling supervision (the Aware Goods project), stock management (project Smart Shelf), office augmentation (project Smart Office) and customer-oriented ambient intelligence (project Ozone).

A common characteristic among the majority of the projects listed above is that they require combined research in both communication protocols and middleware components. We believe this pattern will remain common over the next few years. Another characteristic is that research is driving the emergence of a myriad of, often disparate, middleware component that, sooner or later, will need to be integrated in order to support a wide range of applications.

The next section discusses forthcoming technological improvements that can and, very likely, will affect the development of mobile systems. Section 5.3 briefly outlines current approaches at the level of data-link and hardware. The following two sections examine current RTD in ad-hoc networks and context-aware systems. Section 5.6 outlines middleware services developed and employed in mobile systems. The last section of this chapter summarizes future trends in the area.

5.1 Forthcoming Technological Improvements

Mobile devices are often characterised by enumerating their constraints in comparison with fixed, desktop computers. This includes the lack of computational power, bandwidth and most remarkably, the small (restricted) autonomy, which results from the small capacity of existing batteries. Research has been pursuing solutions to these constraints. The next few years a new generation of devices should emerge that realises the best of both worlds: the computational power and bandwidth of desktop computers with the weight of mobile devices. We can identify two important future breakthroughs:

- Bandwidth: forthcoming standards are bringing to the wireless media bandwidths hardly achievable in wired LANs a few years ago. This is the case of the IEEE802.11a and IEEE802.11g standards which propose bandwidths up to 54Mb/s for Wireless LANs. Higher bandwidths are already in use in Line-Of-Sight wireless links. The mobile phone industry is also promising that 4G network bandwidth will approach 100Mb/s.

- Fuel Cells will be one of the new generation of power supply for mobile devices. Besides providing longer autonomy, users will be able to refuel the devices anywhere. Fuel cells are not yet ready for mass production but the latest results are promising and experts believe that the technology will soon be able to enter the market.

Solving the power limitations of mobile devices would have a significant impact on all the remaining characteristics. Computational and storage capacity will be able to grow at a pace comparable to desktop computers. The constraints imposed on the number of messages exchanged by wireless devices and on the transmission power will also become less relevant. This will impact transmission range, bandwidth and link level protocols. On the other hand, mobile devices are inevitably different from desktop computers in size, weight and interface. Disconnection periods, interference and high error rates will continue to characterize the wireless communication medium. It is expected that the diminishing importance of power constraints will shift the direction of ongoing research.

At the same time not all wireless devices will be equally affected by these changes. Sensors are a particular class of devices that aim at providing the context information required by many mobile applications. Sensors will not get substantial benefits from fuel cells because it will not be practical to refuel thousands or millions of devices, massively deployed over large areas or embedded in the environment. Therefore, research on power saving protocols is likely to continue though with a more limited scope of application.

5.2 Research at the Data-link Level and Hardware

Current market directions indicate the future dominance of a few data-link level protocols. In particular, the IEEE802.11 family of protocols has already seen a significant amount of deployment. It is however worth noting that there are a number of disadvantages associated with these protocols.

Collision Avoidance protocols are unpredictable in the sense that devices cannot foresee the moment at which they will be able to send or receive some message. Therefore, devices must stay in the idle state, constantly listening to the medium. This is a resource consuming operation that contributes to the fast draining of battery life. Battery life can be conserved in sleep state, a power saving mode that temporarily disables network cards. Switching between the idle and the sleep state requires the data-link level protocol to schedule the transmissions of all devices and let the receiver know in advance the moment at which the transmission will take place.

Additionally, the unpredictability associated with collision avoidance protocols prevents applications from bounding message latency. Messages can suffer arbitrary delays, for example, due to successive collisions. This can significantly impact the performance of both soft and hard real-time applications.

Both energy saving and real-time constraints have favoured the emergence of alternative protocols. In this context, the Time-Division Multiple Access (TDMA) protocol emerges as an interesting alternative. Research toward this direction is being pursued by the EYES, Smart-Its and TBMAC projects. The R-Fieldbus project is concerned with the problems posed by real-time applications, which are adapting real-time solutions designed for the wired infrastructure to the mobile environment.

Sensors will play a fundamental role as context providers for the upcoming mobile applications. How sensors will collect the information and provide it to the application and other sensors continues to be an open subject, which is being addressed by projects such as CAMS, DSoS and SeNDT. Programmable sensors are an

interesting research trend that aims at developing a generation of sensors whose actions can be customized according to the desired application domain. Existing prototypes developed in the scope of the Relate and Smart Its project use a software development kit for defining programs in the C language. Code can then be uploaded using a wireless communication protocol such as TDMA or Bluetooth.

5.3 Ad Hoc Network

In [Al-Bar and Wakeman 2001] the three main factors affecting the performance of mobile applications are categorised: device constraints, network constraints and mobility constraints. Device constraints are often related to the portability requirements of resource-constrained heterogeneous mobile devices. The communication model of the device may introduce network constraints characterised by constrained bandwidth, power consumption and latency. Mobility constraints are introduced by the movement of users (physical space constraints), information (information space constraints) and devices (connection space constraints). These constraints do not limit traditional or static networked applications to the same extent or the same severity as mobile applications. Wireless communication faces more obstacles than wired communication because the surrounding environment interacts with the signal, blocking signal paths and introducing noise and echoes [Forman and Zahorjan 1994]. As a result wireless communication is characterised by lower bandwidths, higher error rates and more frequent disconnections, increased by mobility. Ad hoc wireless networks include and increase the challenges of infrastructure wireless networks as the communication relies on the peer-to-peer connectivity of the autonomous mobile nodes. Here we pay particular attention to the challenges of mobile ad hoc wireless networks. The absence of a fixed infrastructure means that nodes in an ad hoc network communicate directly with one another in a peer-to-peer fashion. The mobile nodes themselves constitute the communication infrastructure – a node acts as both a router and an end host. As nodes move in and out of range of other nodes, the connectivity and network topology changes dynamically [Wang and Li 2002].

The assumption of permanent connectivity and synchronous communication of traditional middleware is not applicable in this infrastructureless, dynamic mobile network. The rate of link failure in an ad hoc network is directly related to node mobility: greater mobility increases the fluctuations in link quality, the volume of topological updates, the time spent processing the updates (e.g., for route discovery protocols), and congestion due to increased update transmissions and retransmissions. Realising QoS guarantees is challenging and complex in this dynamic, unpredictable environment. The CORTEX project is exploring the fundamental theoretical and engineering issues that must be addressed to deliver real-time, context-aware collaborative applications composed of collections of sentient objects, i.e., mobile intelligent software components, which interact with each other and demand predictable and sometimes guaranteed QoS in a dynamic mobile wireless network.

The CORTEX project requires guaranteed real-time event-based communication between the mobile sentient objects. Previous research on event-based middleware for wireless networks has mainly focused on nomadic applications, which are characterised by mobile nodes using the wireless network primarily to connect to a fixed infrastructure network, such as the Internet, that may experience periods of disconnection while moving between points of connectivity. The collaborative applications available via CORTEX requires a novel event-based communication model which does not rely on the presence of any separate infrastructure, e.g. the model developed in the STEAM project, and which defines temporal and dependable management of QoS constraints, e.g., the Generic Event ARchitecture (GEAR) project [Verissimo and Casimiro 2003], which uses perfect timely failure detection such as supported by a timely computing base (TCB) [Verissimo and Casimiro 2002].

Minimizing end-to-end latency is critical to achieving the timeliness requirements of real-time event-based communication. Collisions cause unpredictable latency for medium access that is unacceptable in real-time event-based communication, where each mobile node must have time-bounded access to the wireless medium to transmit real-time events. In the TBMAC project the first time-bounded MAC protocol for multi-hop wireless ad hoc networks has been developed.

It is foreseen that future mission-critical real-time communication systems will be comprised of networked components that will act autonomously in responding to a myriad of inputs to affect and control their surrounding environment. The advances made by such projects as CORTEX, STEAM and TBMAC will enable a new generation of collaborative applications in areas such as intelligent vehicles, mobile robotics, smart buildings and traffic management.

In mobile wireless networks the available bandwidth is very limited and some wireless devices have severe energy constraints, relying for example on battery power. Current research into power management for extended battery life, for example [Gomez et al 1999] has focused on stand-alone mobile devices. With the vision of collaborative and ubiquitous computing environments, increased emphasis will be placed on the power

management of a collection of mobile devices. The WhATT project is investigating power management in "smart space" environments, and the EYES project is developing new architectural schemes and communication protocols for energy efficiency in limited power sensor networks.

We should mention, however, that there is a concern coming from industry that some of the existing models used in academic research are not always complex enough to grasp all the important characteristics of real life systems, which clearly impedes their applicability. This is understandable considering the novelty of the area and the complexity of the systems but there is clearly a need for further research and for better coordination with the industrial product providers.

5.4 Context-awareness

A context-aware system can be aware of its environment in such a way as to integrate itself better into this environment. Context-awareness is of particular importance in mobile computing, where the operating environment is constantly changing due to the mobility of devices and the characteristics of wireless communication technologies. In such environments, context-awareness can enable applications to respond intelligently to variable bandwidth, unreliable connections and the varied economy of different connections.

One of the greatest challenges in context-aware computing, and one that has not yet been adequately addressed, is the provision of middleware and services to support the application developer. The major problem lies in providing generic support for the acquisition and use of multiple fragments of information gleaned from (potentially unreliable) multi-modal sensors in a mobile environment. The Sentient Object Model project defines software abstractions that ease the use of sensor information, and associated actuation, by context-aware applications. The CORTEX project uses the sentient object model for interaction between sentient objects or the environment possibly by stigmergic coordination[20] [Fitzpatrick et al 2002].

An additional facet of context-awareness for CORTEX is infrastructural awareness, primarily the QoS achievable by the infrastructure. CORTEX requires highly adaptive behaviour accounting for dynamically varying network parameters. As pervasive environments and ubiquitous computing become a reality, the differences between infrastructure and wireless networks will become ever more transparent to the user, i.e. it will become increasingly difficult to distinguish wireless ad-hoc infrastructures with wired, fixed ones. A mobile user will expect context rich information regardless of current physical location or network structure. The CAMS project is investigating extracting meaningful context from sensor-rich environments to enable context-sensitive applications. The CAMS middleware processes the data flow either by augmentation or fusion to generate an event stream that is monitored by applications, essentially creating a smart environment. The CASCO project is investigating how the sharing of context in ubiquitous environments can support cooperation in a smart environment and the security and privacy issues that arise. Facilitating the development of these types of smart environment requires diverse areas of research - from computer vision, to distributed computing, to human computer interaction. The AIMS project investigates the use of computer vision to detect and track objects and people as they move about a smart space and, in particular, how computer vision can be used in conjunction with other types of sensors to acquire context information from the environment. Research into smart environments is in fact escalating especially in relation to ubiquitous computing, and is discussed in the next section.

Developments in context-awareness enhance the services provided by context-aware applications to mobile users. An example is the location-dependent dissemination of multimedia information to mobile users at particular locations of relevance. With advances in mobile computing, users now expect and demand uninterrupted multimedia applications with guaranteed QoS regardless of location. The CARMEN project aims to combine the developments in context-awareness, particularly location-awareness, with guaranteed QoS by delivering a multimedia application development framework for applications that reside in a wireless ad hoc network to support and manage the dissemination of multimedia hearsay[21].

The services offered by context-aware applications may require dynamic adaptation as the execution environment, the application resources and user demands change, in a possibly erratic and unpredictable manner. When an application needs to adapt it is often not because the core problem domain of the application has changed, but rather that a non-functional requirement or behaviour of some object within the application needs

[20] stigmergic coordination describes a form of asynchronous interaction and information exchange among entities mediated by an active environment [Almeida and Verissimo 1998]

[21] multimedia notes left for users in particular locations

to change [Keeney and Cahill 2003]. The Chisel project is developing an open framework for dynamic adaptation of services using reflective techniques in a policy-driven, context-aware manner. The objective is to make mobile-aware dynamic changes to the behaviour of the middleware and to support the addition of new unanticipated behaviours at run-time, without changing or stopping the middleware or applications using it. Guaranteeing continuous communication in a mobile computing environment encapsulating heterogeneous devices and networks requires dynamic adaptation, and at a user level, adaptable mobile applications. At the application layer, aspect-oriented programming projects, for example WEAVE.NET, and reflective projects, for example CHISEL, take a policy-driven approach allowing different application-specific and user-specific polices to control the dynamic adaptation of component behaviour including the addition of non-functional behaviours. Dynamic adaptation at any level requires additional intelligence or context awareness to allow better integration with the dynamic environment.

5.5 Middleware Research

The development of any non-trivial mobile application using context information or any form of wireless communication today requires a significant level of expertise from the application developer. Most of the complexity derives from the need to interface a myriad of devices, communication media and link conditions. The programmer's task would become significantly simpler if she were provided with the appropriate software development tools. At a generic level, several research projects have been pursuing the goal of providing to the application developer a more adequate framework for mobile application development. Different frameworks have been developed with different emphases including multimedia (see CARMEN), context-awareness (see CASCO), location-awareness (see FLARE and Nexus), smart-spaces (see GRASS), Mobile Ad hoc NETworks (MANETs) (see MICS), and ambient intelligence (see project Ozone). In the following we will provide outline descriptions of some of these frameworks.

Service discovery and interfaces

At any given moment, a mobile device may be in reach of a broad and varying range of other devices, with different characteristics and purposes. To discover all these devices, and the most appropriate mechanisms to interface with them, is a challenging task. The diversity of interfaces may be very large and a single device may provide the same information through more than one interface. This makes it infeasible to assume that all the drivers and communication protocols, required to retrieve all useful context information provided by the available sensors, are pre-loaded on every mobile device. Therefore, mechanisms are required that allow mobile devices to reconfigure themselves at run-time and download the appropriate interfaces when needed. The eXtensible Markup Language (XML) provides a promising approach to solving this problem. Jini has also been successfully tested for service discovery in one test-bed application in the scope of the CORTEX project. However, a definitive solution has yet to be found.

Event-driven communication

Short-range wireless communication will perform an important role in future mobile applications. Different devices, owned by the same or different users, will be required to exchange information among themselves and with nearby sensors. Sensors will be responsible for providing context information to users. The most common example of context information is location but the possible types of information can be as diverse as information about user movement, heartbeat rate, traffic information, temperature, humidity, etc.

Often, interactions between devices in a mobile environment are only possible, and relevant, for short periods of time, namely, while the devices are in transmission range. Therefore, it is important to avoid complex handshake procedures, which may consume a significant part of the available bandwidth, power and time.

Event-driven communication appears as an appropriate mechanism for such loosely coupled interactions. Existing research on event-based middleware for wireless networks has mainly focused on what may be termed *nomadic applications*. Such applications are characterized by the fact that mobile nodes make use of the wireless network primarily to connect to a fixed network infrastructure, such as the Internet, but may suffer periods of disconnection while moving between points of connectivity. Such applications typically employ *infrastructure networks*. As a result, most of this work has concentrated on handling disconnection while entities move from one access point to another [Bacon et al 2000][Cugola et al 2001][Podnar et al 2002][Sutton et al 2001][Huang and Garcia-Molina 2001]. In contrast, relatively little work has focused on accommodating *collaborative applications*. These applications are composed of mobile nodes that use the wireless network to communicate with each other within some common geographical area. Although these applications may use infrastructure networks, they will often use *ad hoc networks* to support communication without the need for a separate

infrastructure. Consequently, this collaborative style of application allows loosely coupled components to communicate and collaborate in a spontaneous manner. Collaborative applications may have requirements concerning real-time constraints for timeliness and reliability, for example communication between mobile robots or inter-vehicle communication. Timely and reliable communication is essential if applications in these domains are to be realised.

Application of the event-driven communication paradigm for mobile computing raises scalability problems. It is necessary to prevent receivers from being flooded with information which needs to be parsed in order to retrieve the useful items. To reach the destination, events may need to be forwarded by several intermediary nodes. Adequate policies concerning event dissemination and filtering need to be defined to prevent resource wasting. The Sentient Objects, STEAM and CAMS projects are addressing these problems. In this context, one of the most important concerns is event filtering. Devices should be able to specify which events they are interested in receiving and processing using, for instance, declarative rules. Three classes for event filtering are under definition: location (e.g. limit the event dissemination to devices closer than some distance to the sender), functional (e.g., filter based on the subject of the message) and non-functional (e.g. filter based on the quality of service to be provided).

STEAM [Meier and Cahill 2002][Meier and Cahill 2003] is an event-based middleware designed for use in wireless ad hoc networks and is intended for applications that include a large number of highly mobile application components typically distributed over a large geographical area. Unanticipated interaction between nearby components is supported, enabling a component to dynamically establish connections to other components within its current vicinity. This allows components representing real world objects currently located within the same geographical area to deliver event notifications at the location where they are relevant.

STEAM differs from most other event-based middleware in that its architecture does not rely on the presence of any separate infrastructure. The middleware is exclusively collocated with the collaborative application components and does not depend on centralised or intermediate components. Decentralised techniques for discovering peers and for filtering of event notifications are supported. Event notifications may be filtered at both the producer and the consumer side or may be filtered implicitly. Filters may be applied to a range of functional and non-function attributes associated with an event notification including subject, content, and context, such as geographical location. Combining distributed event notification filters is beneficial to the precision of filtering, allowing a component to define the subset of event notifications in which it is interested using multiple criteria, such as meaning, time, location, and quality of service. Event notification filtering in general and STEAM's approach of combining filters in particular improves system scalability by limiting forwarding of event notifications. To support ad hoc networks, event notification filters may be used to define geographical areas within which certain event notifications are valid, hence bounding the geographical scope within which these event notifications are propagated. Such geographical scopes represent a natural way to identify event notifications of interest for mobile components. Geographical scoping is essentially filtering of event notifications using the space criteria and consequently increases system scalability further. Bounding the dissemination range of event notifications improves the predictable behaviour of the middleware.

To achieve real-time event-based communication in a dynamic mobile ad hoc wireless network the unpredictability inherent in the environment must be reduced. STEAM uses a predictive technique for achieving timeliness and reliability for real-time event-based communication in ad hoc wireless networks [Hughes and Cahill 2003]. This approach is the first to directly address this issue and essentially relies on predictive techniques to alleviate the impediments to real-time event-based communication that are characteristic of mobile ad hoc environments. It essentially allows the non-functional requirements of event notifications to be mapped to the quality of service zone that is defined by the associated geographical scope. A proactive technique for reserving the required network resources based on predictions on the future behaviour of mobile entities and indeed proximities is used to route messages from a producer to consumers with a high probability.

Classical event/object models are usually software oriented. As such, when transported to a real-time, embedded systems setting, their harmony is cluttered by the conflict between, on the one side, end/receive of “software” events (message-based), and on the other side, input/output of “hardware” or “real-world” events, (register based). The Generic Events Architecture (GEAR) [Verissimo and Casimiro 2003], which emerged from the CORTEX Project, allows the seamless integration of physical and computer information flows. The GEAR architecture introduces the unifying concept of *generic event*, be it derived from the Boolean indication of a door opening sensor, from the electrical signal embodying a network packet (at the WLAN aerial) or from the arrival of a temperature event message. GEAR defines an event layer, the COSMIC (Cooperating SMart devICes) middleware, which is responsible for event propagation in the whole system, through several *Event Channels (EC)*. In fact, this layer plays a fundamental role in securing the functional and non-functional (e.g. reliability

and timeliness) properties of the envisaged applications. COSMIC event channels can be of several synchrony classes, ranging from non real-time to hard real-time classes.

Hermes [Pietzuch and Bacon 2002] is a large-scale, event-based middleware that uses XML for event transport among a network of brokers. Hermes is built on a peer-to-peer overlay network. Programmers in end systems may use a standard programming language, such as Java, and XPath is used to support automatic conversion. Hermes provides distributed filtering for notifications and advertisements, using peer-2-peer to define rendezvous, and an event composition service.

The SCRIBE system developed at MS Cambridge is a generic, scalable and efficient event notification system. It provides application-level multicast based on the publish-subscribe paradigm. SCRIBE is self-organizing, flexible, highly scalable and efficient. Applications of SCRIBE include instant messaging/presence notification; news broadcast and stock quote dissemination.

Challenges of future distributed event-based systems include support for applications that typically rely on a heterogeneous communication system comprised of networks with largely different quality characteristics ranging from highly predictable real-time buses to wireless ad-hoc networks in which mobility of nodes result in frequent link failures. Event-based systems accommodating such applications should seamlessly disseminate relevant information generated by deeply embedded controllers to all interested entities in the global network regardless of the characteristics of the specific networks involved.

Distribution transparency

One classical goal of operating systems and middleware designed for distributed systems is to hide the fact that the system is distributed and to provide the illusion, to the application programmer and user, of a single powerful machine. In this case, one says that distribution is “transparent”. In such systems, resources like computation, memory or storage may be provided by different hosts without the user being aware of which resources are being used. Applying this concept to mobile computing greatly augments the range of possible applications. Mobile devices may delegate part of the tasks on different devices, facilitating load-balancing and power consumption distribution. Also, users could commute from one device to the other while continuing to have access to the same applications and data, without having to explicitly synchronize them. Users would then be able to work on the device providing the most appropriate interface in their neighbourhood. Such a complex environment is being pursued in the scope of the 2WEAR project. Load and resources balancing is addressed in the scope of the WhATT project.

A particular mechanism that helps to hide distribution, and that is currently the fundamental building block in most distributed architectures, is the remote procedure call (RPC) mechanism. RPCs attempt to mask differences between the client, issuing the call, and the server, executing it. With the generalization of mobile computing, it will be important to hide the problems associated with mobility from both wired and mobile computers. Middleware must mask temporary disconnections from mobile applications. Possible approaches are buffering RPC invocations for transmission as soon as connectivity is re-established. However, mobility and, more specifically IP address changes, can also raise problems when fixed hosts attempt to perform RPCs to mobile hosts. The solutions provided by Mobile IP may not be fully satisfactory for specialized references such as those used by CORBA or DCOM. This problem has been addressed by the VIVIAN and ALICE projects.

Group communication

Group communication is a paradigm providing useful consistency properties for applications requiring multi-participant interactions (see Chapter 10). In the wired setting, group communication has proven to be extremely useful in many application domains. It is likely that similar applications, with the same requirements, will also emerge for the wireless domain. However, traditional membership protocols designed for the wired environment perform badly with a large number of nodes and need to be executed whenever one node leaves or joins the group. Mobile computing is therefore an adverse environment for strongly consistent multi-party interactions, such as virtual synchronous protocols. Effort has been concentrating on the definition of new membership criteria and useful weakened semantics. Current attempts include using a WAN-of-CAN structure to group nodes so as to reflect potentially heterogeneous network architectures, a potential scenario encompassed in the CORTEX project.

Real-time

The high error rates and the mobility of the devices may introduce significant delays in communication. The large variations in observed latency make the wireless medium particularly adverse for real-time applications. Attempts to mitigate these problems can be found at the middleware level, for instance, by adapting dynamically

the communication protocols and the applications in response to changes in the execution environment. Future mobile applications will benefit from middleware capable of, for example, introducing Forward Error Correction (FEC) codes on demand, adapting frame rate and/or frame resolution and regulating wireless network traffic, all according to the existing conditions (see the project Fault-tolerance in Audio/Video Communications via Best Effort Networks). Pro-activity will also play a significant role by anticipating the changes on the network conditions (see project STEAM). There is also ongoing research pursuing the definition of hybrid solutions, that involve allocating part of the spectrum for a real-time control component.

Security

Developments in mobile computing and in particular ubiquitous computing foresee a massively networked environment supporting a large population of diverse but cooperating autonomous mobile entities. This infrastructure introduces new security challenges that are not adequately addressed by existing security models and mechanisms. Secure collaboration in ubiquitous and pervasive computer systems requires security models to accomplish mutual trust whilst minimising the need for human interaction. The potential size of the infrastructure means that the security policy must encompass billions of potential collaborators. Mobile computational entities are likely to become disconnected from their home network, thus the mobile entities must have the ability to make fully autonomous security decisions; they cannot rely on a specific security infrastructure such as certificate authorities and authorisation servers. In addition the dynamism of the infrastructure means that mobile entities offering services may be confronted with requests from entities that they have never met before, and mobile entities will need to obtain services within environments that are unfamiliar and possibly hostile.

The SECURE project is investigating these security challenges to develop a self-configuring security system based on the human notion of trust, to provide mutual trust between ubiquitous devices, which must share information and work together to present an unobtrusive interface to their users. Various security models have been proposed for different types of distributed application requirements and numerous types of execution environments. Adding code to the application to authenticate principals, authorise operations and establish secure communication among distributed software components typically reinforces these security models. This code is often application and context-specific, which makes it difficult to integrate applications with each other [Abendroth and Jensen 2003]. The CUCHULAINN project proposes a new unified access control mechanism that supports most existing security models and offers a number of additional controls that are not normally provided by security mechanisms.

Adaptability

One possibility for coping with mutable network conditions is to delegate the tuning of run-time parameters to a specialized component that can react quickly and provide applications with the most appropriate configuration for the executing context. Examples of adaptation parameters are the level of security, the use of Forward Error Correction codes or the frame rate of a video stream. Failing to react appropriately to changing network conditions may result on inefficiencies at several levels including power consumption, bandwidth use, or user dissatisfaction. Adaptability is a vertical concern which has already been referred to in several of the sections above. Research results show that adaptability will play a fundamental role in future applications. Furthermore, automatic reconfiguration may remove from the application developer the burden of foreseeing all possible combinations of run-time parameters that are relevant at execution time. Instead, such a task can be delegated to specialized, self-configuring middleware. At different levels, several CaberNet members' projects have been addressing this subject. This is the case of the CHISEL, 2WEAR, CORTEX, COMCAR and the "Fault-tolerance in Audio/Video Communications via Best Effort Networks" projects.

Information retrieval

To operate properly, computers are likely to require a large amount of information, provided by a large number of devices dispersed over distinct locations. Most of the devices will be hidden in everyday objects like clothes, watches, desks, and vehicles or in the surrounding environment. The ability to retrieve, select and interpret information will be one of the challenges for the near future.

Information retrieval is challenging because one needs to discover and interoperate with the relevant devices, without being overwhelmed by communication with non-relevant information providers. It is also possible that the required information will be provided by more than one device, which would require the selection of the most appropriate source, based on criteria such as location, security, etc. The exchange of information will

require negotiation, possibly with third parties, of the data-link level and routing protocol. The communication should be made using the more appropriate configuration for the existing conditions (e.g. error rate, traffic) which are subject to change without notice and tolerate temporary disconnection periods. Some kinds of information may require special properties, like timely or guaranteed delivery, confidentiality, or may require contributions from several participants (such as movement updates in a multi-player game). The understanding of retrieved information is impaired by the use of different formats, syntactic rules and semantic conventions among the different sources. Location information for example can be absolute (as provided by a GPS receiver) or relative to some origin. Accuracy can also vary depending on the precision of the device.

Software licensing

The emergence of new platforms such as mobile computing will revolutionise how software is licensed and managed. The Pervasive Software Licensing (PerSL) project is investigating use of Web Services as a platform for pervasive computing, with a focus on the application area of software licensing for mobile devices.

5.6 Future Trends

Pervasive computing is maturing from its origins as an academic research area to becoming highly commercial [Banavar et al 2000]. Ubiquitous and pervasive computing is concerned with bringing about the “disappearance” of the computer, embedding it naturally so that computer technology can be used transparently, as part of our daily lives. The vision of ubiquitous computing requires the development of devices and technologies, which can be pervasive without being intrusive, and form a smart environment.

In some sense, mobile computing is revisiting many classical problems of distributed computing in a specific context characterized by limited resources and intermittent connectivity. Given the advances foreseen in the underlying technology, it is likely that some of these constraints will vanish: mobile devices of the future will have more available power, memory, disc and bandwidth than some of their current fixed counterparts. However, ongoing research has identified a number of challenges specific to mobile computing that are likely to remain or even be exacerbated in the future.

One is that the mobile environment is characterized by a large diversity of interactions and execution conditions. Therefore, it is very unlikely that the sort of static, carefully pre-tuned, solutions that were so successful in the wired world will succeed in the mobile arena. Thus, adaptive and reconfigurable systems assume a role of extreme relevance in this context.

Another intrinsic characteristic of mobile applications is the need to interact constantly with the environment, collect information from a wide range of surrounding sensors, and modify the environment by controlling different actuators. This calls for the development of loosely coupled interaction schemes, such as event dissemination and exchange. On the other hand, the massive use of these highly unstructured communication patterns may adversely affect the performance of the devices and make it extremely difficult, if not impossible, to enforce any sort of quality of service in the communication. The right balance of these conflicting forces is still an issue for ongoing research.

Finally, on a more immediate and practical level, it is extremely important to define middleware architectures for mobile environments that gain the same level of wide acceptance as similar constructs have reached in the wired world (for instance, CORBA, .NET, etc). A common set of interfaces and services would certainly constitute a solid ground for a faster emergence of sophisticated mobile applications that could be deployed in a wide range of devices.

Part III. System Architecture

6 Operating Systems

The traditional image of operating systems is changing because processors are now integrated in a variety of objects and environments: mobile phones, smart cards, sensors, etc. These environments impose specific requirements and constraints: security, availability, fine-grain control of the usage of resources, flexibility, and adaptability. Another change is brought about by the development of closely coupled processor networks and clusters. In the following sections, we have singled out a few emerging topics of operating systems research.

6.1 New Architectures and Paradigms

The requirements of adaptability and flexibility have contributed to a move towards a "deconstruction" of the traditional operating system. This trend started in the 1980s with the emergence of micro-kernel architectures such as Mach and Chorus. However, microkernels did not bring all the expected benefits: there were found too rigid (not leaving enough flexibility to resource allocation policies) and suffered from performance problems. Starting in the mid 1990s, a new current of research has attempted a finer grain decomposition of the operating systems kernels. Projects such as Exokernel/Aegis, Spin, OS-Kit, Vino, and others have introduced the idea of a highly flexible, "componentised" operating system, which could even be dynamically composed to respond to changes in its environment. The main challenge is to preserve the security and integrity of the systems in the face of dynamic change.

Think (acronym of "Think Is Not A Kernel") [Senart et al 2002][Charra and Senart 2002] is a software framework (under development in the Sardes Project) inspired by the exokernel model. However, Think goes one step further than the original exokernel: its interface does not provide any abstraction at all, but essentially reifies the underlying hardware interface; and it provides a lightweight software framework for developing the basic functions of an operating system (naming, binding and communication). Think is intended to be used for embedded systems, in which flexibility and a small footprint are strong requirements. Starting from the original Think system [Fassino et al 2002], a lightweight component structure has been developed to provide the system with capabilities for dynamic composition and reconfiguration. Experimental embedded applications have been developed with Think in the area of mobile robots and active networks. The main work on Think has been done in collaboration between the Sardes project and France Telecom Research and Development.

Resource Management

Resource management is an important function of an operating system. In traditional OS, the resource management policy is entirely controlled by the system. However, the flexibility and adaptability requirements imposed by the new applications and environments have led to more interaction between the users and the system itself. The system may provide "hooks" through which information on resource usage and requirements may flow, both ways, between the users and the kernel. An early example is "scheduler activations", a mechanism through which part of the task of thread scheduling may be delegated to the application, while not compromising the system safety. Other examples are related to memory resources and, more recently, to energy management.

Storage and File Systems

The challenge of file system design is to take the best advantage of the underlying storage hardware in order to maximize performance, while ensuring availability and data security. Examples of progress are log-structured file systems, which improve performance through more efficient writes, and distributed file systems, which exploit the overall capacity of networked storage. Recent architectures tend to explore this latter path further, e.g., by efficiently using the aggregated storage capacity of a local area network or a cluster. This may be combined with the use of caching. There is a trend toward considering the file system as an autonomous subsystem, with decentralized "intelligence", and architectural advances, such as network-attached devices, go in the same direction.

The Sardes project is now investigating a form of distributed storage for clusters [Hansen 2001a][Hansen and Jul 2001][Hansen 2001b][Hansen and Lachaize 2002], with the following goals: optimizing the global sharing of devices within a cluster (possibly using remote DMA facilities); and exploring the use of component-based architecture for I/O systems. It is expected that the extensibility and scalability of I/O in clusters, and the usage ratio of the resources will be improved. A modular I/O infrastructure called Proboscis has been developed to allow global I/O management within a cluster: a node may access to I/O devices located on another node. The current version may be used on both Ethernet and SCI networks. Measurements show that remote access does

not entail significant performance loss, and that distributing storage management over a cluster is a workable solution. Under extreme load conditions, the slowdown of a node is limited to 10 to 20% (in execution time) for SCI networks and 50% for Ethernet networks.

It is important to realise that the volume of data to be accessed is dramatically growing and in the future there will be a need to store and to deal with data in various effective ways, allowing, for example, for fast searches, etc, by using indexing and other ways of data structuring.

Security and Trust

Security has always been an important issue in operating systems. The importance of security is increasing with the advent of more open and dynamically changing systems. Active tracks of research are concerned with the security of "imported" code, for which formal methods ("proof carrying code") are being explored. Other projects investigate various forms of trusted kernels, with different levels of trust being attached to the layers of the system.

Within the Sardes project a protection framework has been recently incorporated in the Think system [Rippert and Stefani 2002a][Rippert and Stefani 2002b]. The work was done on four types of resources: processor, RAM, disk storage, and network connections. The framework is able to protect against the following attacks: service denial by packet flooding or by attack on the disk channel; memory corruption, direct or by insertion of malicious code.

Operating Systems for Constrained Environments

Following the trend towards pervasive and ubiquitous computing, specialised systems are being developed, for which customized operating systems need to be designed. Two examples are operating systems for mobile robots, and operating systems for sensor networks. In both cases, the system must take into account strong environmental constraints such as energy consumption, memory footprint, communication bandwidth, etc., so that sensor networks and their operating systems are now a fast developing area of research (e.g., projects such as TinyOS).

Cluster Operating Systems

Clusters of closely coupled computers have emerged as a low-cost alternative to high-performance computers, for a variety of applications. Initially limited to high-performance scientific computing, the application domain of clusters now includes information management, e.g., web servers. The challenge of a cluster operating system is to exploit as efficiently as possible the aggregated resources of the cluster. The function of observation and administration (e.g., reconfiguring the cluster in case of partial failure) is also more developed than in a single-machine OS.

The Sardes project has investigated the use of clusters for data management [Cecchet et al 2002][Cecchet and Marguerite 2003], specifically, the focus is on improving the performance of the J2EE server architecture through the use of clusters. There are two main tracks of work: configurable load balancing mechanism for Enterprise Java Beans (EJB); and replicating databases for availability and performance. The latter track has led to the development of C-JDBC, a software framework that allows any application using the standard JDBC interface to transparently access a database replicated on a possibly heterogeneous cluster.

Virtual Machine Monitors

One key future direction is, as often the case in systems research, a reincarnation of a hot-topic of the past: virtual machine monitors (VMM). Research projects in the United States (e.g. Denali at the University of Washington) and within Europe (e.g. the XenoServers project at the University of Cambridge) are reapplying VMM techniques to modern architectures with a view to supporting micro-services: components of Internet-scale applications requiring only a small fraction of the resources of a server**.** As part of the XenoServers project a hypervisor XenoMon has been developed which securely multiplexes the resources of a machine between a number of overlaying *guest* operating systems. Performing the resource division task at such a low level allows the support of multiple OS types on a single machine.

6.2 Future Trends

This is, in summary, the set of future trends and challenges which OS researchers and developers will be facing:

- Developing flexible componentized OSs
- Preserving security and integrity of the systems in the face of dynamic changes
- Efficient using the aggregated storage capacity of a local area and a cluster
- Developing OSs for constrained environments
- Exploiting as efficiently as possible the aggregated resources of a cluster
- Introducing advanced features supporting flexibility, openness and mobility into Virtual Machine Monitors.

Other intriguing avenues for future operating systems research include building tiny embedded systems for sensor network (or other extreme network systems), and support for “ad hoc supercomputing” in the Grid.

7 Distributed Object and Component Technologies

Distributed object technology takes the form of distributed platforms and abstractions that enable interactions between objects which are deployed on separate machines and networks. The basic conceptual underpinnings of this technology were worked out several years ago in the ISO Reference Model for Open Distributed Processing (RM-ODP) [ISO 1995]. Seamlessness and (selective) transparency are important goals of this technology: platforms typically try to provide the illusion that objects are accessed identically (or near identically) whether they are remote, or are within the same address space as the caller. Language and system independence are also widespread goals: platforms such as the OMG's CORBA make it possible for objects to be made available for remote invocation regardless of the language in which they are written or the system environment (e.g., operating system, CPU type) in which they exist. CORBA is now a well-established technology and has reached a degree of technical maturity. Similar platforms such as Microsoft's DCOM[22] and Sun's Java RMI are in a similar state of development. These environments also come with sets of generic services such as object trading, persistence, transactions, event propagation, media streaming, replication, etc. These are fairly widely used (some more than others) although in certain cases they can be criticised as being limited in scope and application (e.g., the replication services).

More recently, there has been a strong trend to raise the abstraction level of distributed object platforms by exploiting the notion of *components*. These are third-party deployable software modules that can be composed in ways unforeseen by their developers to produce desired behaviour. Component-based distributed systems also automate much of the manual housekeeping that must be performed by the application programmer in a typical distributed object environment. These factors lead to environments in which it is arguably easier and quicker to develop reliable, scalable, extensible distributed applications. Examples of commercially available component-based platforms are the OMG's CORBA Component Model (CCM), Microsoft's COM, COM+ and .NET and Sun's Enterprise Java Beans (EJB).

This chapter is structured as follows. First, Section 7.1 briefly characterises the state of the art in current object/component platforms. Then section 7.2 offers a broader perspective on the field that includes current research activities in distributed object/component systems in CaberNet and elsewhere. Finally, Section 7.3 identifies areas that we believe will become foci of attention in this area in the future.

7.1 Overview

All of models mentioned above support an integrated container-based environment for automating the management of generic, non-functional, properties such as transactions, security, persistency and event notification. Only one of the models, CCM, is not tied to a particular language (like Java) or operating system (like Windows). CCM was designed to align closely with the EJB specification and, apart from the language independence issue, these component models can broadly be considered conceptual equivalents. Both support different types of components which automatically determine the available container interfaces and the policies for managing the components' state and persistency (component-managed or container-managed). Furthermore, CCM and EJB define three approaches to demarcate transactions (cf. client-managed, container-managed and server-managed) while COM+ supports only automatic transactions (MTS-managed). Moreover, COM+ defines only one type of component. This leads to a simpler programming model, but also leads to limited expressiveness and deep dependence on the MTS environment. Despite being more difficult and complex to learn and manage, CCM and EJB architectures may be considered more flexible and open than COM+ which builds on top of the operating system services. Nevertheless, COM+ is a binary standard that allows the integration of several languages without compromising the performance.

Another significant aspect is the recurrent use of meta-information for describing the structure and behaviour of components. Meta-information is widely used in CCM (e.g. the interface repository), EJB (e.g. bean descriptors) and COM+ (e.g. type libraries) but is particularly visible in .NET where the metadata is embedded in the image files and then extracted using reflection to reason about the system and control assembly, enforce security, manage serialisation, perform compiler optimisations, etc. The combination of meta-information and reflection is an interesting approach for managing type evolution.

[22] there are clear indications that MS is now abandoning this work moving its interest to Web Services as the basic integration framework (see chapter 8)

Finally, it must be emphasised that these component models are inherently heavyweight and complex. For example, in their present form they are not suitable for deployment in most embedded environments. Nevertheless, they exhibit many potentially interesting features that would clearly be of interest to the developers of embedded systems. What is required is research to make such features available in component model environments that are considerably more lightweight and, probably, can be tailored to specific environments on a highly configurable 'what you want is what you pay for' basis.

7.2 Current and Recent Work

When taking a closer look at what practical distributed-object technology actually entails at present, it can be argued that we are not seeing distribution playing a main role, but actually only remote objects. Remote objects are characterized by the fact that their state is not distributed, but form indivisible units that are hosted by a single server, along with the code for performing state operations. Remote objects provide almost complete access transparency, meaning that clients residing at a different location can reference, access, and invoke remote objects just as if they were local objects. One can, however, argue that a truly distributed object is one whose state can be (transparently) distributed across multiple sites, and that the remote object is just a special case of a distributed object.

When observing current trends in distributed-object technology, we observe that attention has started to focus on the transparent distribution of an object across multiple sites. One of the first steps in this direction, exemplified in CaberNet by the ALICE project, is to provide support for mobile objects. Instead of having an object be permanently hosted by a single server, allowing an object to transparently migrate between different servers enhances location transparency, one of the different forms of distribution transparency identified by RM-ODP. There are various problems with supporting mobile objects. Prominent is the fact that moving an object requires that each of its hosts can support the execution of an object's associated methods (this is not a problem in Java RMI, but it is an issue in language independent environments such as CORBA).

A more difficult problem is that of allowing clients to maintain their binding with an object while it is moving. In its simplest form, a binding consists of an object reference containing precise information on where and how to contact an object. In practice, an object reference is location dependent: it typically contains the IP address of the machine executing an object server, the port number to which the server is listening, and a server-specific identifier for the referenced object. As soon as an object moves, all this information becomes obsolete, rendering the object reference useless. To solve this problem, different solutions have been proposed. Within CaberNet, the most prominent one is the Globe Location service which maintains, for each object, a mapping from a location-independent identifier to the object's (current) contact address. The location service is extensively described in [Ballintijn 2003]. Alternative solutions include forwarding pointers, broadcasting, and home-based approaches, which are all excellently surveyed in [Pitoura and Samaras 2001]. The main and common drawback of these alternative solutions is that they suffer from scalability problems. A recent, novel solution, that has not yet been extensively applied to mobile objects, is to adopt a flexible home-based solution deploying a peer-to-peer lookup service in structured overlay networks [Balikrishnan et al 2003].

Location transparency for distributed objects, even if they are mobile, can be largely regarded as a well-known problem with a number of satisfactory to good solutions. There is more to distribution transparency, however. One of the more important and more difficult issues regarding the distribution of an object across multiple sites, is that of dealing with replication. Replication is widely applied for different reasons: fault tolerance and scalability. Replication for fault tolerance is largely an issue of handling group communication, a topic that has received much attention, notably in Europe (see Chapter 10). Within CaberNet, much work in this area has been done at the University of Bologna as part of the Jgroup project. In this project, a programming abstraction is provided that allows a client to invoke methods on a group of objects, although the group itself is hidden behind an interface. In other words, to the client the group appears as just another object. A detailed description of Jgroup can be found in [Montresor 2000].

This approach towards transparently replicating remote objects has also been adopted for fault tolerance in CORBA (see [Moser et al 1998] for ideas on using object groups for fault tolerance in CORBA; the underlying concepts, however, go back to the ANSA project which preceded RM-ODP). One of the issues here is that different applications require different approaches towards replication. In other words, it was soon recognized that the middleware for replication should provide different replication policies, and that these policies should be separated from the mechanisms to enforce them.

The separation of polices and mechanisms has long been practiced in the operating systems communities, but has by now also found its way in the development of object-based middleware. *Reflection* is a key technique in this

regard. In a reflective object-based system, an application developer is generally offered the means to intercept invocations and take any additional measures before or after the interception takes place. In CORBA, reflection is supported by means of interceptors [Narasimham et al 1999] but more advanced approaches have been developed as well.

Research into reflective middleware is a major issue at Lancaster University, where efforts have been underway to develop reflection into open, object-based middleware using ODP's model of computation (see [Blair et al 2001] for an overview of the OpenORB project). It has also been a major concern in the Globe project at the Vrije University Amsterdam, where it was used to separate replication issues from object functionality [van Steen et al 1999]. Despite the need for separating concerns, incorporating mechanisms such as reflection into (object based) middleware has caused much debate concerning the role that middleware should have in supporting distributed systems. Middleware originated from the need to integrate different client-server applications [Bernstein 1996]. As such, an important goal was to provide support for hiding the intricacies of distribution of data, processes, and control from application developers and integrators. This view of providing distribution transparency has been formalized within the RM-ODP. Basically, it suggests that middleware should incorporate the mechanisms and the policies that realize transparency.

However, as experience with middleware grew, it became clear that simply hiding everything was not such a good idea. The problem is that full transparency is often interpreted as enforcing specific distribution polices, thus allowing few if any adjustments to satisfy application needs. As it turned out, a “one-size-fits-all” approach by which a single policy is used for all applications was simply too restrictive. Distribution has always been a difficult beast to tame.

A stricter separation between mechanisms and policies was badly needed. As a consequence, middleware research shifted its attention towards providing mechanisms, such as reflection, leaving the policies for the applications to decide. Developers and integrators were now suddenly confronted with having to solve problems related to distribution for which they hoped the middleware would provide solutions.

Of course, it is not a question of throwing back distribution problems to developers and integrators, but instead of creating awareness that functionality can and should be separated from distribution issues. Reflection allows us to build middleware systems that can easily be augmented with libraries implementing distribution policies. For developers, the issue is not to implement their own policies, but instead to select the appropriate policies for their applications. Middleware can provide default policies, but facilities should be available to override these defaults.

It should also be said that, in research at Lancaster University and elsewhere, reflection is seen as more than “just” a technique for separating concerns between policy and mechanism. It has been used, for example, to support run-time reconfiguration in middleware platforms so that these platforms can be more reactive and adaptive in hostile dynamic environments such as mobile computing, ad hoc networking, and real-time environments. For example, reflection is a major theme in the CORTEX project. CORTEX introduces a programming model inspired by the earlier sentient computing paradigm, based on the concept of sentient objects, which are distributed, mobile, and autonomous components. Sentient objects exist at very different levels of abstraction. At the lowest level, such objects might represent simple sensors or actuators capable of generating or consuming events. At a slightly higher level of abstraction, a tightly-coupled embedded system that integrates many such simple objects connected via a field bus might represent a single sentient object by component-aware composition, itself capable of generating and/or consuming events as a component of a larger system.

Current research indicates that we have come a long way in understanding reflective middleware, but there is still much to learn concerning the development and incorporation of distribution policies. In commercial middleware, the separation of functionality and distribution still has a long way to go, but the need will become more prevalent as distributed applications advance into the next decade.

Some of the concerns raised by industry with respect to component technologies are as follows. Their perception sometimes is that CORBA is too fragile to cope with changes, or to allow for adaptivity for dealing with faults, so that that it best suits inter-organisational integration of components. Moreover, CORBA solutions are not scalable. Some companies find EJBs too difficult to use because this technology is not really successful in hiding details of complex functionalities it provides. Another concern is difficulties in developing new services providing administration of EBJ containers.

7.3 Future Trends

Where are distributed-object technology and its supporting models heading? The trend has been set towards lowering the granularity of objects, while at the same time making them less location dependent (e.g., mobile), and distributing them across a geographically dispersed area. In other words, we are looking at scalability issues concerning the number of objects and their geographical distribution. New challenges concerning the development of flexible and lightweight middleware lie ahead of us. For example, we might in the future see middleware systems that are much more configurable and reconfigurable: platform instances might be generated from libraries of fine-grained components on a per-application basis. Such componentisation (along with reflection) is a likely basis for a systematic approach to runtime reconfiguration that might allow middleware platforms to thrive in hostile environments like mobile computing, ad hoc networking, etc. Note that this purported trend is a generalisation and extension of the current practice of components at the application level (e.g. in .NET and EJB) to the middleware and, perhaps even the operating system levels.

Another important trend is that such component-based configurability might be driven by high-level models or architectures which can automatically generate appropriate component configurations. The OMG's Model Driven Architecture is an important activity in this area.

Quality of Service support will remain an active area of research. Although it could be argued that we have a fair understanding how to specify QoS, devising the appropriate mechanisms to implement and enforce QoS policies is still very much an open research question. Reflection will help, but should be considered as the assembly language for policy enforcement.

The future will also see diversification of the application areas in which middleware technologies are applied. These technologies started out being applied in traditional client-server architectures and have been primarily deployed in business-oriented application domains. As mentioned above, however, there is increasing interest in applying these technologies in mobile computing and ad hoc networking. Real-time computing (see Chapter 4) is another promising area (the OMG has a RealTime CORBA specification but this suffers from a rather static and inflexible approach which might be considerably improved by an application of the above-mentioned fine-grained componentisation ideas). Other promising areas are programmable networking (e.g. in the recently initiated NETKIT project at Lancaster University), and flexible overlay networks (e.g. in the companion Open Overlays project).

Another possible, and in many ways complementary, direction of future development, with a particular interest from Microsoft, is that of active data. This may revolutionise (or, at least, dramatically change) the way that systems are integrated by associating services with the data. The traditional message-passing paradigm will be replaced with a higher level abstraction: active data components providing effective and consistent access, so that system integration will be accomplished by federating multiple data centres.

Finally, there is great potential in appropriately combining middleware technologies with Web Services platforms (see Chapter 8). This is also true for Grid platforms, which are built on Web Services foundations. Standard component platforms such as EJB, CCM and .NET are already used as back-end technologies in these environments, but we see additional scope for integration between the two families of technologies. For example, the generic services already available in CORBA would add considerable value to existing SOAP/WSDL based systems which are currently rather primitive in terms of services offered. The contribution can be both in terms of wrapping existing standards and in terms of contributing experience and expertise as the Web Services community starts to grapple with the provision of generic services (as it inevitably will). More radically, we see great benefits in the use of flexible middleware technologies in underpinning the basic Web Services platforms themselves (e.g., in terms of providing selection of underlying protocols and QoS, and in supporting configuration and reconfiguration as discussed above).

8 Service-oriented Architecture: Web Services

A surging interest in the area of Service Oriented Architectures (SOA) [Ferguson et al 2003] is coming from all the parties involved: researchers, various branches of industry, users, software developers, academia, etc. SOA introduces a powerful solution that enables the development of open flexible adaptive systems in which components have a high degree of autonomy. Web Services (WS) provide a technology enabling a Service-Oriented Architecture (SOA). A general SOA is built using a collection of autonomous services, identified by their references (URLs in WS), with interfaces documented using an interface description (e.g. Web Service Description Language – WSDL – WS), and processing well-defined XML messages. SOA is a natural complement to the object-oriented, procedural and data centric approaches. The first SOA for many people was with the use DCOM or ORBs based on the CORBA specification. But SOA develops these ideas much further. A SOA differs from the previous integration technologies in one key aspect: binding. Services interact based on what functions they provide and how they deliver them.

Following [Ferguson et al 2003] we outline the main defining characteristics of the SAO:

1. **Services are described by schema and contract not type.** Unlike previous systems, the SOA model does not operate on the notion of shared types that require common implementation. Rather, services interact based solely on contracts (e.g. WSDL/BPEL4WS for message processing behaviour in WS) and schemas (WSDL/XSD for message structure in WS). The separation between structure and behaviour and the explicit, machine-verifiable description of these characteristics simplifies integration in heterogeneous environments. Furthermore, this information sufficiently characterizes the service interface so that application integration does not require a shared execution environment to create the messages structure or behaviour. The service-oriented model assumes a fully distributed environment where it is difficult, if not impossible, to propagate changes in schema and/or contract to all parties that have encountered a service.

2. **Service compatibility is more than type compatibility.** Procedural and Object-oriented designs typically equate type compatibility with semantic compatibility. Service-orientation provides a richer model for determining compatibility. Structural compatibility is based on contract (WSDL and optionally BPEL4WS in WS) and schema (XSD in WS) and can be validated. Moreover, the advent of WS-Policy (WS) provides for additional automated analysis of the service assurance compatibility between services. This is done based on explicit assertions of capabilities and requirements in the form of WS-Policy statements.

3. **Service-orientation assumes that bad things can and will happen.** Some previous approaches to distributed applications explicitly assumed a common type space, execution model, and procedure/object reference model. In essence, the "in-memory" programming model defined the distributed system model. Service-orientation simply assumes that the services execute autonomously and there is no notion of local execution or common operating environment. For this reason, an SOA explicitly assumes that communication, availability, and type errors are common. To maintain system integrity, service-oriented designs explicitly rely on a variety of technologies to deal with asynchrony and partial failure modes.

4. **Service-orientation enables flexible binding of services.** One of the core concepts of SOA is flexible binding of services. More traditional procedural, component and object models bind components together through references (pointers) or names. An SOA supports more dynamic discovery of service instances that provides the interface, semantics and service assurances that the requestor expects. In procedural or object-oriented systems, a caller typically finds a server based on the types it exports or a shared name space. In an SOA system, callers can search registries (such as UDDI in WS) for a service. The loose binding with respect to the implementation of the service that enables alternative implementations of behaviour can be used to address a range of business requirements.

The next sections survey current and recent work on SOA, focusing more specifically on the Web Services Architecture, and are followed by a brief overview of future trends in the area.

8.1 Web Services

The Web has led to the introduction of a significant number of technologies towards interoperability, relevant to most areas of computer science, over the last few years. In this context, the Web Service Architecture is expected to play a prominent role in the development of the next generation of distributed systems due to its architectural support for integrating applications over the Web, which makes it particularly attractive for the

development of multi-party business processes. This is further witnessed by the strong support from industry and the huge effort in the area.

The Web Service architecture targets the development of applications based on the XML standard [W3C-XML], hence easing the development of distributed systems by enabling the dynamic integration of applications distributed over the Internet, independently of their underlying platforms. According to the working definition of W3C Consortium [W3C-SG]: *Web service is a software system identified by URI, whose public interfaces and bindings are defined and described by using XML*. This allows its definition to be discovered by other software systems. These systems may then interact with the Web Service in a manner prescribed by its definition by using XML-based messages conveyed by Internet protocols.

Currently, the main constituents of the Web Service architecture are the following: (i) WSDL (Web Services Description Language) that is a language based on XML for describing the interfaces of Web Services [W3C-WSDL]; (ii) SOAP (Simple Object Access Protocol) that defines a lightweight protocol for information exchange [W3C-SOAP]. The Web Service architecture is further conveniently complemented by UDDI (Universal Description, Discovery and Integration) that allows specification of a registry for dynamically locating and advertising Web Services [UDDI].

There already exist various platforms that are compliant with the Web Service architecture, including .NET [MS-NET], J2EE [SUN-J2EE] and AXIS [APACHE-AXIS]. However, there clearly is a number of research challenges regarding support for the development of distributed systems based on Web Services as surveyed in the next sections of this chapter.

8.2 Web Services Composition

Composing Web Services relates to dealing with the assembly of autonomous components so as to deliver a new service out of the components' primitive services, given the corresponding published interfaces. In the current Web Service architecture, interfaces are described in WSDL and published through UDDI. However, supporting composition requires further addressing: (i) the specification of the composition, and (ii) ensuring that the services are composed in a way that guarantees the consistency of both the individual services and the overall composition. This puts forward the need for a high-level specification language of Web Services that is solely based on the components of the Web Service architecture and that is as far as possible declarative. Defining a language based on XML then appears as the base design choice for specifying the composition of Web Services.

A first requirement for the XML-based specification of Web Services is to enforce correct interaction patterns among services. This lies in the specification of the interactions of services with their users that are assumed by the Web Service's implementation for the actual delivery of advertised services. In other words, the specification of any Web Service must define the observable behaviour of a service and the rules for interacting with the service in the form of exchanged messages. Such a facility is supported by a number of XML-based languages: WSCL [W3C-WSCL], BPEL [BPEL] and WSCI [W3C-WSCI]. The W3C Web Services Choreography Working Group [W3C-CHOR] aims at extending the Web Services architecture with a standard specifying Web Services choreographies. The specification of a choreography language is actually in preparation.

Given the specification of choreographies associated with Web Services, the composition (also referred to as orchestration) of Web Services may be specified as a graph (or process schema) over the set of Web Services, where the interactions with any one of them must conform to the associated choreographies. The specification of such a graph may then be: (i) automatically inferred from the specification of individual services as addressed in [Narayanan and McIllraith 2002], (ii) distributed over the specification of the component Web Services as in the XL language [Florescu et al 2002], or (iii) be given separately as undertaken in [BPEL], [BPML], [Casati et al 2001], [Fauvet et al 2001], [Tartanoglu et al 2003] and [Yang and Papazoglou 2002].

The first approach is quite attractive but restricts the composition patterns that may be applied, and cannot thus be used in general. The second approach is the most general, introducing an XML-based programming language. However, this makes the reuse of composed Web Services in various environments more complex since this requires retrieving the specification of all of the component Web Services prior to the deployment of the composed service in a given environment. On the other hand, the second approach quite directly supports the deployment of a composed service from its specification by clearly distinguishing the specification of component Web Services (comprising primitive components that are considered as block-box components and/or inner composite components) from the specification of composition. Execution of the composed service may then be realized by a centralized service provider or through peer-to-peer interactions [Benatallah et al 2002].

8.3 Web Services Composition and Dependability

Transactions have been proven successful in enforcing dependability in closed distributed systems. The base transactional model that is the most used guarantees ACID (atomicity, consistency, isolation, durability) properties over computations. However, such a model is hardly suited for making the composition of Web Services transactional for at least two reasons: (i) the management of transactions that are distributed over Web Services requires cooperation among the transactional support of individual Web Services –if any-, which may not be compliant with each other and may not be willing to do so given their intrinsic autonomy and the fact that they span different administrative domains; (ii) locking accessed resources (i.e., the Web Service itself in the most general case) until the termination of the embedding transaction is not applicable to Web Services, again due to their autonomy, and also to the fact that they potentially have a large number of concurrent clients that will not stand extensive delays.

Enhanced transactional models may be considered to alleviate the latter shortcoming. In particular, the split model (also referred to as open-nested transactions) where transactions may split into a number of concurrent sub-transactions that can commit independently allows reduction of the latency due to locking. Typically, sub-transactions are matched to the transactions already supported by Web Services (e.g., transactional booking offered by a service) and hence transactions over composed services do not alter the access latency as offered by the individual services. Enforcing the atomicity property over a transaction that has been split into a number of sub-transactions then requires using compensation over committed sub-transactions in the case of sub-transaction abortion. Using compensation comes along with the specification of compensating operations supported by Web Services for all the operations they offer. Such an issue is in particular addressed by BPEL [BPEL] and WSCI [W3C-WSCI].

However, it is worth noting that using compensation for aborting distributed transactions must extend to all the participating Web Services (i.e., cascading compensation by analogy with cascading abort). An approach that accounts for the specification of the transactional behaviour of Web Services from the standpoint of the client in addition to the one of the service is proposed in [Mikalsen et al 2002]. This reference introduces a middleware whose API may be exploited by Web Services' clients for specifying and executing a (open-nested) transaction over a set of Web Services whose termination is dictated by the outcomes of the transactional operations invoked on the individual services. Finally, more general solutions are undertaken in [WS-C] and [WS-CF], which allow the abstract specification of coordination protocols among Web Services, including the specification of the coordination in the presence of failures. Dependability then relies on the exploitation of specific coordination types, such the ones defined in [WS-T] and [WS-TXM]. An approach that is specifically based on forward error recovery is presented in [Tartanaglu et al 2003]: it allows the structuring of composite Web Services in terms of coordinated atomic actions. A Web Service Composition Action (WSCA) is defined in this work by a set of cooperating participants that access several Web Services. Participants specify interactions with composed Web Services, stating the role of each Web Service in the composition. Each participant further specifies the actions to be undertaken when the Web Services with which it interacts signal an exception, which may be either handled locally to the participant or be propagated to the level of the embedding action. The latter then leads to coordinated exception handling according to the exceptional specification of the WSCA.

The work discussed above concentrates on the specification of the dependable behaviour of Web Services. Complementary research is undertaken in the area of transaction protocols supporting the deployment of transactions over the Web, while not imposing long-lived locks over Web resources. Existing solutions include THP (Transaction Hold Protocol) from W3C [W3C-THP] and BTP from OASIS [OASIS-BTP]. The former introduces the notion of tentative locks over Web resources, which may be shared among a set of clients. A tentative lock is then invalidated if the associated Web resource gets acquired. The BTP protocol introduces the notion of cohesion, which allows defining non-ACID transactions by not requiring successful termination of all the transaction's actions for committing.

8.4 Web Service Validation and Verification

The expected future omnipresence of Web Services requires sophisticated methodologies for the engineering of more and more Web Service protocols and their implementations. All open applications will provide Web Services as internet-friendly interfaces to enable their use as components in distributed systems. They require application specific protocols. Additionally many common issues will have to be addressed with general-purpose infrastructure protocols, which need to be of high quality, due to their wide adoption. Their dependability requires their correct implementation.

Unfortunately today's Web Service based protocols and their implementations are frequently of low quality. Protocols are underspecified and have errors. Many Web Services do not implement their protocols correctly, due to bugs, misunderstandings of protocols, or ease of implementation. This raises interoperability issues, which might have an effect only in complex scenarios. The reasons for this situation are insufficient methodologies for protocol engineering and adoption.

The required methodologies fall into different categories. Appropriate specification techniques are required for the protocols. Protocols have to be checked for consistency before they are implemented. Parts of the implementation can be generated from specifications. Verification can be applied to prove the Web Service's correctness accompanying the implementation phase. Existing implementations can then automatically be validated for conformance to the specification. An interesting example of the work in the area is RTD by the Telematics Department of the Technische Universität Hamburg-Harburg that addresses the required methodologies in the three following projects.

The "Formal Description and Model Checking of Web Services Protocols" project addresses the correctness of Web Service protocols. It uses a very advanced specification and reasoning approach that is well supported by tools called Temporal Logic of Actions (TLA) [Lamport 2003a][Lamport 2003b]. The use of TLA+ as a specification language allows a precise statement of the protocol in clear mathematical terms. Furthermore, the TLC tool allows model checking of the specification against various consistency properties to detect semantic flaws of the protocol design in the earliest phase possible. As a starting point the Atomic Transactions protocol of the WS-Transaction specification [WS-T] were chosen for a case study. Further protocols will be checked for their correctness in the realm of the Web Service protocols in the future.

The "Automatic Validation of Web Services" project focuses on validating Web Services after implementation. Automatic validation means the process of checking if occurring message flows conform to their specifications [Venzke 2003]. This is performed by a general-purpose validator, which observes messages exchanged by communication partners and analyses the conformance to the specification. Immediately detecting non-conformance allows corrective actions before putting risk on the systems dependability.

The "Distributed Infrastructure for Reputation Services" project applies the two developed methodologies to protocol engineering and is concerned with Reputation Services that are frequently layered on top of Peer-to-Peer networks. Reputation Services build trust between trading partners on global e-marketplaces by collecting and aggregating the ratings that participants give on past transactions. They require the project to develop Web Service based protocols, which need to be specified, checked for consistency and their implementation to be validated.

8.5 Web Services for Sensor Node Access

Gathering information about environmental conditions and states can be seen as a highly distributed system [Lindsey et al 2001]. Sensor nodes of different kind operate in the field and collect data such as temperature, wind strength, and pH-values, etc. They can count animals as well as the amount of traffic or the fluctuation of passengers in a city. Currently, these sensors are more or less small devices that act autonomously, fulfilling their tasks for a special research or commercial purpose. They are integrated into a special application that knows how to handle and address these sensor nodes.

With the upcoming IP v6 and the further improvements in the creation of small devices with low power consumption and highly integrated circuits, it will be possible to massively equip our environment with sensor networks and intelligent data acquisition agents (leading to pervasive computing, see Chapter 5). Current research in wireless sensor networks (preformed for example within project EYES, CAMS and Aware Goods) covers issues such as hardware and software architecture [Pottie and Kaiser 2000][Elson and Estrin 2001a], energy saving [Sinha et al 2000][Shih et al 2001], protocols [Heinzelman 2000][Sohrabi et al 2000], addressing [Heidemann et al 2001], routing [Shih et al 2001][Sohrabi et al 2000], topology discovery, location mechanisms [Bulusu et al 2001], and time synchronisation [Elson and Estrin 2001b], as well as the description of sensor capabilities, data formats, data types, and data exchange. Future research activities should also consider data security and accounting, and standardised interfaces to access sensors nodes within our environment.

Integrating Web Services technology into sensor node software and sensor node design opens new perspectives in this area [Hillenbrand et al 2003a][Hillenbrand et al 2003b]. Describing the sensor nodes and sensor data using XML dialects and offering service methods using WSDL makes it very easy for application developers to access sensor node data and integrate it into applications. This creates an open system in which heterogeneous clients (concerning programming language, operating system and purpose) can easily access sensor data in order

to operate on it. A standardised sensor node description such as SensorML [ESSL] even allows interoperability between sensors from different providers.

As sensors can be mobile, they always provide some kind of location information. Research in recent years has developed several techniques to obtain location information from within an application. It now has to be arranged that all these different solutions can be used seamlessly. Additionally, research in lookup mechanisms to find sensor nodes according to their location and service semantics will provide a possibility of identifying and finding suitable sensor nodes.

Urban planning is one of the research fields in which a standardised architecture for sensor nodes would provide large improvements [Hillenbrand et al 2003a][Hillenbrand et al 2003b]. It is necessary to find a balance between the three areas of society, economy and ecology (addressed by the keyword "sustainable development") and to detect interdependencies between them to understand how they influence each other. But it is not possible to build a city and check its state and progress over the time. Thus, simulation is currently the only reasonable way to gain information about interdependencies and the three-dimensional effects of urban planning (perhaps even by using augmenting reality techniques). And research will be focused on identifying and creating distributed and semantically well-defined service components (e.g. for data gathering, simulation and visualisation) for urban planning that can be automatically found on the Internet to be integrated in urban planning processes. Issues such as single sign on, security and accounting will be seamlessly integrated in these service-oriented computing infrastructures.

A similar development called Smart Personal Object technology Initiative[23] – SPOT is now undertaken by MS Research. This is aimed at improving the function of everyday objects through the incorporation of software. The work focuses on making everyday objects, such as clocks, pens, key-chains and billfolds more personalized and more useful through the use of special software.

8.6 Web Services for Grid

In recent years, there has been a lot of interest in *Grid computing* from both the research and the business sectors. This increased interest has led to the formation of the Global Grid Forum (GGF[24]), a forum for the discussion of Grid-related ideas and the promotion of enabling technologies. One of its main activities is the creation of a standards-based platform for Grid Computing with emphasis on interoperability.

Despite the interest in Grid computing, the term "Grid" does not have a widely accepted definition and it means different things to different users groups and application domains. The following list contains just a few of the views and is by no means exhaustive:

- **Virtual organizations**. The Grid is seen as the collection of enabling technologies for building virtual organizations over the Internet.
- **Integration of resources.** The Grid is about building large-scale, distributed applications from distributed resources using a standard implementation-independent infrastructure.
- **Universal computer**. According to some (e.g., IBM-GRID[25]), the Grid is in effect a universal computer with memory, data storage, processing units, etc. that are distributed and are used transparently from applications.
- **Supercomputer interconnection**. The Grid is the result of interconnecting supercomputer centers together and enabling large-scale, long-running scientific computations with a very high demand regarding all kinds of computational, communication, and storage resources.
- **Distribution of computations**. Finally, there are those who see cycle-stealing applications, such as SETI@HOME, as typical Grid applications without any requirements for additional, underlying technologies.

---

[23] http://www.microsoft.com/presspass/features/2002/nov02/11-17SPOT.asp

[24] http://www.ggf.org

[25] http://www.ibm.com/grid

No matter how one sees the Grid, it is apparent that distributed computing plays a significant role in its realization. Moreover it is clear that the Grid addresses a great number of distributed computing related issues (e.g., security, transactions, resource integration, high-performance interconnects, location transparency, etc.) applied at a very large scale.

Currently, the Grid community is working towards the definition of open standards for building interoperable Grid applications, while, at the same time, a great number of research projects investigate approaches to Grid computing. In the following part of the chapter, the Open Grid Services Architecture is briefly described and then three research projects are introduced as examples of the diverse approaches in Grid computing.

8.6.1 Open Grid Services Architecture

GGF is defining an architecture for Grid computing (as it is perceived by the relevant working group) based on the concept of a *Grid Service*. This blueprint is defined by the Open Grid Services Architecture (OGSA) working group as a set of fundamental services whose interfaces, semantics, and interactions are standardised by other GGF working groups. OGSA plays the coordinating role for the efforts of these groups. It identifies the requirements for e-business and e-science applications in a Grid environment and specifies the core set of services and their functionality that will be necessary for such applications to be built, while the technical details are left to the groups.

While OGSA has adopted a services-oriented approach to defining the Grid architecture, it says nothing about the technologies used to implement the required services and their specific characteristics. That is the task of the Open Grid Services Infrastructure (OGSI), which is built on top of Web services standard technologies. Figure 8.1 shows the relation between OGSA, OGSI, and the Web services standards. Also, a list, which is not exhaustive, of candidate core services is presented. The OGSA working group is currently working towards standardising the list of services.

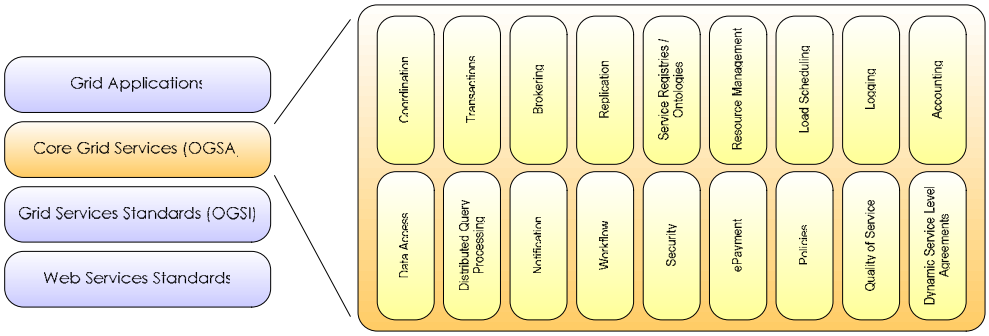


Figure 8.1: A potential set of services for the OGSA platform

8.6.2 Current and Recent Work

OGSA-DAI

Open Grid Services Architecture Data Access and Integration (OGSA-DAI) is a project that aims to provide a component library for accessing and manipulating data in a Grid for use by the UK and international Grid community. It also provides a reference implementation of the emerging GGF recommendation for Database Access and Integration Services (DAIS). The project develops a middleware glue to interface existing databases, other data resources and tools to each other in a common way based on OGSA. As part of this glue a simple integration of distributed queries to multiple databases. OGSA-DAI will also interact with other Grid standards and software to support replication services and complex workflows.

OGSA-DAI provides a number of new OGSI services that are added to the base services facilitated by OGSI. They include:

- *Grid Data Service* (GDS) – which provides standard mechanism for accessing data resources;

- *Grid Data Service Factory* (GDSF) – which creates a GDS on request; and
- *Database Access and Integration Service Group Registry* (DAISGR) – which allows clients to search for GDSFs and GDSs that meet their requirements.

The OGSA-DAI services are designed to be highly configurable. They take as input complex XML documents. In the case of the GDS such a document can specify a number of parameterised database operations such that the parameters can be instantiated within a particular document or in subsequent documents submitted to the same GDS. This functionality could support complex client interactions or allow a GDS to be configured to allow only a carefully restricted set of operations on its database.

The users of the OGSA-DAI software are, at least in the first instance, applications developers who will use the components to build data access applications. The intention is to allow a new, standard way to access data resources and tools. OGSA-DAI builds on top of the Globus Toolkit 3, which is a reference implementation of the OGSI specification and OGSA-DAI makes use of the implementations (and proposals) for functionality such as notification, and security.

An explicit aim of the project from the outset was to re-use as much existing technology as possible by introducing a new, standard way to access data resources and tools. OGSA-DAI builds on top of the Globus Toolkit 3, which is a reference implementation of the OGSI specification and OGSA-DAI makes use of the implementations (and proposals) for functionality such as notification, and security.

myGrid

The myGrid project develops an open source high-level service-based middleware to support *in silico* experiments in biology. *In silico* experiments are procedures using computer-based information repositories and computational analysis adopted for testing hypothesis or to demonstrate known facts. In myGrid the emphasis is on data intensive experiments that combine use of applications and database queries. The user is helped to create workflows (a.k.a. experiments), sharing and discovering others' workflows and interacting with the workflows as they run. Rather than thinking in terms of data grids or computational grids, myGrid is built around the concept of *Service Grids*, where the primary services support routine *in silico* experiments. The project goal is to provide middleware services as a toolkit to be adopted and used in a "pick and mix" way by bioinformaticians, tool builders and service providers who in turn produce the end applications for biologists. The target environment is open which means that services and their users are decoupled and that the users are unknown to the service provider.

The myGrid middleware framework employs a service-based architecture prototyped with Web Services. The middleware services are intended to be collectively or selectively adopted by bioinformaticians, tool builders and service providers. The primary services to support routine *in silico* experiments fall into four categories:

- services that are the **tools** that will constitute the experiments, that is: specialised services such as AMBIT text extraction [Gaizauskas et al 2003] and external third party services such databases, computational analysis, simulations, etc., wrapped as Web Services by Soaplab [Senger et al 2003] if required
- services for **forming and executing experiments**, that is: workflow management services [Addis et al 2003], information management services, and distributed database query processing [Alpderim et al 2003]
- **semantic services** for discovering services and workflows, and managing metadata, such as: third party service registries and federated personalized views over those registries [Lord et al 2003], ontologies and ontology management [Wroe et al 2003]
- services for supporting the **e-Science scientific method** and best practice found at the bench but often neglected at the workstation, specifically: provenance management [Stevens et al 2003] and change notification [Moreau et al 2003].

ARION

The ARION system [Houstis et al 2002][Houstis et al 2003] provides a service-based infrastructure designed to support search and retrieval of scientific objects, and capable of integrating collections of scientific applications including datasets, simulation models and associated tools for statistical analysis and dataset visualization. These

collections may reside in geographically disperse organizations and constitute the system content. On-demand scientific data processing workflows are also actively supported, in both interactive and batch mode.

The underlying computational grid used in the ARION system [Houstis et al 2002] is composed of geographically distributed and heterogeneous resources, namely, servers, networks, data stores and workstations, all resident in the member organizations that provide the scientific content and resources. ARION provides the means for organizing this ensemble so that its disparate and varied parts are integrated into a coherent whole. Hence, ARION can be viewed as the middleware between users, the data they wish to process and the computational resources required for this processing. In addition, the system offers semantic description of its content in terms of scientific ontologies and metadata information. Thus, ARION provides the basic infrastructure for accessing and deriving scientific information in an open, distributed and federated system.

To achieve these goals the ARION project has brought together two closely related technology trends that are undergoing continuous development and have reached an acceptable level of maturity, namely the Semantic Web and the Grid. The gain from this promising coupling is the combination of large-scale integration of resources with a universally accessible platform that allows data to be shared and processed by automated tools as well as by people. The system demonstration scenarios involve environmental applications (offshore to near shore transformation of wave conditions, synthetic time series and monthly statistical parameters, coupled ocean-atmosphere models etc.).

8.7 Future Trends

The development of dependable distributed systems based on the Web Services Architecture is an active area of research that is still in its infancy. A number of research challenges are thus yet to be addressed to actually enable the dependable composition of Web Services. Issues include the thorough specification of individual Web Services and of their composition, so as to ensure the dependability of the resulting systems as well as to allow the dynamic integration and deployment of composed services. Also, associated dependability mechanisms should be devised to enable full exploitation of the dependability properties enforced by individual Web Services and also to deal with the specifics of the Web. Another issue relates to allowing the deployment of Web Services on various platforms, ranging from resource-constrained devices to servers.

An extremely important area of RTD in SOA is that of bringing semantics to the Web. To this end the W3C has created a working group on Semantic Web[26] to work on the representation of data on the World Wide Web and on giving these data well-defined meaning.

Another important area of future research is developing rigorous methodologies supporting Web composition. TLA+ among other techniques will achieve an important role in supporting unambiguous communication between protocol engineers and implementers. This will make it possible for Web Services to be automatically validated to detect non-conformance to the specification, while testing and in productive distributed systems. This will also lead to developing a proven set of general purpose, high quality protocols that can be combined modularly to address common issues.

[26] http://www.w3.org/2001/sw/

9 Mobile Agents

The idea of mobile code in general has gained unprecedented attention in recent years in the form of *mobile agents*. The mobile agent notion is generally viewed to have risen from two fields, on the one hand artificial intelligence research with its notion of an agent as a piece of software acting autonomously and "on behalf of its owner". On the other hand, the pervasiveness of networking greatly increased interest in the idea of mobile code. Combining these two influences results in the concept of a mobile agent as a piece of software that autonomously acts, and when necessary moves, within a computer network, performing some function on behalf of its owner. Software as diverse as adaptive user interfaces, network management components, and Web search engines have all been described as agents. In this chapter, we define a *mobile agent* as a process with the ability to move itself during its own execution in all essential parts from one host machine to another within a heterogeneous network while preserving its internal state.

A mobile agent consists of *code* and *state*, contains its own *thread of control*, and is able to voluntarily move its code and state between host machines, an act usually termed *migration*. A migration includes all "essential" parts of an agent so that it can continue to function autonomously after migration has taken place. An important distinction results from what exactly is to be included in the migrated state of the process: while all practical mobile agents are able to take their global data with them on migration, only some also preserve the execution context of their thread of control. Five properties can be recognized as characteristic, distinguishing mobile agents from various related computing concepts:

- **Generality**. While it appears at first sight that mobility is the interesting new property added to the plain and old concept of executable code, the question of what exactly is meant by "code" deserves attention as well. Code could be considered as a continuum, ranging from a simple set of commands which cause a server to execute a limited set of operations, through "programs" written in a specialist language for a particular application (such as SQL queries to a database server) to programs written in a full general purpose programming language. For the purposes of this chapter, we shall consider mobile code to be in this most general form: code in a Turing-complete programming language with a programming interface to the host system which is sufficiently rich to enable a wide range of applications. In other words, the code can be used for general programs. Note that this characterization leaves the issue of the code's level of abstraction open; in theory, it can be anything from native machine code for a physical microprocessor to expressions in an abstract declarative programming language. In practice, however, there are certain languages that are more suitable than others for this purpose.

- **Asynchrony**. While it is conceivable to execute a piece of mobile code as part of the flow of control of the hosting run-time system, in the case of mobile agents most systems have opted for the most general model of executing each agent in its *own thread of control*. This corresponds to the idea of activity and autonomy conveyed in the agent term, mirrored in practice by the increased programming convenience of threads, for instance when dealing with concurrency or external events.

- **Voluntary mobility and location awareness.** As defined above, a mobile agent is a "process with the ability to move itself... from one host machine to another", that is, the act of migration is decided and initiated by the agent itself, not by the underlying system. This enables the agent to implement its own strategy for choosing its itinerary. An inevitable and important consequence of voluntary mobility is *location awareness*: A mobile agent is aware of where it is and where it chooses to go, possibly dynamically based on its experiences so far (as in distributed searching). This is a crucial difference compared with designs where migration is imposed on the process, such as traditional load balancing systems, where the process is moved in reaction to environment conditions. The same feature also distinguishes mobile agents from general mobile object systems, where objects typically do not trigger their own migration, but are moved at the request of the processes using these objects

- **Heterogeneity**. The advantage of mobile code is particularly apparent in networks of limited bandwidth (such as wide area or mobile networks) or having a dynamic and diverse service and device structure, invariably consisting of heterogeneous nodes. Accordingly, nearly all mobile agent systems assume a heterogeneous network and provide mechanisms to migrate executing agents across heterogeneous machines, and hence, some compatibility layer is needed to to with this inherent heterogeneity.

- **Crossing protection domains.** Wide area networks typically comprise nodes or domains of nodes belonging to more than one organization or individual, each with their own security sensitive resources and policies, comprehensively termed a *protection domain*. A mobile agent effectively transfers code between protection domains, which constitutes, along with the generality property discussed earlier, a security threat

even more fundamental than the numerous and well known threats inherent in traditional networked computing. A conceptual consequence of this property is that a mobile agent must bear some form of identifiable authority to allow visited protection domains to attribute a level of trust to this agent, as is required for deciding on what actions by the agent to authorize in this protection domain.

Mobile code introduces numerous new requirements on a run-time system, the most obvious of these being portability and security of code execution. Dedicated and potentially complex run-time systems are required for mobile agents, usually termed *mobile agent systems* (MAS). Such a mobile agent system must be present on any machine hosting agents.

Portability of mobile agent code is essential in practice, as it is not realistic in a large-scale network such as the Internet to maintain different versions of each agent for each host platform it might potentially encounter in the network. Accordingly, the agent must be shipped in a common representation which is then locally executed by the mobile agent system, using a kind of interpretation, compilation, or a combination of both.

The second obvious requirement when accepting mobile code into a host system is security. Execution of the visiting code must be closely monitored by the MAS to prevent intentional or unintentional damage, spying, overuse of resources, or any other of the well-known security threats of distributed computing. On the other hand, it becomes clear that the agent itself might require protection from the foreign host as well, against spying or manipulation of its data and code.

In addition to portability and security, various other functions are required from a mobile agent system to make it practical, such as inter-agent communication and agent control and management, fault tolerance, group communication, directory services, name service, key management, etc.

9.1 Current and Recent Work

This section discusses the Mole system from the University of Stuttgart and the Ara system from the University of Kaiserslautern, both of which have been developed by the CaberNet partners. These are well-known systems, representative of developments in the area.

The Mole mobile agent system

Mole [Baumann et al 1999] supports agents written in Java, provides weak migration, and offers elaborate communication and distributed programming support, including message passing, RPC-style invocations, a session concept for both message passing and remote method invocation, events, group communication, and agent termination and orphan detection. Furthermore, concepts for reliability and agent security have been researched using Mole as a testbed, though they have not been incorporated into the core system. Many demonstration applications have been implemented, and Mole has been used by several industrial partners (e.g. DaimlerChrysler, Siemens, Tandem).

Mole agents do not hold references to any external objects, they are “object islands”; such objects and services are accessed by Java RMI instead, even if they are local. This facilitates the implementation of agent state capturing (using standard Java object serialization on the agent object), but incurs the RMI overhead for any system call, as system functionality is available through special “system” agents only. Being uniformly based on RMI, all Mole communication mechanisms are available equally for local and remote communication.

The places where agents stay and move between are not designated by physical DNS host names as in most MASs; instead, Mole has abstract *locations* for this purpose, and these are resolved by a suitable name service to physical IP addresses and port numbers. The intention behind this level of indirection is to hide the physical structure of the network.

Mole agents are either addressed through their name (a globally unique id every agent receives at creation time), or by way of a *badge*, a symbolic and not necessarily unique name that an agent may assume and drop at will, including several different badges at the same time. Badges are intended to be used like roles that agents may play in performing a certain task; this makes sense whenever the identity of a communication partner is not important, but only the function it performs, indicated by the badge it is “wearing”, such as “a result collecting agent for the XYZ program”. Agents may assume, drop, copy, or pass on a badge. Each Mole location offers a function to find all agents whose badges match a “badge predicate”, a boolean expression of badges. The badge concept allows agents to register their offered services (assume a badge), to deregister (drop the badge), and to be located (return all agents which fulfill a certain badge predicate).

A Mole location may provide a *directory* of local agents and their services; services are indexed by flat symbolic names, although it is not clear why such a directory is needed as well as the badge mechanism.

Messages in Mole are asynchronous regarding both sending and receiving: on message arrival, the system will start a dedicated thread to execute the `receiveMessage` method of the target agent; if no such agent exists at this location, the message is stored for some time and then returned to the sender. For both message passing and remote method invocation (RMI), Mole offers to embed them within a communication *session*. Sessions must be explicitly initiated by both participating agents, by naming the desired peer and its location. *Group communication* in Mole is built on events and the interesting concept of making the group object a central and intelligent event mapper: all group members send and receive events from the group object, and the group object has some internal state and rules allowing an arbitrarily flexible mapping of input to output events.

Mole supports weak migration: on issuing a migration, with the destination location as an argument, the MAS will call the `stop` method of the agent, which the agent may override to perform application-dependent migration preparations. The agent is then subjected to standard Java object serialization (which is straightforward because of the island concept), transferred to the destination location, and restarted.

For security reasons, application agents cannot directly access any platform services, but must do so by contacting system agents offering the desired functions. This provides high flexibility considering the fact that the system agents can be arbitrarily complex in their policies, but may raise performance concerns. This approach provides the additional option of a system agent changing its policy regarding a specific agent during the time of that agent's visit.

Locations can restrict the kind of agents they accept to a specific Java type. The intention of this feature is not quite clear; if the allowed type is a Java class with certain predefined and final (that is, non-overridable) methods, the location could indeed make some assumptions about the agent's behaviour; on the other hand, not all methods can really be predefined (as such an agent would not be very useful in practice), and once control enters an application-defined method, no assumptions about the agent's behaviour can be made.

For each agent, the Mole system may control its consumption of several quantitative resources, including CPU and wall clock time, local and remote communication, and the number of child agents created. The system agents responsible for resources such as remote communication may perform the enforcement and accounting whenever an application agent desires a resource access.

A novel approach to mobile agent code protection based on code obfuscation was developed in the context of the Mole system.

Mole pioneered the use of Java for mobile agent systems, and has its strong point in the flexible, powerful, and convenient support for remote communication and distributed programming. On the implementation side, however, the decisions to base all local communication and any system call on the Java RMI mechanism, and to create a dedicated thread for each message delivery, raise some concerns regarding the performance of Mole. Despite some concerns regarding security (e.g. in Mole, authentication, for instance, seems to be completely lacking), the Mole research explored many important issues in mobile agent system design.

The Ara mobile agent system

The Ara ("Agents for Remote Action") system [Peine 2002] supports a number of different languages for agent programming. The Ara programming model consists of agents moving between and staying at places, where they use certain services, provided by the host or other agents, to perform their task. A place is physically located on some host machine, and may impose specific security measures on the agents staying at that place. Agents may migrate to another place at will, resuming execution from exactly the same point in their program execution. With this in mind, agents may be programmed much like conventional programs in all other respects, i.e. they work with a file system, user interface and network interface. The Ara architecture deliberately abstains both from high-level agent-specific concepts, such as support for intelligent interaction patterns, as these are left to the application, and from complex distributed services such as found in distributed operating systems, as these conflict with the mobility and autonomy of mobile agents.

Most MASs use the same basic solution for portability and security: agents are run on some virtual machine, usually an interpreter and a run-time system, which both hides the details of the host system architecture as well as confining the actions of the agents to that restricted environment. Ara does this as well. Concretely, Ara agents are programmed in some interpreted language and executed within an interpreter for this language, supported by a special run-time system for agents, called the *core* in Ara terms. The relation between core and

interpreter is characteristic here: isolate the language-specific issues (e.g., how to capture the C-specific state of an agent programmed in the C programming language) in the interpreter, while concentrating all language-independent functionality (e.g., how to capture the general state of an Ara agent and use that for moving the agent) in the core. To support compatibility with existing programming models and software, Ara does not prescribe an agent programming language, but instead provides an interface to attach existing languages. This makes it possible to employ several interpreters for different programming languages, running on top of the common, generic core, which makes its functions uniformly available to agents in all languages. Apart from migration support, the core's main functions include agent management, communication, persistency, and security mechanisms.

The functionality of the system core is kept to the absolute minimum, with higher-level services provided by dedicated server agents. The complete ensemble of agents, interpreters and core runs as a single application process on top of an unmodified host operating system. To demonstrate the viability and generality of this design, interpreters for three rather diverse languages have been adapted to the Ara core, covering a wide spectrum of applications: The *Tcl* scripting language, *C/C++* and *Java*.

Agents are executed as a separate *thread of control* while in one place, and are transparently transformed into a portable representation whenever they choose to move. The system employs additional threads for certain internal tasks ("system threads") in order to modularize the architecture. Employing threads as opposed to host operating system processes keeps the agent management completely under control of the core, achieves superior performance, and facilitates portability.

Adapting a given interpreter for some programming language to the Ara core requires the definition of calling interfaces (*stubs*) in this language for the functions of the core API, and conversely the implementation of certain functions for interpreter management (*upcalls*) for use by the core. The job of the stubs is mostly a matter of data format conversions and similar interface mappings. Regarding the interpreter upcalls, the most prominent functions are those for the extraction of an executing interpreter's state as it is needed for the migration of the agent being interpreted, and conversely for the restoration of such a state on arrival of a migrated agent. Finally, the interpreter has to assist the core in the preemptive execution of the agent programs by performing regular calls to a core function for time slice surveillance. This design ensures that execution contexts will always be machine independent.

Considering that one of the main motivations for mobile agents was to avoid remote communication, Ara emphasizes local agent interaction. Two simple and efficient mechanisms are available for this, synchronous message passing and a tuple space. For message passing, the core provides the concept of a *service point*: a symbolically named rendezvous point where agents can interact as clients and servers through an *n:1* synchronous exchange of request and reply messages of arbitrary format. The other communication mechanism between Ara agents is the tuple space, a persistent storage space for key/value pairs which can be used as a means of asynchronous communication, which has proven to be particularly appropriate to the asynchronous execution typical of mobile agent applications.

Ara security builds on authentication of agent users, manufacturers, and hosts by digital signatures. On top of this, a flexible scheme of places with local authorization policies allows a fine-grained, but conceptually simple authorization of mobile agents visiting a host. Agent transfers may be performed over encrypted and authenticated connections. The central authorization concept in Ara is the *allowance* of an agent. An allowance is a vector of access rights to various system resources, such as files, CPU time, RAM, or disk space. The elements of such a vector constitute resource access limits, which may be quantitative (e.g. for CPU time) or qualitative (e.g. for the network domains to where connection is allowed). The system core will ensure that an agent never oversteps its allowance. An agent migrating to a remote place specifies the allowance it desires for its task there, and the place in turn decides what allowance should actually be given to the applicant. This decision is performed in the form of an arbitrary application-defined function per place to compute the local allowance, given an applicant's credentials and desires.

The security design thus guards against the dangers of eavesdropping, impersonation, local denial of service, and general undue accesses by agents, and ensures the security of the host in a flexible, convenient, and efficient way.

9.2 Future Trends

Mobile agents as a distinct field of research is now more than ten years old. This is a long time in the computing world, yet despite this, the field is strikingly immature. There is no consensus on fundamental issues, such as the right scope and requirements of a mobile agent system, or the relation of mobile agents to networked computing frameworks, and even the general value of the concept is still debated. One of the reasons for this is that there

are still hardly any convincing, let alone commercial, applications of mobile agents. This is mainly because many individual problems solved by mobile agents could also be solved by different means, such as more powerful protocols, sophisticated caching, reconfigurable servers, and richer parameterisation of services. It is true that from a software engineering point of view, one uniform concept, such as mobile agents, is preferable over a number of individual solutions. However, no single application requires *all* of the above features. Thus, even if the remaining technical problems of mobile agents were to be solved at some time in the future, it might turn out by then that a number of satisfactory custom solutions have already firmly established themselves and effectively closed the window of opportunity for mobile agents.

After a flurry of research activity both in industry and academia during the mid and late 1990s, a sort of consolidation process is happening now. A number of notable commercial mobile agent systems have been abandoned (such as Telescript, Odyssey, or Aglets[27]), and the more recent mobile agent research projects seem to concentrate on specific aspects and niches rather than attempting to build complete mobile agent systems. This does not necessarily mean that all ideas developed in the context of mobile agents will be eventually discarded; rather, this process might result in a better understanding of the useful ideas and experiences and incorporating them into new designs with somewhat different goals. In the remaining part of the chapter we will show why we believe that mobile code has a bright future as a system mechanism in coming middleware frameworks, whereas the notion of a mobile agent as an autonomous entity, as well as the mobile agent term, has an uncertain future.

Mobile agents vs. mobile code

In the early years of the field, the conceptual difference between mobile agents and mobile code was often not clearly recognized and appreciated, and the properties of one were often used to reason about the other. Most of the advantages quoted for mobile agents, such as efficiency by local interaction and flexibility by dynamic reconfiguration, are actually properties of mobile code. The genuine advantage of mobile agents over mere mobile code is that not only is the functionality mobile, but so also is the *control* over the placement of this functionality. Mobile agents thus offer an additional form of flexibility over mobile code. However, this mobile control is a fairly abstract kind of flexibility, and it seems that at least today its added value can only be judged in the context of specific applications. Of course it is always good to have another design dimension at hand, even if it is not really known yet what structures can be built using it; however, as this additional flexibility incurs a cost (such as more complicated security and control) which affects the other design dimensions as well, the specific added value of this flexibility needs to be worth this cost. In the case of the added value of mobile control of mobile functionality (that is, of mobile agents) over mere mobile code, we feel that this has not been satisfactorily demonstrated thus far.

Apart from such conceptual reasoning, there is also practical evidence favouring simple mobile code over mobile agents. The first idea that comes to mind when searching several sites using mobile agents might be one agent visiting each site in turn, an assumption apparently underlying many popular mobile agent application scenarios (buy tickets to a show, then reserve a table at a restaurant, then have flowers delivered and the like). However, an agent picking up data on its way will, on each hop, carry along the data found on each previous node, causing a total network load in the order of $O(n^2)$ when visiting n servers and picking up about the same amount of data on each. When taking into account that security issues (e.g. integrity of dynamically generated data) are considerably complicated by introducing multi-hop migrations, these observations suggest that single-hop designs might be generally favourable where applicable, not necessarily only for parallelizable tasks. Single-hop agents, however, are already rather close to pure mobile code. It fits the picture that the most well known applications of mobile code so far are single-hop: the Java applet and the Jini service proxy object.

Mobile code, however, can deliver many of the described benefits even without being embedded in a mobile agent, in particular the fundamental benefits of handling unanticipated interaction and weak connectivity. For this reason, we see a bright future for mobile code as a system mechanism in future middleware frameworks. Evidence for this can be found in the role of mobile code in current middleware such as EJBs, and, even more, in the .NET middleware and similar concepts expected to dominate the networked computing for the foreseeable future. Mobile agents, by contrast, we see as a technology in waiting, at best: if and when applications are conceived which actually put the flexibility of mobile control to good use, then this technology, which largely already exists, will be actually deployed and further improved for practical usability.

Future run-time support for mobile code

---

[27] Aglets is still continuing as a volunteer-supported open source project, but has been abandoned as a potential commercial product.

Even simple mobile code, however, needs a sophisticated run-time system for portable and secure execution. On the whole, mobile agent systems (or any run-time system for mobile code) provide functions for agents which are similar to those provided by operating systems for application processes, augmented by increased portability and security requirements. It seems appropriate to ask if this partial duplication of functionality should be removed by integrating mobile code functionality into the operating system and abandoning the notion of a separate mobile agent or code system. From an abstract systems design point of view, this seems only consequential when regarding mobile code applications as networked applications which are not really fundamentally different from other applications on a modern, networked machine. However, the still considerable divergence of opinions about mobile code makes it hard to standardize on a common set of mobile code functionality for operating systems at this time; nevertheless, this seems a realistic goal for the future.

Bottom line

Mobile agents are not the panacea to all problems of networked computing. Some of the hardest problems of distributed computing basically remain, only somewhat mitigated, such as partial failures, and some are even intensified, such as distributed control or security. However, mobile code does provide a useful means to attack certain problems of networked computing, such as network latency and bandwidth consumption, access to local interfaces, and, probably most influential in the long run, dynamic remote reconfiguration. The technology and experience acquired by building mobile agent systems such as Ara will contribute to this.

Even though we believe the mobile agent notion will slowly disappear over the coming years, the future of mobile code as a building block of networked computing frameworks looks bright today. Computer networking will become more and more pervasive, and network interaction will become more and more dynamic and unanticipated, while bandwidth will remain scarce for the foreseeable future, considering all the potential applications of, for example, mobile multimedia and virtual or augmented reality. The unique ability of mobile code to handle unanticipated interaction and weak connectivity will make it a certain part of the technology that will make all this happen.

Part IV. Advanced Features

10 Group Communication

The group communication (or groupware) paradigm allows the provision of reliable and high-available applications through replication. *Groups* are the key abstraction of group communication. A group consists of a collection of *members* (i.e., processes or objects) that share a common goal and actively cooperate in order to reach it.

During the last decade, several experimental and commercial group communication systems have appeared. Although the services provided by them present several differences, the key mechanisms underlying their architectures are the same: a *group membership service* integrated with a *reliable multicast service*. The task of the group membership service is to keep members consistently informed about changes in the current membership of a group through the *installation* of *views*. The membership of a group may dynamically vary due to voluntary requests to join or leave a group, or to accidental events such as member's crashes. An installed view consists of a collection of members and represents a perception of the group's membership that is shared by its members. In other words, there has to be agreement among members on the composition of a view before it can be installed. The task of a reliable multicast service is to enable members of a group to communicate by multicasting messages. Message deliveries are integrated with view installations as follows: two members that install the same pair of views in the same order deliver the same set of messages between the installations of these views. This delivery semantics, called *view synchrony*, enables members to reason about the state of other members using only local information such as the current view composition and the set of delivered messages.

Primary Partition Group Communication

Two classes of group communication services have emerged: *primary-partition* and *partitionable*. A primary-partition group communication service attempts to maintain a single agreed view of the current membership of a group. Members excluded from this primary view are not allowed to participate in the distributed computation. Primary-partition group communication services are suitable for non-partitionable systems, or for applications that need to maintain a unique state across the system. The most notable examples of primary-partition group communication services are Isis[28] and Phoenix[29].

Partitionable Group Communication

In contrast, a partitionable group communication service allows multiple agreed views to co-exist in the system, each of them representing one of the partitions into which the network is subdivided. Members of a view are allowed to carry on the distributed computation separately from the members not included in the view. Partitionable systems are intended for applications that are able to take advantage of their knowledge about partitionings in order to make progress in multiple, concurrent partitions. As we have said in the introduction, applications with these characteristics are called *partition-aware*. Examples of applications that can benefit from a partition-aware approach can be found in areas such as computer-supported cooperative work (CSCW), mobile systems, and weak-consistency data sharing.

Dynamic Crash No-Recovery

This model is the basis for most implementations and was used by the first commercial groupware, Isis. In this model, processes cannot formally recover, instead the membership of the group is dynamic: when a process crashes, it is excluded, when it recovers, it joins under a new identity – this is called a new incarnation. The current membership information is called a view and is consistent across all members.

The main advantage of this model is simplicity: algorithms in the crash no-recovery model are relatively easy to implement, and the handling of view information is usually implemented using group communication primitives. The model has several drawbacks:

- Specifying dynamic crash no-recovery communication primitives is still an ongoing debate.
- The crash no-recovery model requires processes to be artificially crashed and restarted in case of a false failure suspicion. This is very expensive.
- The crash no-recovery model has trouble coping with a crash of a majority of processes.

[28] http://www.cs.cornell.edu/Info/Projects/Isis/

[29] http://lsewww.epfl.ch/projets/phoenix/

Crash Recovery Model

This model allows processes to crash and recover during calculations. While more powerful and more flexible, this model has mostly been considered in theoretical research and there have been no real toolkits produced. Beside complexity, the main issue of this model is that to tolerate a crash of all processes, usage of stable storage is required. As stable storage is typically implemented using hard disk technology, it is very slow.

10.1 Current and Recent Work

Database Replication

The distributed systems community has proposed many replication algorithms, most of them based on group communication. Unfortunately, most of this work has been theoretical and has been done considering only individual operations. This makes such distributed algorithms unsuitable for databases, since databases need concurrency control at a level higher than that of the individual operation.

Database replication has been traditionally used to provide exclusively high availability or scalability. Replica consistency is enforced by eager replication, which guarantees that at the end of a transaction all replicas have the same state. Jim Gray noted in [Gray et al 1996] that traditional eager replication protocols were not scalable. In contrast, lazy replication protocols propagate the updates of a transaction to the rest of the replicas after the transaction has committed and, therefore provide better response time than eager protocols.

Gray's paper triggered some research in both eager and lazy protocols. On one hand research was initiated to provide lazy replication with high levels of freshness [Pacitti and Simon 2000] and consistent lazy replication [Breitbart et al 1999]. On the other hand, it was studied how to overcome this apparent impossibility of scalable eager replication using group communication. One of the early efforts in this direction was the DRAGON project. This project was aimed at implementing database replication schemas based on advanced communication protocols developed by the group communication community [Kemme and Alonso 2000a][Kemme and Alonso 2000b].

The implementation of eager data replication protocols follow either a white, black, or grey box approach, depending on whether the implementation is performed within the database, outside the database, or outside the database with the addition of some functionality to the database, respectively. The white box approach was taken by Postgres-R [Kemme and Alonso 2000a][Kemme and Alonso 2000b] showing the feasibility of attaining scalable eager data replication. After this seminal work, given the inherent complexity of combining replication with the many optimizations performed within databases, the idea of replicating databases at the middleware level (outside the database) was explored by grey box [Jimenez-Peris et al 2002a] and black box approaches [Amir and Tutu 2002][Rodrigues et al 2002].

Middle-R [Jimenez-Peris et al 2002a] is a middleware for database replication being developed in the context of the Adapt project. Middle-R adopts a grey box approach to enable a scalability close to the one achieved by the white box approach [Kemme and Alonso 2000a]. This grey box approach requires two services from the database: One service to get the updates performed by a transaction, and another service to install these updates. These two services enable asymmetric processing [Jimenez-Peris et al 2003a] at the middleware level. Asymmetric processing consists in processing a particular transaction at one of the replicas and then propagating the updated tuples to the remaining replicas. Asymmetric processing boosts scalability of data replication as shown analytically in [Jimenez-Peris et al 2003a] and experimentally in [Jimenez-Peris et al 2002a]. Fully processing a transaction implies to parse and analyze the SQL statement, generate the query plan, possibly perform many reads and finally install some updates. Most of this work is saved at the other replicas, which only install the updated tuples. This saving provides the spare capacity required to boost scalability (even under workloads with a significant number of update transactions). In contrast, the black box approach does not rely in any service exported by the database [Amir and Tutu 2002][Rodrigues et al 2002]. This approach has the advantage of being independent of underlying database system. However, this seamlessness is not for free due to its inherent limitation in scalability [Jimenez-Peris et al 2003a].

All the aforementioned solutions are based on reliable uniform total-ordered multicast. This fact simplified the replication protocols but at the price of increasing transaction latency. To overcome this shortcoming it was observed that in local area networks, total order was achieved spontaneously when sites were not saturated [Kemme et al 2003]. This spontaneous total order can be exploited by delivering optimistically total ordered multicast messages with a high probability of success. This optimistic delivery allows overlapping the transaction execution with the establishment of the message total order. This overlapping effectively masks the

latency introduced by the total ordered multicast and was exploited in [Kemme et al 2003] to adopt an optimistic replication technique. However, this kind of optimism was not robust [Jimenez-Peris and Patiño-Martínez 2003b], meaning that when optimistic assumptions do not hold, transactions are aborted. During peak loads many messages might be lost and resent. In this situation the system would enter into thrashing behaviour, aborting a high percentage of transactions. Robust optimism [Jimenez-Peris and Patiño-Martínez 2003b] has been proposed as a way to overcome the shortcoming of traditional optimistic approaches during periods when optimistic assumptions do not hold. Robust optimism calls for additional safeguards that guarantee a non-thrashing behaviour during those periods. Robust optimism has been used successfully in [Patiño-Martínez et al 2000] in database replication protocols, with a negligible abort rate when optimistic assumptions do not hold. It has also been used to decrease probabilistically the inherent latency of non-blocking atomic commitment from three to two rounds [Jimenez-Peris et al 2001].

The Adapt project is concerned with developing support to achieve adaptive basic and composite Web Services. In the case of basic services, Adapt aims to provide support for adaptiveness at the different tiers of application servers, among them the database tier. As part of this research, adaptiveness is being introduced into Middle-R. Different kinds of adaptation are being studied: failures, recoveries, changes in the workload, variations in the available resources, and to changes in the quality of service (QoS) demanded by clients.

Replication already adapts to server failures by introducing redundancy. Traditional approaches to database replication have always assumed that recovery was performed off-line. That is, when a new or failed replica is added to the system, the whole system is stopped to reach a quiescent state and then, the state is transferred to the new replica from a working one. However, this approach contradicts the initial goal of replication: high availability. Some seminal work has suggested how to attain online recovery of replicated databases [Kemme et al 2001] using a white box approach. In [Jimenez-Peris et al 2002b], online recovery is studied at the middleware level. For adaptation to changes in the workload, Middle-R is being enriched with dynamic load balancing algorithms to redistribute the load evenly among replicas. For adaptation to variations on the available resources Middle-R is being enhanced with an adaptive admission control. The adaptation mechanisms being introduced in Middle-R in the context of Adapt are summarized in [Milan et al 2003].

Communication Frameworks

The implementation of group communication in the presence of process crashes and unpredictable communication delays is a difficult task. Existing group communication toolkits are very complex. The CRYSTALL project is interested in the development of (i) semantic foundations (or models) of group communication and (ii) implementations of group communication that would correspond closely to the model. The general objective is to facilitate the understanding of the behaviour of the system and the verification of its correctness.

In the CRYSTALL project, a group communication toolkit is designed and implemented using a modular approach. Group communication services are implemented as separate protocols, and then selected protocols are combined using a protocol framework. In addition to the possibility of constructing systems that are customized to the specific needs of an application, we have direct correspondence between the group communication model (defined as a set of abstractions) and its actual implementation. We currently experiment with implementing group communication using the Cactus and Appia frameworks.

Integrated Groupware

Group communication has seen little acceptance outside academia, possible causes are poor performance, complex models and a lack of usage examples for real applications, but also a lack of standard. There are numerous toolkits, but they are not interoperable. Because of this, applications designed for a particular groupware are not portable and linked to the fate of an academic prototype.

To solve this issue the main approach is to integrate groupware facilities in standard communication toolkits, such as middleware systems. The middleware infrastructure increasingly integrates services and concepts that originate from other communities: transactions, persistence, named-tuples. As group communication primitives typically offer message broadcasts, they are well suited to be integrated with message oriented middleware systems (MOM). Group communication concepts can also be implemented in RPC style middleware, typically to support replicated invocations.

The challenge is to integrate group communication primitives in a transparent way inside the communication toolkit and to expose all control mechanism in a standard way. The JMS standard offers a good mean to interface the broadcasting primitives [Kupsys et al 2003] and directory services such as LDAP are suited for interfacing the view membership service [Wiesmann et al 2003]. As new middleware concept and architecture emerge, like

for instance Web-services or Grids, new challenges and new opportunities for integration arise. Emphasis is set on lightweights components and interoperability. The goal is to build complex system using of the shelf components.

Ad-hoc Groupware

Group communication toolkits have until now concentrated on fixed networks. While group communication primitives would doubtless be useful in an ad-hoc network, implementing them is a difficult task. Because of node mobility and intermittent links, implementing low-level functionality such as routing is already difficult, so implementing more powerful primitives is quite a challenge.

On the other hand it is common to assume that the nodes of an ad-hoc network have access to information which are usually unavailable in fixed network: the location, speed and bearing of a node. Using this information, it should be possible to define new primitives that offer strong properties.

While there are many group-communication toolkits, the Franc project aims at building an ad-hoc oriented framework for both implementation and simulation. Theoretical research aims at building accurate models for wireless networks and defining cost and time boundaries for broadcast primitives.

Correct Modular Protocols

Initial group communication systems such as Isis or Ensemble were monolithic, that is new protocols could not be added easily to the system. Recently, systems such as Cactus and Apia have been built using modular architectures. In these frameworks, new protocols can be defined by combining existing or new micro-protocols. This approach aims at bringing flexibility and modularity to group communication systems. While new protocols can indeed be implemented more easily on these frameworks, proving that these protocols are correct and deadlock-free is a difficult task [Mena et al 2003].

The goal of the Crystal project is to build a modular group communication framework that offers facilities to prove that algorithms are correct in a rigorous fashion [Wojciechowski et al 2002]. The framework also helps to ensure that micro-protocols are deadlock free.

Groupware – Application Interaction

While a lot of research has been done to improve the group communication toolkit itself, research on how to use group communication has often been relegated to toy examples. In the context of the Dragon project, it became clear that group communication could be used to build replicated databases, but that the use of the group communication primitives was not straightforward. Research on how group communication primitives can be used to replicate real applications is needed, as there is no clear understanding of what exact primitive is needed for a specific application. Different applications have very different requirements, both in terms of fault-tolerance and needed semantics.

One important issue in this respect is the lack of end-to-end properties of group communication systems as highlighted by Cherion and Skeen in [Cheriton and Skeen 1993]. In particular, group communication as defined today cannot be used to implement 2-safe database replication [Wiesmann and Schiper 2003].

Security and Trustworthiness Issues

However efficient and sophisticated a group communication system is, if it lacks vital features such as security and trustworthiness evaluation of group members there is a considerable chance of its failure in a real world business setting. Group member communication in many cases must be confidential, integrity preserving and even non-repudiatable. Furthermore, group members might be particularly interested in interacting with other members who do what they say and abstain from those who are untrustworthy [Fahrenholtz and Bartelt 2001]. That is why project DISRS (Distributed Infrastructures for Secure Reputation Services) is concerned with

- defining necessary requirements and metrics pertaining to distributed systems that support reputation services,
- providing a prototype infrastructure that meets these requirements and metrics
- implementing a secure reputation service using modern cryptography [Fahrenholtz and Lamersdorf 2002].

Within this project it is planned to conduct an end user trial to gather users' view on the prototype system and to

assess how it encourages secure communication and interaction of group members. The focus is on employing Peer-to-Peer technology. The implementation of the prototype system will avail itself of current enterprise frameworks such as Microsoft .NET and SUN J2EE.

10.2 Future Trends

Group communication has been a hot topic for about a decade now. Unfortunately, this powerful paradigm has not become mainstream in the industrial field. The main reason for the little success of group communication technology may be identified as the lack of integration with existing paradigms. For example, in the database area (and more generally, in all enterprise technologies), transactionality is the most important property. But few if any of existing group communication toolkits are integrated with a standard transactional service. We firmly believe that, in order to promote the adoption of this paradigm, better integration with existing technologies is needed. In this sense, projects such as DRAGON and ADAPT are an attempt in this direction, integrating group communication into the database world and the enterprise world represented by Java 2, Enterprise Edition.

Current groupware systems are built on a certain system model. As new technologies emerge and new computing architectures gain acceptance, group communication systems will need to be adapted:

- **High-Performance Groupware**. One of the basic assumptions of group communication is that the network is slow. This assumption is already false in cluster settings where CPU contention is more an issue than network usage. With the advent of gigabit networking and given the fact that network bandwidth increases at a much faster pace than processing power, new group communication models will be needed to cope with such a changed environment.

- **Network Oriented Architecture**. Today's groupware systems are built around the assumption of computing nodes linked by a network. Processing and storage is done on the nodes, and the network simply transports messages. With the advent of network-attached storage (NAS) and more and more advanced networking equipment, this assumption is becoming less and less appropriate. Building groupware systems that rely on NAS and smart networking equipment will be an important challenge in the next few years.

Regarding the application of group communication to replicated databases we foresee two future directions: replication across wide area networks and adaptive replication. The research problems in the first area have to do with the latency of multicast in wide area networks that should be minimized. Some researchers are exploring overlay networks [Amir et al 2000] whilst others are resorting to optimistic delivery [Vicente and Rodrigues 2002][Sousa et al 2002].

11 Distributed Multimedia Systems

Distributed multimedia platforms allow distributed users to access, process and display media of many different types within an integrated environment. Their task is to support distributed multimedia applications by offering services to transport and handle continuous and discrete data in an integrated manner. Today the handling of interactive or streaming audio and video data is a central aspect of most multimedia applications. In future, one can expect that handling continuous data will be as natural as the handling of discrete data. For example, the integration of video and audio for teleconferencing will be a common service for interactive applications.

Early multimedia systems research illustrated the pitfalls of developing directly on top of system specific devices and services. Systems took a long time to develop and were not portable. Thus middleware platforms have been investigated, providing several basic services specifically for multimedia applications.

The main service offered by distributed multimedia platforms is support for media exchange. As a result, multicast and broadcast support is still a topic of current research work as well as that of the utilization of Quality of Service mechanisms for the network infrastructure. Moreover, security and digital rights management support are expected functionalities of distributed multimedia platforms.

Being a middleware component, distributed multimedia platforms make the system specific devices and services transparent to the applications. But an even greater challenge is to make the specific properties of the network infrastructure that is being used transparent to the applications. Current research work in this context is support of wireless networks, peer-to-peer, and broadcast overlay networks as well as the support of different QoS mechanisms.

11.1 Current and Recent Work

11.1.1 Networks

Current CaberNet work in the field of networking focuses on quality of service, multicast, mobility, and wireless networks.

Quality of Service

The processing of multimedia data by distributed platforms involves several components and requires the availability of different types of resources. If one or more components, involved in the processing of multimedia data, is not able to provide the required resources, then this will have a negative impact on the resulting media quality. Therefore the processing of multimedia data requires an end-to-end Quality of Service (QoS). In order to ensure this, two types of components and resources are considered separately today: the network which has to provide sufficient transmission capacity and end systems which have to provide sufficient processing capacity [Vogt et al 1998][Coulson el al 1998].

Considering the network QoS, different mechanisms for the reservation of resources – and therefore for realizing QoS – have been developed. These mechanisms differ by providing QoS for each data stream or for aggregated streams only. The latter is also called Class of Service. Though several mechanisms exist and are also available in real hardware and software (e.g. support for RSVP or DiffServ) their use is not common practice today. There are still open research issues considering the whole process of (dynamically) providing QoS by the network (projects DANCE, COMCAR and HELLO) [Henrici and Reuther 2003].

Considering QoS as the end-to-end characteristic of the system, the main problem is that of the reservation of CPU time for the processes which handle the multimedia data. Because current PC and server architectures were not designed to handle time-dependent data, it is usually not possible with standard operating systems to reserve CPU resources. Nevertheless, several solutions exist, but these are used only for specialized applications. For desktop PCs the common practice today is to depend on the availability of sufficient CPU resources (projects GOPI and HELLO) [Reuther and Hillenbrand 2003].

Reservation of resources requires that sufficient resources are available and affordable. Multimedia systems should be able to adapt to environments where the ideal amount of resources is not available or cannot be guaranteed. Considering the network QoS there are two main approaches for the distribution of multimedia data via computer networks:

- Download/offline distribution

- Real-time delivery using streaming technology.

While download is relatively uncritical regarding QoS requirements, the streaming approach requires in time delivery of the media data from a media server to the streaming client at a defined data rate. Due to this fact, streaming traffic in IP-based networks was considered to be non-elastic. The development of layered streaming techniques or scalable encoders/decoders is relaxing this requirement by allowing rate adaptation in conjunction with transmission control techniques for streaming media. This results in streaming systems that can be designed to be TCP- friendly by reactively or proactively reacting to congestion on the network path between client and server. A reduction of the sending rate in the case of congestion results in lower quality at the client terminal (project "Fault-tolerance in Audio / Video Communications via Best Effort Networks").

Wireless networks

Wireless network technologies such as UMTS/GPRS, Wireless LAN, and Bluetooth, and the resulting availability of high bandwidth mobile access networks, provide a platform for high quality mobile media distribution. Spurred only in part by the industrial interest in commercial distribution of multimedia content, the research community is focussing on the following aspect: How to deliver multimedia content via all-IP networks built out of multiple wireless and wired networking technologies?

Since wireless networks have transport and error characteristics different from those of wired networks, it is required that streaming systems adapt their transmission control as well as error correction technique to the transmission medium used. The main reason for this requirement is that transmission control schemes based on the principle of "handling congestion" do not work in an optimal sense in a wireless environment. In the case of a streaming client moving between different wireless and wired access technologies an automatic adaptation is required (project R-Fieldbus).

Mobility

Nowadays, network structures are relatively static. But mobile devices such as PDAs, handhelds, digital cellular phones, and notebooks that connect to the Internet at different locations and using different wireless media are becoming more and more widespread. Users wish to benefit from the possibility of accessing information everywhere with reasonable quality. Networks and the applications using them have to cope with this new demand [Kroh et al 2000].

Techniques such as "Mobile IP" offer the possibility to communicate with a mobile user as if it was a statically connected one. When the user moves from one network to another a switchover occurs and all data streams need to be redirected. As multimedia platforms often depend on a reliable network service, the interruptions caused by switchovers must be kept as short and transparent as possible (projects @HA, FABRIC and OZONE).

11.1.2 Content Distribution

The oldest and simplest form of content distribution is downloading: whenever media is needed it is copied from the content provider. This form of delivery is now often not appropriate. Equipped with powerful underlying network services, applications are able to stream multimedia data making data available quasi in real-time for the user. As stated above, the network must guarantee a certain quality of service or the application must adapt to the current network characteristics for this to work. Another problem is the large amount of data that has to be distributed. Multicast technology is required to stream media from one source to many receivers efficiently. With intelligent caching and mirroring techniques it is possible to limit needed network bandwidth further. While broadcasting or multicasting requires an appropriate infrastructure, these techniques are adequate mainly if there are a few static senders only, e.g., TV broadcasters. Other techniques such as peer-to-peer systems, are able to distribute content from many different sources. Broadcasting and multicasting require the support of the network infrastructure. Since the world wide Internet cannot be changed easily to support specific new services, overlay networks were used to realize services such as broadcasting and peer-to-peer [Metzler et al 1999][Akamine et al 2002].

Broadcasting of Multimedia Data

In the context of streaming media delivery using IP-based networks, broadcast scenarios such as streaming of live sporting events or live concerts to a huge number of receivers are of serious commercial interest. They are not possible in today's Internet without using multicast or content internetworking techniques.

The fact that the average Internet user is not connected to an IP-multicast infrastructure, together with the emerging interest on bandwidth intensive Internet broadcasting services, have opened the door to alternatives

acting at the application layer, namely peer-to-peer or overlay networks as a platform for application layer multicast.

In general, application layer multicast approaches use overlay networks to organize the nodes of a multicast group. One-to-many or many-to-many transport of data to the members of this group is performed by building data delivery trees on top of the Overlay. For data transport, unicast transmission is used. This allows features such as reliability (i.e., error control), congestion control and security to be deployed in a natural way on top of such overlays. A drawback of the Application Layer Approach is that it is impossible to completely prevent multiple Overlay edges from traversing the same physical link and thus some redundant traffic on physical links is unavoidable. The central challenge of the Overlay approach can be formulated as a question: How do end systems with limited topological information cooperate to construct good overlay structures? An answer to these questions involves defining:

- Application-specific metrics for measurement of Overlay performance
- Choosing an application-specific control and data delivery topology for the Overlay
- Development of distributed algorithms for Overlay organisation and maintenance (i.e. keep the overlay performing even if the network situation is changing) and measurement.

Grid and Peer-to-Peer

The success of peer-to-peer file interchange platforms such as Napster, Gnutella, Morpheus, eDonkey and Kazaa was accompanied by several research projects. They targeted the development of new schemes for organising nodes in peer-to-peer networks (CAN, Tapestry, Chord) or speeding up the search request in peer-to-peer networks.

The Grid is a type of parallel and distributed system that enables sharing, selection, and aggregation of geographically distributed “autonomous” resources dynamically at runtime depending on their availability, capability, performance, cost, and users' quality-of-service requirements. With the Globus toolkit, the basic communication protocols have been implemented and gained wide use. Research today lies within the definition of the Open Grid Service Architecture (OGSA), which embraces Web Service technology (see Chapter 8) to provide the standardized basic Grid services that organize and manage Grid resources and Grid nodes.

The goal of Grid research in the following years will be to enable applications to benefit from this higher level Grid architecture. It will be crucial to define commonly usable Grid services in different areas - services that enable application developers and providers to adopt their products and solutions to the Grid. Concerning distributed multimedia platforms, different Grid services for multimedia data (e.g., storage, delivery, and manipulation) will be identified and offered through the Grid. Topics such as authentication, authorisation and accounting will be of interest, as well as Grid-enabled multimedia software components that can be used to build distributed multimedia applications upon (projects Anthill and GridKIT).

11.1.3 Security and Accounting

Digital Rights Management

Codes for audio and video content such as mp3 for audio and MPEG-4 or its derivates such as DivX for video in combination with high bandwidth Internet access have enabled the exchange of music and video via the Internet. While server base solutions are more likely to be used to distribute legal content, distributed systems operating without anyone taking responsibility for them, i.e. being liable for their consequences, such as many peer-to-peer systems, are very extensively used to exchange data without respect to ownerships and copyrights. A prominent example for this was the peer-to-peer system Napster. Though mostly video and audio media is illegally copied, other digital media such as books are also affected by this development.

Since it seems to be nearly impossible to control the distribution of digital media in the world wide Internet, the media itself will be protected against unregulated duplication and usage. Therefore Digital Rights Management (DRM) and security mechanisms are essential parts for multimedia platforms. The challenge of the development of DRM is to fulfil the protection requirements of the copyright owners as well as the usability requirements of the users (project LicenseScript) [Wong and Lam 1999][Perring et al 2001].

Authentication, Authorization, Accounting

Users accept more and more to be charged for valuable services. Therefore the demand for accounting systems is rising. For this, users need to authenticate themselves to the service provider, so that no other user can use services in their stead. Also the Service provider will authorize the users to use a specific service.

Considering communication services then another reason for using authentication, authorization and accounting (AAA) arises. When different levels of Service are available (per flow or per class) then it is inevitable to use mechanisms for AAA. Otherwise users will tend to always request the best service level resulting in useless blocking of resources. This means that offering different QoS requires the use of AAA for communication services. Several research groups and the IETF are currently improving mechanisms for AAA (project NIPON) [Zhang et al 2003].

11.1.4 Integration, Convergence and Standardization

Middleware

Early multimedia systems research illustrated the pitfalls of developing directly on top of system specific devices and services. Systems took a long time to develop and were not portable. Middleware platforms are to decouple devices and services from each other, ideally in a platform independent way. Interfaces abstracting from technical details also enable convergence of different underlying technologies. This becomes even more important today because of the increasing number of different multimedia platforms, especially mobile devices.

Ideally, a middleware platform should enable the application to adapt to the characteristics of the current network, for instance the available bandwidth or utilization of QoS mechanisms. Besides reacting to the network characteristics and situation, the middleware should respond to changing user needs. In the COMCAR project, streaming media is adapted to changing network characteristics by the middleware according to the QoS profiles and rule sets specified by the application. Middleware platforms may even offer support specifically for multimedia applications, for instance provisioning of authoring models (see projects FABRIC and MAVA).

Web Services and XML

At higher layers, established middleware concepts are CORBA, COM and DCOM. Web-Services are a newer approach (see Chapter 8). Based on standardized building blocks such as XML and SOAP, they will offer support for the integration of different media types and provide platform independent services. Scalability and security are two significant topics as well as those of automatic service discovery and assembly. Using Web Services in the context of multimedia is a current research topic. The dynamic integration of Web Services into multimedia platforms will lead to more flexibility and extensibility.

XML is also used in open and vendor-independent media formats or wherever structured data needs to be exchanged. Solutions based on XML are becoming widespread, but due to lack of agreed upon standards many proprietary schemata exist (the PATIA project).

Standards

The availability of standards is important, especially for the development of communication systems since such systems are usually not used as local stand-alone solutions. All aspects of communication should be standardized: the communication protocols, the data formats or more specifically the media codecs as well as application-specific data such as control data or description languages (e.g. Synchronized Multimedia Integration Language - SMIL) which will be handled by distributed multimedia platforms).

When developing new communication applications and systems the availability of (preferably open) standards is essential. It is the task of the research community to develop the fundamentals for open standards.

Standardizations bodies such as IETF, ISMA, and 3GPP allow the development of interoperable streaming platforms for industry as well as for research. Based on the ISMA standard for MPEG-4 based audio/video streaming the Open Multimedia Streaming Architecture Project (OMSA) founded by the “Bundesministerium für Bildung und Forschung” (BMBF), is successfully working in the areas of

- Development of Audio/Video codecs for streaming, such as H.264/AVT and AAC, and
- Optimisation of the transport of multimedia data via all-IP Networks and broadcast techniques.

The intent of this work is to support the development of an open MPEG-4 based open streaming architecture.

The members of the OMSA project are the Fraunhofer institutes HHI in the area of video codecs, FOKUS in the area of networking, and IIS in the area of audio codecs.

11.2 Future Trends

Computing power becomes cheaper and devices more and more mobile and ubiquitous. This circumstance raises the demands for distributed multimedia systems enormously: any type of multimedia content and services needs to be made available anywhere, anytime, and on various devices.

Today, as presented in the previous section, the technological foundations for approaching this goal are under research and beginning to evolve. Media applications have already moved from text to images to video. However, the research community has to face many challenges to change the status of multimedia from “hype” to an established and pervasive technology.

11.2.1 Current Developments

Based upon research results of past years, multimedia platforms will become part of everyday life. Therefore, the already developed pieces of the puzzle need to be put together. Service quality and performance as well as scalability are important issues to be considered.

Deployment of Mobility, QoS-Features, and Middleware

Already developed technology needs to be deployed and become established in order to form a building block for multimedia applications and new research. Examples are the deployment of IPv6, multicast, and quality-of-service in the Internet. Operating systems need native support for handover and roaming to assist mobile computing. Middleware hides technological details and network characteristics from the application while still providing required feedback on network status. Other requirements are performance, extensibility, and security. For maximum interoperability and seamless internetworking suitable, open standards need to be defined.

High range of system capabilities

More and more mobile devices are capable of handling multimedia data. Currently cell phones are used to present audio and video data or even to record such data. PDAs are devices which can be used for nearly the same applications as desktop computers and many notebooks have integrated wireless network support. With ubiquitous computing even more different devices will be available. So the type and capabilities of devices that will be used as platforms for multimedia communication will increase. With this development the capabilities of the systems used as a basis for distributed multimedia platforms diverge.

Distributed multimedia platforms must take into account that they may be used on systems with similar capabilities to those of a ten-year-old computer (e.g. cell phones) as well as on modern high performance desktop PCs. So the mechanism for adapting distributed multimedia platforms to different environments must be improved to support the trends towards higher mobility and pervasiveness.

Utilization of Web-Service Technologies

Web-Services enable one to “plug-in” functionality provided by any server in the Internet. It is expected that these concepts will enhance many applications that may use these services. Also, distributed multimedia platforms may use Web Services to offer enhanced services or to outsource some functionality. For example, a multimedia platform could use a service for conversion of data types or for storing information in remote databases.

Convergence

Today, television and sound radio informs and entertains using a unidirectional data flow, and audio and video data are often still analog. Mobile phones are mainly used for communication and desktop PCs allow interactive Internet browsing. These fields of application will overlap more and more in the future, and their differences decrease. All data flows will be digital, providing higher spectral efficiency.

With higher bandwidth in networks and larger computational resources even in inexpensive devices, multimedia will become ubiquitous in more and more appliances. For instance, wearable devices will emerge thus increasing the need for powerful distributed multimedia platforms. Consumer electronics will not take the form of isolated

devices any more but rather will employ communicative ones, providing multimedia entertainment to the user (e.g., the Ozone project).

New fields of application

Distributed multimedia platforms will enter into new domains. For instance, billboards will be replaced by multimedia display panels providing new possibilities for marketing. Vehicles such as cars, trains, and aircraft will be equipped with multimedia devices. Driver or pilot multimedia systems will be assisted as intuitively as possible, and passengers will be entertained. Distance learning will become commonplace, whereby distributed multimedia platforms will help learners and tutors to stay in touch.

11.2.2 Future Developments

Distributed multimedia systems will become ubiquitous. Like previous steps from text to images to video, transition will progress further to 3D and virtual reality. All aspects of human life, i.e., working, living, learning, supply, and recreation, will be affected by multimedia systems providing services and augmenting human capabilities. This pervasiveness will raise new demands in scalability and integration of multimedia platforms.

Human centered systems

Besides solving technological issues, usability and intuitive handling need particular consideration. The focus needs to be put on people instead of technology, i.e. multimedia systems need to adapt to human requirements. “Calm” technology needs to be designed that does not bombard the user with information but stays in the background and raises attention in a level matching to the particular circumstance. Therefore, systems need to be context aware, automatically sensing which and what amount of information is currently required to give optimal assistance. Further attention will need to be paid to the social impact of ubiquitous multimedia systems[30].

Service-Orientation and Integration

The Web-Service concept enables the use of network services without having knowledge of their implementation or the platform being used. Users describe the required service and are able to use any service provider that fulfils the requirements. This is a high level of abstraction, which enables a high degree of flexibility and simplifies the usage of services. Distributed multimedia platforms already make specific platforms and network details transparent for the applications. But application programmers still have to deal with several technical details. Ideally, an application programmer will define only what should be done and the platform being used will automatically determine how this can be achieved. This also requires that the platform be able to detect limits of the given environment (firewalls, low network bandwidth, low CPU resources, etc.) and be able to adapt to this situation automatically and transparently to the application.

Systems that nowadays are independent or only loosely coupled will interact with each other in a seamless way. Technical details will be hidden from users and only the services to be fulfilled for the user will be exhibited.

Tele-Immersion

The goal of tele-immersion is to create the illusion that users at geographically dispersed places share the same physical space, i.e., tele-immersion should support communication and interaction between humans and devices with respect to all human senses. For example, people in different places should be able to meet in a virtual room and they should be able to do all the things they could do in a real meeting-room. This includes construction of a “virtualized reality”, i.e., construction of virtual worlds out of real scenes in real-time. Tele-immersion enables simulation of a certain environment and the usage of devices. Virtual environments (VE) can also be used in art, construction, and entertainment for artificial creations that look and feel real.

Tele-immersion has high demands on computer interfaces (especially graphics) and computer networking. Imagine viewing a basketball game from the viewpoint of your favourite player or even from the viewpoint of the ball. Within a tele-immersive scenario many types of information must be integrated, transported and processed in real-time. Besides video and audio data there are several additional information types which may be used to support human communication: position of objects and persons within the scenario, movement of objects and people, detailed information of faces, gesture or even smell. Further, all interaction with objects within a scenario must be supported. Therefore, tele-immersion is the ultimate multimedia application, placing extreme requirements on a distributed multimedia platform[31].

---

[30] http://www.extra.research.philips.com/euprojects/ozone/

[31] http://www.advanced.org/teleimmersion.html

E-Science

In the near future, digitally enhanced science will provide services and tools for researchers of various different kinds that enable them to make massive calculations, store and retrieve loads of data, collaborate across boundaries, and simulate, analyse, and visualize their experiments using the Internet (especially by employing Grid and peer-2-peer technology) in ways that are currently only imaginable. Distributed multimedia platforms will be a major part in this future research (called e-science) in the areas of collaboration, simulation and visualization. They need to make use of standardized interfaces to become self-organizing, self-configuring, self-healing and thus secure and reliable.

12 Network Storage Services

Most work on distributed storage systems has focused on the client-server model: the storage service is provided by a small set of reliable, highly-available and powerful server machines that are accessed by a large number of clients.

The client-server model for storage systems has two drawbacks. The reliability and availability of the server machines rests heavily on the ability and honesty of the system administrators that manage the storage servers. An incompetent or malicious system administrator can discard all the data stored by a server or corrupt the data in ways that are hard to detect. Ironically, system administration is increasingly outsourced or handled by temporary employees. Additionally, the model relies on expensive server machines with high bandwidth network connections. Administering these machines to achieve high-availability is also expensive.

12.1 Current and Recent Work

Recent research on large-scale peer-to-peer storage attempts to address both issues. The basic idea behind this research is to harness the aggregate storage capacity and bandwidth of a large number of inexpensive machines to eliminate the reliance on a small number of system administrators and reduce hardware and maintenance costs. These can be infrastructure machines placed strategically in the network to implement a large-scale storage utility or, in the extreme, it is possible to eliminate servers completely and implement algorithms that coordinate the client machines to provide the storage service.

Peer-To-Peer

There are several interesting research issues that need to be addressed to implement a usable peer-to-peer storage system: fault-tolerance, security, self-management, and scalability. Since peers are inexpensive machines without dedicated system administration, the reliability, availability, and security of individual peers is significantly lower than what can be achieved with a well-managed storage server. Therefore, it is necessary to develop appropriate replication algorithms to achieve the desired levels of reliability and availability, and to secure these algorithms against incompetent or malicious users. Additionally, these systems must be self-organizing to keep administration costs down. They must be able to work with machines joining and leaving the system continuously. Finally, these systems must be able to store very large amounts of data on behalf of a very large number of users. This scalability is necessary when implementing an Internet-scale storage utility or a storage system for a large corporation.

There have been a large number of projects on peer-to-peer storage systems over the last three years. Examples of such systems include OceanStore [Kubiatowicz et al 2000], Farsite [Bolosky et al 2000][Adya et al 2002], CFS [Dabek et al 2001], PAST [Rowstron and Druschel 2001b], Mnemosyne [Hand and Roscoe 2002] and XenoStore (see Pasta [Moreton et al 2002]). This chapter provides an overview of two CaberNet projects: Farsite and Past. These projects illustrate two interesting points in the design space: Farsite provides full file system semantics with shared mutable files and directories, while Past only provides storage for immutable objects but scales to larger systems.

Farsite [Bolosky et al 2000][Adya et al 2002] was designed to eliminate the need for storage servers in a large corporation. It coordinates the machines in the corporation to implement a reliable, highly available, and secure storage system. It is designed to scale up to 100,000 machines.

Any authorized user can publish a traditional hierarchical file system namespace in Farsite. It does this by choosing a set of nodes from the pool of available machines to implement the root directory. These nodes join a replica group that runs a practical Byzantine fault tolerant state machine replication algorithm (BFT [Castro and Liskov 2002]). This algorithm allows a group with 3f+1 replicas to work as a single correct machine provided at most f of replicas fail or are compromised by an attacker. The group maintains directory data for the namespace it manages but it does not store file contents, which represent the bulk of file system data. Instead, the group stores secure digests (e.g., SHA-1[32]) of the latest versions of files and the identity of f+1 machines that store copies of the file. The machines that store copies of the file are chosen by the group from the pool of available machines.

[32] "Secure hash standard", National Institute of Standards and Technology, Federal Information Processing Standards Publication 180-1, April 1995.

To achieve scalability, a replica group can delegate a portion of the namespace that it manages to a new replica group when the old group becomes overloaded. The new group is selected by the members of the old group from the pool of available machines. Both groups maintain pointers to each other to enable navigation in the directory structure and cross-directory operations. Additionally, scalability is achieved by using leases and client caching of directory and file data.

[Rowstron and Druschel 2001b] describe a peer-to-peer archival storage utility called PAST that was designed to scale to the Internet. Files in PAST are immutable and can be shared at the discretion of their owner. PAST is built on top of Pastry [Rowstron and Druschel 2001a], which is a structured overlay that maps application keys to overlay nodes. Nodes have identifiers that are selected randomly from the same numeric space as keys. Pastry provides applications with a primitive to send a message to a destination key. The message is delivered to the node whose identifier is numerically closest to the key. Messages are delivered in O(log N) hops and nodes maintain only O(log N) routing state. Additionally, Pastry balances the key management and routing load evenly over all the nodes in the overlay.

PAST derives the key that identifies a file from the file contents using a secure hash function. It uses Pastry's routing primitive to store a file on the node that is responsible for its key. Additionally, the file is replicated on the set of k live nodes with identifiers numerically closest to the key. Pastry maintains information about these nodes and notifies applications when they change. PAST uses these notifications to move replicas to new nodes when nodes fail or new nodes join the system. Copies of the file are retrieved by sending a retrieve request to the file's key using Pastry's routing primitive. This provides reliability and high-availability and distributes storage load evenly over all nodes.

Pastry enables applications to intercept messages at each overlay hop. PAST uses this functionality to implement a dynamic caching mechanism to handle hotspots. Popular files are cached along Pastry routes towards the file's key to shield the nodes responsible for storing the replicas from excessive load. Security is achieved by using public-key cryptography and there has also been recent work on securing Pastry's routing mechanism and enabling the storage of mutable objects in PAST [Castro et al 2002].

There is also work on network attached storage devices that replace servers by a large number of less expensive devices connected by a very fast network in a cluster. This work does not address the issue of trust on system administrators but it reduces hardware costs and maintenance costs because it enables incremental scalability and repair. Ongoing work to correctly define the interfaces to network-attached storage is taking place mainly in the industrial arena through the auspices of the National Storage Industry Consortium (NSIC). Network storage interfaces can be classified by the amount of functionality provided by the device. High-level file interfaces such as NFS [NFS 1989] and CIFS [CIFS 1996] provide most of the functionality of file systems, and hence provide protection; low-level block-based interfaces such as iSCSI [Satran] provide the functionality of disks, because of their flexibility any file or database system can use them; it is intermediate-level interfaces that have been the subject of more recent research. Such interfaces attempt to combine the safety of high-level interfaces with the flexibility of low-level ones. Object-based interfaces [Gibson and van Meter 2000][Mesnier 2002] provide a middle ground between the two extremes, but they still provide such a high-level interface that devices must implement many functions which can benefit from application-specific policies. Commercial offerings of Storage Area Networks (SANs) also abound, with vendors such as IBM and HP shipping mature products.

Content Delivery Networks

A parallel research direction in the Network Storage Service area investigates architectures and techniques to improve the so-called “Quality of Experience” (QoE) [CacheFlow] that is the user satisfaction in accessing the content. Today the Content Delivery Networks (CDNs) are considered to be one of the most promising technologies in this area. The CDNs key approach to decreasing the user response time (the parameter that mainly contributes to the QoE) is to distribute content over servers (called *surrogates*) located close to the users. When a request for accessing content (e.g., an HTTP request) arrives, the CDN Request-Routing System (RRS) redirects it to the *best* surrogate, which provides a fast delivery of the content required. In general, the more surrogates are present, the better access and delivery service a CDN can provide. However, increasing the number of surrogates may imply high costs and requires a careful estimation of the number of resources necessary to provide a given delivery service. A different solution for a more flexible architecture consists in renting the resources on-demand. The idea of using federations of CDNs, also named Content Distribution Internetwork (CDIs) [Green et al 2002][Day et al 2003] was born to this end. A CDI allows different CDNs to share resources so as to enable each CDN to employ a large number of surrogates.

The CDI technology is a new research domain and the main ideas in this field are outlined in the RFCs and Internet drafts produced by the IETF CDI Working Group [IETF]. A number of aspects in this area should be investigated further to make this approach useful. Specifically, issues related to the Request-Routing

Internetworking System (RRIS) [Cain et al 2002], i.e., the system responsible for redirecting the client request to the *best* CDN, have been recently analysed. The RRIS bases its decision on the performance data that are collected from the other CDNs. Such data represent the performance (e.g., response time, network delay, etc.) a CDN is able to deliver a given content in a given region in a given moment.

Due to the importance of the performance data, RIEPS (Routing Information Exchange Protocol for a Star Topology) [Turrini and Ghini 2003], a protocol that allows performance data to be exchanged among CDNs, has been implemented. The critical point of this type of protocol is to find out the right trade-off between the freshness of the information and the network load. The first parameter is quite important, as the performance data is totally useless if it does not reflect the real situation. On the other hand, the performance data that are usually considered in making forwarding decisions have a high variability. Thus maintaining the data performance coherent with the real situation may be quite expensive. To investigate the RIEPS behaviour, an experimental evaluation is needed. Moreover, since RIEPS is a protocol that allows the information necessary for each CDNs to perform request forwarding to be exchanged, it will be evaluated taking into consideration the effectiveness of the forwarding decisions based on the performance data that it exchanges. The RIEPS evaluation will also be performed with a larger scale approach. This means, in particular, that more than two CDNs will be considered. If a source CDN interacts with more than one destination, CDNs it may need different information from different CDNs. For instance, if the source CDN in a given moment realises that destination CDN A is providing much better performance compared with destination CDN B, it can ask CDN B to send performance data only if they are similar to the ones of the CDN A.

12.2 Future Trends

An issue that needs to be addressed for peer-to-peer network storage to be viable is the design of an incentive mechanism for peers to cooperate. Studies of Gnutella found that a large fraction of the peers are freeloaders, i.e., they use resources from the common pool without ever contributing any resources [Adar and Huberman 2000]. This is not a problem for storage systems inside a corporation or on infrastructure machines but it must be addressed when peers are random machines in the Internet. Additionally, more work is needed on systems that support mutable shared objects at Internet scale. Finally, it is necessary to deploy these systems and evaluate them in real everyday use. A detailed comparative analysis between peer-2-peer systems and traditional client-server architecture is necessary in order to outline the real potentialities of both systems and foresee their future development. This comparison should be done considering both the different nature of the two systems (goals, content managed, architecture) and also the performance provided (user response time, usability and resource availability).

In the CDI area many research directions are possible; specifically since RIEPS represents one of the first attempts of research in the CDI scenario, the interaction between RIEPS and other CDI components would be worth investigating.

More generally, we can expect to see the commoditization of storage, and more precisely the integration of storage services into the network model. Some initial work here should focus on logistical networking, that is, considering storage and communication as tightly coupled problems and designing solutions accordingly. Other challenges for the future include building effective personal storage networks, enhancing data availability in times of network outage or partition, increasing information security, and raising the semantic level of access interfaces.

13 Network and Distributed System Management

Network and Distributed Systems management is a broad discipline covering all the adaptation and control activities performed on a system after it is deployed to ensure its correct and optimal functioning. In addition, Network and Systems management also covers activities such as provisioning and capacity planning which are performed during the deployment phase. Research in network and systems management can be broadly subdivided in two areas: (i) the design of systems management infrastructures which includes the development of new techniques, tools and architectures for systems management and (ii) the application of those techniques to various domain specific areas such as network quality of service (QoS), multimedia, and cellular telecommunications.

However, to be able to *manage* a system it is necessary to monitor its behaviour and to perform control actions which adapt its behaviour and functioning. Thus, although partial monitoring of a system can be achieved non-intrusively, monitoring and especially control activities can never be entirely separated from the design of the system itself. Thus, many of the projects focussing on network and systems design invariably comprise and use innovations in management techniques for making them adaptable and re-configurable. In particular, many traditional *management* services and techniques such as instrumentation, event handling, management by delegation and policy-based management are being considered as important components of the design of mobile and pervasive infrastructures which rely on asynchronous event notifications to adapt their functioning. Therefore, many of the projects mentioned in this chapter overlap with those in other chapters. In particular, projects on network QoS are also addressed under network design and multimedia (see Chapter 11), and projects on pervasive and mobile systems often encompass adaptation techniques and security projects which cover security management in their scope.

13.1 Current and Recent Work

There are continuing research efforts towards providing guaranteed Quality of Service and reliable communications in multiservice networks. Current investigations focus on Traffic Engineering (including load measurement and modelling) [Wolfinger and Kühn 2002][Wolfinger et al 2002] policy-based management, and the provision of adaptive and programmable networks. Policy-based management is seen as an important technique towards providing uniform control of network resources and driving the adaptation of QoS mechanisms according to changes in application requirements, variations in network usage or in response to failures and performance degradations [Lymberopoulos et al 2003]. Advances have been made towards policy specification [Damianou et al 2001], policy deployment [Dulay et al 2001], the implementation of tools for policy-based systems [Damianou et al 2002] and policy analysis and refinement [Bandara et al 2003]. The ANDROID project has been using policy-based management of proxylets which execute in the application layer within servers in the network and hence provide an Application Level Active Network. Application level adaptation techniques such as adaptive video encoding and information dispersal have also been investigated [Richter 2001].

In parallel with the work on network QoS, substantial effort is being devoted towards the development of reliable, fault tolerant architectures and control mechanisms for a variety of application areas including the automotive industry. A significant theme across many of the ongoing studies is the development of new techniques and infrastructures capable of providing adaptive behaviour at both the application and the network level in large distributed systems. This work is becoming even more critical in mobile and ubiquitous systems which are characterised by numerous dynamic changes in network connectivity, usage context and service availability.

13.2 Future Trends

Work towards providing QoS in multiservice networks will certainly continue although there is a need for better integration between the network provisioning, traffic engineering, and management systems.

Increasing emphasis will be placed on the development of mobile and ubiquitous computing environments (see Chapter 5). Entities in these environments must exhibit a high degree of “intelligence” by adapting both their requirements and their behaviour to frequent changes in their environment and the context in which they evolve. Frequently, entities in these environments such as laptops, PDAs or embedded computers have limited resources and limited network communication capabilities. Management in such environments is a significant challenge as

the need for adaptability, programmability and context awareness is exacerbated. To address these issues new techniques for providing dynamically adaptable control in embedded devices are necessary and highly dynamic management frameworks which combine a variety of adaptation mechanisms and techniques must be defined. The scale and widespread deployment of such systems will require new self-managing devices which can collaborate within the context of *self-managed* and dynamic coalitions. A new facet in ubiquitous systems is the need for context-aware management and management of context services [Hegering et al 2003].

The main challenge in the years to come is therefore to identify common architectural patterns and techniques for self-management that can be extensible and scale down to dynamic coalitions of embedded devices, e.g., for body area networks and scale up to large enterprise systems.

Increasing security concerns have emphasised the need for environments where security is not the result of a static configuration manually changed by a system administrator but where the security system continuously adapts and reacts to events such as failures, intrusions and interactions with other systems. This requires a better integration between the management and the security systems and renewed efforts towards adaptive security management.

Conclusions

While core techniques such as policy-based management and monitoring are steadily progressing, research in Distributed Systems Management seems at first sight to be decreasing. However, this is a mistaken impression which is due to the fact that adaptivity techniques are playing a more important role in the newer mobile, dynamic, context-aware and autonomous systems. If research in the early 90's was characterised by problems of scalability and in the late 90's by problems of interoperability, the first decade of the new millennium is characterised by the challenges of **adaptability** and **autonomous** ("autonomic" in IBM's lexicon) evolution. It is therefore not surprising that these rely on paradigms of constrained programmability (policy) and detection of change (monitoring).

14 Control and Coordination in Dynamic Virtual Organisations

Organisations are increasingly using the Internet to offer their own services and to utilise the services of others. The availability of a wide variety of services and resources over the Internet creates new opportunities for providing value added, inter-organizational services by composing multiple exiting services into new Composite Services (see Chapter 8). This naturally leads to resource sharing across organisational boundaries. An inter-organisational business relationship is commonly referred to as a virtual organisation (VO). Whether in the context of large-scale scientific experiments, eCommerce, eGovernment or any other collaborative effort, organisations need cost-effective ways of finding, purchasing and managing services performed by other organisations. It is therefore necessary for organisations to be able to set-up and manage business links with other organisations in a rapid, dynamic and flexible manner. A VO, however, blurs the distinction between 'outsiders' and 'insiders', yet organisations forming a VO will want to preserve their individual autonomy and privacy. A central problem in VO management is therefore that of how organisations can regulate access to their resources by other organisations in a way that ensures that their individual policies for information sharing are honoured. Regulating access to resources by other organisations is made difficult as the organisations might not trust each other. Organisation will therefore require their interactions with other organisations to be strictly controlled and policed. How can this be achieved? First we need to understand trust management issues in open systems (see [Grandison and Sloman 2000] for review of trust-related issues and the iTrust project).

Trust is a vital component of every business interaction. The Oxford dictionary defines trust as “Firm belief in the reliability or truth or strength of an entity”. Following [Gerck 1998], we consider a trust relationship of the form: **A** *trusts* **B** *on matters of* **X** *at epoch* **T**

Here, *A* and *B* may be people, computers and their specific resources and services, or even small or large organisations that admit to *trust relationships*. In the proposition, *A* is placing a trust relationship (*dependence*) on *B* with respect to matters of *X*. Such matters constitute the set of *rights and obligations* of *A* with respect to *B*, such that *B* permits access to its specific resources (services) to *A* provided that *A* fulfils specific *obligations* (*conditions*) laid down by *B*. Epoch *T* represents the period during which both *A* and *B* observe the well-being of the their trust relationship without incidence of failure.

In the paper-based world, businesses have been conducted using contracts. The concept and the use of contracts are not new to today’s society. Legal contracts can be traced back to ancient times [Halsall 1999]. There are records that indicate that legal contracts were used by the Mesopotamians circa 2300-428 BC for selling and purchasing slaves, land and crops. Also, there is evidence that shows that the Mesopotamians knew how to write and sign legal contracts for hiring services such as houses and workers; and for establishing partnerships between two or more land-owners.

To form and manage VOs, we need to emulate electronic equivalents of the contract-based business management practices; this means: (i) relationships between organizations for information access and sharing will need to be regulated by *electronic contracts*, defined in terms of a variety of *service level agreements* (SLAs), and then enforced and monitored; and (ii) organizations will need to establish appropriate *trust relationships* with each other and implement corresponding access control policies before permitting interactions to take place.

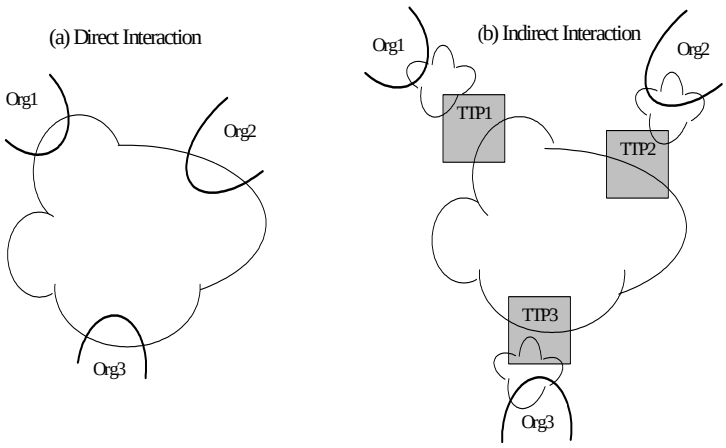
We note that development of infrastructure support for VO management is attracting significant interest from the research communities active in Grid, middleware and Web Services domains [Gerck 1998][Foster et al 2001][Shrivastava 2003][OGSA][Ouzounis and Tschammer 2001][Keller et al 2002]. In the next section we define middleware-related research problems that need to be tackled.

14.1 Current and Recent Work

Middleware for regulated interactions

This is the layer of software that regulates the interaction between two (or more) contracting parties who, despite mutual suspicion, wish to interact and share information with each other. We would like to know if it is possible to come up with a reasonably small set of generic services that can be used to support arbitrarily complex interactions between organizations. Figure below shows two basic interaction styles, where a cloud represents middleware services shared between the organizations: (a) organizations share trusted middleware services to enable direct interactions with each other; and (b) no trusted middleware services are shared between organizations, so interactions take place through trusted third parties (TTPs) acting as intermediaries. A given contractual relationship could be implemented by either of the two interaction styles. In a dynamic setting, it may

be that interactions initially take place through TTPs and once sufficient trust has been established, organizations agree to interact directly. What services would be required? From the viewpoint of each party involved, the overarching requirements are (i) that their own actions meet locally determined policies; and that these actions are acknowledged and accepted by other parties; and (ii) that the actions of other parties comply with agreed rules and are irrefutably attributable to those parties. These requirements imply the collection, and verification of non-repudiable evidence of the actions of parties who interact with each other.



Law-Governed Interaction (LGI) [Minsky and Ungureanu 2000] is an example of middleware for regulated interactions; LGI is a message exchange software layer that allows a group of distributed agents to interact over a communication medium, honouring a set of previously agreed upon rules (a law). Laws are enforced by controllers which are trusted entities conceptually placed between each agents and the communication medium. Another approach is the B2Bobject [Cook et al 2002] middleware that resembles a transactional object replica management system where each organization has a local copy of the object(s) to be shared. Any local updates to the copy by an organization ("proposed state changes" by the organization) are propagated to all the other organizations holding copies in order for them to perform local validation; a proposal comprises the new state and the proposer's signature on that state. Each recipient produces a response comprising a signed receipt and a signed decision on the (local) validity of the state change. All parties receive each response and a new state is valid if the collective decision is unanimous agreement to the change. The signing of evidence generated during state validation binds the evidence to the relevant key-holder. Evidence is stored systematically in local non-repudiation logs. It is assumed that each organization has a local set of policies for information sharing that is consistent with the overall information sharing agreement (contract) between the organizations. The safety property of the B2BObject system ensures that local policies of an organization are not compromised despite failures and/or misbehaviour by other parties; whilst the liveness property ensures that if all the parties are correct (not misbehaving), then agreed interactions would take place despite a bounded number of temporary network and computer related failures.

The middleware architectures described above are suitable for enabling 'direction interactions' to take place between organizations. However, it should be noted that, in the presence of misbehaving organizations, liveness guarantees cannot be provided. If liveness guarantees are required under such conditions, then interactions need to take place through TTPs, as depicted in the figure ('indirect interaction'). Any such interaction protocol is likely to be based on the notion of 'fair exchange'.

Fair exchange protocols play an important role in application areas where protocol participants require mutual guarantees that an exchange of data items has taken place in a specific manner. An exchange is fair if a dishonest participant cannot gain any advantage over honest participants by misbehaving. Practical schemes for fair exchange require a trusted party that essentially plays the role of a notary in the paper-based schemes. Two-participant fair-exchange protocols that make use of a trusted third party have been studied extensively in the literature (e.g., [Zhou and Gollmann 1997][Pfitzmann et al 1998][Asokan et al 1998]); these protocols maintain fairness (which also implies liveness) even if the dishonest participant can tamper with the protocol execution in an unrestricted (malicious) manner. They however require that an honest participant's node execute the protocol correctly – suffering no failures. To be of practical use, these protocols need to be made fault tolerant; a fair exchange protocol is fault-tolerant if it ensures no loss of fairness to an honest participant even if the participant's node experiences failures of the assumed type. Such protocols have been investigated recently [Liu et al 2000][Ezhilchelvan and Shrivastava 2003].

Contract monitoring and enforcement services

At a higher level (above the regulated interaction middleware), one would like services for contract management. Electronic contract management services should provide ways for representing contracts in terms of rights and obligations so that they can be enforced and monitored. It is clearly not possible to prevent organisations within a VO from misbehaving and attempting to cheat on their agreed trust relationships. The best that can be achieved is to ensure that all contractual interactions between such organisations are funnelled through their respective contract management services, and that all other non-contractual interactions are disallowed.

Of course, there is no easy way of automatically transforming a contract written in a natural language into an executable version, but several systematic, semi-automatic approaches have been proposed and are under investigation [Marjanovic and Milosevic 2001][Abrahams and Bacon 2001][Daskalopulu et al 2001][Molina-Jimenez et al 2003]. For example, a systematic way of representing contracts as finite state machines (FSMs) and how rights and obligations can be monitored is described in [Molina-Jimenez et al 2003]. This work needs to be progressed further as it is not yet clear how such a framework can be deployed dynamically and made to respond to changes. For example, it is anticipated that an organisation might offer a given service to business partners under slightly different contractual terms that are negotiated dynamically. To cope with this complexity, the organisation should have tools for automatic generation of customised electronic contracts from the generic version. Note that contract monitoring and enforcement services themselves will be implemented according to the interaction patterns depicted in the previous diagram: a monitoring and enforcement service could be run on behalf of a VO on a TTP or in a decentralised manner. Another area needing further work is integration of advanced role based access control techniques, such as those developed in OASIS [Bacon et al 2001] that provide extended notions of appointments, for delegation of role-playing, and of multiple, mutually aware domains for mobile roles, that can be re-located and still able to communicate without confusion.

Service composition and enactment

The availability of contract management services as discussed above creates a safe and secure way for organisations to form VOs that provide new composite services (CSs). Clearly there needs to be a common standard between organizations for specifying, publishing, finding and composing CSs. Indeed, emerging Web Services standards such as SOAP, UDDI, WSDL etc. are a step in this direction (see Chapter 8). Unfortunately, they do not yet address issues of stateful services that can be made available, scalable and adaptive. Some of these issues are being addressed by several industry led efforts aimed at developing (often competing!) standards for specifying, composing and coordinating the execution of CSs; these include ebXML/OASIS [ebXML], Web Services architecture work at W3C [WSA] and Rosettanet [Rosettanet]. The ongoing work on Open Grid Services Architecture [OGSA] could provide a more unifying framework. In any case, certain basic facilities need to be developed.

High-level tools are required for service creation and management, including a service description language with exception handling facilities for consistency management. The fact that individual services may have clean “ACID” transactional semantics does not necessarily imply that CSs will have the same: it has long been realized that not all interactions can be structured as ACID transactions. In many cases, a CS will be composed of a mixture of transactional and non-transactional services, and CS functions will be required to preserve user specific consistency properties in the presence of a variety of exceptions. For this reason, a CS can be very complex in structure, containing many temporal and data-flow dependencies between its constituent services.

Not only do we need a high level language for service description capable of expressing these dependencies we also need a run time environment where constituent services can be invoked respecting these dependencies. Service invocations should be possible despite the temporary unavailability of these services, for instance due to processor or network failures, among others. Workflow management systems provide a suitable approach for CS specification (as a business process) and enactment. Contract management must be made part of the business processes of the organizations involved. An organization’s business processes can be divided into two broad categories, namely, the business processes that are internal to the organization and the ‘contract management processes’ that involve interactions with trading partners. A difficult problem is that of coordinating multiple workflows in a decentralised manner. Most commercial workflow systems are inherently centralised and do not appear to be suitable. There are several industry led efforts aimed at developing standards for Web Service coordination. The specifications are still evolving and often ill defined [van der Aalst 2003].

14.2 Future Trends

Presence of a wide variety of services and resources over the Internet creates new opportunities for providing value added, inter-organizational services by dynamically composing multiple exiting services into new Composite Services (CSs). Dynamic service creation leads to business interactions that cross-organisational boundaries. The organisations forming a CS might not necessarily trust each other, so an important challenge is concerned with developing infrastructure support for interactions between two or more potentially mutually distrusting parties. The infrastructure should support interaction in a hostile environment: interactions between organisations over open networks. Given these observations, the development of middleware for composite service provisioning that takes into account the requirements of inter-organisational interactions poses very challenging research problems that have been highlighted here.

Part V. Software Engineering

15 Rigorous Design

This chapter is concerned with the design, modelling and analysis of distributed systems using formal methods and the use of these methods in a "rigorous" style. The chapter is, by nature, very general, and other chapters in this report have shown how formal methods are used within the particular topic areas. For example, see Chapter 3 on Distributed System Security, and Chapter 4 on Real-Time Systems.

The term "formal methods" covers a range of techniques based on notations which have precise (mathematical) meaning and which thus facilitate formal or rigorous reasoning about designs and implementations of systems. It is important to make clear that there is no claim here that formal methods can achieve perfection nor that they remove the need for other approaches to achieving dependable systems: soundly based specifications and designs ought be one component in achieving dependability; fault-tolerance, testing and measurement are complementary necessities. All engineering disciplines benefit from mathematical underpinnings; software engineering is no exception. It is pointed out here that the key potential of formal approaches is in the design process.

Distributed systems have a high complexity because they nearly always exhibit concurrent, distributed, real-time behaviour. This combination of factors is likely to be found in an increasing number of critical hardware and software systems as a consequence of the inevitable growth in scale and functionality and the trend to interconnect systems in networks. The likelihood of subtle, difficult to detect, errors in this class of systems is even greater than in systems without concurrency and real-time requirements. The problems are further exacerbated because it is often the case that human interaction with the system must be considered.

With traditional quality control measures, such as peer review and testing alone, it is not possible to obtain sufficient statistical information about the adequacy of software for complex and critical systems in the presence of rare but possibly catastrophic errors. This is because in order to find rare errors in general an unthinkable amount of testing or simulation would be required to detect such errors [Liggesmeyer et al 1998][Rushby 1993].

The key to obtaining pay off from the use of formal methods is to use them throughout the design process. It is for this reason that this chapter is entitled "Rigorous Design". This argument is given in more detail in [Jones 1996]. The main justification for this view is that it is incredibly difficult to *create* a correct system and that this is where the effort should go. There is no point in creating an admittedly faulty system and then trying to remove the errors. It has been pointed out above that testing is inadequate but even a heroic attempt to prove correctness after a faulty design is an incredible waste of effort.

Other engineering disciplines such as the design of airframes would never throw something together then attempt to sort out the mess; software design also has to learn to progress from a clear specification to an implementation by a series of justified steps. Only in this way can we avoid the "scrap and rework" which is where productivity is lost in software creation. (This is not to say that there is no place for formal tools at the end of the implementation process: model checking has achieved popularity because its use requires less expertise in the formalism itself – this topic is discussed in detail in Section 15.3 below.)

Another key advantage of using formalism throughout the design process is that it will help identify and remove unnecessary complexity in the architecture of a system. It is precisely complexity which causes so much of the difficulty in obtaining correctness and confidence in correctness. The need to reason rigorously about a design will identify simplifications long before subsequent work is based on it. An interesting approach, developed within the DSoS project, which combines rigorous design (notably the use of Jones' rely-/guarantee-conditions) with fault tolerance is described in [Hayes et al 2003].

Before going into more detail, it is worth closing off one other misplaced dream. It is *not* wise to look for automatic translations of specifications into implementations: [Hayes and Jones 1989] gives technical reasons for this; [Hoare et al 2004] points out that is exactly the diversity between a posited design and the specification thereof that uncovers errors. The error might even be in the specification but it is the tension between two views which is key to progress towards a higher quality product.

15.1 Formal Specifications

The key to any development should be a specification. If the specification is written in a natural language such as English it will inevitably contain ambiguities (one can again make comparisons with other major engineering projects which would never rely solely on text to define a complex system).

There are many formal specification notations (such as VDM, Z, VVSL, Larch, RAISE, B) which have been reasonably widely used in significant systems; there are other notations which can be used to define aspects of a system's behaviour (CSP, CCS, pi-calculus, temporal logic, TLA). As well as providing the advantage of precision, the creation of a formal specification is likely to clean up one's ideas of a system and its architecture. Not only do post-conditions (of operations or actions) provide a way of documenting "what rather than how" but the intelligent use of abstract objects (e.g. in the description of the state of a system) enhances the understanding of a system because precision can be achieved without committing to implementation decisions.

It is interesting to speculate why the obvious advantages of formal specifications are not universally recognised. As far as the engineers themselves are concerned, it would appear that abstraction is actually difficult for some people: a trained implementer needs a course in "anti-programming" to appreciate that it is *abstraction* that is looked for in a specification – and some never see the point. As far as management is concerned, a focus on the early design stages leaves them nervous that there is nothing tangible (i.e. running code) visible. It is as though programmers are expected to have half of the airplane flying by the halfway point in a project to reassure management that the whole project is "on time".

It is of course true that requirements all too often change during a software project – to argue that this is a justification for getting to code early is difficult. One cannot help wondering whether it is judged by some as better *not* to have an assessment of the impact of changes!

A good example of developing formalisms for specific areas is the Mikado project which aims to develop new formal models for both the specification and programming of large-scale, highly distributed mobile systems [LaCoste 2002]. Another project ("A Constructive Framework for Partial Specifications"), covering the challenging area of the specification of complex systems, develops a general, practical and constructive framework for the composition of partial specifications which together capture the specification for the total system [Bowman et al 2002].

15.2 Design

A rigorous design process is one in which the steps can be formalised.

We will use here terminology from the VDM literature [ISO 1995] but the points apply equally to RAISE [RAISE 1995] or B [Abrial 1996]. Starting from a formal specification, one is likely first to *reify* the abstract objects: a designer proposes (more) concrete objects which are close(r) to the eventual implementation language. In a completely formal justification of a design, the relationship is recorded by *retrieve functions* and the *adequacy* of the representation is proved. It is then necessary to show a formal *commutativity* property (wrt the retrieve function) for each operation. In a *rigorous* design, a design inspection might approve a step of design based solely on the retrieve function and satisfactory answers to questions where the reviewers have any doubts.

Formalisability shows that there are extra steps of formalisation that can be resorted to where doubt persists.

A similar story exists for the later stages of the design process where (once the data representation has been designed and justified) algorithms are designed to satisfy the *pre-post conditions*. Again, in a completely formal process, *proof obligations* can be extracted and even discharged using a theorem prover like Isabelle or PVS. In a *rigorous* approach, a single invariant might be recorded and accepted at a walkthrough.

A useful resource for literature on formalism in general (and rigorous methods in particular) is the long running series of conferences which were initially known as "VDM-Europe" and more recently as "Formal Methods Europe". All of the proceedings since 1987 have been published by Springer-Verlag in their influential *Lecture Notes in Computer Science* series. The conferences are distinctive in that they are a place of real interchange between (applicable) theory and practical use.

Formal methods can be used as a complementary quality control measure that can reveal inconsistencies, ambiguities and incompleteness, and several other shortcomings of system designs early on in the development process. This reduces significantly the accidental introduction of design errors in the development of the software leading to higher quality software and cost reduction in the testing and maintenance phases of system development. Formal methods and their related software tools have been used extensively and successfully in the past in a variety of areas including protocol development, hardware and software verification, embedded/real-time/dependable systems and human safety, demonstrating that great improvements in system behaviour can be realised when system requirements and design have a formal basis.

15.3 Model Checking

In the following we highlight some of the recent advances and challenges in the development of *model checking*. Model checking is a verification technique in which efficient algorithms are used to check, in an automatic way, whether a desired property holds for a finite model of the system. Very powerful logics have been developed to express a great variety of system properties and high-level languages have been designed to specify system models.

One of the major advantages of model checking, in comparison with e.g. theorem proving, is the possibility to automatically generate counter-examples that provide designers with important debugging information [Clarke et al 2002]. Another advantage is the fact that the verification is automated so that much of the mathematical detail of the verification is hidden from the designer. This reduces the amount of training that is required in order to use model-checking tools. The cost effectiveness of this technique could be further improved by its use earlier in the software development.

15.3.1 Current and Recent Work

Model checking is becoming widely used in industry (e.g. IBM, Intel, Microsoft, Motorola, SUN Microsystems, Siemens, Bell labs) and is being applied to numerous industrial case studies and standards ranging from hardware, operating systems, compilers, communication protocols, to control systems, embedded real-time systems and multi-media application software. Evidence of the high relevance of model checking for the dependability community is the presence of a workshop on model checking in 2003 at the International Conference on Dependable Systems and Networks, the most important conference on dependability.

The main problem of model checking is the well-known *state-space explosion problem*. However, techniques have been developed to alleviate this problem. Most notable is the use of appropriate abstraction techniques that allow models of systems to be verified with an essentially unlimited number of states.

A further important issue is the relation between the abstract models used for specification and verification and the final implementation of the system. Formal testing is an approach that forms a bridge between these models and the implementation. It re-uses formal models for automatically generating/synthesizing relevant test-suites for component, integration and system testing and for actually executing such tests against real system implementations [Brinksma and Tretmans 2001].

With the recent increase in efficiency of model checking techniques there is renewed interest in its direct application to software written in real programming languages such as C and Java. This could enormously enhance the capability of error detection in software code but it depends critically on the development of appropriate heuristics and automatic abstractions to guide the search for possible errors in the enormous state space generated by the software [Groce and Visser 2002].

For an ever increasing class of systems their functional correctness cannot be separated from their “quantitative correctness”, e.g. in real-time control systems, multimedia communication protocols and many embedded systems.

Hybrid Systems

Hybrid systems are characterized by a combination of discrete and continuous components: they take the form of a discrete controller embedded in an analogue environment, where the analogue part of the system may concern variations of the ambient temperature of the system, the volume of coolant in a chemical process, or, more simply, the passage of time (between system events).

Formal models for hybrid systems typically take the form of a finite-state automaton (representing the discrete component) equipped with a finite set of real-valued variables (representing the continuous component) [Alur et al 1993]. The underlying model is therefore a state transition system; this differs from control theory which is based on continuous dynamical systems where state variables evolve according to differential equations. The values of the variables may influence the transitions among the states of the automaton (for example, by enabling particular transitions for choice), and, conversely, the transitions between such states and the passage of time when control remains within the states can affect the values of the variables (for example, a transition may reset the values of some continuous variables to a particular value, and differential equations describe how the variables change over time when control resides in a given state). A classification of models of hybrid automata along with their theory is introduced in [Henzinger 1996] [Henzinger 2000]. A subclass of hybrid automata

called the timed automata [Alur and Dill 1995], which admits real-valued clocks that increase at the same rate as time as the only continuous variables, has been applied successfully to model and verify real-time systems.

The presence of real-valued variables in formalisms for hybrid systems means that the underlying semantic model of such formalisms is infinite-state. This makes their automatic analysis difficult, if not infeasible. A breakthrough result concerning the development of automatic verification methods for real-time systems was made by Alur and Dill [Alur and Dill 1995], who presented a technique for obtaining a faithful finite-state representation of timed automata. Unfortunately, the decidability of basic sub-tasks of verification such as reachability is undecidable for many classes of hybrid automata (for example, the reachability problem for timed automata equipped with one clock which can be stopped in some states and restarted in others is, in general, undecidable), although some decidability results have been developed.

Model checking tools for timed automata, such as UPPAAL and KRONOS, implement decidable algorithms, whereas semi-decidable model checking algorithms are implemented in model checkers of hybrid automata, such as the tool HYTECH. The development and implementation of efficient algorithms for timed automata has made possible the verification of several (moderately-sized) industrial case studies.

Stochastic Systems

Many systems exhibit stochastic dynamics, either in the form of discrete probabilistic behaviour that is the outcome of coin tossing, or as described by continuous probability distributions (the probability of the system moving to a given state within a specified time is given by a probability density function). Probabilistic models, typically some variants of discrete or continuous Markov processes, are those whose transition relation is probabilistic. Probabilistic modelling is used to represent and quantify uncertainty; as a symmetry breaker in distributed co-ordination problems; to model unreliable or unpredictable behaviour; and to predict system behaviour based on the calculation of performance characteristics.

The now established field of performance evaluation aims to develop formalisms and tools for modelling systems and analysing their performance measures, as a means to support the process of design and engineering. The analysis involves building a probabilistic model of the system being considered, typically a continuous time Markov chain (CTMC), but often more general probability distributions are needed. Such models can be derived from high-level descriptions in stochastic process calculi or Petri nets. The model serves as a basis for analytical, simulation-based or numerical calculations which result in steady-state or transient probabilities and the associated performance measures (resource utilisation, average call waiting time, etc). The focus is on quantitative characteristics, including measurement and testing, and covers a broad spectrum of issues.

Probabilistic model checking is an extension of model checking techniques to probabilistic systems. An example of this is found in the “Probabilistic Model Checking of Mobile Ad Hoc Network Protocols” project [Kwaitkowska et al 2003]. As in conventional model checking, a model of the probabilistic system, usually in the form of a discrete or continuous time Markov chain (DTMC/CTMC) or a Markov decision process (MDP), is built and then subjected to algorithmic analysis in order to establish whether it satisfies a given specification. The model checking procedure combines traversal of the underlying transition graph with numerical solutions. Model checking of non-probabilistic systems has developed very quickly from first algorithms into implementations of industrially relevant software tools. In contrast, model checking of probabilistic systems has not followed the same path: although the algorithms and proof systems have been known since the mid-1980s, little implementation work has been done until recently, culminating in the development of tools PRISM [Kwiatkowska et al 2002] (for DTMCs/CTMCs and MDPs) and ETMCC [Hermanns et al 2003] (for DTMCs/CTMCs), with the help of which a number of systems have been studied (IEEE 1394 FireWire, Crowds anonymity protocol, etc). As in the case of conventional model checking, state space explosion is a particular difficulty, and has been tackled through an adaptation of symbolic (BDD based), methods and parallel, distributed or disk-based techniques. These issues have been investigated with project Automatic Verification of Randomized Distributed Algorithms.

15.3.2 Future Trends

Completely formal methods (and related tools) can be used most profitably to analyse parts of systems that are most critical. The development of guidelines for the identification of those system parts, the level of formality that is required and the selection of the most appropriate method to analyse its properties need to be addressed. There is strong evidence that rigorous approaches pay off not only in getting more quickly to a correct product but also result in cleaner, more maintainable, architectures.

Particular attention must be paid to the development of models that can take user interaction with the system into account. Also here quantitative models play an important role when it comes to descriptions of human performance aspects relevant for the overall evaluation of critical systems.

The use of different methods brings about the issue of the relation between models used for different verification purposes (e.g. simulation models, models used for model checking and theorem proving but also formalisms used for data intensive versus those for control intensive problems). This becomes particularly relevant in the case of combined verification.

Training of designers, integration of tools and methods in the design process and the development of course ware to allow a broad take-up of formal methods in industry and elsewhere are fundamental for the transfer of available knowledge to the daily practice of software developers.

Hybrid Systems

We expect significant advances concerning compositionality, hierarchical modelling, refinement and object orientation. In the timed automata case, understanding of the relationship with control theory and the unification of theories is an achievable goal. Much of the future effort will be directed towards methods of analysis of timed and hybrid systems, including verification by model checking and simulation, and the scaling up of these methods to industrial size examples. The successful development of timed automaton model checking tools such as UPPAAL will be consolidated in two respects: on one hand, there will be further advances in the development and implementation of efficient algorithms for timed automata verification; on the other hand, methods for the verification of extensions of the basic timed automata model for example, with parameters, probabilities, and costs/rewards will be implemented and applied to real-life systems. Progress in this direction has been made by project Verification of Quality of Service Properties in Timed Systems.

Stochastic Systems

The challenges for this area include the modelling of spatial mobility for example based on the spi-calculus or hybrid automata. This topic is being investigated by the projects: Probabilistic Model Checking of Mobile Ad Hoc Network Protocols and A Future Of Reliable Wireless Ad hoc networks of Roaming Devices.

Compositionality, modularity and hierarchical notations will be studied and we anticipated progress to be made. Furthermore, counterexamples generated by model checkers provide designers with valuable information about possible problems in their designs and the extension of these examples to probabilistic model checking is a key issues that need to be addressed. These topics are included in the aims of project Automated Verification of Probabilistic Protocols with PRISM.

The development of convenient and powerful logics to express relevant quantitative dependability measures in a concise and formal way is key to future systematic and possibly automatic, analysis of dependability aspects of critical systems. In addition, the development of logics to express in addition cost and reward based measures can greatly enhance the number of interesting measures relevant for dependable systems.

16 Software Architectures for Distributed and Dependable Systems

Architectural representations of systems have been shown to be effective in assisting the understanding of broader system concerns by abstracting away from details of the system [Garlan et al 2002]. This is achieved by employing architectural styles that are appropriate for describing systems in terms of *components*, the interactions between these components - *connectors*, and the properties that regulate the composition of components - *configurations*. Thus components are units of computation or data store, while connectors are units of interaction among components or rules that govern that interaction.

Defining notations for the description of software architectures has been one of the most active areas of research in the software architecture community. Regarding the overall development process, the *Architecture Description Languages* (ADLs) that have been proposed so far are mainly concerned with architecture modelling during the analysis and design phase. However some existing ADLs enable system implementations to be derived and deployed, provided there is an available implementation of the system primitive components and connectors. Hence, although most efforts are conducted independently, we may envisage the production of architecture-based development environments that ease the building of robust software systems since one of the contributions of the software architecture field lies in providing methods and tools for the thorough design and assessment of the system architecture. In particular, a major objective in the definition of ADLs is to provide associated CASE tools, which enable tasks underpinning the development process to be automated. In this context, a special emphasis has been put on the usage of formal methods and associated tools, whose application to the analysis of complex software systems is eased due to the focus on system architectures that are abstract and concise. As a result, work in the software architecture community provides a sound base ground towards easing the development of distributed systems, as summarized in Section 16.2, which more specifically focuses on middleware-based distributed systems.

The dependability of systems is defined as the reliance that can justifiably be placed on the service the system delivers [Avizienis et al 2003]. Dependability has become an important aspect of computer systems since everyday life increasingly depends on software. Although there is a large body of research in dependability, architectural-level reasoning about dependability is only just emerging as an important theme in software engineering. This is due to the fact that dependability concerns are usually left until too late in the development process. Additionally, the complexity of emerging applications and the trend of building trustworthy systems from existing untrustworthy components are causing dependability concerns be considered at the architectural level. Examples of the untrustworthy components that might be the basic building blocks of new software systems are commercial off-the-shelf components (COTS), legacy systems, and component systems of systems-of-systems. Several recent dissemination initiatives, in terms of workshops, have specifically tackled some of the issues related to the area of software architectures and dependable systems. The first one was the Workshop on Architecting Dependable Systems (WADS 2002) that was held during the International Conference on Software Engineering (ICSE 2002) [de Lemos et al 2002a][de Lemos et al 2002b]. As an outcome of this Workshop, a book has been published containing extended versions of selected papers presented in the Workshop, together with some invited papers [de Lemos et al 2003b]. The second initiative, perhaps in a broader sense of dependability, was the ACM SIGSOFT Workshop on Self-Healing Systems (WOSS 2002) that was held during the Foundations of Software Engineering Conference (FSE 2002) [Garlan et al 2002]. The final initiative was the Workshop on Software Architectures for Dependable Systems (WADS 2003) that was held during the International Conference on Software Engineering (ICSE 2003) [de Lemos et al 2003a][de Lemos et al 2003d] A clear outcome from these workshops has been the recognition that this area is an emergent area in which a lot of work remains to be performed, and that there should be more collaboration between the international communities of dependability and software architectures for cross fertilization of ideas and exchange of knowledge and experiences. Against this background, Section 16.2 summarises ongoing effort on architecting dependable systems, considering the four categories of dependability presented in Chapter 2.

As suggested above, the architecting of distributed and dependable systems raises a number of challenges concerning the provision of methods and tools that effectively facilitate systems development. Evolution in distributed and dependable systems, e.g. the task of developing systems that enable the ambient intelligence vision, introduces additional challenges. These are briefly surveyed in Section 16.3.

16.1 Software Architectures for Distributed Systems

The development of distributed software systems is recognised as a complex task: in addition to the development of the system-specific parts, issues raised by distribution management should be addressed. However, since the early 90s, the development of such systems has been simplified through the emergence of standardised software infrastructures that offer solutions to problems frequently met in application families. Such infrastructures are provided by component and middleware technologies. Briefly stated, components correspond to the building blocks of distributed systems, and may be easily composed during interaction. This corresponds well to the notion of component used in architectural descriptions, except it is closely coupled with some middleware technology that is a middleware layer lying between the application and the network operating system, and providing reusable solutions to problems such as heterogeneity, interoperability, security, transactions, fault tolerance etc. Middleware and component technologies are now exploited for the development of most distributed systems (see Chapter 7). This leads to refining the notions of architectural building blocks as well as to examining the architecture-based design of middleware underpinning the development of distributed software systems.

Matching Architectural and Middleware Building Blocks

The building blocks of distributed software systems relying on some middleware infrastructure fit quite naturally with those of software architectures. Hence, the development of such systems can be assisted with an architecture-based development process in a straightforward way. This is already supported by a number of ADL-based development environments targeting system construction such as Darwin (from Imperial College) and Aster (from INRIA) developed by the CaberNet partners. In this context, the architectural components correspond to the application components managed by the middleware, and the architectural connectors correspond to the supporting middleware. Most of the work on the specification of connectors has focused on the characterisation of the interaction protocols among components whilst connectors abstracting middleware embed additional complex functionalities (e.g., support for provisioning fault tolerance, security, transactions).

The above concern has led the software architecture community to examine the specification of the non-functional properties offered by connectors. For instance, these are specified in terms of logic formulae in [Issarny et al 1998b], which further enables synthesising middleware customised to the application requirements as supported by the Aster ADL [Issarny et al 1998a]. Dually, domain-specific ADLs such as C2 [Taylor et al 1996], target the description of architectures based on connectors enforcing specific interaction patterns, which may not be directly supported by middleware infrastructures. This issue has been investigated in [Dashofy et al 1999], which explores the applicability of middleware infrastructures to the construction of domain-specific software architectures.

Another issue that arises when integrating existing components, as promoted by middleware infrastructures, is that of assembling components that rely on distinct interaction patterns. This aspect is known as architectural mismatch [Garlan et al 1994b] and is one of the criteria substantiating the need for connectors to be first-class entities in architecture description. The abstract specification of connector behaviour as for instance supported by the Wright ADL [Allen and Garlan 1997] enables reasoning about the correctness of component and connector composition with respect to the interaction protocols that are used. However, from a more pragmatic standpoint, software development is greatly eased when provided with means of solving architectural mismatches, something which further promotes software reuse.

Connectors implemented using middleware infrastructures actually abstract complex software systems comprising not only a broker and proxies but also services for enhanced distribution management. Hence, middleware design deserves as much attention as the overall system design and must not be treated as a minor task that simply involves reliance on some middleware infrastructure. Architecture-based design is again of significant assistance here. In particular, existing ADLs facilitate the description of middleware architectures as addressed in [DiNitto and Rosenblum 1999]. In addition, since middleware architectures build upon well-known solutions regarding the enforcement of non-functional properties, the synthesis of middleware architectures that comply with the requirements of a given application may be partly automated through a repository of known middleware architectures [Zarras 2000]. In the same way, such a priori knowledge about middleware architectures facilitates the safe dynamic evolution of the middleware architectures in response to environmental changes, by exploiting both the support for adaptation offered by novel reflexive middleware infrastructures and the rigorous specification of software architectures enabled by ADLs [Blair et al 2000].

Concluding Remarks

Results in the area primarily lie in the definition of ADLs that allow the rigorous specification of the elements composing a system architecture which may be exploited for aiding in the system design and in particular in the assessment and construction of software systems.

Ongoing research work focuses on closer coupling with solutions that are used in practice for the development of software systems. This includes integration of ADLs with the now widely accepted UML standard for system modelling [Issarny et al 2002]. From this perspective, one issue that remains is whether architecture description should only be given in terms of UML diagrams with a possible extension of the UML language, or be a combination of ADL-based specifications and UML diagrams. Practically, the former approach should be encouraged so as to ensure the actual usage of solutions aimed at easing architecture-based development of software systems. However, it is not yet obvious that this is achievable, although growing concern for architecture description in industry should lead to adequate UML support. Still in this direction, coupling with OMG's model-driven architecture should be much beneficial.

Another area of concern when considering the development of actual distributed systems is that of exploiting middleware infrastructures and in particular the CORBA standard for systems development. This issue has already deserved a great deal of attention and there exist architecture-based development environments that do ease the design and construction of middleware from middleware infrastructures. However, addressing all the features enabled by middleware within the architecture design is not yet fully covered. For instance, this would require capturing the composition of, possibly interfering, middleware services enforcing distinct non-functional properties. Another area of ongoing research work from the standpoint of architecture specification relates to architectural evolution as required by emerging applications, including those based on the Internet and/or aimed at mobile computing. In this context, it is mandatory to enable the design of system architectures that can adapt to the environment.

16.2 Software Architectures for Dependable Systems

The state-of-the-art is presented in the context of current achievements, in terms of the four means by which dependability is attained (see Chapter 2) which we briefly recall: *Fault prevention*, also known as rigorous design, aims at preventing the occurrence or the introduction of faults; *Fault removal*, also known as verification and validation, aims at reducing the number or severity of faults; *Fault tolerance*, aims at delivering correct service in the presence of faults and *Fault forecasting*, also known as system evaluation, aiming at estimating the present number, the future incidence, and the likely consequences of faults.

Fault Prevention

Fault prevention is concerned with all the development activities that introduce rigor into the design and implementation of systems for preventing the introduction of faults or their occurrence during operation.

One of the bases for obtaining rigor in the design of software systems is the usage of formal or rigorous notations to represent them. In other words, architectural models should be the starting point of any development. Thus dependability attributes of system components should be clearly documented, together with the static and dynamic properties of their interfaces. Also, as part of the models, assumptions should be documented about the required and provided behaviour of the components, including their failure assumptions. Although UML, and now UML 2.0 [Björkander and Kobryn 2003], are the de facto standards in terms of notations for describing system designs, there are several languages that would be more appropriate for representing the architecture of systems [Medvidovic and Taylor 2000]. Nevertheless, industry still heavily relies on UML for obtaining models of their business, architectures, and design, and as well as the metamodels that allow defining dialects that are appropriate for describing their applications. Although some of these architectural languages have formal semantics associated with them, in general, there is a lack of a formal underpinning that would allow direct manipulation of their notations [Bernado and Inverardi 2003].

In addition to representations, rigor in the process of relating representations is equally important. For instance, there is the question of how to derive architectural strategies that will be able to enforce the security policies that exist for preventing the introduction of vulnerabilities. Moreover, once an architectural model is obtained, how can one instantiate that model into an executing platform, guaranteeing at the same time that all the dependability properties are maintained. Current work is being carried out that relates, in a consistent way, dependability concerns from the early to late stages of software engineering by following the principles of Model Driven Architecture (MDA) [Rodrigues et al 2003].

The approaches associated with the prevention of architectural mismatches (design faults) attempt to protect a component, or the context of that component, against potential mismatches by adding to the structure of the system architectural solutions based on integrators (more commonly known as wrappers). The assumption here is that the integrators are aware of all incompatibilities between system components [Rodriguez et al 2002] (the DSoS project).

Fault Removal

Fault removal is concerned with the verification and validation techniques, such as, inspection, model checking, theorem proving and testing, for reducing the number or the severity of faults.

Most of the approaches for removing faults from architectural representations have centred on model-checkers [de Lemos 2001][Zarras et al 2003] (the latter within the DSoS project). In this approach, models of dependable systems are checked against properties that must hold for the system to behave correctly. Depending on the system and the type of properties to be analysed, different model-checkers can be used.

When building systems from existing components, it is inevitable that architectural mismatches might occur. The approaches associated with the removal of this type of design fault are those that aim at localising architectural mismatches during the integration of arbitrary components [Gacek 1998]. Existing approaches for identifying architectural mismatches are aimed at the development of software, either during the composition of components while evaluating the architectural options [Gacek 1998], or during architectural modelling and analysis [Egyed et al 2000].

Fault Tolerance

Fault tolerance is concerned with mechanisms and techniques that ensure the system continues to deliver correct service in spite the presence of faults. Most of the work in architecting dependable systems has been done in this area, including tolerance against malicious faults [Verissimo et al 2003] (cf the MAFTIA project), i.e., intrusion tolerance. To leverage the dependability properties of systems, solutions are needed at the architectural level that are able to guide the structuring of undependable components into a fault tolerant architecture.

In one of the approaches, an idealised architectural component is defined with structure and behaviour equivalent to that of the idealised fault-tolerant component concept [de C. Guerra et al 2003a]. This approach was later extended to deal with COTS software components [de C. Guerra et al 2003b] (partially within the DOTS project). An example of an architectural solution that makes use of compensation, to ensure dependable system composition and evolution when upgrading components, is an approach that makes use of diversity between old and new versions of components [Medvidovic et al 2003]. Fault treatment can also be achieved at the architectural level, and a good example of such an approach is the use of architectural mechanisms that allow a system to adapt at run time to varying resources, system errors and changing requirements [Garlan et al 2003].

Another run-time software repair solution that is architectural-based relies on events and connectors to achieve required structural flexibility to reconfigure the system on the fly. The architectural repair in performed atomically [Oriezy et al 1999][Dashofy et al 2002]. Considering that architectural mismatches cannot all be either prevented or removed, an alternative solution is to tolerate them [de Lemos et al 2003c]. In the context of large-scale systems, such as Internet applications, techniques are needed to coordinate concurrency control and failure occurrences in open environments [Tartanoglu et al 2003] (the DSoS project). Outside the context of fault tolerance, compensation has been used to ensure dependable system composition and evolution when upgrading components, by employing an approach that makes use of diversity between old and new versions of components [Medvidovic et al 2003].

A novel architecture has been proposed to support the dynamic resource management in real-time dependability-critical distributed systems [Porcarelli et al 2003]. The solution relies on adaptive system reconfiguration, a technique which employs a hierarchy of resource managers for providing fault tolerance and scalability, and allows statistical modelling to be performed to facilitate decision-making related to system reconfiguration. For the same kind of problem, however in the context of an open system, an approach was proposed that extends the basic Jini architecture [Tichy and Giese 2003]. Such approach makes use of service-discovery mechanisms as a means to provide essential services once there is a failure in one of the system components, by rediscovering lost components or finding other components that provide the same services [Dabrowski and Mills 2002]. Also in the context of distributed systems, and perhaps more forward looking, is the notion of self-organizing software architecture, in which the objective is to minimize the degree of management for supporting change whilst preserving the architectural properties of the system [Georgiadis et al 2002].

Most of the contributions from the software engineering community are presented as self-healing approaches [Garlan et al 2002]. The differences between self-healing and fault-tolerance were discussed in a panel organized during WADS'03 [de Lemos et al 2003a].

Fault Forecasting

Fault forecasting is concerned with the estimation of the presence, the creation and the consequences of faults. So far, very little work has been done in the area of architectural fault forecasting, although the importance of field has been identified for a while [Klein et al 1999]. One of the interesting examples of the current work in the area is on the quantitative evaluation of dependability attributes which is performed by transforming UML diagrams into Timed Petri Net models [Majzik et al 2003].

Also as part of fault forecasting is the analysis of service degradation, and an approach has been presented in which graceful degradation is presented as a system property [Shelton and Koopman 2003]. For measuring the degradation of services in the presence of multiple component faults, the architectural representation of systems plays a major role.

In terms of forecasting faults at the level of individual components, the DBench project aims to define a conceptual framework and an experimental environment for benchmarking the dependability of COTS components. Once actual dependability measures are available, these can be incorporated into architectural representations of system to be used on their evaluation.

16.3 Future Trends

Future directions in the field of software architectures for dependable and distributed systems can be conjectured from two perspectives, design and run time. The fundamental issue for design time is tool support to represent and analyse the different properties of a system, both qualitatively and quantitatively. This would encompass the need to have a common representation of the different architectural description languages to support portability across specialised tools. A problem arising from such capability is how to integrate the analysis performed from different viewpoints. For that, solution such as aspect architectures might be a way forward. From the perspective of rigorous development, support should be provided that allows the efficient generation of components integrators (or wrappers) based on the identified incompatibilities during component integration. This would enable to build, efficiently, systems from existing components. Another important contribution for the development of systems from their architectural representation would be the ability to generate architectural solutions directly from system policies. This would allow the efficient tailoring of existing system architectures into specific applications.

In terms of run time, the major capability of future dependable and distributed systems, from the perspective of their architecture, will be the provision of autonomy. Systems, in the future, should be able to handle changes, both internal and external, without human intervention. For achieving such capability, there is a need for structural flexibility that would facilitate the architectural reconfiguration of systems. This however raises several issues, not only related to the process of decision making during run-time, but also the need for identifying criteria and mechanisms that would guarantee stable architectures. In the context of the Internet, a basic requirement for achieving such flexibility, perhaps in the context of Web Services, is the need to have open architectures that would support heterogeneous architectural solutions (see Chapter 8). If future systems will be capable of reconfiguring themselves in response to changes that might occur, then mechanisms have to be envisaged that permit to introspect the architectural elements for observing their internal structure, not only for the purpose of detecting changes, but also for analysing the feasibility, or even, the optimality of alternative architectural actions.

There is clearly growing interest in industry in using appropriate mechanisms for dealing with software architectures. This is caused by growing understanding that it is important to work with an abstract view on the system, to do it right from the beginning of development, to analyse it, to allow for future flexibility and openness by choosing the right architecture at the earlier phases of development.

The viability of future distributed systems, such as ambient computing and Web Services, will inevitably depend on the architectural modelling and analysis of such systems due to their inherent complexity and scale. If system developers are not equipped with the appropriate mechanisms, techniques and tools the technological evolution will be impaired.

Part VI. Conclusions

17 Conclusions and Recommendations

17.1 Future Systems and Trends in RTD

The vast majority of future computer-based systems will be distributed systems of various scales built on a variety of interconnection mechanisms using a wide range of development paradigms and techniques. The defining characteristics of these emerging systems are as follows:

- Openness – the systems will be assembled and reassembled online out of component systems which are found dynamically, having been developed without precise knowledge of the context in which they would be employed
- Ubiquity – presence of computer systems everywhere; this prediction is based on massive recent advances in technology, and the assumption that these advances will continue to be made
- Mobility of devices, users and code
- Adaptivity developed in order to deal with the changing environment, system and subsystem behaviour, characteristics, and requirements
- Growing system complexity caused by new types of interdependencies between systems and their components, the growing complexity of the underlying infrastructures and number of system components and layers, and the need to build new systems out of existing and often running systems
- The ever increasing role of dependability due to the continuously growing reliance that society, organisations and the individual place on computer-based systems
- Great variety of faults, threats and vulnerabilities that systems have to deal with, often simultaneously, including human (especially malicious) and organisational faults, application software faults, and system faults occurring during or even caused by online modifications and mismatches of components when composed.
- Future systems will require means to deal with broken trust and confidence, broken safety, broken privacy, broken confidentiality, etc.
- The growing use of COTS, legacy components, third-party code, external services
- New application domains in which distributed systems will be employed.

In future, not only will the focus be on system integration rather than on development from scratch, but integration will also be of a new kind: flexible, dynamic, open and dependable. Consequently, a number of challenges will need to be addressed, including:

- Development of novel software engineering solutions. Advanced development techniques covering several phases of the development process will be proposed to allow disciplined and rigorous reasoning about design models. In particular
    - Software architectural solutions that address the issues of system robustness and trustworthiness, system openness (allowing for system integration, composability, building systems of systems), system reconfiguration via openness, and accommodation of legacy elements.
    - Rigorous design methods. The focus will need to be on developing cost–effective compositional methods for analysing system models and developing tools and tool engines with better interfaces capable of tracking all development decisions (see, for example, Part 3, Appendix E [AMSD 2003]). It will be important for work to be carried out not only on traditional fault avoidance approaches but also on the rigorous analysis of system fault tolerance measures. The concern often raised by industry that the application of existing methods usually requires specially trained people will have to be addressed.
- Making service-oriented computing into a practical widely used and accepted solution. There are many indications that this will be the predominant paradigm. But there are still many concerns to be

addressed, including service-oriented architecture applicability, ease of use (including development of various tools), engineering applications based on service-oriented computing, investigation of performance-related issues, gaining wider experience, finding better ways to use this approach in conjunction with all previous solutions, developing specialised solutions for various application domains, etc.

- Development of systematic approaches in order to deal with the legacy components in a predictable dependable and tractable fashion

- Finding solutions that will allow wide and flexible run time choices. To support this, exhaustive well-structured information about system and component requirements, characteristics, parameters, architecture and design decisions will have to be systematically used at run time. This will, for example, allow for run time dependable system adaptation and evolution [Jones and Randell 2004]

- Proposing novel dependability-ensuring techniques supporting explicit handling of system dependability characteristics through the system development process. These will allow the end-to-end characteristics to be analysed, and the dependability requirements to be met by construction

- Developing advanced mechanisms suitable for ensuring fault tolerance in future systems which may face faults of new and unanticipated types. These will include fault tolerance during adaptation and reconfiguration, involving users in recovery, fault tolerance during upgrading and of upgraded systems, coordinated recovery of several interdependent systems (layers, infrastructures, components), etc.

17.2 Industry Vision[33]

From industry's viewpoint the main concern in RTD of future distributed and dependable systems is their complexity. The complexity of present-day systems is becoming a major obstacle to their development and maintenance. Thus, radical approaches are required to render systems that are more designable, deployable, dependable, manageable, maintainable and evolvable. Integrating systems built from separate subsystems is still an open issue. Software architecture, componentisation, aspect-oriented approaches, and virtualisation are general solution spaces but there are fundamental problems to be overcome before complexity management can be considered solved. Moreover the entire software life cycle, ranging from means of empowering software designers and developers, to methodologies for flexible and autonomous fault tolerance and adaptation, should be encompassed in solutions to complexity. Unfortunately many existing programming models are too complex and error-prone to use. The challenge is in developing solutions that while being more clever and sophisticated, are also easier to use. There is a very strong industry feeling that evolution or migration from present systems to new architectures is a difficult and crucial issue, and one which the research community should be addressing more vigorously.

More specifically, the industry view often is that “service-oriented architecture” (SOA) is likely to play a significant role in future systems, but that currently central Web Services technologies (e.g. SOAP, HTTP, WSDL), will not necessarily or exclusively remain at the heart of things. It is becoming clear for many industrial players that CORBA had not been as successful as was initially hoped, primarily due to its perceived complexity, brittleness and poor scalability. It is hoped that current SOA initiatives will do better.

Trust (e.g., based on reputation or third party recommendation) is also highlighted by many industrial parties as a key concept for dealing with security-related aspects of future systems. It is perceived as a more general and tractable way of viewing such concerns than present approaches.

Ensuring dependability of future systems is a serious concern for many industrial parties. Some of the issues which are often mentioned are lack of network reliability and security, insufficient quality of the existing services and components, poor flexibility and adaptivity of the fault tolerance mechanisms, and difficulties in using advanced dependability enabling techniques.

In the area of persistent data management, some industrial organisations (including MicroSoft) are considering an “active data objects” approach, whereby the structuring and grouping of data items (e.g., files structured in directories) should become an increasingly dynamic process—i.e. computed on the basis of meta-data and

[33] This subsection is based on Geoff Coulson’s summary of the discussions hold during a meeting of the CaberNet Links-to-Industry Forum (London, December 3, 2003)

predicates over a flat database. But it is considered that significant research is required to enable such systems in a scaleable wide-area environment.

17.3 Recommendations

Based on this critical analysis of the current RTD of distributed and dependable systems, several recommendations can be made with respect to achieving the vision presented in the document, bearing it in mind that Europe should aim to be a leading world force in the area:

- Industry-academia cooperation. Our analysis clearly shows that more efforts and resources should be devoted to supporting industry-academia cooperation and to fostering successful transfers of knowledge from academia to industry. These two worlds live at a different pace, obey very different rules and measure their success in different ways – and this is how it should remain. Existing research programmes and instruments do not always help to bridge them in the best possible way. Additional support should be given to industry willing to invest in application of recent research results, aware of the fact that its and society's future prosperity lies in developing and using new technologies.

- Young researchers. Young researchers from academia need strong links with industry, business, universities, policy makers, software developers and users, to help in building collaborations of the future. They should gain experience and skills not only in writing research papers but also in preparing short proposals and in marketing research.

- Foundation research. Basic foundation research in the area of distributed and dependable systems should not be neglected in favour of short-term aims. Such research is investment into Europe's future.

- Marketing and promoting best European research. There are many areas in which European researchers, in general, and the CaberNet partners in particular, are carrying out world-class research. CaberNet has had some successes in marketing and promoting the best European research. But more work is needed to make the work of all the CaberNet partners known. More generally, future projects of a similar type should promote open project workshops, participation of leading researchers from outside Europe in project events, and support for partners' closer involvement in international standardisation bodies.

- International research. Focused and direct support (including financial) should be given to organising world-class conferences in Europe. Future programmes should encourage participation of world-class universities from outside Europe.

As the CaberNet project comes to an end, we firmly believe that special attention should be paid to its legacy. Among other things, CaberNet has been extremely successful in:

- supporting young researchers (particularly through its Radicals programme, but also through its short-term visits programme).

- marketing and promoting research conducted by the project members. CaberNet has collected a unique bank of knowledge on state of the art results in RTD and established an infrastructure for disseminating this knowledge via dedicated Internet mechanisms.

The project members are determined to do their best to maintain the tradition of Radicals workshops and the Project infrastructure for as long as possible. It would be highly beneficial if these activities were supported in the future. CaberNet needs to promote its results and the individual results of its members in RTD of distributed and dependable systems more widely with a special focus on transferring them to industry and on emphasising Europe's position and vision in the area.

References

Chapter 2

[Abdellatif 2001] O. Abdellatif-Kaddour, P. Thevenod-Fosse, H. Waeselynck. Adaptation of simulated annealing to property-oriented testing for sequential problems, *2001 International Conference on Dependable Systems and Networks (DSN'2001). Fast abstracts*, Göteborg (Sweden), 1-4 June 2001, pp. B82-B83.

[Abou El Kalam 2003a] A. Abou El Kalam, R. E. Baïda, P. Balbiani, S. Benferhat, F. Cuppens, Y. Deswarte, C. Saurel, G. Trouessin. Security Models and Policies for Health and Social Information and Communication Systems. In *1st French-speaking Conference on Management and Engineering of Hospital Systems (GISEH 2003)*, (Lyon, France), pp. 268-277, 2003 (in French).

[Abou El Kalam 2003b] A. Abou El Kalam, R. El Baïda, P. Balbiani, S. Benferhat, F. Cuppens, Y. Deswarte, A. Miège, C. Saurel, G. Trouessin. Organization Based Access Control. In *4th IEEE Workshop on Policies for Distributed Systems and Networks (POLICY-2003)*, (Como, Italy), pp. 120-131, IEEE CS Press, 2003.

[Aidemark 2002] J. Aidemark, J. Vinter, P. Folkesson, J. Karlsson. Experimental Evaluation of Time-redundant Execution for a Brake-by-wire Application. *International Conference on Dependable Systems and Networks (DSN-2002)*, Washington DC, USA, June 2002.

[AMSD 2003] A dependability roadmap for the information society in Europe. Accompanying Measure System Dependability. IST Project 2001-37553. 3 parts. 2003. http://www.am-sd.org/

[Anderson et al 2003] T. Anderson, M. Feng, S. Riddle, A. Romanovsky. Protective Wrapper Development: A Case Study. In *Proc. 2nd International Conference on COTS-Based Software Systems, ICCBSS 2003*. Ottawa, Canada, February 2003. pp. 1 - 14. LNCS 2580, Springer. 2003.

[Arlat 1999] J. Arlat, Y. Crouzet, Y. Deswarte, J.C. Laprie, D. Powell, P. David, J.L. Dega, C. Rabéjac, H. Schindler, J.F. Soucaille. Fault tolerant computing. *Encyclopedia of Electrical and Electronic Engineering*, Vol.7, Ed. J.G. Webster, Wiley Interscience, ISBN 0471139467, 1999, pp. 285-313.

[Arlat 2000] J. Arlat, J.-P. Blanquart, T. Boyer, Y. Crouzet, M.-H. Durand, J.-C. Fabre, M. Founau, M. Kaâniche, K. Kanoun, P. Le Meure, C. Mazet, D. Powell, F. Scheerens, P. Thévenod-Fosse, H. Waeselynck, *Software components and dependability - integration of COTS*, 159p., Hermès Science, Paris, 2000 (in French).

[Avizienis 2001] A. Avizienis, J.-C. Laprie, B. Randell. Fundamental Concepts of Dependability. Technical Report 739, pp. 1-21, Department of Computing Science, University of Newcastle upon Tyne, 2001.

[Baier 2002a] C. Baier, H. Hermanns, B. Haverkort, J.-P. Katoen. Automated Performance and Dependability Evaluation using Model Checking. In *Performance Evaluation of Complex Systems: Techniques and Tools*, (M. Calzarossa and S. Tucci, Eds.), (Rome, Italy), Lecture Notes in Computer Science, 2459, pp.261-89, Springer, 2002.

[Baier 2002b] C. Baier, J.-P. Katoen, H. Hermanns, B. Haverkort. Simulation for Continuous-time Markov Chains. In *Concurrency Theory (CONCUR)*, (L. Brim, P. Jancar, M. Kretinzki, A. Kucera, Eds.), (Brno, Czech Republic), Lecture Notes in Computer Science, 2421, pp. 338-54, 2002.

[Baxter 2003] G. D. Baxter, K. Tan, S. Newell, P. R. F. Dear, A. Monk. Analysing Requirements for Decision Support in Neonatal Intensive Care. *Archives of Disease in Childhood*, 88 (1), p. .A46, 2003.

[Beder 2001] D.M. Beder, B. Randell, A. Romanovsky, C. M. F. Rubira-Calsavara. On Applying Coordinated Atomic Actions and Dependable Software Architectures for Developing Complex Systems. *Int. Symp. on Object-oriented Real-time Distributed Computing*, Margeburg, Germany, May 2001, IEEE, 4, pp. 103-112, 2001.

[Bell 2001] A. Bell, B. R. Haverkort. Serial and Parallel Out-of-Core Solution of Linear Systems Arising from Generalised Stochastic Petri Net Models. *High Performance Computing 2001*, Seattle, USA, April 22-26, 2001

[Besnard 2003] D. Besnard. Building Dependable Systems with Fallible Humans. In *5th CaberNet Open Workshop*, Porto Santo, Portugal, November 5-7, 2003.

[Besnard and Arief 2003] D. Besnard, B. Arief. Computer security impaired by legitimate users. To appear in *Journal of Computers & Security*.

[Besnard et al 2003] D. Besnard, D. Greathead, G Baxter. When mental models go wrong. Co-occurrences in dynamic, critical systems. To appear in International Journal of Human-Computer Interaction, 2003.

[Besnard and Greathead 2003] D. Besnard, D. Greathead. A cognitive approach to safe violations. To appear in *Cognition, Technology & Work*, 2003.

[Betous-Almeida 2002] C. Betous-Almeida, K. Kanoun. Stepwise Construction and Refinement of Dependability Models. *2002 International Conference on Dependable Systems & Networks (DSN'2002)*, Washington (USA), 23-26 June 2002, pp. 515-526 .

[Bohnenkamp 2003] H. Bohnenkamp, P. van der Stok, H. Hermanns, F. Vaandrager. Cost-Optimisation of the IPv4 Zeroconf Protocol. In *Int. Symp. on Dependable Systems and Networks (DSN 2003)*, (San Francisco, CA, USA), pp. 531-540, IEEE CS Press, 2003.

[Bondavalli 1998] A. Bondavalli, F. D. Giandomenico, F. Grandoni, D. Powell,, C. Rabéjac. State Restoration in a COTS-based N-Modular Architecture. *1st Int. Symp. on Object-Oriented Real-Time Distributed Computing (ISORC'98)*, Kyoto, Japan, 1998, pp. 174-183.

[Bondavalli 2000a] A. Bondavalli, S. Chiaradonna, F. Di Giandomenico, F. Grandoni. Threshold-Based Mechanisms to Discriminate Transient from Intermittent Faults. *IEEE Transactions on Computers*, 49(3) March 2000, pp. 230-245.

[Bondavalli 2000b] A. Bondavalli, I. Mura, S. Chiaradonna, R. Filippini, S. Poli, F. Sandrini. DEEM: a Tool for the Dependability Modeling and Evaluation of Multiple Phased Systems. *International Conference on Dependable Systems and Networks (DSN2000)*, New York, NY, USA, IEEE Computer Society Press. June 2000, pp. 231-236.

[Bondavalli 2001] A. Bondavalli, M. Dal Cin, D. Latella, I. Majzik, A. Pataricza, G. Savoia. Dependability Analysis in the Early Phases of UML Based System Design. *Journal of Computer Systems Science and Engineering*, Vol. 16, pp. 265-275, 2001.

[Buchacker 2001] K. Buchacker, V. Sieh. Framework for Testing the Fault-Tolerance of Systems Including OS and Network Aspects. *High-Assurance System Engineering Symposium (HASE 2001)*, IEEE, Boca Raton, Florida, 2001, pp. 95-105.

[Buchacker 2003] K. Buchacker, M. D. Cin, H. J. Höxer, R. Karch, V. Sieh, O. Tschäche. Reproducible Dependability Benchmarking Experiments Based on Unambiguous Benchmark Setup Descriptions. *Int. Conf. on Dependable Systems and Networks (DSN 2003)*, (San Francisco, CA, USA), pp.469-78, IEEE CS Press, 2003.

[Carreira 1998] J. Carreira, H. Madeira, J. Gabriel Silva. Xception: A Technique for the Experimental Evaluation of Dependability in Modern Computers. *IEEE Transactions on Software Engineering*, 24(2): 125-136 (1998)

[Casimiro 2002] A. Casimiro, P. Veríssimo, "Generic Timing Fault Tolerance using a Timely Computing Base", *International Conference on Dependable Systems and Networks (DSN 2002)*, Washington D.C., USA, June 2002

[Chevalley 2001a] P. Chevalley, P. Thévenod-Fosse, "An empirical evaluation of statistical testing designed from UML state diagrams : the flight guidance system case study", *12th International Symposium on Software Reliability Engineering (ISSRE'2001)*, Hong Kong, 27-30 November 2001, pp. 254-263.

[Chevalley 2001b] P. Chevalley, "Applying mutation analysis for object-oriented programs using a reflective approach", *8th Asia-Pacific Software Engineering Conference (APSEC 2001)*, Macau (Chine), 4-7 December 2001, pp. 267-270.

[Clarke et al 2003] K. Clarke, J. Hughes, D. Martin, M. Rouncefield, I. Sommerville, C. Gurr, M. Hartswood, R. Procter, R. Slack, A. Voss,. Dependable red hot action. In K. Kuutti, E. H. Karsten, G. Fitzpatrick, P. Dourish, & K. Schmidt, (Eds.) Proceedings of the *European Conference on Computer Supported Cooperative Work*, Helsiniki, Finland, pp. 61-80, 2003.

[Correia 2002] M. Correia, P. Veríssimo, N. F. Neves, "The Design of a COTS Real-Time Distributed Security Kernel", *4th European Dependable Computing Conference*, Toulouse, France, October 2002.

[Cukier 1999] M. Cukier, D. Powell, J. Arlat, "Coverage estimation methods for stratified fault-injection", *IEEE Transactions on Computers*, 48, 7, pp. 707-723, July 1999.

[David 2003] P. David, H. Waeselynck (Eds.), *Open Source Software and Dependability*, 234p., Hermès Science, Paris, 2003 (in French).

[de Lemos et al 2003] R. de Lemos, C. Gacek, A. Romanovsky. Architectural Mismatches Tolerance. *Architecting Dependable Systems*. Lecture Notes in Computer Science 2677. Springer. Berlin, Germany. 2003. pp. 175-193.

[Dearden 2000] A. Dearden, M. Harrison, P. Wright, "Allocation of Function: Scenarios, Context and the Economics of Effort", *International Journal of Human-Computer Studies*, 52, pp.289-318, 2000.

[Denaro 2003] G. Denaro, L. Lavazza, M. Pezze, "An Empirical Evaluation fo Objetc Oriented Metrics in Industrial Setting", in *5th CaberNet Open Workshop*, Porto Santo, Portugal, November 5-7, 2003.

[Deswarte 1991] Y. Deswarte, L. Blain, J.-C. Fabre, "Intrusion Tolerance in Distributed Systems", in *Symp on Research in Security and Privacy*, (Oakland, CA, USA), pp.110-21, IEEE Computer Society Press, 1991.

[Deswarte 2001] Y. Deswarte, N. Abghour, V. Nicomette, D. Powell, "An Internet Authorization Scheme using Smart Card-Based Security Kernels", in *International Conference on Research in Smart Card (e-Smart 2001)*, (I. Attali and T. Jensen, Eds.), (Cannes, France), Lecture Notes in Computer Science, 2140, pp. 71-82, 2001.

[Deswarte 2003] Y. Deswarte, J.-J. Quisquater, A. Saidane, "Remote Integrity Checking — How to Trust Files Stored on Untrusted Servers", in *6th IFIP TC-11 WG 11.5 Working Conference on Integrity and Internal Control in Information Systems (IICIS 2003)*, (Lausanne, Switzerland), Kluwer Academic Publishers, 2003.

[Dobson 1986] J. E. Dobson, B. Randell, "Building Reliable Secure Systems out of Unreliable Insecure Components", in *Conf on Security and Privacy*, (Oakland, CA, USA), pp. 187-193, IEEE Computer Society Press, 1986.

[Elmenreich 2002] W. Elmenreich, P. Peti, "Achieving Dependability in Time-Triggered Networks by Sensor Fusion", *6th IEEE International Conference on Intelligent Engineering Systems (INES)*, May 2002, Opatija, Croatia

[Essamé 1999] D. Essamé, J. Arlat, D. Powell, "PADRE : A Protocol for Asymmetric Duplex Redundancy", *7th IFIP International Working Conference on Dependable Computing for Critical Applications (DCCA-7)*, San Jose (USA), 6-8 January 1999, pp. 213-232

[Fabre 1999] J.C. Fabre, F. Salles, M. Rodriguez, J. Arlat, "Assessment of COTS microkernels by fault injection", *7th IFIP International Working Conference on Dependable Computing for Critical Applications (DCCA-7)*, San Jose (USA), 6-8 January 1999, pp. 19-38

[Fota 1999] N. Fota, M. Kaâniche, K. Kanoun, "Incremental approach for building stochastic Petri nets for dependability modelling", *Statistical and Probabilistic Models in Reliability*, Eds. D.S. Ionescu, N. Limnios, ISBN 0-8176-4068-1, Birkhauser, 1999, pp. 321-335

[Fraga 1985] J. Fraga, D. Powell, "A Fault and Intrusion-Tolerant File System", in *IFIP 3rd Int Conf on Computer Security*, (J. B. Grimson, H.-J. Kugler, Eds.), (Dublin, Ireland), Computer Security, pp.203-18, Elsevier Science Publishers B.V. (North-Holland), 1985.

[Haverkort 2001] B.R. Haverkort, R. Harper, "Performance and Dependability Modelling Techniques and Tools", special issue of *Performance Evaluation*, Volume 44, Issues 1-4, 2001

[Haverkort 2002] B.R. Haverkort, L. Cloth, H. Hermanns, J. P. Katoen, C. Baier, "Model checking performability properties", *IEEE Int'l Conference on Dependable Systems and Networks*, June 2002 , pp. 103-112

[Hollnagel 1987] E. Hollnagel, Information and reasoning in intelligent decision support systems. *International Journal of Man-Machine Studies*, 27, pp. 665-678, 1987.

[Höxer 2002] H.-J. Höxer, V. Sieh, V., K. Buchacker, "UMLinux - A Tool for Testing a Linux System's Fault Tolerance", *LinuxTag 2002*, Karlsruhe, Germany, June 6.-9. 2002.

[ISTAG 2001] ISTAG, "Scenarios for Ambient Intelligence in 2010", Final Report, February 2001.

[Kaâniche 2001] M. Kaâniche, K. Kanoun, M. Rabah, "A framework for modeling availability of e-business systems", *10th IEEE International Conference on Computer Communications and Networks (IC3N'2001)*, Scottdale (USA), 15-17 octobre 2001, pp. 40-45

[Kaâniche 2002] M. Kaâniche, J.-C. Laprie, J.-P. Blanquart, "A Framework for Dependability Engineering of Critical Computing Systems", *Safety Science*, 9 (40), pp.731-52, December 2002.

[Kaâniche 2003] M. Kaâniche, K. Kanoun, M. Martinello, "User-Perceived Availability of a Web-based Travel Agency", in *International Conference on Dependable Systems and Networks (DSN'2003). International Performance and Dependability Symposium*, (San Francisco, CA, USA), pp. 709-718, IEEE CS Press, 2003.

[Kanoun 1999] K. Kanoun, M. Borrel, T. Morteveille, A. Peytavin, "Availability of CAUTRA, a subset of the French air traffic control system", *IEEE Transactions on Computers*, 48, 5, pp.528-535, May, 1999

[Kanoun 2002] K. Kanoun, H. Madeira, J. Arlat, "A Framework for Dependability Benchmarking", in *Supplement of the 2002 Int. Conf. on Dependable Systems and Networks (DSN-2002)*, (Washington D.C., USA), pp. F7-F8, 2002.

[Kemme 2003] B. Kemme, F. Pedone, G. Alonso, A. Schiper, M. Wiesmann, "Using Optimistic Atomic Broadcast in Transaction Processing Systems", *IEEE Transactions on Knowledge and Data Engineering*, 15 (3), pp. 1018-32, July 2003.

[Killijian 2000] M.O. Killijian, J.C. Fabre, "Implementing a reflective fault-tolerant CORBA system", *19th IEEE Symposium on Reliable Distributed Systems (SRDS 2000)*, Nuremberg (Germany), 16-18 October 2000, pp.154-163

[Lawrie 2002] T. Lawrie, C. Gacek, "Issues of Dependability in Open Source Software Development", *Software Engineering Notes*, 27 (03), pp. 34-38, May 2002.

[Leeman 2003] M. Leeman, M. Leeman, V. D. Florio, G. Deconinck, "A Flexible Library for Dependable Master-Worker Parallel Programs," in *11th Euromicro Workshop on Parallel and Distributed Processing (PDP2003)*, 2003.

[Littlewood 2000] B. Littlewood, P. Popov, L. Strigini, "Assessment of the Reliability of Fault-Tolerant Software: a Bayesian Approach", *19th International Conference on Computer Safety, Reliability and Security (SAFECOMP'2000)*, Rotterdam, the Netherlands, Springer, 2000

[Littlewood 2001a] B. Littlewood, P. Popov, L. Strigini, "Design Diversity: an Update from Research on Reliability Modelling", in *Safety-Critical Systems Symposium*, (Bristol, U.K.), Springer, 2001.

[Littlewood 2001b] B. Littlewood, P. Popov, L. Strigini, "Modelling Software Design Diversity - a Review", *ACM Computing Surveys*, 33 (2), pp.177-208, June 2001.

[Littlewood 2002] B. Littlewood, P. Popov, L. Strigini, "Assessing the Reliability of Diverse Fault-Tolerant Software-Based Systems", *Safety Science*, 40, pp. 781-796, 2002.

[Lung 2003] L. C. Lung, M. Correia, N. F. Neves, P. Veríssimo, "A Simple Intrusion-Tolerant Reliable Multicast Protocol using the TTCB", in *21º Simpósio Brasileiro de Redes de Computadores*, (Natal, Brazil), 2003.

[Marsden 2001] E. Marsden, J.C. Fabre, "Failure mode analysis of CORBA service implantations", *IFIP/ACM International Conference on Distributed Systems Platforms*, Heidelberg (Germany), 12-16 November 2001

[Marsden 2002] E. Marsden, J.-C. Fabre, J. Arlat, "Dependability of CORBA Systems: Service Characterization by Fault Injection", in *21st IEEE Symposium on Reliable Distributed Systems (SRDS'2002)*, (Osaka, Japan), pp.276-85, IEEE CS Press, 2002.

[Meling 2002] H. Meling, A. Montresor, O. Babaoglu, B. E. Helvik. Jgroup/ARM: A Distributed Object Group Platform with Autonomous Replication Management for Dependable Computing, University of Bologna, Italy, Technical Report UBLCS 2002-12, October 2002.

[Mena 2003] S. Mena, A. Schiper, P. T. Wojciechowski, "A Step Towards a New Generation of Group Communication Systems", in *Middleware 2003*, 2003.

[Mersiol 2002] M. Mersiol, J. Arlat, D. Powell, A. Saidane, H. Waeselynck, C. Mazet, "FAST: a Prototype Tool for Supporting the Engineering of Socio-technical Systems", in *3rd European Systems Engineering Conference*, (Toulouse, France), pp. 33-40, 2002.

[METT 1993] Ministere de l'Equipement, des Transports et du Tourisme, *Rapport de la commission d'enquête sur l'accident survenu le 20 Janvier 1992 près du Mont Sainte Odile (Bas-Rhin) à l'airbus A.320 immatriculé F-GGED exploité par la compagnie Air Inter*, 1993.

[Montresor 2001] A. Montresor, R. Davoli, O. Babaoglu, "Enhancing Jini with Group Communication", in *ICDCS Workshop on Applied Reliable Group Communication (WARGC 2001)*, (Phoenix, AZ, USA), 2001 (Also appears as Technical Report UBLCS 2000-16, December 2000, Revised January 2001).

[Montresor 2002] A. Montresor, H. Meling, O. Babaoglu, "Towards Adaptive, Resilient and Self-Organizing Peer-to-Peer Systems". *Proceedings of 1st International Workshop on Peer-to-Peer Computing*, Pisa, Italy, May 2002

[Mostéfaoui 2001] A. Mostéfaoui, S. Rajsbaum, M. Raynal, "Conditions on Input Vectors for Consensus Solvability in Asynchronous Distributed Systems", *33rd Annual ACM Symposium on Theory of Computing*, July 6-8, 2001, Heraklion, Crete, Greece. ACM, 200

[Mura 2001] I. Mura, A. Bondavalli, "Markov Regenerative Stochastic Petri Nets to Model and Evaluate the Dependability of Phased Missions", *IEEE Transactions on Computers*, 50, 12, December 2001, pp.1337-1351

[Nestmann 2003] U. Nestmann, R. Fuzzati, M. Merro, "Modeling Consensus in a Process Calculus", in *CONCUR 2003*, Springer, 2003.

[Pinho 2002] L. Pinho, F. Vasques, "Transparent Environment for Replicated Ravenscar Applications", *7th International Conference on Reliable Software Technologies - Ada-Europe 2001*, Vienna, Austria. June 2002.

[PITAC 1999] PITAC, "Information technology research: investing in our future", Report to the President, February 1999.

[Popov 2002a] P. Popov, "Reliability Assessment of Legacy Safety-Critical Systems Upgraded with Off-the-Shelf Components", in *SAFECOMP'2002*, (Catania, Italy), Springer, 2002.

[Popov 2002b] P. Popov, L. Strigini, J. May, S. Kuball, "Estimating Bounds on the Reliability of Diverse Systems", *IEEE Transactions on Software Engineering* 2002.

[Powell 1988] D. Powell, G. Bonn, D. Seaton, P. Veríssimo, F. Waeselynck, "The Delta-4 Approach to Dependability in Open Distributed Computing Systems", *18th IEEE Int. Symp. on Fault-Tolerant Computing Systems (FTCS-18)*, Tokyo, Japan, 1988, pp. 246-251.

[Powell 1999] D. Powell, J. Arlat, L. Beus-Dukic, A. Bondavalli, P. Coppola, A. Fantechi, E. Jenn, C. Rabejac, A. Wellings, "GUARDS : a generic upgradable architecture for real-time dependable systems", *IEEE Transactions on Parallel and Distributed Systems*, 10, 6, pp. 580-599, June 1999

[Powell 2001] D. Powell (Ed.), *A Generic Fault-Tolerant Architecture for Real-Time Dependable Systems*, Kluwer Academic Publishers, N°ISBN 0-7923-7295-6, 2001, 242 p.

[Powell 2002] D. Powell, "Carnasie Line, the 'French Touch' under Broadway — Safety of the New York subway", CNRS Info, no. 401, pp .17-18, 2002 (in French).

[Rodriguez 2000] M. Rodriguez, J. C. Fabre, J. Arlat, "Formal specification for building robust real-time microkernels", *21st IEEE Real-Time Systems Symposium (RTSS2000)*, Orlando (USA), 27-30 November 2000, pp. 119-128

[Rodriguez 2002] M. Rodriguez, A. Albinet, J. Arlat, "MAFALDA-RT: a tool for dependability assessment of real-time systems", *2002 International Conference on Dependable Systems & Networks (DSN'2002)*, Washington (USA), 23-26 June 2002, pp. 267-272

[Rodriguez 2003] M. Rodriguez, J.-C. Fabre, J. Arlat, "Building SWIFI Tools from Temporal Logic Specifications", in *International Conference on Dependable Systems and Networks (DSN'2003). Dependable Computing and Communications Symposium*, (San Francisco, CA, USA), pp. 95-104, IEEE CS Press, 2003.

[Ruiz-Garcia 2001] J.C. Ruiz-Garcia, P. Thévenod-Fosse, J. C. Fabre, "A strategy for testing MetaObject Protocols in reflective architectures", *2001 International Conference on Dependable Systems and Networks (DSN'2001)*, Göteborg (Sweden), 1-4 July 2001, pp. 327-336

[Saïdane 2003] A. Saïdane, Y. Deswarte, V. Nicomette, "An Intrusion Tolerant Architecture for Dynamic Content Internet Servers", in *First ACM Workshop on Survivable and Self-Regenerative Systems (SSRS'03)*, (Fairfax, VA, USA), 2003.

[Simache 2001] C. Simache, M. Kaâniche, "Measurement-based availability analysis of Unix systems in a distributed environment", *12th International Symposium on Software Reliability Engineering (ISSRE'2001)*, Hong Kong, 27-30 November 2001, pp. 346-355

[Simache 2002] C. Simache, M. Kaâniche, A. Saidane, "Event Log Based Dependability Analysis of Windows NT and 2K Systems", in *Pacific Rim International Symposium on Dependable Computing (PRDC'2002)*, (Tsukuba, Japan), pp. 311-15, IEEE CS Press, 2002.

[Sommerville 2003] I. Sommerville, D. Martin, M. Rouncefield, "Informing the RE Process with Patterns of Cooperative Interaction", *International Arab Journal of Information Technology*, 1 (1) 2003.

[Steiner 2002] W. Steiner, M. Paulitsch, "The Transition from Asynchronous to Synchronous System Operation: An Approach for Distributed Fault-Tolerant Systems", *International Conference on Distributed Computing Systems (ICDCS 2002)*, Vienna, Austria, July 2-5, 2002.

[Tan 2003] K. Tan, G.D. Baxter, K. G. Brownlee, S. J. Newell, P. R. F. Dear, S. Smye, "Fuzzy Logic Expert System for Ventilation of the Newborn Infant", *Archives of Disease in Childhood*, 88 (1), p. A47, 2003.

[Tartanoglu 2003] F. Tartanoglu, V. Issarny, A. Romanovsky, N. Levy. Coordinated Forward Error Recovery for Composite Web Services. In *22nd Symposium on Reliable Distributed Systems (SRDS*). Florence, Italy. 2003. pp. 167-176

[Urbán 2001] P. Urbán, X. Défago, A. Schiper, "Neko: A Single Environment to Simulate and Prototype Distributed Algorithm". In *15th Int. Conf. on Information Networking (ICOIN-15)*, (Beppu City, Japan), 2001.

[Valdes 2002] A. Valdes, M. Almgren, S. Cheung, Y. Deswarte, B. Dutertre, J. Levy, H. Saidi, V. Stavridou, T. E. Uribe. An Architecture for an Adaptive Intrusion Tolerant Server. In *Security Protocols Workshop*, (Cambridge, UK), 2002.

[Verissimo 2002] P. Verissimo, A. Casimiro. The Timely Computing Base Model and Architecture. *IEEE Transactions on Computers*, Special Issue on Asynchronous Real-Time Systems. 51, 8, Aug 2002

[Vieira 2003] M. Vieira, H. Madeira. Benchmarking the Dependability Benchmark of Different OLTP Systems. In *Int. Conf. on Dependable Systems and Networks (DSN 2003)*, (San Francisco, CA, USA), pp. 305-130, IEEE CS Press, 2003.

[Whitten and Tikkar 1999] A. Whitten, J. D. Tygar. Why Johnny can't encrypt: A usability evaluation of PGP 5.0, *Proceedings of 9th USENIX security symposium*, Washington DC, USA 1999.

[Xu et al 2002] J. Xu, B. Randell, A. Romanovsky, R. J. Stroud, A. F. Zorzo, E. Canver, F. von Henke. Rigorous development of an Embedded Fault-Tolerant System Based on Coordinated Atomic Actions. IEEE Transactions on Computers (Special Issue on Fault Tolerance), 51, 2, pp. 164-179. 2002.

[Zarras 2001] A. Zarras, V. Issarny. Automating the Performance and Reliability Analysis of Enterprise Information Systems. *16th IEEE International Conference on Automated Engineering (ASE2001)*, pp. 350-354, November, 2001, San Diego CA, USA. LNCS 2218, Middleware 2001, Springer, 2001, pp. 216-231.

Chapter 3

[Abadi and Jurjens 2001] M. Abadi, J. Jurjens. Formal eavesdropping and its computational interpretation. In Fourth International Symposium on Theoretical Aspects of Computer Software (TACS2001), LNCS. Springer-Verlag, 2001.

[Abadi and Rogaway 2002] M. Abadi, Ph. Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). Journal of Cryptology, 15(2), pp. 103–127, 2002.

[Adelsbach et al 2002] A. Adelsbach, D. Alessandri, C. Cachin, S. Creese, Y. Deswarte, K. Kursawe, J. C. Laprie, D. Powell, B. Randell, J. Riordan, P. Ryan, W. Simmonds, R. Stroud, P. Verissimo, M. Waidner, A. Wespi. Conceptual Model and Architecture of MAFTIA. Project MAFTIA IST-1999-11583 deliverable D21. 2002 http://www.research.ec.org/maftia/deliverables/D21.pdf.

[Agat 2000] J. Agat. Transforming out timing leaks. In 27th Principles of programming languages (POPL), pp. 40–53, Boston, Massachusetts, Jan 2000. ACM Press, New York.

[Alur et al 1998] R. Alur, T. Henzinger, F. Mang, S. Qadeer, S. Rajamani, S. Tasiran. MOCHA: Modularity in model checking. In Computer Aided Verification, pp. 521–525, 1998.

[Alvisi et al 2000] L. Alvisi, D. Malkhi, E. Pierce, M. K. Reiter, R. N. Wright. Dynamic Byzantine quorum systems. In: Proceedings of the IEEE International Conference on Dependable Systems and Networks. (2000) pp. 283-292.

[Anderson 2001] R. J. Anderson. Security Engineering: A guide to building dependable distributed systems. John Wiley & Sons Inc, New York, 2001.

[Ateniese et al 2000] G.Ateniese, M. Steiner, G. Tsudik. New multi-party authentication services and key agreement protocols. IEEE J. of Selected Areas on Communications 18, 2000

[Avizienis 2001] A. Avizienis, J.-C. Laprie, B. Randell. Fundamental Concepts of Dependability. Technical Report 739, pp. 1-21, Department of Computing Science, University of Newcastle upon Tyne, 2001.

[Bellare 1997] M. Bellare. Practice-oriented provable security. In Information Security, First International Workshop, ISW ’97, LNCS 1396, pp. 221–231, 1997.

[Bellare and Kohno 2003] M. Bellare, T. Kohno. A theoretical treatment of related-key attacks: Rka-prps, rka-prfs, and applications. In EUROCRYPT 2003, LNCS 2656, pp. 491–506, 2003.

[Biham and Shamir 1997] E. Biham, A. Shamir. Differential fault analysis of secret key cryptosystems. In B. S. Kaliski Jr., editor, 17th Advances in Cryptology (CRYPTO), volume LNCS 1294, pp. 513–525, Santa Barbara, California, Aug 1997. Springer-Verlag, Berlin.

[Black et al 2002] J. Black, P. Rogaway, T. Shrimpton. Encryption scheme security in the presence of key-dependent messages. In Selected Areas in Cryptography — SAC’02, LNCS. Springer-Verlag, 2002.

[Burrows et al 1990] M. Burrows, M. Abadi, R. Needham. A logic of authentication. ACM Transactions on Computer Systems, 8(1), pp. 18–36, 1990.

[Canvel et al 2003] B. Canvel, A. Hiltgen, S. Vaudenay, M. Vuagnoux. Password interception in a SSL/TLS channel. In D. Boneh, Ed., Advances in Cryptology (CRYPTO), Santa Barbara, California, Aug 2003. Springer-Verlag, Berlin.

[Castro and Liskov 1999] M. Castro, B. Liskov. Practical {Byzantine} fault tolerance. In: Proceedings of the Third Symposium on Operating Systems Design and Implementation. (1999)

[Chari et al 1999] S. Chari, C. Jutla, J. R. Rao, P. Rohatgi. A cautionary note regarding evaluation of AES candidates on Smart-Cards. In 2nd Advanced Encryption Standard (AES) Candidate Conference, Rome, Italy, 1999.

[Deswarte 1991] Y. Deswarte, L. Blain, J.-C. Fabre, "Intrusion Tolerance in Distributed Systems", in *Symp on Research in Security and Privacy*, (Oakland, CA, USA), pp.110-21, IEEE Computer Society Press, 1991.

[Dobson and Randell 1986] J. E. Dobson, B. Randell, "Building Reliable Secure Systems out of Unreliable Insecure Components", in *Conf on Security and Privacy*, (Oakland, CA, USA), pp. 187-193, IEEE Computer Society Press, 1986.

[Dolev and Yao 1983] D. Dolev, A. C. Yao. On the security of public key protocols. IEEE Transactions on Information Theory, 29(2), pp. 198–208, 1983.

[English and Hamilton 1996] E. English, S. Hamilton. Network security under siege: the timing attack. IEEE Computer, 29(3):95–97, Mar 1996.

[Fraga 1985] J. Fraga, D. Powell, "A Fault and Intrusion-Tolerant File System", in *IFIP 3rd Int Conf on Computer Security*, (J. B. Grimson, H.-J. Kugler, Eds.), (Dublin, Ireland), Computer Security, pp.203-18, Elsevier Science Publishers B.V. (North-Holland), 1985.

[Goldreich 1997] O. Goldreich. On the foundations of modern cryptography. Lecture Notes in Computer Science, 1294. pp. 46–74, 1997.

[Handschuh and Heys 1998] H. Handschuh, H. M. Heys. A timing attack on RC5. In Selected Areas in Cryptography (SAC), LNCS 1556, pp. 306–318, Kingston, Canada, Aug 1998. Springer-Verlag, Berlin.

[Herrmann et al 2002] P. Herrmann, L. Wiebusch, H. Krumm. State-Based Security Policy Enforcement in Component-Based E-Commerce Applications. In: 2nd IFIP Conference on E-Commerce, E-Business & E-Government (I3E), pp. 195-209, Lisbon, October 2002. Kluwer Academic Publisher.

[Herrmann 2003a] P. Herrmann. Formal Security Policy Verification of Distributed Component-Structured Software. To appear in: 23rd IFIP International Conference on Formal Techniques for Networked and Distributed Systems (FORTE'2003), IFIP, LNCS, Berlin, September/October 2003, Springer-Verlag.

[Herrmann 2003b] P. Herrmann. Trust-Based Protection of Software Component Users and Designers. In: 1st International Conference on Trust Management, pp. 75-90, LNCS 2692, Heraklion, May 2003, Springer-Verlag.

[Hevia and Kiwi. 1998] A. Hevia, M. Kiwi. Strength of two data encryption standard implementations under timing attacks. In C. L. Lucchesi, A. V. Moura, Eds, 3rd Latin American Symp. on Theoretical Informatics (LATIN), pp. 192–205, Campinas, Brazil, Apr 1998. Springer-Verlag, Berlin.

[Hiltunen et al 2001] M. Hiltunen, R. Schlichting, C. A. Ugarte. Enhancing survivability of security services using redundancy. In: Proceedings of the IEEE International Conference on Dependable Systems and Networks, pp. 173-182, 2001.

[Kihlstrom et al 2001] K. P. Kihlstrom, L. E. Moser, P. M. Melliar-Smith. The Secure Ring group communication system. ACM Transactions on Information and System Security 4 pp. 371—406, 2001.

[Kocher 1996] P. C. Kocher. Timing attacks on implementations of Diffe-Hellman, RSA, DSS and other systems. In N. Koblitz, editor, 16th Advances in Cryptology (CRYPTO), LNCS 1109, pp. 104–113, Santa Barbara, California, Aug 1996. Springer-Verlag.

[Laud 2001] P. Laud. Semantics and program analysis of computationally secure information flow. In Programming Languages and Systems: 10th European Symposium on Programming, ESOP 2001 Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2001 Genoa, Italy, April 2-6, 2001, volume 2028 of LNCS, pp. 77–91. Springer-Verlag, 2001.

[Laud 2002] P. Laud. Encryption cycles and two views of cryptography. In NORDSEC 2002 - Proceedings of the 7th Nordic Workshop on Secure IT Systems (Karlstad University Studies 2002:31), pp. 85–100, 2002.

[Laud and Corin. 2003] P. Laud, R. Corin. Sound computational interpretation of formal encryption with composed keys. In Information Security and Cryptology - ICISC 2003, 6th International Conference, LNCS. Springer-Verlag, 2003.

[Lazic 1999] R .S. Lazic. *A Semantic Study of Data Independence with Applications to Model Checking*, DPhil thesis, Oxford University Computing Laboratory, April 1999.

[Lowe 1997] G. Lowe. Casper: A compiler for the analysis of security protocols. In 10th Computer Security Foundations Workshop (CSFW), pp. 18–30, Rockport, Massachusetts, Jun 1997. IEEE CS Press.

[Lück et al 1999] I. Lück, M. Schönbach, A. Mester, H. Krumm. Derivation of Backup Service Management Applications from Service and System Models. In R. Stadler, B. Stiller (eds.), Active Technologies for Network and Service Management, Proc. DSOM'99, pp. 243-255, Springer-Verlag, LNCS 1700, 1999.

[Lück et al 2001] I. Lück, C. Schäfer, H. Krumm. Model-based Tool-Assistance for Packet-Filter Design. In: M. Sloman, E. Lupu, J. Lobo (Eds.), Proceedings of the IEEE Workshop on Policies for Distributed Systems and Networks (Policy 2001), pp. 120-136, Bristol, UK, LNCS 1995, 2001. Springer-Verlag.

[Lück et al 2002] I. Lück, S. Vögel, H. Krumm. Model-Based Configuration of VPNs. In: 8th IEEE/IFIP Network Operations and Management Symposium (NOMS2002), pp. 589-602, Florence, April 2002. IEEE Computer Society Press.

[Malkhi et al 2001] D. Malkhi, M.K. Reiter, D. Tulone, E. Ziskind. Persistent objects in the Fleet system. In: Proceedings of the 2nd {DARPA} Information SurvivabilityConference and Exposition (DISCEX II). 2001.

[May et al 2001] D. May, H. Muller, N. P. Smart. Non-Deterministic processors. In V. Varadharajan and Y. Mu, Eds, 6th Australasian Conf. (ACISP) – Information Security and Privacy, volume LNCS 2119, pp. 115–129, Sydney, Australia, Jul 2001. Springer-Verlag. 2001.

[Micciancio and Warinschi 2003] D. Micciancio, B. Warinschi. Completeness theorems for the abadi-rogaway language of encrypted expressions. To appear. 2003.

[Paulson 1998] L. Paulson. The inductive approach to verifying cryptographic protocols. Journal of Computer Security, 6, pp. 85–128, 1998.

[Pfitzmann et al 2000] B. Pfitzmann, M. Schunter, M. Waidner. Cryptographic Security of Reactive Systems, Notes in Theoretical Computer Science (ENTCS), March 2000. http://www.elsevier.nl/cas/tree/store/tcs/free/noncas/pc/menu.hym.

[Reiter 1995] M.K.Reiter. The {Rampart} toolkit for building high-integrity services. In: Theory and Practice in Distributed Systems. LNCS 938, Springer Verlag, 1995.

[Ryan et al 2000] P. Y. A. Ryan, S. A. Schneider, A. W. Roscoe, G. Lowe, M. H. Goldsmith. Modelling and Analysis of Security Protocols, Pearson Education 2000.

[Schunter 2000] M. Schunter. SEMPER: Secure Electronic Marketplace for Europe. In LNCS 1854. Springer Verlag. 2000.

[Verissimo et al 2003] P. E. Veríssimo, N. F. Neves, M. P. Correia. Intrusion-Tolerant Architectures: Concepts and Design. In: Architecting Dependable Systems. LNCS 2677, Springer Verlag, 2003.

Chapter 4

[Aldea Rivas and Gonzalez Harbour 2002a] M. Aldea Rivas, M. Gonzalez Harbour. “Application-Defined Scheduling in Ada”, IRTAW 11, Ada Letters, XXII (4), pp. 77-84. 2002.

[Aldea Rivas and Gonzalez Harbour 2002b] M. Aldea Rivas, M. Gonzalez Harbour. POSIX-Compatible Application-Defined Scheduling in MaRTE OS. Proceedings of 14th Euromicro Conference on Real-Time Systems, Vienna, Austria, IEEE Computer Society Press, June 2002

[Allen et al 1983] J. Allen, K. Kennedy, C. Porterfield, J. Warren. Conversion of Control Dependence to Data Dependence. Proceedings of the 10th ACM Symposium on Principles of Programming Languages. pp. 177-189, 1983.

[Anantaraman et al 2003] A. Anantaraman, K. Seth, K. Patil, E. Rotenberg, F. Mueller. Virtual Simple Architecture (VISA): Exceeding the Complexity Limit in Safe Real-Time Systems, Proceedings of the International Symposium on Computer Architecture, 2003.

[Aydin et al 2001] H. Aydin, R. Melhem, D.Mossé and Pedro Mejia Alvarez ``Determining Optimal Processor Speeds for Periodic Real-Time Tasks with Different Power Characteristics", ECRTS01 (Euromicro Conference on Real-Time Systems), Delft, Holland, 2001

[Bauer et al 2001] G. Bauer, H. Kopetz, P. Puschner. Assumption Coverage under Different Failure Modes in the Time-Triggered Architecture. In: Proceedings of the 8th IEEE International Conference on Emerging Technologies and Factory Automation, pp.333-341, 2001.

[Kopetz 1993] Kopetz, H, and Reisinger, J. The Non-Blocking Write Protocol NBW: A Solution to a Real-Time Synchronization Problem. In Proceedings of the 14th IEEE Real-Time Systems Symposium, p.131-137, 1993.

[Kopetz 1997] Kopetz, H. Real-Time Systems: Design Principles for Distributed Embedded Applications. Kluwer Academic Publishers, 1997.

[Kopetz 1998] Kopetz, H. The Time-Triggered (TT) Model of Computation. In Proceedings of the 19th IEEE Real-Time Systems Symposium, p.168-177, 1998.

[Kopetz 2000] Kopetz, H. Software Engineering for Real-Time: A Roadmap. In Proceedings of the 22nd International Conference on Software Engineering, p.201-211, 2000.

[Kopetz 2003] Kopetz, H, and Bauer, G. The Time-Triggered Architecture. Proceedings of the IEEE, Volume 91, Number 1, p.112-126, 2003.

[TTTech] TTTech Computertechnik AG. Homepage of TTTech: www.tttech.com.

[Bernat and Burns 2001] G. Bernat, A. Burns. "Implementing a Flexible Scheduler in Ada", LNCS, 2043, pp. 179-190. 2001

[Bernat et al 2002] G. Bernat, A. Colin, S. Petters, Analysis of Probabilistic Hard Real-Time Systems. Proceedings of the IEEE Real-Time Systems Symposium, 2002.

[Bollella et al 2000] G. Bollella, B. Brosgol, P. Dibble, S. Furr, J., Gosling, D. Hardin, M. Turnbull. "The Real-Time Specification for Java", Addison Wesley, 2000.

[Burns and Wellings. 2002] A. Burns, A. J. Wellings. "Accessing Delay Queues", IRTAW 11, Ada Letters, XXII(4), pp. 72-76. 2002

[Burns et al 2003] A. Burns, B. Dobbing, T. Vardanega. "Guide for the Use of the Ada Ravenscar Profile in High Integrity Systems", YCS-2003-348. 2003

[Carnahan and Ruark 1999] L. Carnahan, M. Ruark, (Eds). "Requirements for Real-time Extensions for the Java Platform", NIST Publication 5000-243, 1999 http://www.nist.gov/rt-java, last accessed 20/8/2002.

[Colin and Puaut 2000] A. Colin, I. Puaut. Worst Case Execution Time Analysis for a Processor with Branch Prediction. Real-Time Systems, 18 (2/3). Pp. 249-274, 2000.

[Critical Software 2003] Critical Software, SA, "Call-off Order 01: SFMECA and Code Inspections on SCOS-2000 2.3e Evaluation Report", Report CSW-RAMS-2003-TNR-1515, DL-RAMS01-08, Version 1.3, October 31, 2003 (ftp.estec.esa.nl/pub/tos-qq/qqs/RAMSFrameContract/SW_FMECAonSCOS2000andCodeInspections/).

[de la Puente and Ruiz 2001] J. A. de la Puente, J. F. Ruiz. "Open Ravenscar Real-Time Kernel – Operations Manual", March 2001.

[ECSS 2002] ECSS Secretariat, "ECSS-E-40B, ECSS Space Engineering, Software Space Engineering Draft B", ESA-ESTEC Requirements & Standards Division, Noordwijk, The Netherlands, February 2002 (http://www.ecss.nl/).

[Engblom and Ermedahl 2000] J. Engblom, A. Ermedahl. Modeling Complex Flows for Worst Case Execution Time Analysis. Proceedings of the 21st IEEE Real-Time Systems Symposium. pp. 163-174, 2000.

[Erpenbach and Altenbernd 1999] E. Erpenbach, P. Altenbernd Worst-Case Execution Times and Schedulability Analysis of Statchart Models. Proceedings of the 11th Euromicro Conference on Real-Time Systems. Pp. 70-77, 1999.

[Fohler 1995] G. Fohler, Joint scheduling of distributed complex periodic and hard aperiodic tasks in statically scheduled systems, 16th IEEE Real-Time Systems Symposium (RTSS '95) Dec 1995.

[Hecht et al 1997] H. Hecht, M. Hecht, S. Graff. Review Guidelines for Software Languages for the Use in Nuclear Powerplant Systems. Technical Report NUREG/CR-6463, Rev. 1, U.S. Nuclear Regulatory Commission, 1997.

[JCR 2000] Java Community Process, JSR50. "JSR 50: Distributed Real Time Specification", http:// www.jcp.org/jsr/detail/50.jsp, 2000, last accessed 20/8/2002.

[J-Consortium 2000] J-Consortium. Realtime Core Extensions", Revision 1.0.14, http://www.j-consortium.org/rtjwg/index.html, 2000, last accessed 20/8/2002.

[Kirner et al 2002] R. Kirner, R. Lang, G. Freiberger, P. Puschner. Fully Automatic Worst-Case Execution-Time Analysis for Matlab/Simulink Models. Proceedings of the 14th Euromicro Conference on Real-Time Systems. pp. 31-40, 2002.

[Kopetz 1993] Kopetz, H, and Reisinger, J. The Non-Blocking Write Protocol NBW: A Solution to a Real-Time Synchronization Problem. In Proceedings of the 14th IEEE Real-Time Systems Symposium, p.131-137, 1993.

[Kopetz 1997] H. Kopetz. Real-Time Systems: Design Principles for Distributed Embedded Applications, Kluwer Academic Publishers, 1997.

[Kopetz 1998] H. Kopetz. The Time-Triggered (TT) Model of Computation. In Proceedings of the 19th IEEE Real-Time Systems Symposium, p.168-177, 1998.

[Kopetz 2000] H. Kopetz. Software Engineering for Real-Time: A Roadmap. In Proceedings of the 22nd International Conference on Software Engineering, p.201-211, 2000.

[Kopetz 2003] H. Kopetz, G. Bauer. The Time-Triggered Architecture. Proceedings of the IEEE, Volume 91(1) p.112-126, 2003.

[Krause and Hartmann 1999] K. H. Krause, W. Hartmann. “RT JAVA Proposal", 1999 http://www.j-consortium.org/rtjw

[Kwon et al 2002] J. Kwon, A. J. Wellings, S. King. “Ravenscar-Java: A High-Integrity Profile for Real-Time Java, Java Grande, pp. 131-140. 2002.

[Lipari and Baruah 2000] S. Lipari, S. K. Baruah "Efficient Scheduling of Multi-Task Applications in Open Systems" IEEE Proceedings of the 6th Real-Time Systems and Applications Symposium, Washington DC, June 2000.

[Liu and Layland 1973] C.L. Liu, J.W. Layland, “Scheduling Algorithms for Multiprogramming in Hard Real-Time Environment”, Journal of the ACM, No. 1, Vol. 20, pp. 40-61, 1973.

[Lundqvist and Stenström 1999] T. Lundqvist, P. Stenström. Timing Anomalies in Dynamically Scheduled Microprocessors. Proceedings of the 20th IEEE Real-Time Systems Symposium. Pp. 12-21, 1999.

[Mok et al 2001]. A. Mok, X. Feng & D. Chen , Resource Partition for Real-Time Systems, 7th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 2001).

[Moreira et al 2003] F. Moreira, R. Maia, D. Costa, N. Duro, K. Hjortnaes, P. Rodriguez, “Static and Dynamic Verification of Critical Software for Space Applications”, In proc. of the Data Systems In Aerospace conference 2003 (DASIA’03), Prague, Czech Republic, June 2-6, 2003.

[Petters and Färber 1999] S. Petters, G. Färber. Making Worst Case Execution Time Analysis for Hard Real-Time Tasks on State of the Art Processors Feasible. Proceedings of the 6th IEEE International Conference on Real-Time Computer Systems and Applications. Pp. 442-449, 1999.

[Puschner and Burns 2002] P. Puschner, A. Burns. Writing Temporally Predictable Code. Proceedings 7th IEEE International Workshop on Object-oriented Real-time Dependable Systems. Pp. 85-91, 2002.

[Puschner 2002a] P. Puschner. Is Worst-Case Execution-Time Analysis a Non-Problem – Towards New Software and Hardware Architectures. Proceedings 2nd Euromicro International Workshop on WCET Analysis, Technical Report, Department of Computer Science, University of York, 2002.

[Puschner 2002b] P. Puschner. Transforming Execution-Time Boundable Code into Temporally Predictable Code. Kleinjohann, B., Kim, K.H. (Kane), Kleinjohann, and L., Rettberg, A. Editors, Design and Analysis of Distributed Embedded Systems, pp. 163-172, Kluwer Academic Publishers, 2002.

[Puschner 2003] P. Puschner. Algorithms for Dependable Hard Real-Time Systems. Proceedings 8th IEEE International Workshop on Object-oriented Real-Time Dependable Systems. 2003.

[Puschner and Koza 1989] P. Puschner, C. Koza, “Calculating the Maximum Execution Time of Real-Time Programs”, *Real-Time Systems*, vol. 1, pp. 159-176, 1989.

[Ramamamritham1990] K. Ramamritham: Scheduling Complex Periodic Tasks, Intl. Conference on Distributed Computing Systems, June 1990.

[Rodríguez et al 2003] M. Rodríguez, N. Silva, J. Esteves, L. Henriques, D. Costa, “Challenges in Calculating the WCET of a Complex On-board Satellite Application”, In proc. of the 3rd Int. Euromicro Workshop on Worst-Case Execution Time Analysis, Porto, Portugal, pp. 3-6, July 1, 2003.

[Regehr and Stankovic 2001] J. Regehr, J. Stankovic, "HLS: a framework for composing soft real-time systems", Proc. RTSS 2001, London 2001.

[Sprunt et al 1989]] B. Sprunt, L. Sha, J. Lehoczky. Aperiodic task scheduling for hard real-time systems. Journal of Real-Time Systems, 1(1):27--60, 1989.

[Spuri and Buttazzo 1996] M. Spuri, G. Buttazzo, "Scheduling Aperiodic Tasks in Dynamic Priority Systems", The Journal of Real-Time Systems, Vol. 10, No. 2, pp. 179-210, March 1996.

[Tindell et al 1994] K. Tindell, A. Burns and A.J. Wellings: An Extendible Approach for Analysing Fixed Priority Hard Real-Time Tasks, Real-Time Systems, Vol. 6(2), pp. 133-151 (March 1994).

[TTTech] TTTech Computertechnik AG. Homepage of TTTech: www.tttech.com.

[Wellings 2004] A. J. Wellings. “Concurrent and Real-Time Programming in Java”, Wiley, 2004 (to appear).

Chapter 5

[Abendroth and Jensen 2003] J. Abendroth, C. D. Jensen. "A Unified Security Framework for Networked Applications". Computer Security Track of the Symposium of Applied Computing (SAC2003), 2003, pp. 351-357.

[Al-Bar and Wakeman 2001] A. Al-Bar, I. Wakeman. "A Survey of Adaptive Applications in Mobile Computing". International Conference on Distributed Computing Systems, Arizona, USA, 2001, pp. 246-251.

[Almeida and Verissimo 1998] C. Almeida, P. Verissimo "Using light-weight groups to handle timing failures in quasi-synchronous systems." The 19th IEEE Real-Time Systems Symposium. Pp. 430-439, Madrid, Spain. 1998.

[Bacon et al 2000] J. Bacon, K. Moody, J. Bates, R. Hayton, C. Ma, A. McNeil, O. Seidel, M. Spiteri, "Generic Support for Distributed Applications," *IEEE Computer*, vol. 33, pp. 68-76, 2000.

[Banavar et al 2000] 11 G. Banavar, J. Beck, E. Gluzberg, "Challenges: An Application Model for Pervasive Computing". Sixth Annual Conference on Mobile Computing and Networking (MobiCom), 2000

[Belaramani et al 2003] N. M. Belaramani, Y. Chow, V. W.-M. Kwan, C-L Wang, F. C. M. Lau, "A Component-based Software Architecture for Pervasive Computing". Intelligent Virtual World: Technologies and Applications in Distributed Virtual Environments, World Scientific Publishing Co., 2003.

[Cugola et al 2001] G. Cugola, E. D. Nitto, A. Fuggetta, "The JEDI Event-Based Infrastructure and its Application to the Development of the OPSS WFMS," *IEEE Transactions on Software Engineering (TSE)*, vol. 27, pp. 827-850, 2001.

[Fitzpatrick et al 2002] A. Fitzpatrick, G. Biegel, S. Clarke, V. Cahill. "Towards a Sentient Object model". in Workshop on Engineering Context-Aware Object-Oriented Systems and Environments (ECOOSE), Seattle, WA, USA, November, 2002.

[Forman and Zahorjan 1994] G. H. Forman, J. Zahorjan, "The Challenges of Mobile Computing". IEEE Computer, 27, 4, pp. 38-47, April 1994.

[Gomez et al 1999] J. Gomez, A. T. Campbell, M. Naghshineh, C. Bisdikian. "Power-aware Routing in Wireless Packet Networks". In 6th IEEE International Workshop on Mobile Multimedia Communications (MOMUC'99), San Diego, 15-17 November 1999.

[Huang and Garcia-Molina 2001] Y. Huang, H. Garcia-Molina, "Publish/Subscribe in a Mobile Environment," in *Proceedings of the Second ACM International Workshop on Data Engineering for Wireless and Mobile Access (MobiDe'01)*. Santa Barbara, CA, USA, 2001, pp. 27-34.

[Hughes and Cahill 2003] B. Hughes, V. Cahill, "Towards Real-time Event-based Communication in Mobile Ad Hoc Wireless Networks," in *Proceedings of 2nd International Workshop on Real-Time LANS in the Internet Age 2003 (ECRTS/RTLIA03)*. Porto, Portugal, 2003, pp. 77-80.

[Keeney and Cahill 2003] J. Keeney, V. Cahill, "Chisel: A Policy-Driven, Context-Aware, Dynamic Adaptation Framework". in Fourth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY2003), Lake Como, Italy, 4-6 June, 2003, pp. 3 -14.

[Meier and Cahill 2002] R. Meier, V. Cahill, "STEAM: Event-Based Middleware for Wireless Ad Hoc Networks," in *Proceedings of the International Workshop on Distributed Event-Based Systems (ICDCS/DEBS'02)*. Vienna, Austria, 2002, pp. 639-644.

[Meier and Cahill 2003] R. Meier, V. Cahill, "Exploiting Proximity in Event-Based Middleware for Collaborative Mobile Applications," in *Proceedings of the 4th IFIP International Conference on Distributed Applications and Interoperable Systems (DAIS'03)*. Paris, France: Springer-Verlag Heidelberg, Germany, 2003.

[Pietzuch and Bacon, 2002] P. Pietzuch, J. Bacon, "Hermes: A Distributed Event-Based Middleware Architecture," in *Proceedings of the International Workshop on Distributed Event-Based Systems (ICDCS/DEBS'02)*. Vienna, Austria, 2002, pp. 611-618.

[Podnar et al 2002] I. Podnar, M. Hauswirth, M. Jazayeri, "Mobile Push: Delivering Content to Mobile Users," in *Proceedings of the International Workshop on Distributed Event-Based Systems (ICDCS/DEBS'02)*. Vienna, Austria, 2002, pp. 563-570.

[Sutton et al 2001] P. Sutton, R. Arkins, B. Segall, "Supporting Disconnectedness – Transparent Information Delivery for Mobile and Invisible Computing," in *Proceedings of the IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2001)*. Brisbane, Australia: IEEE CS Press, 2001, pp. 277-285.

[Verissimo and Casimiro 2002] P. Verissimo, A. Casimiro. "The Timely Computing Base Model and Architectures". Transaction on Computers - Special Issue on Asynchronous Real-Time Systems, 51 (8), August, 2002.

[Verissimo and Casimiro, 2003] P. Verissimo, A. Casimiro, "Event-Driven Support of Real-Time Sentient Objects". In 8th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems, Guadalajara, Mexico, January 2003.

[UMTS] Universal Mobile Telecommunications System Forum. http://www.umts-forum.org

[Wang and Li 2002] K. Wang, B. Li. "Efficient and Guaranteed Service Coverage in Partitionable Mobile Ad-hoc Networks". in IEEE Joint Conference of Computer and Communication Societies (INFOCOM'02), New York City, New York, June 23-27 2002, pp. 1089-1098. 2002.

Chapter 6

[Cecchet et al 2002] E. Cecchet, J. Marguerite, W. Zwaenepoel. Performance and Scalability of EJB Applications. Proceedings of OOPSLA'02, 2002, Seattle, WA, USA.

[Cecchet and Marguerite 2003] E. Cecchet, J. Marguerite, C-JDBC: Scalability and High Availability of the Database Tier in J2EE environments, the 4th ACM/IFIP/USENIX International Middleware Conference (Middleware), 2003, Rio de Janeiro, Brazil, June.

[Fassino et al 2002] J.-P. Fassino, J.-B. Stefani, J. Lawall, G. Muller. THINK: A Software Framework for Component-based Operating System Kernels. Proceedings of Usenix Annual Technical Conference, 2002, Monterey (USA), June 10-15, 2002.

[Senart et al 2002] A. Senart, O. Charra, J.-B. Stefani. Developing Dynamically Reconfigurable Operating System Kernels with the THINK Component Architecture. Proceedings of the workshop on Engineering Context-aware Object-Oriented Systems and Environments, in association with OOPSLA'2002. 2002, Seattle, WA USA November 4-8.

[Charra and Senart 2002] O. Charra, A. Senart. ThinkRCX: un noyau de systeme d'exploitation extensible pour Lego RCX. Actes des Journees sur les Systemes a Composants Adaptables et Extensibles, Grenoble (France), 17-18 October. pp. 239-244.

[Rippert and Stefani 2002a] C. Rippert, J.-B. Stefani. THINK: A Secure Distributed Systems Architecture. Proceedings of the 10th ACM SIGOPS European Workshop", 2002.

[Rippert and Stefani 2002b] C. Rippert, J.-B. Stefani. Building Secure Embedded Kernels with the THINK Architecture. Proceedings of the Workshop on Engineering Context-aware Object-Oriented Systems and Environments, in association with OOPSLA'2002. November 4-8, Seattle, WA (USA) 2002.

[Hansen 2001] J. S. Hansen. Flexible Network Attached Storage using Remote DMA, Proceedings of Hot Interconnects 9, Stanford University. Pp. 51-55, August 22-24. IEEE. 2001.

[Hansen and Jul 2001] J. S. Hansen, E. Jul. Prioritizing Network Event Handling in Clusters of Workstations. Proceedings of the 7th International Euro-Par Conference LNCS Vol. 2150, pp. 704-711, Manchester, 2001, August 28-31. Springer.

[Hansen 2001b] J. S. Hansen. I/O Buffer Management for Shared Storage Devices in SCI-based Clusters of Workstations. Proceedings of SCI Europe 2001, 4th International Conference on SCI-based Technology and Research, October1—3, Dublin. pp. 47-54, 2001,

[Hansen and Lachaize 2002] J. S. Hansen, R. Lachaize. Using Disks in a Cluster as a High Performance Storage System. Proceedings of IEEE International on Cluster Computing (Cluster 2002), 2002, Chicago, IL, (USA).

Chapter 7

[Balikrishnan et al 2003] H. Balikrishnan, M. F. Kaashoek, D. Karger, R. Morris, I. Stoica. Looking up Data in P2P CACM. 46, 2, pp. 43-48. 2003.

[Ballintijn 2003] G. Ballintijn. Locating Objects in a Wide-area System. PhD thesis. Vrije Universiteit Amsterdam, October 2003.

[Bernstein 1996] P. Bernstein. Middleware: A Model for Distributed System Services. CACM, 39, 2, pp. 87-98, 1996.

[Blair et al 2001] G.S. Blair, G. Coulson, A. Andersen, L. Blair, M. Clarke, F. Costa, H. Duran-Limon, T. Fitzpatrick, L. Johnston, R. Moreira, N. Parlavantzas, K. Saikoski. The Design and Implementation of OpenORB v2. IEEE DS Online, Special Issue on Reflective Middleware, 2, 6, 2001.

[ISO 1995] ISO. Open Distributed Processing Reference Model. International Standard ISO/IEC IS 10746. 1995.

[Montresor 2000] A. Montresor. System Support for Programming Object-Oriented Dependable Applications in Partitionable Systems. PhD thesis. University of Bologna, 2000.

[Moser et al 1998] L .E. Moser, P .M. Mellior-Smith, P. Narasimhan. Consistent Object Replication in the Eternal System. Theory and Practice of Object Systems. 4, 2, pp. 81-92, 1998.

[Narasimham et al 1999] P. Narasimham, L .E. Moser, P. M. Mellior-Smith. Using Interceptors to Enhance CORBA. IEEE Computer. 32, 7, pp. 62-68, 1999.

[Pitoura and Samaras 2001] E. Pitoura, G. Samaras. Locating Objects in Mobile Computing. IEEE Transactions. tkde-13 4, pp. 571-592. 2001.

[van Steen et al 1999] M. van Steen, P. Homburg, A. S. Tanenbaum, Globe: A Wide-Area Distributed System, IEEE Concurrency. 7, 1, January, pp. 70-78, 1999.

Chapter 8

[Addis et al 2003] M. Addis, T. Oinn, M. Greenwood, J. Ferris, D. Marvin, P. Li, A. Wipat. Experiences with e-Science workflow specification and enactment in bioinformatics. Proceedings of the 2nd NEReSC All-hands meeting, Leeds, 14th of April 2003. Newcastle upon Tyne. UK. 2003. http://www.neresc.ac.uk/events/2ndallhands/

[Alpdemir et al 2003] M. N. Alpdemir, A. Mukherjee, N. W. Paton, P. Watson, A. A. A. Fernandes, A. Gounaris, J. Smith. OGSA-DQP: A Service-Based Distributed Query Processor for the Grid. Proceedings of the 2nd NEReSC All-hands meeting, Leeds, 14th of April 2003. Newcastle upon Tyne. UK. 2003. http://www.neresc.ac.uk/events/2ndallhands/

[APACHE-AXIS] The Apache Software Foundation. Apache Axis. http://ws.apache.org/axis.

[Benatallah et al 2002] B. Benatallah, M. Dumas, M-C. Fauvet, F.A. Rabhi. Towards patterns of Web Services composition. In Patterns and skeletons for parallel en distributed computing. Springer, 2002.

[BPEL] Business Process Execution Language for Web Services Version 1.1. BEA, IBM, Microsoft, SAP, and Siebel, 2003. http://www-106.ibm.com/developerworks/library/ws-bpel.

[BPML] A. Arkin. Business Process Modeling Language, BPML 1.0 Last Call Working Draft, 2002.

[Bulusu et al 2001] N. Bulusu, J. Heidemann, D. Estrin. Adaptive Beacon Placement, Proceedings of the 21st International Conference on Distributed Computing Systems (ICDCS-21), pp. 489-498, Phoenix, Arizona, USA, April 2001.

[Casati et al 2001] F. Casati, M. Sayal, M.-C. Shan. Developing E-services for composing E-services. In Proc. Of CAISE'2001, LNCS 2068, pp. 171-186, 2001.

[Elson and Estrin 2001a] J. Elson, D. Estrin. Random, Ephemeral Transaction Identifiers in Dynamic Sensor Networks, Proceedings of the 21st International Conference on Distributed Computing Systems (ICDCS-21) April 16-19, 2001, Phoenix, Arizona, USA.

[Elson and Estrin 2001b] J. Elson, D. Estrin. Time Synchronization for Wireless Sensor Networks, Proceedings of the 2001 International Parallel and Distributed Processing Symposium (IPDPS), Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing. 2001.

[ESSL] Earth System Science Lab: Sensor Model Language, http://vast.uah.edu/SensorML/

[Fauvet et al 2001] M-C. Fauvet, M. Dumas, B. Benatallah, H-Y. Paik. Peer-to-peer traced execution of composite services. In proc. Of TES'2001, LNCS 2193, pp. 103-117, 2001.

[Ferguson et al 2003] D. F. Ferguson, T. Storey, B. Lovering J. Shewchuk. Secure, Reliable, Transacted Web Services: Architecture and Composition. IBM Corporation. MS and IBM. Technical Report. September 2003.

[Florescu et al 2002] D. Florescu, A. Grunhagen, D. Kossmann. XL: An XML language for web service specification and composition. In Proceedings of the WWW'02 Conference, 2002.

[Gaizauskas et al 2003] R. Gaizauskas, M. Hepple, N. Davis, Y. Guo, H. Harkema, A. Roberts, I. Roberts. AMBIT: Acquiring Medical and Biological Information from Text. Proceedings of the 2nd NEReSC All-hands meeting, Leeds, 14th of April 2003. Newcastle upon Tyne. UK. 2003. http://www.neresc.ac.uk/events/2ndallhands/

[Heidemann et al 2001] J. Heidemann et. al. Building Efficient Wireless Sensor Networks with Low-Level Naming, Proceedings of the Symposium on Operating Systems Principles, pp. 146-159. Chateau Lake Louise, Banff, Alberta, Canada, ACM. October, 2001.

[Heinzelman 2000] W. B. Heinzelman. Application-Specific Protocol Architectures for Wireless Networks, Ph.D. thesis, Massachusetts Institute of Technology, 2000.

[Hillenbrand et al 2003a] M. Hillenbrand et. al: Web Services for Sensor Node Access, 5th CaberNet Plenary Workshop, November 2003, Porto Santo, Portugal.

[Hillenbrand et al 2003b] M. Hillenbrand et. al: XML basierte Middleware für mobile Sensoren, Berliner XML-Tage, October 2003, Berlin, Germany.

[Houstis et al 2002] C. Houstis, S. Lalis, V.Christophides, D. Plexousakis, E. Vavalis, M. Pitikakis, K. Kritikos, A. Smardas, Ch. Gikas, "A Service infrastructure for e-Science: the case of the ARION system", 14th International Conference on Advanced Information Systems Engineering (CAiSE 2002), E-Services and the Semantic Web workshop (WES2002), Toronto, Canada, May 27-28, 2002, LNCS 2512, pp. 175-187. 2002

[Houstis et al 2003] C. Houstis, S. Lalis, M. Pitikakis, G. Vasilakis, K. Kritikos, A. Smardas. "A grid service-based infrastructure for accessing scientific collections: The case of the ARION system", The International Journal of High Performance Computing Applications, Vol. 17, No 3, pp. .269-280. 2003.

[Kuno et al 2001] H. Kuno, M. Lemon, A. Karp, D. Beringer. Conversations+Interfaces= Business logic. In Proc. Of TES' 2001, LNCS 2193, pp. 30-43, 2001.

[Lamport 2003a] L. Lamport: *Specifying Systems – The TLA+ Language and Tools for Hardware and Software Engineers*. Addison-Wesley, 2003. ISBN 0-321-14306-X.

[Lamport 2003b] L. Lamport: *TLA - The Temporal Logic of Actions*. Webpage for TLA. Microsoft Corporation, 2003. http://research.microsoft.com/users/lamport/tla/tla.html

[Lindsey et al 2001] S. Lindsey, C. Raghavendra, K. Sivalingam. Data Gathering in Sensor Networks using the Energy Delay Metric, International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing, (San Francisco, CA), Apr. 2001.

[Lord et al 2003] P. Lord, C .Wroe, R. Stevens, C. A. Goble, S. Miles, L . Moreau, K. Decker, T. Payne, J. Papay. Semantic and Personalised Service Discovery. Proceedings of the 2nd NEReSC All-hands meeting, Leeds, 14th of April 2003. Newcastle upon Tyne. UK. 2003. http://www.neresc.ac.uk/events/2ndallhands/

[Mikalsen et al. 2002] T. Mikalsen, S. Tai, I. Rouvellou. Transactional attitudes: Reliable composition of autonomous Web services. In Proc. of the Workshop on Dependable Middleware-based Systems in DSN'2002, 2002.

[Moreau et al 2003] L. Moreau, X. Liu, S. Miles, A. Krishna, V. Tan, R. Lawley. myGrid Notification Service. Proceedings of the 2nd NEReSC All-hands meeting, Leeds, 14th of April 2003. Newcastle upon Tyne. UK. 2003. http://www.neresc.ac.uk/events/2ndallhands/

[MS-NET] Microsoft. .NET. http://msdn.microsoft.com/net/.

[Narayanan and Mc Ilraith 2002] S. Narayanan, S. McIlraith. Simulation, verification and automated composition of web services. In Proceedings of the WWW'02 Conference, 2002.

[OASIS-BTP] OASIS Committee Specification. Business Transaction Protocol, Version 1.0, 2002.

[Pottie and Kaiser 2000] G .J. Pottie, W. J. Kaiser. Wireless Integrated Network Sensors, Communications of the ACM, vol. 43, no. 5, pp. 551-8, May 2000.

[Romanovsky et al 2002] A. Romanovsky, P. Periorellis, A. F. Zorzo. On Structuring Integrated Web Applications for Fault Tolerance. In Proceedings of the 6th International Symposium on Autonomous Decentralised Systems (ISADS 2003), Pisa, Italy, April 2003. pp. 99-106.

[Senger et al 2003] M. Senger, P. Rice, T. Oinn. Soaplab - a unified Sesame door to analysis tools. Proceedings of the 2nd NEReSC All-hands meeting, Leeds, 14th of April 2003. Newcastle upon Tyne. UK. 2003. http://www.neresc.ac.uk/events/2ndallhands/

[Shih et al 2001] E. Shih et al. Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks, The seventh annual international conference on Mobile computing and networking 2001, July 16 - 21, 2001, Rome Italy. Pp. 272 – 287.

[Sinha et al 2000] A. Sinha, A. Wang, A. P. Chandrakasan. Algorithmic Transforms for Efficient Energy Scalable Computation, Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED), 2000.

[Sohrabi et al 2000] K. Sohrabi, J. Gao, V. Ailawadhi, G. J. Pottie. Protocols for self-organization of a wireless sensor network, IEEE Personal Communications, 7, 5, pp. 16-27, Oct. 2000.

[Stevens et al 2003] R Stevens, M Greenwood, and CA Goble. Provenance of e-Science Experiments – experience from Bioinformatics. Proceedings of the 2nd NEReSC All-hands meeting, Leeds, 14th of April 2003. Newcastle upon Tyne. UK. 2003. http://www.neresc.ac.uk/events/2ndallhands/

[Tartanoglu et al 2003] F. Tartanoglu, V. Issarny, A. Romanovsky, N. Levy. Coordinated Forward Error Recovery for Composite Web Services. In Proceedings of the 22nd Symposium on Reliable Distributed Systems (SRDS'2003)., Florence, Italy, October 2003.

[UDDI] UDDI, Version 2.0, API Specification. Technical report, 2002. http://www.uddi.org.

[Venzke 2003] M. Venzke: *Automatic Validation of Web Services.* In: Proceedings of the 8th CaberNet Radicals Workshop. Hotel Eden Roc, Ajaccio, Corsica, October 5 - 8, 2003. http://www.ti5.tu-harburg.de/Publication/2003/Paper/Venzke03b/Venzke_CaberNet03.pdf

[SUN-J2EE] Sun Microsystems Inc. Java 2 Platform, Enterprise Edition (J2EE). http://java.sun.com/j2ee/.

[W3C-SOAP] W3C. Soap. Version 1.2. Part 0: Primer. W3C Recommendation. 2003. http://www.w3.org/TR/soap12-part0/

[W3C-THP] W3C. Tentative Protocol Part 1: White Paper. WSC Note, 2001.

[W3C-WSCI] W3C. Web services choreography interface (WSCI), version 1. W3C Note, 2002. http://www.w3.org/TR/wsci/.

[W3C-WSCL] W3C. Web services conversation language (WSCL), version 1.0. W3C Note, 2002. http://www.w3.org/TR/wscl10/.

[W3C-WSDL] W3C. Web services description language (WSDL), version 2.0. Part 1: Core Language. W3C Working Draft, 2003. http://www.w3.org/TR/wsdl20

[W3C-XML] W3C. Second Edition of the Extensible Markup Language (XML). 1.0 Specification. W3C Recommendation. http://www.w3.org/TR/2000/REC-xml-2001006. 2000.

[W3C-SG] W3C. Web Services Glossary. W3C working draft. 14 November 2002. http://www.w3.org/TR/2002/WD-ws-gloss-20021114/

[Wroe et al 2003] C. Wroe, R. Stevens, C. Goble, A. Roberts, M. Greenwood. A suite of DAML+OIL ontologies to describe bioinformatics web services and data. *International Journal of Cooperative Information Systems*, 12(2). Pp. 197–224, 2003.

[WS-C] Web Services Coordination. (WS-Coordination). BEA, IBM and Microsoft, 2002. http://www.ibm.com/developerworks/library/ws-coor/

[WS-CF] Web Services Coordination Framework (WS-CF) version 1.0. Arjuna Technologies, Fujitsu Limited, IONA Technologies, Oracle Corporation and Sun Microsystems. 2003. http://www.oasis-open.org/committees/ws-caf.

[WS-T] Web Services Transaction (WS-Transaction), BEA, IBM and Microsoft, 2002. http://www-106.ibm.com/developerworks/webservices/library/ws-transpec/

[WS-TXM] Web Services Transaction Management (WS-TXM) version 1.0. Arjuna Technologies, Fujitsu Limited, IONA Technologies, Oracle Corporation and Sun Microsystems. 2003. http://www.oasis-open.org/committees/ws-caf.

[Yang and Papazoglou 2002] J. Yang, P. Papazoglou. Web component: A substrate for web service reuse and composition. In Proceedings of CAISE'02, pp. 21-36, 2002.

Chapter 9

[Baumann et al 1999] J. Baumann, F. Hohl, K. Rothermel, M. Strasser. *Mole - Concepts of a Mobile Agent System*. In *Mobility: Processes, Computers, and Agents*. D. Milojicic, F. Douglis, R. Wheeler, (Eds). pp. 535-556, Addison-Wesley and the ACM Press. 1999.

[Peine 2002] H. Peine. *Run-Time Support for Mobile Code*, Doctoral. Ph.D. dissertation. University of Kaiserslautern, Germany. 2002.

Chapter 10

[Amir et al 2000] Y. Amir, C. Danilov, J. Stanton. A Low Latency, Loss Tolerant Architecture and Protocol for Wide Area Group Communication. Int. Conf. on Dependable Systems and Networks (DSN), New York, 2000.

[Amir and Tutu 2002] Y. Amir, C. Tutu. From Total Order to Database Replication. IEEE ICDCS 2002.

[Breitbart et al 1999] Y. Breitbart, R. Komondoor, R. Rastogi, S. Seshadri, A. Silberschatz. Update Propagation Protocols for Replicated Databases. ACM SIGMOD Conference 1999, pp. 97-108. 1999.

[Cheriton and Skeen 1993] D. R. Cheriton, D. Skeen. Understanding the Limitations of Causally and Totally Ordered Communication in B Liskov, editor, *Proceedings of the 14th Symposium on Operating Systems Principles*, volume 27, pp. 44-57 Asheville, North Carolina, 1993, ACM Press New York, NY, USA.

[Fahrenholtz and Bartelt 2001] D. Fahrenholtz, A. Bartelt. Towards a Sociological View of Trust in Computer Science. In M. Schoop, R. Walzcuch, eds, *Proceedings of the 8th Research Symposium on Emerging Electronic Markets*, 2001, Maastricht, The Netherlands.

[Fahrenholtz and Lamersdorf 2002] D. Fahrenholtz, W. Lamersdorf Transactional Security for a Distributed Reputation Management System. In K. Bauknecht, A. Min Tjoa, G. Quirchmayr, (Eds) *Proceedings of the 3rd International Conference on Electronic Commerce and Web Technologies. Aix-en-Provence*, 2002. pp. 214–224, Springer-Verlag Heidelberg.

[Gray et al 1996] J. Gray, P. Helland, P. E. O'Neil, D. Shasha. The Dangers of Replication and a Solution. ACM SIGMOD Conference 1996, pp. 173-182.

[Jimenez-Peris et al 2001] R. Jimenez-Peris, M. Patiño-Martínez, G. Alonso, S. Arevalo. A Low-Latency Non-blocking Commit Service. Distributed Computing Conf. (DISC), pp. 93-107, Lisboa, Portugal. 2001.

[Jimenez-Peris et al 2002a] R. Jimenez-Peris, M. Patiño-Martínez, B. Kemme, G. Alonso. Improving the Scalability of Fault-Tolerant Database Clusters. IEEE Int. Conf on Distributed Computing Conf. ICDCS 2002, pp. 477-484. 2002.

[Jimenez-Peris et al 2002b] R. Jimenez-Peris, M. Patiño-Martínez, G. Alonso. An Algorithm for Non-Intrusive, Parallel Recovery of Replicated Data. IEEE Int. Symp. on Reliable Distributed Systems (SRDS), pp. 150-159. Oct. 2002, Osaka, Japan. 2002.

[Jimenez-Peris et al 2003a] R. Jimenez-Peris, M. Patiño-Martínez, G. Alonso, B. Kemme. Are Quorums an Alternative to Data Replication? ACM Transactions on Database Systems, Vol. 28, N. 3, Sept. 2003, pp. 257-294.

[Jimenez-Peris and Patiño-Martínez 2003b] R. Jimenez-Peris, M. Patiño-Martínez. Towards Robust Optimistic Approaches. In Future Directions in Distributed Computing, LNCS-2584, pp. 45-50, Springer, 2003.

[Kemme and Alonso 2000a] B. Kemme, G. Alonso. Don't Be Lazy, Be Consistent. Postgres-R, A New Way to Implement Database Replication. VLDB'00, pp. 134-143, Cairo, Egipt, 2000.

[Kemme and Alonso 2000b] B. Kemme, G. Alonso. A New Approach to Developing and Implementing Eager Database Replication Protocols. ACM Trans. on Databases 25(3), pp. 333-379, 2000.

[Kemme et al 2001] B. Kemme, A. Bartoli, O. Babaoglu. Online Reconfiguration in Replicated Databases Based on Group Communication. IEEE Symp. On Dependable Systems and Networks (DSN), pp. 117-130, Goteborg, Sweden, 2001.

[Kemme et al 2003] B. Kemme, F. Pedone, G. Alonso, A. Schiper, M. Wiesmann Using Optimistic Atomic Broadcast in Transaction Processing Systems. IEEE Trans. on Data and Knowledge Engineering. N. 4. Vol. 15. 2003.

[Kupsys et al 2003] A. Kupsys, S. Pleisch, A. Schiper, M. Wiesmann. Towards JMS-compliant group communication. Technical Report 200353, École Polytechnique Fédérale de Lausanne, Switzerland, September 2003.

[Mena et al 2003] S. Mena, A. Schiper, P. T. Wojciechowski. A Step Towards a New Generation of Group Communication Systems. Technical Report IC-2003/01, Faculté Informatique et Communications, École Polytechnique Fédérale de Lausanne, January 2003.

[Milan et al 2003] J. Milan, R. Jimenez-Peris, M. Patiño-Martínez. Towards Dynamically Adaptive Database Replication. 5th CaberNet Plenary Workshop, 2003.

[Pacitti and Simon 2000] E. Pacitti, E. Simon. Update Propagation Strategies to Improve Freshness in Lazy Master Replicated Databases. VLDB Journal 8(3-4), pp. 305-318, 2000.

[Patiño-Martínez et al 2000] M. Patiño-Martínez, R. Jimenez-Peris, B. Kemme, G. Alonso. Scalable Replication in Database Clusters. Distributed Computing Conf. (DISC) 2000, pp. 315-329, Toledo, Spain, 2000.

[Rodrigues et al 2002] L. Rodrigues, H. Miranda, R. Almeida, J. Martins, P. Vicente. Strong Replication in the GlobData Middleware. In Workshop on Dependable Middleware-Based Systems (part of DSN 2002).

[Sousa et al 2002] A. Sousa, J. Pereira, F. Moura, R. Oliveira. Optimistic Total Order in Wide Area Networks. IEEE Symp. On Reliable Distributed Systems, SRDS'02. IEEE CS Press, 2002.

[Vicente and Rodrigues 2002] P. Vicente, L. Rodrigues An Indulgent Uniform Total Order Algorithm with Optimistic Delivery. IEEE Symp. On Reliable Distributed Systems (SRDS), pp. 92-101. Oct., 2002, Osaka, Japan. 2002.

[Wiesmann and Schiper 2003] M. Wiesmann, A. Schiper. Beyond 1-safety and 2-safety for replicated databases: Group-safety. Technical Report IC/2003/18, École Polytechnique Fédérale de Lausanne, Switzerland, March 2003.

[Wiesmann et al 2003] M. Wiesmann, X. Défago, A. Schiper. Group communication based on standard interfaces. In *Proceedings of the IEEE International Symposium on Network Computing and Applications (NCA-03).* Pp. 140-147, Cambridge, MA, USA, 2003.

[Wojciechowski et al 2002] P. T. Wojciechowski, S. Mena, A. Schiper. Semantics of Protocol Modules Composition and Interaction. In Coordination 2002 (The Fifth International Conference on Coordination Models and Languages). LNCS 2315, April 2002.

Chapter 11

[Busse et al 1996] I. Busse, B. Deffner, H. Schulzrinne, Dynamic QoS control of multimedia applications based on RTP, Computer Communications, Jan. 1996.

[Vogt et al 1998] C. Vogt, L. C. Wolf, R. G. Herrtwich, H. Wittig, HeiRAT - Quality-of-Service Management for Distributed Multimedia Systems, Multimedia Systems, Vol 6, Nr. 3, 1998, pp. 152-166.

[Coulson el al 1998] G. Coulson, M.W. Clarke, A Distributed Object Platform Infrastructure for Multimedia Applications, Computer Communications, Vol 21, No 9, pp 802-818, July 1998.

[Reuther and Hillenbrand 2003] B. Reuther, M. Hillenbrand. Supervised Media Processing with Real-Time Priority, 7th IASTED International Conference on Internet and Multimedia Systems and Applications; IMSA 2003; August 13-15, 2003; Honolulu; Hawaii; USA.

[Henrici and Reuther 2003] D. Henrici, B. Reuther, 2nd IASTED International Conference on Communications, Internet and Information Technology; CIIT 2003; November 17-19, 2003; Scottsdale; Arizona; USA.

[Kroh et al 2000] R. Kroh, A. Held, M. Aldinger, R. Keller, T. Lohmar, E. Kovacs. High-Quality Interactive and Broadband IP Services for the Vehicle on the Net - The COMCAR Approach - (ITS2000).

[Akamine et al 2002] H. Akamine, N. Wakamiya, H. Miyahara, Heterogeneous Video Multicast in an Active Network, IEICE Transactions on Communications, vol. E85-B, no. 1, pp.284-292, January 2002

[Metzler et al 1999] B. Metzler, T. Harbaum, R. Wittmann, M. Zitterbart. AMnet: Heterogeneous Multicast Services based on Active Networking, Proc. IEEE OpenArch, pp. 98-105, March 1999.

[Perring et al 2001] A. Perrig, R. Canetti, D. Song, J. D. Tygar. Efficient and Secure Source Authentication for Multicast. ISOC Network and Distributed System Security Symposium 2001.

[Wong and Lam 1999] C. K. Wong, S. Lam. Digital Signatures for Flows and Multicasts. IEEE/ACM Transactions on Networking, Vol. 7, No.4, August 1999.

[Zhang et al 2003] G. Zhang, B. Reuther, P. Müller, User Oriented IP Accounting in Multi-user Systems, The 8th IFIP/IEEE International Symposium on Integrated Network Management, Colorado Springs, USA, 2003.

Chapter 12

[Adar and Huberman 2000] E. Adar, B. Huberman, "Free riding on gnutella," First Monday, vol. 5, Oct. 2000.

[Adya et al 2002] A. Adya, W. J. Bolosky, M. Castro, G. Cermak, R. Chaiken, J. R. Douceur, J. Howell, J. R. Lorch, M. Theimer, R. P. Wattenhofer. FARSITE: Federated, Available, and Reliable Storage for an Incompletely Trusted Environment. Proceedings of the 5th OSDI, December 2002. 2002.

[Bolosky et al 2000] W. J. Bolosky, J. R. Douceur, D. Ely, M. Theimer. Feasibility of a Serverless Distributed File System Deployed on an Existing Set of Desktop PCs. Proceedings of the international conference on Measurement and modeling of computer systems, 2000, pp. 34-43.

[CacheFlow] http://www.cacheflow.com/files/whitepapers/wp_shared_vs_dedicate.pdf

[Cain et al 2002] B. Cain, O. Spatscheck, M. May, A. Barbir. '*Request Routing Requirements for Content Internetworking*', IETF draft, draft-ietf-cdi-request-routing-reqs-01.txt, May 2002

[Castro and Liskov 2002] M. Castro, B. Liskov. Practical Byzantine fault tolerance and proactive recovery. ACM Transactions on Computer Systems (TOCS), 20, 4, pp. 398 – 346. 2002

[Castro et al 2002] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, D. S. Wallach, "Security for structured peer-to-peer overlay networks". In Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI'02), Boston, MA, December 2002.

[CIFS 1996] CIFS: A Common Internet File System. Microsoft Interactive Developer Magazine. November 1996. http://www.microsoft.com/Mind/1196/CIFS.htm

[Dabek et al 2001] F. Dabek, M. Kaashoek, D. Karger, R. Morris, I. Stoica. Wide-area cooperative storage with CFS. Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP '01). Banff, Canada. October 2001.

[Day et al 2003] M. Day, B. Cain, G. Tomlinson, P. Rzewski. '*A Model for Content Internetworking (CDI)*', RFC 3466, draft-ietf-cdi-model-02.txt, Feb 2003.

[Gibson and van Meter 2000] G. A. Gibson, R. van Meter. Network Attached Storage Architecture. Communications of the ACM 43(11). November 2000 .

[Green et al 2002] M. Green, B. Cain, G. Tomlinson, S. Thomas, P. Rzewski. '*Content Internetworking Architectural Overview*', IETF draft, draft-ietf-cdi-architecture-01.txt, June 2002

[Hand and Roscoe 2002] S. Hand, T. Roscoe. Mnemosyne: Peer-to-Peer Steganographic Storage. Proceedings of the 1st International Workshop on Peer-to-Peer Systems. Boston, MA. March 2002.

[IETF] http://www.IETF.org/html.charters/cdi-charter.html

[Kubiatowicz et al 2000] J. Kubiatowicz et al. OceanStore: An Architecture for Global-Scale Persistent Storage. Proceedings of the Ninth international Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2000). November 2000.

[Mesnier 2002] M. Mesnier. The Rebirth of Object-Based Storage. Intel Developer Forum Presentation. January 2002. available from ftp://download.intel.com/labs/storage/download/rebirthOSD.pdf

[Moreton et al 2002] T. D. Moreton, I. A. Pratt, T. L. Harris. Storage, Mutability and Naming in Pasta. Proceedings of the International Workshop on Peer-to-Peer Computing at Networking 2002. Pisa, Italy. May 2002.

[NFS 1989] NFS: Network File System Protocol Specification. RFC 1094. Sun Microsystems, Inc. Menlo Park, CA. 1989.

[Rowstron and Druschel 2001a] A. Rowstron, P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems". IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), Heidelberg, Germany, pp. 329-350, November, 2001. 2001.

[Rowstron and Druschel 2001b] A. Rowstron, P. Druschel. Storage Management and Caching in PAST, a Large Scale Persistent Peer-to-peer Storage Utility. Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP '01). Banff, Canada. October 2001. 2001.

[Satran et al 2002] J. Satran, K. Meth, C. Sapuntzakis, M. Chadalapaka, E. Zeidner. iSCSI. 2002. Internet draft available at http://www.ietf.org/internet-drafts/draft-ietf-ips-iscsi-18.txt

[Turrini and Ghini 2003] E. Turrini, V. Ghini, '*A Protocol for exchanging performance data in Content Distribution Internetworks*', 3rd International Conference on Networking, Mar. 2004, To appear.

Chapter 13

[Bandara et al 2003] A. Bandara, E. Lupu, A. Russo. *Using Event Calculus to Formalise Policy Specification and Analysis*, IEEE 4th International Workshop on Policies for Networks and Distributed Systems, Como, June 2003.

[Damianou et al 2001] N. Damianou, N. Dulay, E. Lupu, M. Sloman. The Ponder Policy Specification Language. In Proceedings of the Policy Workshop 2001, Bristol, UK, Jan. 2001, Springer Verlag, Lecture Notes in Computer Science vol. 1995. 2001.

[Damianou et al 2002] N. Damianou, N. Dulay, E. Lupu**,** M. Sloman, T. Tonouchi. *Policy Tools for Domain Based Distributed Systems Management.* IFIP/IEEE Symposium on Network Operations and Management. Florence, Italy, Apr. 2002.

[Dulay et al 2001] N. Dulay, E. Lupu, M. Sloman, N. Damianou. A Policy Deployment Model for the Ponder Language. In Proceedings of the IFIP/IEEE Symposium on Integrated Network Management (IM 2001), Seattle, USA, May 2001.

[Hegering et al 2003] H.-G. Hegering, A. Küpper, C. Linnhoff-Popien, H. Reiser. Management Challenges of Context-Aware Services in Ubiquitous Environments, Proc. IEEE/IFIP Distributed Systems Operations and Management Workshop, Heidelberg, Oct. 2003, Springer Verlag, LNCS-2867. 2003.

[Lymberopoulos et al 2003] L. Lymberopoulos, E. Lupu, M. Sloman An Adaptive Policy Based Framework for Network Services Management, *Plenum Press Journal of Network and Systems Management*, Special Issue on Policy Based Management, Vol 11, No. 3 Sep. 2003.

[Richter 2001] J.P. Richter. "Spezifikations- und Messmethodik für ein adaptives Dienstgütemanagement". GI-Fachtagung "Kommunikation in Verteilten Systemen" (KiVS 2001). Hamburg, Germany. Springer-Verlag. February 2001.

[Wolfinger and Kühn 2002] B. E. Wolfinger, P .J. Kühn (eds.). Special Issue “Internet Traffic Engineering”. Praxis der Informationsverarbeitung und Kommunikation (PIK) 24(2). June 2002.

[Wolfinger et al 2002] B.E. Wolfinger, M. Zaddach, K. Heidtmann, G. Bai. “Analytical Modeling of Primary and Secondary Load as Induced by Video Applications Using UDP/IP”. Computer Communications Journal 25 (11-12). July. 2002. pp. 1094-1102.

Chapter 14

[Abrahams and Bacon 2001] A. S. Abrahams, J. M. Bacon. Representing and Enforcing E-Commerce Contracts Using Occurrences. The Fourth International Conference on Electronic Commerce Research (ICECR-4) Dallas, TX, USA, November 8 - 11, 2001.

[Asokan et al 1998] N. Asokan, M. Schunter, M. Waidner. Asynchronous protocols for optimistic fair exchange. In Proc. IEEE Symposium on Research in Security and Privacy, pp. 86-99, 1998.

[Bacon et al 2001] J. Bacon, K. Moody, W. Yao. Access Control and Trust in the use of Widely Distributed Services. IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001), November 2001, Heidelberg,, Lecture Notes in Computer Science. LNCS-2218, pp. 300-315.

[Cook et al 2002] N. Cook, S. K. Shrivastava, S. M. Wheater. Distributed Object Middleware to Support Dependable Information Sharing between Organisations. Proc. IEEE/IFIP Int. Conf. on Dependable Syst. and Networks (DSN-2002), Bethesda USA, June 2002.

[Daskalopulu et al 2001] A. Daskalopulu, T. Dimitrakos, T. Maibaum. E-Contract Fulfilment and Agents' Attitudes. Proceedings of ERCIM WG E-Commerce Workshop on The Role of Trust in eBusiness, Zurich, October, 2001.

[ebXML] ebXML Collaboration Protocol Profile and Agreement, http://www.oasis-open.org/

[Ezhilchelvan and Shrivastava 2003] P. D. Ezhilchelvan, S. K Shrivastava. Systematic Development of a Family of Fair Exchange Protocols. Seventeenth Annual IFIP WG 11.3 Working Conference on Data and Applications Security, Estes Park, Colorado, August 2003.

[Foster et al 2001] I. Foster, C. Kesselman, S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. Intl J. Supercomputer Applications, 2001.

[Gerck 1998] E. Gerck, *Towards Real-World Models of Trust: Reliance on Received Information*, published on 23rd June 1998 in the mcg-talk list server.

[Grandison and Sloman 2000] T. Grandison, M. Sloman. A survey of trust in Internet applications. IEEE Communications Surveys, Fourth Quarter 2000, www.comsoc.org/pubs/surveys.

[Halsall 1999] P. Halsall. Ancient History Sourcebook: A Collection of Contracts from Mesopotamia, c. 2300-428 BCE, http://www.fordham.edu/halsall/ancient/mesopotamia-contracts.html, March, 1999.

[Keller et al 2002] A. Keller. G. Kar. H. Ludwig, A. Dan, J. Hellerstine. Managing Dynamic Services: a Contract based Approach to a Conceptual Architecture. Proc. Of 8th IEEE/IFIP Network Operations and Management Symposium (NOMS 2002), Florence, April 2002.

[Liu et al 2000] P. Liu, P. Ning, S. Jajodia. Avoiding Loss of Fairness Owing to Process Crashes in Fair Data Exchange Protocols. Proc. *Int. Conf. on Dependable Systems and Network*. pp. 631-640, June 2000.

[Marjanovic and Milosevic 2001] O. Marjanovic, Z. Milosevic. Towards formal modelling of e-contracts. Proc. of 5th IEEE/OMG International Enterprise Distributed Object Computing Conference (EDOC 2001), September 2001, pp. 59-68.

[Minsky and Ungureanu 2000] N. H. Minsky, V. Ungureanu. Law-governed interaction: a coordination and control mechanism for heterogeneous distributed systems. ACM Trans. on Software Engineering and Methodology, 9(3), July 2000, pp. 273-305.

[Molina-Jimenez et al 2003] C. Molina-Jimenez, S. K. Shrivastava, E. Solaiman, J. Warne. Contract Representation for Run-time Monitoring and Enforcement. Proceedings of IEEE Conference on Electronic Commerce (CEC'03), Newport Beach, CA, June 2003.

[OGSA] Open Grid Services Architecture: A Roadmap. Global Grid Forum, Working Paper.

[Ouzounis and Tschammer 2001] V. Ouzounis, V. Tschammer. Towards Dynamic Virtual Enterprises. Towards the E-society: The First IFIP Conference on E-commerce, E-Business, E-Government (I3E 2001), October 3-5, 2001, Zurich, Switzerland, Kluwer Academic Publisher.

[Pfitzmann et al 1998] B. Pfitzmann, M. Schunter, M. Waidner. *Optimal Efficiency of Optimistic Contract Signing*. Proc. *ACM Symp. on Principles of Distributed Computing*. New York, 1998.

[Rosettanet] Rosettanet implementation framework: core specification, V2, Jan 2000. http://rosettanet.org

[Shrivastava 2003] S. K. Shrivastava. Middleware for supporting inter-organizational interactions. In Future Directions in Distributed Computing, LNCS 2584, Springer Verlag, 2003.

[WSA] Web Services Architecture work, http://www.w3.org/2002/ws/

[van der Aalst 2003] W.M.P. van der Aalst. Don't go with the flow: Web services composition standards exposed. IEEE Intelligent Systems, Jan/Feb 2003.

[Zhou and Gollmann 1997] J. Zhou, D. Gollmann. Evidence and non-repudiation. *Journal of Network and Computer Applications*, 20(3), pp. 267-281, 1997.

Chapter 15

[Abrial 1996] J-R. Abrial. The B-Book: Assigning Programs to Meanings. Cambridge University Press, 1996.

[Alur et al 1993] R. Alur, C. Courcoubetis, T. Henzinger, P.-H. Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In: Hybrid systems, LNCS-736, pp. 209-229, Springer-Verlag, 1993.

[Alur and Dill 1995] R. Alur, D. Dill. A theory of timed automata. Theoretical Computer Science, 126, 2, pp. 183-235, 1995.

[Bowman et al 2002] H. Bowman, M.W.A. Steen, E.A. Boiten, and J. Derrick, A formal framework for viewpoint consistency. Formal Methods in System Design, 21, pp. 111-166, September 2002.

[Brinksma and Tretmans 2001] E. Brinksma, J. Tretmans. (Eds). Testing Transition Systems: An Annotated Bibliography. MOVEP 2000 - MOdelling and VErification of Parallel processes, Nantes, France, LNCS, Vol. 2067, Springer-Verlag, pp. 187 - 195, 2001

[Clarke et al 2002] E. Clarke, S. Jha, Y. Lu, H. Veith. Tree-Like Counterexamples in Model Checking. IEEE symposium on Logic in Computer Science (LICS), 2002

[Groce and Visser 2002] A. Groce, W. Visser. Model Checking Java Programs using Structural Heuristics. In: Proceedings of the ACM SIGSOFT 2002 Int. Symposium on Software Testing and Analysis, Software Engineering Notes, 27, 4, 2002.

[Hayes and Jones 1989] I. J. Hayes, C. B. Jones. Specifications are not (necessarily) executable, IEE Software Engineering Journal, Volume 4(6), pp 320-388, 1989.

[Hayes et al 2003] I. J. Hayes, M. A. Jackson, C. B. Jones. Determining the Specification of a Control System from that of its Environment, In FME 2003: Formal Methods, LNCS 2805, Springer Verlag, 2003.

[Henzinger 1996] T. Henzinger. The theory of hybrid automata. In: Proc. LICS'96, pp. 278-292, IEEE Computer Society Press, 1996.

[Henzinger 2000] T. Henzinger. The theory of hybrid automata. In: Proc. Verification of digital and hybrid systems, volume 170 of NATO ASI Series F: Computer and Systems Sciences, pp. 265-292, Springer-Verlag, 2000.

[Hermanns et al 2003] H. Hermanns, J. P. Katoen, J. Meyer-Kayser, M. Siegle, A tool for model checking Markov chains. Software Tools for Technology Transfer, 4, 2, pp. 153 – 172, 2003.

[Hoare et al 2004] C. A. R. Hoare, C. B. Jones, B. Randell. Extending the Horizons of DSE (GC6). CS-TR: 853, School of Computing Science, University of Newcastle, 2004

[ISO 1995] International Standards Organisation. VDM-SL. Draft International Standard, ISO/IEC JTC1/SC22/WG19 N-20, 1995.

[Jones 1996] C. B. Jones. A Rigorous Approach to Formal Methods. IEEE Computer, Volume 29(4), pp 20-21 IEEE Computer Society, 1996.

[Kwiatkowska et al 2002] M. Kwiatkowska, G. Norman, D. Parker, PRISM: Probabilistic Symbolic Model Checker. In: Proc. TOOLS 2002, volume 2324 of LNCS, pp. 200-204, Springer-Verlag 2002.

[Kwiatkowska et al 2003] M. Kwiatkowska, G. Norman, D. Parker, J. Sproston. Performance Analysis of Probabilistic Timed Automata using Digital Clocks. In Proc. Formal Modelling and Analysis of Timed Systems (FORMATS'03), 2003.

[LaCoste 2002] M. Lacoste. Building Reliable Distributed Infrastructures Revisited: a Case Study. International DOA Workshop on Foundations of Middleware Technologies (WFoMT'02), 2002.

[Liggesmeyer et al 1998] P. Liggesmeyer et al. Qualitaetssicherung Software-basierter technischer Systeme - Problembereiche und loesungansaetze. Informatik Spektrum, 21. pp. 249-258, 1998.

[RAISE 1995] The RAISE Method Group. The RAISE Development Method. Prentice-Hall, BCS Practitioner Series, 1995.

[Rushby 1993] J. Rushby. Formal Methods and the Certification of Critical Systems. Technical Report CSL-93-7, 1993. Available on line: http://www.csl.sri.com/papers/csl-93-7/

Chapter 16

[Allen and Garlan 1997] R. Allen, D. Garlan. A Formal Basis for Architectural Connection. ACM Transactions on Software Engineering and Methodology (TOSEM), 6(3), pp. 213-249, 1997.

[Avizienis et al 2003] A. Avizienis, J.-C. Laprie, B. Randell. *Fundamental Concepts of Dependability*. Technical Report 739. Department of Computing Science. University of Newcastle upon Tyne. 2001.

[Bernado and Inverardi 2003] M. Bernado, P. Inverardi. *Formal Methods for Software Architectures*. Lecture Notes in Computer Science 2804. Springer. Berlin, Germany. 2003.

[Björkander and Kobryn 2003] M. Björkander, C. Kobryn. Architecting Systems with UML 2.0. *IEEE Software*. July/August 2003. pp. 57-61.

[Blair et al 2000] G.S. Blair, L. Blair, V. Issarny, P. Tuma, A. Zarras. The Role of Software Architecture in Constraining Adaptation in Component-Based Middleware Platforms. Proc. Middleware'2000: IFIP/ACM International Conference on Distributed Systems Platforms, LNCS 1795, pp. 164-184, 2000.

[Dabrowski and Mills 2002] C. Dabrowski, K. Mills. Understanding Self-Healing in Service-Discovery Systems. *Proc. of the 1st ACM SIGSOFT Workshop on Self-Healing Systems (WOSS'02)*. Charleston, SC, USA. November 2002. pp. 15-20.

[Dashofy et al 1999] E.M. Dashofy, N. Medvidovic and R.N. Taylor. Using Off-The-Shelf Middleware to Implement Connectors in Distributed Software Architectures. Proc. 21st International Conference on Software Engineering (ICSE), pp. 3-12, 1999.

[Dashofy et al 2002] E. Dashofy, A. van der Hoek, R. N. Taylor. Towards Architecture-Based Self-Healing Systems. *Proc. of the 1st ACM SIGSOFT Workshop on Self-Healing Systems (WOSS'02)*. Charleston, SC, USA. November 2002. pp. 21-26.

[de Lemos 2001] R. de Lemos. Describing Evolving Dependable Systems using Co-operative Software Architectures. *Proc. of the IEEE Int. Conference on Software Maintenance (ICSM'01)*. Florence, Italy. November 2001. pp. 320-329.

[de Lemos et al 2002a] R. de Lemos, C. Gacek, A. Romanovsky (Eds.). *Proc. of the ICSE 2002 Workshop on Architecting Dependable Systems*. Orlando, FL. May 2002. http://www.cs.ukc.ac.uk/events/conf/2002/wads/

[de Lemos et al 2002b] R. de Lemos, C. Gacek, A. Romanovsky. ICSE 2002 Workshop on Architecting Dependable Systems (Workshop Summary). *ACM Software Engineering Notes 27(5)*. pp. 77-80. September 2002.

[de Lemos et al 2003a] R. de Lemos, C. Gacek, A. Romanovsky (Eds.). *Proc. of the ICSE 2003 Workshop on Software Architectures for Dependable Systems*. Portland, OR. April 2003. http://www.cs.ukc.ac.uk/events/conf/2003/wads/

[de Lemos et al 2003b] R. de Lemos, C. Gacek, A. Romanovsky (Eds.). *Architecting Dependable Systems*. Lecture Notes in Computer Science 2677. Springer. Germany. 2003.

[de Lemos et al 2003c] R. de Lemos, C. Gacek, A. Romanovsky. Architectural Mismatches Tolerance. *Architecting Dependable Systems*. Lecture Notes in Computer Science 2677. Springer. Berlin, Germany. 2003. pp. 175-193.

[de Lemos et al 2003d] R. de Lemos, C. Gacek, A. Romanovsky. ICSE 2003 Workshop on Software Architectures for Dependable Systems (Workshop Summary). *ACM Software Engineering Notes 28(5)*. November 2003.

[DiNitto and Rosenblum 1999] E. Di Nitto, D. Rosenblum. Exploiting ADLs to Specify Architectural Styles Induced by Middleware Infrastructures. in Proc. 21st International Conference on Software Engineering (ICSE), pp. 13-22, 1999.

[Egyed et al 2000] A. Egyed, N. Medvidovic, C. Gacek. Component-Based Perspective on Software Mismatch Detection and Resolution. *IEE Proceedings on Software 147(6)*. December 2000. pp. 225-236.

[Gacek 1998] C. Gacek. *Detecting Architectural Mismatches during System Composition*. PhD Dissertation. Center for Software Engineering. University of Southern California. Los Angeles, CA, USA. 1998.

[Garlan et al 2002] D. Garlan, J. Kramer, A. Wolf (Eds.). *Proceedings of the 1st ACM SIGSOFT Workshop on Self-Healing Systems (WOSS'02)*. Charleston, SC, USA. November 2002.

[Garlan et al 2003] D. Garlan, S.-W. Cheng, B. Schmerl. Increasing System Dependability through Architecture-based Self-Repair. *Architecting Dependable Systems*. Lecture Notes in Computer Science 2677. Springer. Berlin, Germany. 2003. pp. 61-89.

[Georgiadis et al 2002] I. Georgiadis, J. Magee, J. Kramer. Self-Organising Software Architectures for Distributed Systems. *Proc. of the 1st ACM SIGSOFT Workshop on Self-Healing Systems (WOSS'02)*. Charleston, SC, USA. November 2002. pp. 33-38.

[de C. Guerra et al 2003a] P. A. de C. Guerra, C. Rubira, R. de Lemos. A Fault-Tolerant Software Architecture for Component-Based Systems. *Architecting Dependable Systems*. Lecture Notes in Computer Science 2677. Springer. Berlin, Germany. 2003. pp. 129-149.

[de C. Guerra et al 2003b] P. A. de C. Guerra, C. Rubira, A. Romanovsky, R. de Lemos. Integrating COTS Software Components into Dependable Software Architectures. *Proc. of the 6th IEEE Int. Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'03)*. Hokaido, Japan. May 2003.

[Issarny et al 1998a] V. Issarny, C. Bidan, T. Saridakis, "Achieving Middleware Customization in a Configuration-based Development Environment: Experience with the Aster Prototype", in Proc. 4th International Conference on Configurable Distributed Systems (ICCDS), pp. 275-283, 1998.

[Issarny et al 1998b] V. Issarny, C. Bidan, T. Saridakis. Characterizing Coordination Architectures According to Their Non-Functional Execution Properties", in Proc. 31st Hawaii International Conference on System Sciences (HICSS), pp. 275-285, 1998.

[Issarny et al 2002] V. Issarny, C. Kloukinas, A. Zarras. Systematic Aid for Developing Middleware Architectures. In Communications of the ACM, Issue on Adaptive Middleware, 45, 6, pp. 53-58. June 2002.

[Klein et al 1999] M. Klein, et al. Attribute-Based Architectural Styles. *Proc. of the 1st IFIP Working Conference on Software Architecture (WICSA-1)*. 1999. pp. 225-243.

[Medvidovic et al 2003] N. Medvidovic, M. Mikic-Rakic, N. Mehta. Improving Dependability of Component-Based Systems via Multi-Versioning Connectors. *Architecting Dependable Systems*. Lecture Notes in Computer Science 2677. Springer. Germany. 2003. pp. 37-60.

[Majzik et al 2003] I. Majzik, A. Pataricza, A. Bondavalli. Stochastic Dependability Analysis of UML Designs. *Architecting Dependable Systems*. Lecture Notes in Computer Science 2677. Springer. Germany. 2003. pp. 219-244.

[Medvidovic and Taylor 2000] N. Medvidovic, R. N. Taylor. A Classification and Comparison Framework for Software Architecture Description Languages. *IEEE Transactions on Software Engineering 26(1)*. 2000. pp. 70-93.

[Oriezy et al 1999] P. Oriezy, et. al. An Architecture-Based Approach to Self-Adaptive Software. *IEEE Intelligent Systems 14(3)*. May/June 1999. pp. 54-62.

[Porcarelli et al 2003] S. Porcarelli, M. Castaldi, F. Di Giandomenico, A. Bondavalli, P. Inverardi. An Approach to Manage Reconfiguration in Fault Tolerant Distributed Systems. *Proc. of the ICSE 2003 Workshop on Software Architectures for Dependable Systems*. Portland, OR. April 2003. pp. 71-76.

[Rodrigues et al 2003] G. N. Rodrigues, G. Roberts, W. Emmerich, J. Skene. Reliability Support for the Model Driven Architecture. *Proc. of the ICSE 2003 Workshop on Software Architectures for Dependable Systems*. Portland, OR. April 2003. pp. 7-12.

[Rodriguez et al 2002] M. Rodriguez, J.-C. Fabre, J. Arlat. Wrapping Real-Time Systems from Temporal Logic Specification. In *Proc. Euro-pean Dependable Computing Conference (EDCC-4)*, 2002, Toulouse, France), pp. 253-270.

[Shaw, and Garlan 1996] M. Shaw, D. Garlan. *Software Architecture: Perspectives on an Emerging Discipline*. Prentice-Hall. 1996.

[Shelton and Koopman 2003] C. Shelton, P. Koopman. Using Architectural Properties to Model and Measure Graceful Degradation. *Architecting Dependable Systems*. Lecture Notes in Computer Science 2677. Springer, Germany. 2003. pp. 267-289.

[Tartanoglu et al 2003] F. Tartanoglu, V. Issarny, A. Romanovsky, N. Levy. Dependability in the Web Service Architecture. *Architecting Dependable Systems*. Lecture Notes in Computer Science 2677. Springer. Berlin, Germany. 2003. pp. 3-36.

[Taylor et al 1996] R. N. Taylor, N. Medvidovic, K. M. Anderson, E. J. Whitehead, J. E. Robbins, K. A. Nies, P. Oreizy, D. L. Dubrow. A component- and message-based architectural style for GUI software. IEEE Transactions on Software Engineering, 22 (6), pp. 390-406, June 1996.

[Tichy and Giese 2003] M. Tichy, H. Giese. An Architecture for Configurable Dependability of Application Services. *Proc. of the ICSE 2003 Workshop on Software Architectures for Dependable Systems*. Portland, OR. April 2003. pp. 65-70.

[Veríssimo et al 2003] P. E. Veríssimo, N. F. Neves, M. P. Correia. Intrusion-Tolerant Architectures: Concepts and Design. *Architecting Dependable Systems*. Lecture Notes in Computer Science 2677. Springer. Berlin, Germany. 2003. pp. 90-109.

[Zarras 2000] A. Zarras. Systematic Customization of Middleware, Thèse de Doctorat de l'Université de Rennes I, English version available from http://www-rocq.inria.fr/solidor/doc/doc.html, 2000.

[Zarras et al 2003] A. Zarras, C. Kloukinas, V. Issarny. Quality Analysis of Dependable Systems: A Developer Oriented Approach. *Architecting Dependable Systems*. Lecture Notes in Computer Science 2677. Springer. Germany. 2003. pp. 197-218.

Chapter 17

[AMSD 2003] A dependability roadmap for the information society in Europe. Accompanying Measure System Dependability. IST Project 2001-37553. 3 parts. 2003. http://www.am-sd.org/

[Jones and Randell 2004] C. Jones, B. Randell. Dependable Pervasive Systems. Submitted to DTI Foresight UK. 2004. http://www.foresight.gov.uk/