Lecture Notes in Computer Science 4028

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Jürg Kohlas Bertrand Meyer
André Schiper (Eds.)

Dependable Systems: Software, Computing, Networks

Research Results of the DICS Program

Springer

Volume Editors

Jürg Kohlas
University of Fribourg
Department of Informatics
Bd. de Pérolles 90, CH-1700 Fribourg, Switzerland
E-mail: juerg.kohlas@unifr.ch

Bertrand Meyer
ETH Zurich
Department of Computer Science
Clausiusstrasse 59, CH-8092 Zurich, Switzerland
E-mail: Bertrand.Meyer@inf.ethz.ch

André Schiper
EPFL, Faculté IC, Station 14
CH-1015 Lausanne, Switzerland
E-mail: andre.schiper@epfl.ch

Library of Congress Control Number: 2006929805

CR Subject Classification (1998): D.2, D.4, C.4, B.8

LNCS Sublibrary: SL 2 – Programming and Software Engineering

ISSN 0302-9743
ISBN-10 3-540-36821-3 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-36821-2 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11808107 06/3142 5 4 3 2 1 0

Preface

Civilization relies on a functioning infrastructure, which is more and more enabled through information and communication technologies (ICT). A dependable information infrastructure is thus crucial for the modern society. On the other hand, information and communication systems belong to the most complex artifacts ever built by mankind. The design and operation of these systems are challenging tasks. Theories, methods and tools which help to master the problems encountered in the design process and the management of operations are therefore of utmost importance for the future of information and communication technology.

In view of the relevance of this topic in computer science, the Hasler Foundation launched in 2002 a research program on "Dependable Information and Communication Systems" (DICS). The call for projects was addressed to all Swiss universities. More than 40 short project proposals were submitted. Of these, 18 were selected for a hearing, in which a subset of these projects were selected and invited to submit complete project proposals. Finally, nine projects were selected for funding by the Hasler Foundation. All these projects were also partially supported by the universities involved as well as by other third parties, in particular by the Swiss National Foundation for Research.

The DICS projects started at the end of 2002 and the beginning of 2003. The members of the project teams met twice at workshops organized by the Hasler Foundation. The first one took place in Münchenwiler on March 16 and 17, 2004, the second workshop, which marked the conclusion of the projects, in Löwenberg on October 13 and 14, 2005. Each time more than 40 scientists participated in the workshop. The present volume documents the results of the DICS research program. Of course, the subject of dependable information and communication systems is not exhausted by this program. Much research is still needed. Therefore at the end of 2005, the Hasler Foundation launched a new program on "Managing Complexity of Information and Communication Systems" (MICS), which is intended as a follow-up and an extension of DICS.

The Hasler Foundation's existing endowments derive from the former Hasler AG (1852-1986), a pioneer of the Swiss telecommunications industry. The foundation is committed to promoting high-level research and education in the field of information and telecommunication systems. DICS is one of the research programs launched and supported by the foundation. We refer readers to the website www.haslerfoundation.ch for further information.

The Hasler Foundation thanks the editors and the authors of the present volume for their contributions. The editors also thank C. Schneuwly for assistance in editing. Finally, we thank Springer for accepting this volume in the prestigious *Lecture Notes in Computer Science*. It is our hope that this volume helps

to encourage further research in the crucial field of dependability of information and communication systems.

May 2006 Jürg Kohlas
Chairman of the Scientific Committee of the Hasler Foundation

Final DICS workshop, 13 October 2005: The participants at the training center of the Swiss Federal Railways in Löwenberg (Photo: B. Meyer)

Introduction

For all the marvels that information technology delivers, we should never forget — and the users of our systems should never let us forget — that as the results we produce become ever more impressive, the burden on us to make them perform reliably grows ever heavier, simply because the more people rely on them the more critical any failure can be. This is the whole topic of dependability, a central issue in all disciplines of systems engineering.

No single publication could by itself do justice to this rich and vibrant field of research; the present volume presents a snapshot of some of the most interesting work being performed on this theme by teams in top Swiss universities. It covers three key aspects of dependability:

- Dependable software
- Dependable computing
- Dependable networks

Following this triple focus, the book consists of three surveys in Part I, one on each of these topics, then a collection of research contributions in Parts II, III and IV; each of these parts is devoted again to one of the topics, in the same order.

As the surveys in Part I will show, merely *defining* dependability is already a significant task. Each survey identifies major issues of dependability and presents the state of the art in providing solutions. Meyer's survey on dependable *software* uses a broad brush to explore the various techniques available for increasing the reliability of software systems, from management standards such as CMMI all the way down to static analysis, proofs and tests. Schiper's survey on the general topic of dependable *systems* introduces the fundamental, application-independent techniques in the field, with a special emphasis on replication techniques and the associated communication issues. The last survey, by Kurant, Nguyen and Thiran, addresses the specific area of IP over fiber networks, clearly essential in the future growth of the Internet and other networks; it focuses on techniques for failure location, protection and restoration.

Part II explores some aspects of *dependable software*. In the first chapter of this part, Arslan, Eugster, Meyer and Vaucouleur describe the current state of the SCOOP method — Simple Concurrent Object-Oriented Programming — and its implementation, designed to bring concurrent programming in its various incarnations (from multithreading to Internet programming) to the same level of abstraction and dependability as its sequential counterpart. The result is integrated in the object-oriented framework of the Eiffel language and fundamentally relies on notions of Design by Contract.

For a related set of applications of increasing importance, and in particular Web services, XML has emerged as the communication vehicle of choice, but in

today's practice remains dissociated from programming languages, introducing a detrimental gap. Emir, Maneth and Odersky present the Scala language and framework which combine XML, Web Services and, again, concurrency in the framework of an object-oriented programming language, allowing a seamless integration of these different aspects under a single notational framework. As the reader will undoubtedly note, the design is based on decisions very different from those of SCOOP, providing the opportunity for interesting comparisons of viewpoints.

It is often difficult in a single step to arrive at a correct software solution for a complex problem; hence the idea of development by successive *refinements*. Baar, Marković, Fondement and Strohmeier explore its application to the stepwise development of object-oriented software by introducing the notion of contract refinement and applying it to an extended example. Unlike the previous two contributions this one uses, as its underlying technology, not a programming language but the UML modeling notation.

UML also underlies the final contribution to Part II, by Buchs, Pedro and Lúcio, devoted to the generation of test directly from specifications, expressed in the Fondue subset of UML; subsetting is indeed necessary for defining a precise semantics.

Part III is devoted to *dependable computing*, i.e., dependability at the middleware or system level. The first paper by Bünzli et al. presents recent advances in the context of group communication. Group communication is an abstraction that allows a distributed group of processes to provide a reliable service in spite of possible failures within the group. Reliability is achieved by replication: group communication provides the adequate communication abstraction among the replicas. The paper addresses various aspects of group communication: protocol frameworks used to build group communication stacks, new architectures for group communication stacks, specification of group communication, verification of distributed algorithms related to group communication and verification of group communication stacks.

The second paper, by Gerlach, Schaeli and Hersch, is devoted to dependability in the context of parallel applications. The authors have built a framework called *DPS* (*Dynamic Parallel Schedules*), based on a flow graph, for the development of parallel applications on a cluster of workstations. The paper describes how fault tolerance has been added to the DPS framework using two techniques: backup threads (stateless and stateful) and checkpoints. A backup thread is mapped on a different node than its primary thread, allowing the computation to proceed in case of failures. Checkpointing allows a long-running computation to be restarted from a state different from its initial state.

The last paper of Part III, by Pautasso, Bausch and Alonso, addresses a similar problem in the context of a virtual laboratory, characterized by long-running and large-scale computations on a cluster. Virtual experiments are typically modelled as workflows. The paper describes the *JOpera* workflow system and focusses on its fault tolerant features. The system is able to adapt to processor failures by rescheduling jobs. The system also tolerates failures within its kernel,

by ensuring that process execution resumes in a consistent state after a failure. Moreover, the kernel is able to automatically adapt its configuration to optimally use the available resources.

Part IV is devoted to *dependable networks*. The paper by Ducatelle et al. addresses the problem of failure location and traffic rerouting in large IP-over-fiber and wireless ad hoc networks. Traffic rerouting takes place once the failure has been located. Two algorithms for failure location are presented, one for IP/WDM (Wave-length Division Multiplexing), the other for wireless sensor networks. In a second part, failure restoration by rerouting is addressed. In the context of wireless sensor networks, failure restoration is done by a routing algorithm inspired from ant colonies.

The paper by Erlebach et al. addresses the robustness of the Internet. The authors point out that the traditional model of the Internet as a graph of autonomous system does not capture accurately the way traffic is routed, an important factor of robustness. Traffic routing mainly depends on economic relationships between autonomic systems. Traffic routing can be incorporated using the *valley-free path model*. However, this model makes the evaluation of the robustness computationally more difficult. Complexity and approximation results for disjoint paths and minimum cuts in that model are discussed. Experimental findings concerning the number of vertex-disjoint valid paths and the sizes of minimal cuts are also summarized.

The last paper, by Albrecht, Kuhn and Wattenhofer, is devoted to peer-to-peer (*P2P*) overlay networks. P2P systems are based on common desktop machines ("peers") distributed over a large-scale network such as the Internet. The focus of most research in P2P systems is the development of an efficient lookup operation: given a key, locate the peer responsible for the key. P2P systems are characterized by a high rate of peers joining and leaving the system (called *churns*). The paper describes a robust P2P system that can cope with such a highly dynamic situation. The idea is to maintain a simulated hypercube, and to adapt to churns by rearranging peers or by adjusting the dimension of the hypercube to the number of peers in the system.

May 2006 Bertrand Meyer, André Schiper

Table of Contents

Part I: Surveys

Part II: Dependable Software

Part III: Dependable Computing

Part IV: Dependable Networks