

Hot Topics

LNCS 3460

Ozalp Babaoglu M
Alberto Montresor
Stefano Leonardi
Maarten van Steen

Self-star Pr in Complex Informatio

Conceptual and Pra

Commenced Publication in 1973

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison

Lancaster University, UK

Takeo Kanade

Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler

University of Surrey, Guildford, UK

Jon M. Kleinberg

Cornell University, Ithaca, NY, USA

Friedemann Mattern

ETH Zurich, Switzerland

John C. Mitchell

Stanford University, CA, USA

Moni Naor

Weizmann Institute of Science, Rehovot, Israel

Oscar Nierstrasz

University of Bern, Switzerland

C. Pandu Rangan

Indian Institute of Technology, Madras, India

Bernhard Steffen

University of Dortmund, Germany

Madhu Sudan

Massachusetts Institute of Technology, MA, USA

Demetri Terzopoulos

New York University, NY, USA

Doug Tygar

University of California, Berkeley, CA, USA

Moshe Y. Vardi

Rice University, Houston, TX, USA

Gerhard Weikum

Max-Planck Institute of Computer Science, Saarbruecken, Germany

Ozalp Babaoglu Márk Jelasity
Alberto Montresor Christof Fetzer
Stefano Leonardi Aad van Moorsel
Maarten van Steen (Eds.)

Self-star Properties in Complex Information Systems

Conceptual and Practical Foundations



Springer

Volume Editors

Ozalp Babaoglu
Márk Jelasity
Alberto Montresor
Università di Bologna
Dipartimento di Scienze dell'Informazione
40126 Bologna, Italy
E-mail: {babaoglu,jelasity,montresor}@cs.unibo.it

Christof Fetzer
Technische Universität Dresden
Fakultät Informatik
01062 Dresden, Germany
E-mail: christof.fetzer@inf.tu-dresden.de

Stefano Leonardi
Università di Roma "La Sapienza"
Dipartimento di Informatica e Sistemistica
00198 Rome, Italy
E-mail: leon@dis.uniroma1.it

Aad van Moorsel
University of Newcastle upon Tyne
School of Computing
Newcastle upon Tyne, NE1 7RU, UK
E-mail: aad.vanmoorsel@newcastle.ac.uk

Maarten van Steen
Vrije Universiteit Amsterdam
Department of Computer Science
1081 HV, Amsterdam, The Netherlands
E-mail: steen@cs.vu.nl

Library of Congress Control Number: 2005925758

CR Subject Classification (1998): C.2.4, C.2, D.2, F.1, F.2, I.2.11, H.4

ISSN 0302-9743
ISBN-10 3-540-26009-9 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-26009-7 Springer Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

Springer is a part of Springer Science+Business Media
springeronline.com

© Springer-Verlag Berlin Heidelberg 2005
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11428589 06/3142 5 4 3 2 1 0

Preface

Information systems can be complex due to numerous factors including scale, decentralization, heterogeneity, mobility, dynamism, bugs and failures. Deploying, operating and maintaining such systems can be not only very difficult, but also very costly. A flurry of recent activity has been directed at this problem, and future information systems are envisioned as self-configuring, self-organizing, self-managing and self-repairing. Collectively, we call these properties self- \star properties.

This book is a “spin-off” of a by-invitation-only Bertinoro workshop on self- \star properties in complex systems which was held in summer 2004 in Bertinoro, Italy. The Self-star workshop brought together researchers and practitioners from different disciplines and with different backgrounds to discuss complex information systems. The theme of the workshop was to identify the conceptual and practical foundations for modeling, analyzing and achieving self- \star properties in distributed and networked systems. Partly based on these discussions, we solicited papers from the workshop participants and a set of invitees for this book.

We sought original contributions in which authors explicitly take a position concerning requirements, usefulness, potential and limitations of technologies for self- \star properties of complex systems. This position needed to be founded on research results that were put clearly in context with respect to the position statement. We strongly encouraged visionary statements, thought-provoking ideas, and exploratory results that will help the reader form her or his own opinions on the importance of self- \star properties in current and future complex information systems.

We structured the book according to our goal of having such visionary statements. The first part of this book contains a set of separate 1-page summaries of the positions taken by the various authors. This gives the reader a chance to get a quick overview of the various positions. The second part of the book contains the full papers that explain in more detail the positions taken by the different authors.

Without further ado, we wish you a pleasant and stimulating read.

Bologna, Dresden, Rome,
Newcastle upon Tyne, Amsterdam
February 2005

Ozalp Babaoglu
Márk Jelasity
Alberto Montresor
Christof Fetzer
Stefano Leonardi
Aad van Moorsel
Maarten van Steen

Organization

Organizing Committee

Ozalp Babaoglu	University of Bologna, Italy
Márk Jelasity	University of Bologna, Italy
Alberto Montresor	University of Bologna, Italy
Christof Fetzer	Technical University of Dresden, Germany
Stefano Leonardi	University of Rome “La Sapienza,” Italy
Aad van Moorsel	University of Newcastle upon Tyne, UK
Maarten van Steen	Free University of Amsterdam, The Netherlands

Referees

Vinay Aggarwal	Márk Jelasity	Aad van Moorsel
Ozalp Babaoglu	Zbigniew Jerzak	Maarten van Steen
Luca Becchetti	Stefano Leonardi	Andrea Vitaletti
Christof Fetzer	Alberto Montresor	Berthold Vöcking

Special Thanks

We would like to thank the sponsors of Self-star 2004 for making the Workshop possible:

FET Open Project BISON

FET Integrated Project DELIS

FET Open Project COSIN

BICI: Bertinoro International Center for Informatics

UNESCO Office Venice, Regional Bureau for Science in Europe (ROSTE)

We are grateful to Dr. Dum of the European Commission for his enthusiasm and support of research in “Complex Systems” through the projects BISON, COSIN and DELIS. We would also like to thank the University Residential Centre of Bertinoro for hosting the Workshop.

Table of Contents

The Self-star Vision	1
----------------------------	---

Self-organization

Evolving Fractal Gene Regulatory Networks for Graceful Degradation of Software <i>Peter J. Bentley</i>	21
---	----

Evolutionary Computing and Autonomic Computing: Shared Problems, Shared Solutions? <i>A.E. Eiben</i>	36
---	----

Self-★ Topology Control in Wireless Multihop Ad Hoc Communication Networks <i>Wolfram Krause,</i>	49
--	----

Emergent Consensus in Decentralised Systems Using Collaborative Reinforcement Learning <i>Jim Dowling,</i> <i>Eoin Curran,</i>	63
--	----

The Biologically Inspired Distributed File System: An Emergent Thinker Instantiation <i>Sergio Camorlinga,</i>	81
---	----

Evolutionary Games: An Algorithmic View <i>Spyros Kontogiannis,</i>	97
--	----

Self-awareness

Model Based Diagnosis and Contexts in Self Adaptive Software <i>Paul Robertson,</i>	112
--	-----

On the Use of Online Analytic Performance Models in Self-managing and Self-organizing Computer Systems <i>Daniel A. Menascé,</i> <i>Honglei Ruan</i>	128
--	-----

Prediction-Based Software Availability Enhancement <i>Felix Salfner, unther Ho mann,</i>	143
---	-----

Making Self-adaptation an Engineering Reality <i>Shang-Wen Cheng</i>	158
An Online Control Framework for Designing Self-optimizing Computing Systems: Application to Power Management <i>Nagarajan K andasamy,</i> <i>John P. Hayes</i>	174
Self-management of Systems Through Automatic Restart <i>Katinka Wolter</i>	189
Fundamentals of Dynamic Decentralized Optimization in Autonomic Computing Systems <i>Tomasz Nowicki,</i>	204
 Self-awareness vs. Self-organization	
The Conflict Between Self-* Capabilities and Predictability <i>Rogério de Lemos</i>	219
Self-aware Software – Will It Become a Reality? <i>Peter Andras,</i>	229
 Supporting Self-★	
A Case for Design Methodology Research in Self-* Distributed Systems <i>Indranil Gupta,</i> <i>Chris Devaraj,</i> <i>es Morales,</i>	260
Enabling Autonomic Grid Applications: Requirements, Models and Infrastructure <i>M. Parashar,</i>	273
Pandora: An Efficient Platform for the Construction of Autonomic Applications <i>Simon Patarin,</i>	291
Spatial Computing: The TOTA Approach <i>Marco Mamei,</i>	307
Towards Self-managing QoS-Enabled Peer-to-Peer Systems <i>Vana Kalogeraki,</i> <i>Demetris Zeinalipour-Yazti</i>	325

Peer-to-Peer Algorithms

Cooperative Content Distribution: Scalability Through Self-organization <i>Pascal Felber,</i>	343
Design and Analysis of a Bio-inspired Search Algorithm for Peer to Peer Networks <i>Niloy Ganguly,</i>	358
Multifaceted Simultaneous Load Balancing in DHT-Based P2P Systems: A New Game with Old Balls and Bins <i>Karl Aberer,</i>	373
Robust Locality-Aware Lookup Networks <i>Ittai Abraham,</i>	392
Power-Aware Distributed Protocol for a Connectivity Problem in Wireless Sensor Networks <i>Roberto Montemanni,</i>	403
Self-management of Virtual Paths in Dynamic Networks <i>Poul E. Heegaard,</i>	417
Sociologically Inspired Approaches for Self-*: Examples and Prospects <i>David Hales</i>	433
Author Index	447

The Self-star Vision

Achieving various self- \star properties has been a grand challenge of computer science and engineering since the building of the first computer. The latest reincarnation of this challenge is due to the fact that large, complex and dynamic information systems have suddenly become a key part of the infrastructure of modern societies. Accordingly, it has become very important to be able to build, manage, and exploit these systems in the most efficient way possible. In other words, these systems have to become self- \star .

We are now in the process of finding out how to deal with this challenge. It is a complex problem because information systems are deployed over a very wide range of environments from wireless sensor networks to powerful supercomputing clusters, from home networks to the entire Internet. It is very likely that to meet this challenge, we need to draw ideas from many disciplines that have been dealing with the design or explanation of large and/or complex systems, such as the space shuttle, an operating system, the Internet, human intelligence, an ecosystem, evolution, living organisms, etc. The goal of this book is to explore the widest possible range of ideas and approaches that can potentially be relevant in understanding how to build self- \star information systems. We invited experts from different disciplines and asked them the same questions:

Why and where do you think self- \star properties are important? How can your field of research contribute to these goals? What are the most promising directions to explore in the future?

This chapter contains a selection of the most visionary answers we received. To make navigation easier, we organized the contributions into the following, rather fuzzy, categories:

Self-organization. A typical approach in this group emphasizes the importance of *emergence* and *self-organization* of fully distributed and independent components. Biological and social (agent-based.) inspiration belongs here. For some fields, like in wireless ad hoc networks, this approach is a must, in other fields it is an interesting alternative. This approach is often also *radical* in that it proposes to build systems from scratch as opposed to enhance existing systems.

Self-awareness. This approach seeks to achieve self- \star properties via a *meta-model* of the system: self- \star is an add-on. An external component controls the system via a control loop, or a decision mechanism based on the (perhaps pre-computed) model of the system and its environment. This approach is often incremental, and therefore more practical, in that it allows the extension of existing systems with little or no modification.

Self-organization Versus Self-awareness. A small but interesting set of approaches explicitly investigate the relation (synergies, differences) between the first two approaches and therefore cannot be classified in either of those.

Supporting Self-*. We included here contributions that aim at developing methodologies, environments, tools, abstractions, systems support; in short, things that make it easier (or possible) to develop self- \star systems. These approaches are often highly ambitious, because their scope is quite generic.

We hope that you will find browsing the visions below as much fun as we have.

Self-organization

Natural Technology

Peter Bentley

Every salamander is a superhero. It is seemingly invulnerable to harm. Cutting off a leg won't stop it. It will grow a new one back, complete with bones, muscles and blood vessels. Cutting off its tail also has no real effect. A few weeks later it will have grown a new one. Try blinding it, and it will regrow the damaged parts of its eye. No stake through the heart or silver bullet will stop our superhero salamander. Damage its heart and it'll regrow that too.

The salamander is doing more than simple healing here. It is able to regenerate parts of itself. Even when major structures (such as a whole leg) are lost, it can regrow them from nothing. It's positively miraculous.

Salamanders can do these things and we can't because their cells have evolved to be a little bit different to ours. They're not the only ones. Plants are also clever in similar ways. You can amputate all of the limbs of a tree and in a couple of years it will grow a new set. Cut a leg (or "ray" as it is called) from a starfish and it will grow a new one — and the ray you've cut off will grow a new body. Snails can even regrow their entire head if they lose it.

Imagine trying to design a piece of technology that could do the same thing. A car that healed itself after a crash. A house that regrew its roof after damage from a storm. A computer that repaired itself after its circuits had been fried. Such ideas are so beyond our current abilities that they seem ludicrous. We can't design such technologies. We don't know how. Not only that, we can't even design technology that is several orders of magnitude simpler and have it work.

There's something going wrong with contemporary design. Complexity is overwhelming us. For certain types of design, it has already overwhelmed us — software is now too complex to work reliably. Economies are too complex to manage reliably. Societies are too complex to govern reliably.

But we are surrounded by complexity that is orders of magnitude more complex than all these examples. Life is designed by evolution at molecular scales and grows to macro scales. Every cell in our bodies contains a molecular computer made from genes and proteins, which instructs the cell what to do. We have hundreds of billions of cells in us. We are the most complex entities in the known universe — and yet we work.

I believe that complexity should not cause our designs to fail. We must look at successful complex designs in nature and learn from them. We should attempt

to discover not only how they work, but how they are formed. Nature does not perform conscious design, but her designs are better than anything a human is capable of designing. I aim to discover how this happens by working with biologists and modelling the processes of evolution and development. I aim to harness these processes by enabling new forms of technology that can evolve and develop.

I strongly believe that one day we will achieve some of the capabilities of living organisms in our technology. We will have self-adaptive, self-designing, self-building, self-repairing, ... self- \star devices. This “natural technology” is not just a pip-dream of ambitious scientists. It is necessary for us to progress in our technology. It will be the only way we can construct complex devices that work reliably in the future.

Evolutionary Computing

A.E. Eiben

The main point advocated here is that evolutionary computing is one of the key technologies that can help meeting some of the grand self- \star challenges. The arguments backing up this point roughly fall into three categories.

The first argument considers evolutionary computing as a technology that can be used to solve optimisation problems. EC is widely applicable, it requires almost no assumptions about the problem to be solved, an evolutionary solver can be usually developed with limited efforts and it produces good quality solutions at acceptable computational costs for a wide range of problems. The EC community describes the “niche” of EC (i.e., the class of problems where it offers competitive performance) by attributes including: many parameters with highly complex non-linear interactions, many local optima, the presence of noise, dynamically changing circumstances (constraints and/or optimisation objectives), the lack of an analytical problem model, etc. Mind you, it is not suggested that EC is always applicable; for instance, automatic translation would be very “EC-hard”.

The second line of argumentation observes that self- \star properties are mainly required in complex, distributed information systems. In such systems the macro-level behaviour is often a result of complex interactions of numerous of micro-level entities (for instance, system components, parameters, configurations, etc). Therefore, population-based adaptive systems (PAS), such as evolutionary algorithms (EA) or adaptive multi-agent systems, can naturally match the essence of the information systems in question. EAs are inherently distributed and if only one can identify the individual units, define their utility, and specify how to create and test new variants on-the-fly, the system is “evolutionarised”. The example about optimising web services discussed in the paper illustrates this. The use of a PAS, in particular an EA, offers specific advantages for self- \star applications. By the presence of a population there is an inherent ability to adapt to changes, which is a key feature in, for instance, self-healing, self-optimisation, self-reconfiguration, etc.

The third argument is based on the fact that EAs are capable to perform real-time self-optimisation by adjusting (some of) their parameters on-the-fly. Solutions to realise this in evolutionary computing, for instance self-adaptation of EA parameters, could be transportable to the context of autonomic computing. Even though an existing EC trick does not work directly, evolutionary weaponry can very well serve as a source of inspiration.

The second point in my paper concerns the relevance of autonomic computing to evolutionary computing. An evolving system can be seen as a special case of an autonomic system and it can be expected that some solutions invented within autonomic computing can be transferred to EC helping to realise parameter-free evolutionary algorithms.

What Is Self-★?

Wolfram Krause and Martin Greiner

Complex self-★ systems are widespread in nature. Maybe the most fascinating example is given by ourselves and automatically leads to the question “What is life?” Here, self-★ reveals itself on various scales: the biochemical feed-back control system within the microscopic cell and, built on top of that, e.g., the immune system and the brain. We should not hesitate to mention also the next rougher scale, which is represented for example by social networks and our interactions as part of the ecosystem. Herewith it becomes clear that nature defines self-★ as the self-organizing control of highly complex, interacting and networked systems.

Recently physics has started a new branch, which will be key to a generic understanding of self-organization in nature and beyond: the Statistical Physics of complex networks. It discusses the structure and formation of complex networked systems, their dynamics and their function. The new insight which along these lines has already been given for example to gene and metabolic networks constitutes for sure a remarkable highlight and complements the current endeavor to understand the proteomic function of the deciphered, but hieroglyphic (human) genome. Applicationwise, this new physics branch not only focuses on complex networks in nature, but also on social, technical and information systems.

Some of the complex technical systems are beginning to ask similar questions about self-organization and function as the networked nature. Examples are production and logistic networks, traffic networks and communication networks. Their parallelism results from a common interplay between a material flow, consisting of either goods, vehicles or data packets, and an information flow. As these technical networks are up-scaling in size, efficiency requires that the control of the flow interplay should no longer rely on a central approach alone. Elements of decentralized control are becoming increasingly important, which first gather and process local information and then use this for a local feedback control action. In the limit of a purely decentralized control such networked systems become self-organizing.

Our research focuses on information systems in general and wireless communication networks in particular. Coordination and synchronization of different

tasks shall be performed in a distributive manner. In this context, self- \star refers to self-configuration, self-management, self-repair, self-adaption and related topics. But these systems are still far from perfect. A deep understanding of the processes in biological systems might be the basis for new technological methods that allow improved system characteristics. Regarding sensor networks, a distributed, parallel processing of collected data can speed up the data acquisition and reduce the traffic load of the network. One might even think about nodes that can act autonomously and start certain actions, based on data acquired at their own sensors or communicated by neighbors.

However, self- \star has also other meanings. The principles of self- \star are playing an important role in a large variety of other research topics. In solid state physics, self-organization is discussed in the context of the “magic” growth of very specific surface structures, even down to the placement of single atoms. The magical creation of clusters and other macro-molecules are other examples for self- \star processes. Delicate tasks, like the assembly of carbon atoms to a fullerene or nanotube, cannot be performed directly. But given certain environmental conditions, self-organization enters the stage and the process *just happens automatically*.

Cooperative Agents

*Jim Dowling, Raymond Cunningham, Anthony Harrington,
Eoin Curran and Vinny Cahill*

Our vision of self-managing distributed systems is based on building decentralised systems using cooperative agents that can adaptively coordinate their local behaviour to solve system self-management problems. In such a model, each agent gathers information on its own and takes independent decisions on how to behave or adapt itself, but in order to establish and maintain self-managing system properties, agents must also coordinate their local behaviour models through interaction with their neighbours or a shared environment.

The design of decentralised algorithms to build self-managing systems presents a number of challenges. One approach is to use decentralised algorithms that enable consensus between groups of agents on optimal, local behaviour models that produce the desired, system self-management behaviour. The main challenge in designing such algorithms is the establishment of common behaviour models in agents, given the inherent uncertainty and dynamism in the environment, and the impossibility of using centralised techniques. Designers must also not resort to pseudo-centralised techniques, such as using an agent’s local view of the system to infer properties of the system as a whole, as such assumptions in dynamic environments may break down quickly. In designing decentralised systems, we have found as much inspiration in the fields of statistics and optimisation as in natural, self-organising systems. Decentralised algorithms will be important in designing self-managing systems areas such as mobile ad-hoc networks, wireless sensor networks, peer-to-peer systems and very large distributed systems.

Specific challenges in designing these decentralised algorithms for self-managing systems include:

- the representation of an agent’s local view of the world
- techniques that improve the accuracy of the agent’s local view using changes in the agent’s environment and changes in neighbouring agents’ local views
- enabling the establishment of consensus on optimal local behaviour models that can produce self-managing system behaviour

To conclude, decentralised self-managing computing systems should establish and maintain self-management properties in dynamic and uncertain environments with minimal external intervention. Finally, we define a self-managing decentralised system as follows: “A self-managing decentralised system has externally observable self-management properties, that are established and maintained solely by the coordination and adaptation of its agents that execute using only a partial, estimated view of the system, and without knowledge of any system-wide self-management property”.

The Emergent Thinker

Sergio Camorlinga and Ken Barker

Systems research is a large research area within computer science. Systems research works on operating systems, distributed systems, networks, systems management, and many other aspects of computer systems. Algorithms and models have been developed that provide feasible solutions to many of the problems encountered in computer systems within constrained environments. However, the increased complexity of computer systems that are brittle limits the applicability of some solutions. The augmented complexity is due to many factors such as the large number of components, component interdependencies and heterogeneity, systems’ ubiquity (i.e. pervasiveness), and nonstop appearance of new technologies; among other factors. This complexity is calling for better solutions that are scalable and reliable assuming continuous growth in the number of components and component technology, while maintaining the quality of service in the pervasive use of computing components. Terms like self-organization, self-configuration, self-monitoring, self-control and many others “self’s” (i.e. self-★ properties) are being defined as system properties that are needed to cope with the new challenges.

An essential question is how to achieve the self-★ properties necessary to provide services in large computing environments? For this we are proposing the *Emergent Thinker* paradigm. The Emergent Thinker provides a shift in the way we achieve computation by means of *Complex Adaptive Systems* emergent computations (a.k.a. swarm intelligence). The Emergent Thinker paradigm is proposed as an alternative approach for new and current design and implementation challenges in systems research. The solutions that emerge from the simple activities of a swarm’s members and the metaphor variety for complex adaptive systems models in our lives (e.g. biological systems, economic systems, ecologies,

brain, social systems, immune systems, etc.) present a fruitful area to explore for mechanisms and processes with emergent computations that provide self- \star properties.

The *Biologically Inspired Distributed File System* (BPD) implements the Emergent Thinker paradigm and corroborates our hypothesis by means of emergent computations achieved by simulating squirrels biological behaviors. The BPD provides file system services in a complex information system environment that is dynamic, self-organized, *ad-hoc*, and decentralized. The BPD is an initial step for the Thinker. Many pieces remain to be investigated and implemented. The Thinker establishes a computing framework and we use it to prove and instantiate the paradigm with the BPD implementation described later in our paper.

The Emergent Thinker paradigm offers an alternative computational model for complex information systems design; and nature presents the mechanisms required for the emergent thinker computations. Mechanisms and processes remain unknown and waiting for us to uncover. The know-how achieved by applying the Emergent Thinker could lead us to build up new computing algorithms and models (some generic, some specific) by uncovering biological, economic, social and others real systems mechanisms. Thus, a new way to provide computing will emerge and change our thinking in systems research for large computer systems.

Evolutionary Games

Spyros Kontogiannis and Paul Spirakis

Evolutionary Game Theory is a field of science that examines strategic (and perhaps antagonistic) interactions among members of a large population of agents. The individuals base their decision on simple rules. The dynamics of such interactions, under certain conditions, tend to converge to the adaptation of certain strategies that are “more stable” than others. Interestingly, these strategies are also robust against invasion. In the general case, evolutionary dynamics may also lead to chaotic behaviour.

We consider the notions of Evolutionary Game Theory as a solid framework that allows a concrete examination of self-organization procedures in a large number of selfish individuals. This theory originated from a fruitful interaction of Biology, Economics, Game Theory and (rather recently) Computer Science. Our work examines a yet quite unexplored but crucial aspect of Evolutionary Game Theory, namely the combinatorial, algorithmic and computational complexity issues involved.

Indeed, the effort of deriving concrete conclusions in any non-trivial model of antagonistic evolutionary behaviour, poses interesting computational questions and complex decision problems. The further, pragmatic growth of the field seems to depend on how efficiently these questions can be answered, or even approximated. This view seems to invite an interaction of the field with the field of Algorithmics and Computer Science. We highlight such issues here and provide initial evidence of the usefulness of algorithmic way of thought in the strategic evolution domain.

Self-awareness

Self- \star : A New Paradigm

Robert Laddaga and Paul Robertson

Software development technology is critically in need of new paradigms supporting increased robustness. Robustness is of great concern now because our systems are becoming more complex, and because they are increasingly sensing and controlling our physical environment and processes. One such paradigm is self- \star , a collection of properties such as self reconfigurable, self adaptive, self aware, and self checking. We believe that all self- \star systems share the following traits:

- Deferral of design decisions to runtime (a form of late binding)
- Metaprogramming
- Explicit attention to state of the world
- Attention to program state

Software design consists in large part in analyzing the cases the software will be presented with, and ensuring that requirements are met for those cases. A design decision in the context of any possible program state, any possible input, and any possible condition of the environment is inherently more complex than deciding what to do given a specific input in the context of a specific state of the program and the environment. Self- \star systems attempt to use various forms of metaprogramming to enable them to defer decisions to runtime, when attention to the state of the world and the state of the program can be used to reduce the complexity that would otherwise be overwhelming.

Self Adaptive Software (our new self- \star paradigm) monitors its own operation and attempts to correct deviations from required behavior. In the self adaptive architectures we are developing, it accomplishes this by diagnosing the sources of deviant behavior, whether internal program problems, or contextual changes in an embedded program's environment. The software then responds by reconfiguring itself, to use alternate procedures that either correct the malfunction, or perform better in the current context. The diagnosis and reconfiguration are in part accomplished by storing models of the goals, structure, design and requirements of a program such that they are available to the program at run time. Since most of our self adaptive systems are also embedded systems, models of the physical plant which the software controls or effects are also stored and referenced at run time.

Self Adaptive Software provides some advantages over some other self- \star approaches, that utilize a more reactive programming technique, and which are inherently less self aware. these advantages include:

- Explicit management of contexts is used to increase robustness and reduce effective complexity
- Existing functions can be "wrapped" and used in self adaptive architectures
- Use of the descriptions and models in the software motivates their production, and they are also valuable for development and maintenance

- It is easier to control (rather than depend on) emergent behavior
- It is easier to make self aware systems explain their behavior

Important open research issues for Self Adaptive Software are the following:

- Providing assurance of software behavior – convergence on correct solutions, stability, limits on emergent behavior
- Providing tools to assist in development of self evaluation code
- Providing tools to assist in development of descriptions and models of software and controlled systems
- Incorporating learning into Self Adaptive systems.

Online Performance Models

Daniel A. Menascé, Mohamed N. Bennani and Honglei Ruan

Modern computing environments are becoming increasingly complex in nature and exhibit highly varying and unpredictable workloads. Such systems have a multitude of parameters whose settings may significantly affect performance. Under these circumstances, it is virtually impossible for human beings to continuously tune a system's configurable parameters. Therefore, these systems must be self-managing and self-organizing.

We present an approach that consists of a *controller* that continuously determines the configuration that optimizes a given goal function, which is typically a function of the system performance metrics. The size of the state space of possible configurations grows in a combinatorial way with the number of controlled parameters. Therefore, the controller uses combinatorial search techniques to find a configuration for which the value of the goal function is as close as possible to its desired level.

The goal function for a system's current configuration is evaluated as a function of performance metric measurements. However, as the search technique explores the configuration state space, the goal values for potential new configurations have to be computed through the use of models that can predict the value of performance metrics for these configurations. We use analytic performance models—typically queuing network models—of the controlled system to obtain the values of these performance metrics for each configuration.

This *online* use of predictive performance models is a departure from their common use in capacity planning, where performance models are used to analyze and compare scenarios over relatively long (in the order of months) periods of time. In the case of self-configuring and self-managing systems, configurations may have to change very frequently (at a few-minute intervals).

Our technique has been implemented and evaluated in many different settings. In the paper presented in this book we show the effectiveness of the controller on a real Web server. We also show that the controller becomes more effective when the frequency of its invocation is a function of the relative error between the desired performance level and the current performance. The controller effectiveness improves even more if it uses workload intensity forecasting

techniques. The techniques discussed here are shown to be robust to high variability of the interarrival time and service time distributions. Finally, we showed how online performance models can be used to design QoS-aware service oriented architectures.

Proactive Failure Prediction

Felix Salfner, Günther Hoffmann and Miroslaw Malek

Massively distributed, heterogeneous hardware-software systems day by day reach higher and higher complexity levels and their behavior is in parts, especially with respect to certain properties such as dependability and security, unpredictable. A naive belief, that non laboratory systems are deterministic, manually manageable or fault free, is a myth. A number of reasons lead to this increased complexity: rapidly increasing size of individual software components, growing heterogeneity of components, decentralization, emerging new technologies such as ad-hoc reconfiguration and accelerated software development through code reuse techniques. Additionally, fault tolerance techniques and performance boosters can introduce stochastic dynamics, which further complicates matters. This has been leading to a change in the way software systems are perceived and to changing concerns from fault centric — *whether* a system will work to *how well* it will work. There is reason to believe that future systems will grow in complexity making them even more failure prone and unpredictable.

In order to sustain an acceptable level of dependability, radically new methods have to be incorporated in addition to a mix of formal methods, attempts to prove correctness or at least consistency and testing. No amount of proving or testing will at present certify the system's correctness. Therefore, we need to learn to coexist with unpredictable systems and learn to react and adapt in case of unpredictable changes.

Self- \star properties have been proposed as a potential solution to this problem. We propose an aggressive, preventive maintenance by developing automatic stochastic software failure prediction methods which provide for downtime minimization, graceful degradation or outright failure avoidance. In our work we focus on self-managing properties, or in other words *anticipating software systems*. We advocate a proactive approach which capitalizes on probabilistic failure prediction by selecting appropriate methods for avoiding the failure (e.g., load decrease, process retry, failover) or for minimizing the recovery time in case of a crash.

Since failures are stochastic events, it is our belief that the system's behavior can be captured by its probabilistic representation. Machine learning techniques are a class of algorithms that are able to handle the complexity of software systems. They are also capable of finding non-obvious relationships in data and identifying suspicious patterns and deviations from normal behavior. We present two approaches which are Universal Basis Functions (UBF) and Similar Events Prediction (SEP). In our study we model and predict failures of a telecommunication system.

Making Self-adaptation an Engineering Reality

Shang-Wen Cheng, David Garlan and Bradley Schmerl

We posit our vision of a software engineering reality where engineers can develop self-adaptive software-intensive systems cost-effectively. Imagine a world where a software engineer could take an existing software system, specify for a set of properties of interest (1) an objective, (2) conditions for change, and (3) strategies for their adaptation and, within a few man-weeks, make that system self-adaptive where it was not before. An engineer might take an existing client-server system and make it self-adaptive with respect to a specific performance concern such as high latency. He might specify an objective to maintain response latency below some threshold, a condition to change the system if the latency rises above the threshold, and a few strategies to adapt the system to fix the high-latency situation.

Systems increasingly require mechanisms to monitor and adapt themselves to failures or surrounding changes, that is, to *self-adapt*. Currently such capabilities are realized in somewhat limited forms through programming language features such as exceptions and in algorithms such as fault-tolerant protocols. These mechanisms are often highly specific to the application and tightly bound to the code, making them costly to build and difficult to maintain once added. Rather than rely on internal mechanisms, we apply a closed-loop control paradigm using external mechanisms to monitor, model, and control a running system. We further use architectural models to get global system perspective and system constraints, and architectural styles to give us leverage on analysis and guidance for self-adaptation.

To achieve self-adaptation as we envisioned, in addition to mechanisms for monitoring, techniques for diagnosis and problem correction, and capabilities for run-time reconfiguration, we need to make it possible for engineers to use these in cost-effective and principled ways. In particular, we would like to be sure that engineers can augment existing systems to be self-adaptive without having to rewrite them from scratch, that self-adaptation policies and strategies can be used across similar systems, that multiple sources of adaptation expertise can be synergistically combined, and that all of this can be done in ways that support maintainability, evolution, and analysis.

We show how our Rainbow approach, supported by three case studies, fulfills the important properties of *generality*, *composability*, and *tailorability*. Specifically, we focus on the separation between the general parts of Rainbow that can be applied across different styles of systems and different concerns, and the tailorable parts that need to be written to apply Rainbow to specific systems and concerns.

Control-Theoretic Concepts

Nagarajan Kandasamy, Sherif Abdelwahed, Gregory C. Sharp and John P. Hayes

Our ongoing research effort focuses on the theory and practice of designing self-managing distributed computing systems. To operate such systems effectively, multiple performance-related parameters must be continuously tuned to dynamic

operating conditions, and the current state-of-the-art involves substantial human effort. To cope with the complexity expected of future computing systems, it is highly desirable for such systems to manage themselves, given high-level quality-of-service (QoS) objectives from administrators.

Our position is that control-theoretic concepts are applicable to selected and important resource management problems in computing systems, including self-optimization where the system aims to improve its performance and efficiency continuously. Control theory provides a systematic approach to resource management in general settings; if the underlying system is correctly modeled and its operating environment accurately estimated, the actions required to maintain a set QoS level and/or optimize a given utility (cost) function can be derived. It also provides well-established mathematical techniques to analyze system performance and correctness.

We are currently developing online control techniques that allow multiple QoS objectives and operating constraints to be explicitly represented as an optimization problem and efficiently solved at each time step. This approach can manage systems exhibiting both simple and complex nonlinear dynamics. Our research also studies how to build self-healing capabilities within the control framework where the system reconfigures itself in response to hardware (software) resource failures. Stability and feasibility analysis of these control algorithms is another important part of this work. We have, so far, applied this approach to problems such as power management in microprocessors and real-time data processing under a dynamic workloads with promising results. We plan to extend the approach to tackle other important resource management problems in distributed systems such as energy-efficient load balancing in server clusters and dynamic resource provisioning between multiple applications in data centers, among others.

If successful, this research will result in systematic and theoretically sound techniques to design and operate large-scale self-managing computer systems.

Restart

Katinka Wolter

Modern information systems are becoming increasingly complex, powerful and at the same time very widely used. They are being used not only by experts, but in first place by people who quickly want to buy a train ticket, people who want to look up something on the Internet, or people who must write a text. Most users want to have their service delivered by the computer system, but they do not want to know why and when and how the system works.

Obviously, every system will not always work. Computer systems need maintenance, repair and management. Very often, nowadays, some of these can be done remotely, facilitating this task enormously for customers and management service providers.

If the deployment of modern computer systems will continue to spread as it did in the past, management has to become a very simple and easy task. Otherwise there will soon be not enough people available to support all existing computer systems. To what extent a system has 'self' properties will become

a very important characteristic. And 'self' capabilities of systems include as an essential part easy solutions for complex problems.

Our field of research, the 'black-box' restart of jobs is an extremely easy solution for an extremely hard problem. The addressed problem is the reduction of long job execution times. We do not try to understand why jobs take extremely long or fail, we only want to know for a given job, whether it is of the 'extremely long' type. This reduces the problem complexity enormously and makes it amenable to a simple solution such as restart. Such simple and pragmatic solutions are the only way to handle complex information systems in the future.

We take in our work a pragmatic view and propose an algorithm that is designed for one metric, but works for arbitrary completion time problems. It provides on-line a recommendation for when a job should be aborted and restarted, so that system performance is maximised. This will in many cases solve the problem of 'pending jobs' and we see it as a first step on the way to automatic, or semi-automatic system management.

Restart cannot replace all system management, but it can solve some management problems, so that in those situations not even someone is needed to analyse the system and its problem. Some of the management issues are transferred from people to the system itself.

Mathematical Foundations

Tomasz Nowicki, Mark S. Squillante and Chai Wah Wu

Many original concepts and ideas for self- \star properties of complex information systems have come from the natural sciences where there are various examples of self-managing systems. An example, the autonomic function of the human central nervous system is one often cited inspiration. Hence, in the same way that mathematics plays important roles in the natural sciences, mathematical methods must provide the foundations for self- \star systems. In particular, mathematical methods must be exploited in a manner that is as central as it is in, e.g., physics, to best achieve the diverse goals of providing self- \star properties. This is required in at least 3 fundamental areas: The models and methods to analyze and understand self- \star systems; The laws and properties that characterize and predict the complex dynamics of any aspect of a self- \star system; The algorithms and policies to control, manage and optimize any aspect of a self- \star system and its complex dynamics to achieve any objective of interest.

This fundamental and pervasive role for mathematical methods in self- \star systems is important for a variety of reasons. Many different issues and problems associated with self- \star systems essentially reduce, either in part or as a whole, to a much smaller number of fundamental mathematical problems. The corresponding solutions are required across a wide range of temporal and spatial scales in order to meet a common overall goal. These and other complexities of current and future self- \star systems pose new challenges whose solutions must be based on sophisticated mathematical methods. We simply cannot afford approaches based on forcing elementary solutions that do not fit or are inappropriate, as the

consequences of doing so will be disastrous; e.g., consider the chaotic behavior exhibited in various markets which share some of the characteristics of self- \star systems. Furthermore, new mathematical results and methods must be developed as needed, and these mathematical results/methods must also be tailored to the data available and time scale(s) of the problem at hand. These solutions are often best obtained through a combination of different mathematical methods working together in a unified manner. They must also balance the quality of solution with its costs.

In support of this position, our study investigates the mathematical foundations of two fundamental aspects of management in self- \star systems: the optimization of the entire range of autonomic system objectives, and the dynamic control of achieving these optimal solutions. We first establish several important results regarding decentralized optimization, including showing that there is no loss of quality in the (static) solution obtained under a decentralized approach versus a centralized approach. We then turn to study the dynamics of this system in which optimization decisions are made continually over time and at multiple time scales. Our analysis illustrates some of the potential problems and complicated behavior of such continual decentralized optimization when the system environment changes over time, which can include phase transitions, chaos and instability. One of the fundamental problems then is to determine this complex interaction between dynamics and optimization as we consider in our study.

Self-awareness Versus Self-organization

The Conflict Between Self- \star Capabilities and Predictability

Rogério de Lemos

This paper takes the position that autonomy (the basis for enabling self- \star capabilities) and predictability are conflicting aspects when building systems. Starting from the premise that there is a dichotomy on how systems are described, either process or data, this paper argues that uncertainties associated with these forms of system description dictate the ability that a system has in adapting to changes. The difference between process and data representations can be interpreted from the perspective of accuracy and precision. While process descriptions might be precise but not accurate, data descriptions might be accurate but not precise. In process descriptions, the assumptions that allow a process to be realizable introduce uncertainties. However, when these assumptions are discharged more accurate models can be obtained, thus eliminating uncertainties from the system. On the other hand, data descriptions are an abstraction of the actual behaviour of the system, and in some cases for the data to be meaningful it has to undergo through some generalizations. These two issues inevitably lead to the introduction of uncertainties on how systems are described, and it is from these uncertainties that emergent behaviours materialize.

The above argument is supported by two examples on how process and data descriptions of systems can handle predictability and autonomy. In the first ex-

ample, we present an architectural solution based on exceptional handling to tolerate faults. The solution relies on a process description for building adaptable, but deterministic systems. Uncertainties are eliminated from the system behaviour, however the solution is not scalable since exceptional handling based solutions are invariably application dependent, i.e., there is no single mechanism that is able to deal with a general class of faults. For example, it would be feasible to apply such architectural solution to handle intrusions because of the uncertainties associated with these. In the second example, we present an artificial immune system solution (AIS) for the detection of anomalies. The solution relies on a data description for generating error detectors that are able to identify new unexpected circumstances. The rationale associated with this approach is that if systems are to be autonomous when reacting to changes, in this case undesirable, then it is essential that the system should be able to recognise new erroneous states, and adapt its set of detectors accordingly. In this particular context, it has been observed that the generalisation of potential detectors has lead to a decrease on the detection coverage, and an increase in the number of false positives, i.e., false alarms.

The idea of developing systems that rely on both process and data representations, which explores the complementary benefits of these, is not new. Such hybrid systems have mostly been confined to stand alone closed systems, however the challenge ahead is whether the same idea can be applied to more complex systems that are open and collaborative in their nature, and which are expected to show self-★ capabilities and be predictive at the same time.

Self-aware Software

Peter Andras and Bruce G Charlton

Research on self-aware software systems is an active part of computer science almost since its inception. This research led to many interesting theories and applications, but no truly self-aware software system has been developed so far. It appears that a critical barrier that could not be overcome is that software systems are unable to generate appropriately adaptive responses in previously unknown situations.

One possible way towards the development of self-aware software is to imitate natural self-aware systems. The theory of abstract communication systems, built on works of Niklas Luhmann about abstract social systems, provides a very effective framework for the analysis of such systems. Analysing natural self-aware systems, like cells or organisations, reveals features that are critically related to their self-awareness abilities. Such features are: (1) they possess both short- and long-term memories, (2) they have an information subsystem, which processes and creates new memories; (3) they have a set of long-term memory communications representing a self-model that is referenced by identity check communications of the information subsystem; (4) their self-model is adaptive and is changing in response to faulty communications, errors and failures experienced by the system.

We see computer software as a communication system of many components executed on computer hardware. The software in this context is the set of interactions between communication units (e.g., objects). In our view software systems should have comparable features to natural self-aware systems in order to become truly self-aware. Software systems need to expand their memory communications, by creating memories of most interactions between software communication units. The extensive sets of memories will provide the foundations for the development of the information subsystem of the software (see aspect oriented programming). Self-monitoring should be based on memory communications and on identity-check communications of the system. The software system itself should emerge to large extent from identity-check communications. Self-aware software systems should aim to reproduce and expand themselves as communication systems. They should perform their functionality by adapting to their environment and reproducing and expanding within this environment. The software should adapt its self-model, the code of itself, in response to faulty interactions, errors and failures experienced by the system. Building such systems will need a novel 'from within' approach to software development.

Supporting Self-★

Design Methodology

Indranil Gupta, Steven Ko, Nathanael Thompson, Mahvesh Nagda, Chris Devaraj, Ramsés Morales and Jay A. Patel

Today, designing new protocols for self-★ distributed systems such as peer-to-peer systems, autonomic Grid applications, etc., is an extremely challenging task. The only resources available to a researcher designing new protocols are her basic distributed systems knowledge, prior research literature, and the designer's experiences. This "seat of pants" approach to protocol design has resulted in long research project timelines, long lag times to production, and complex final system designs when pieces are assembled.

We believe these shortcomings can be addressed for future systems by populating and enriching a new resource for the protocol designer – Protocol Design Methodologies. Loosely, a protocol design methodology is *an organized, mented set of building blocks, distributed protocols. It is possibly amenable to automated code generation.*

Given a distributed computing problem then, a collection of methodologies can be brought to bear, for either innovating novel protocols, or for composing existing protocols. This results in a more *systematic* approach to protocol design. It augments the creative activity of protocol innovation, rather than stifle it. Methodology research has already matured in other fields such as hardware synthesis, operating systems, and software engineering, etc.

Methodologies for distributed systems can be either *innovative*, i.e., create novel unknown protocols, or *composable*, i.e., combine existing protocols, thus enriching their properties. Methodologies can also be have retroactive or progres-

sive, i.e., they can help in understanding design principles of existing protocols, or create new protocols (or both).

For instance, many protocols derived from natural phenomena suffer from hand-wavy descriptions and analyses. This is due to the lack of systematization. We have created a new innovative and progressive methodology that translates systems of differential equations (that can be used to represent a natural phenomenon) into equivalent distributed protocols. The use of a methodology guarantees that the protocol can be specified formally, and its self-stabilization and fault-tolerance properties can be proved.

Many of the creative distributed protocol ideas, designed by the community over the years, are either never read or never used in a real system. Retroactive and composable methodologies can help systematize the understanding of large classes of protocols, increasing not only chances of their use but also the possibility that they will be composed with other popular protocols.

Autonomic Grid Applications

M. Parashar, Z. Li, H. Liu, V. Matossian and C. Schmidt

Pervasive information and computational Grid environments are inherently large, complex, heterogeneous and dynamic, globally aggregating large numbers of independent computing and communication resources, data stores and sensor networks. Furthermore, emerging Grid applications are similarly complex and highly dynamic in their behaviors and interactions. Together, these characteristics result in application development, configuration and management complexities that break paradigms based on passive components and static compositions and interactions, and impose new requirements on programming systems for Grid applications. Grid programming systems must be able to specify applications that can detect and dynamically respond during execution to changes in both, the state of execution environment and the state and requirements of the application. This requirement suggests that: (1) Grid applications should be composed from discrete, self-managing components, which incorporate separate specifications for all of functional, non-functional and interaction-coordination behaviors. (2) The specifications of computational (functional) behaviors, interaction and coordination behaviors and non-functional behaviors (e.g. performance, fault detection and recovery, etc.) should be separated so that their combinations are composable. (3) The interface definitions of these components should be separated from their implementations to enable heterogeneous components to interact and to enable dynamic selection of components.

Addressing these challenges requires redefining the programming framework to address the separations outlined above. Specifically, it requires (1) static application requirements and system and application behaviors to be relaxed, (2) the behaviors of elements to be sensitive to the dynamic state of the system and the changing requirements of the application and be able to adapt to these changes at runtime, (3) required common knowledge be expressed semantically rather than in terms of names, addresses and identifiers, and (4) the core enabling middleware services be driven by such a semantic knowledge.

In this paper we first investigate the challenges and requirements of programming Grid applications, and present self-managing applications as a means for addressing these requirements. We then introduce Project AutoMate, which investigates autonomic solutions, based on strategies used by biological systems, to realize Grid applications that are capable of managing (i.e., configuring, adapting, optimizing, protecting, healing) themselves.

System-Level Support

Simon Patarin and Mesaac Makpangou

Recently, autonomic computing has received much consideration and many efforts have been put in the various aspects of this emergent domain. However, several years after the identification of this research area (the term “autonomic computing” was first coined in 2001, while “trouble-free systems” had been already mentioned in 1999), autonomic applications have still not modified our day-to-day relationships with a computer.

Although some progresses have made their way into grid computing and enterprise-class software, the standard end-user is left behind. What should expect end-users from a forest fire application, an oil-reservoir application or a self-healing cluster (all examples taken from the most recent literature)? What about an autonomic Web server, an autonomic proxy-cache, an autonomic mail server or an autonomic peer-to-peer file-sharing application instead? This fact is rather paradoxical as the former applications seem much more complex to apprehend rather than the latter ones. Perhaps, this could be risen as an explanation: complex, unpredictable, applications are required to exercise autonomic systems appropriately. But, we do not believe it to be true. Internet is more than enough complex and unpredictable so that any distributed application is a good candidate to demonstrate autonomic capabilities.

According to us, it is a good system-level support for autonomic applications that is currently missing. One that would present the right abstractions to the developers, which would allow them to prototype autonomic applications rapidly. One that would be flexible enough to cope with the diversity and the heterogeneity of current platforms. One that would be efficient in terms of performance, because you simply cannot pretend maintaining any sort of quality of service if the service you propose is of poor quality right from the beginning. And one that would keep simple applications easy to implement. It is our belief that such a basis would allow researchers with different motivations and experiences to put their ideas in practice, free from the painful details of low-level system implementation. As the first simple autonomic elements become available, it will be easier and easier to build more complex system or to plug one’s strategy into an existing element: a bottom-up development strategy does seem appropriate in building autonomic applications.

After having defined how autonomic systems should work and outlined possible approaches, time has come to see them actually working and to make them available to the largest possible audience. We are coming very close to systems

that “just work”, all we need now is the right system support to federate current efforts and reify them.

Spatial Computing

Marco Mamei and Franco Zambonelli

A number of approaches to support self- \star properties in computing are being proposed since the past few years. In general, we fully agree on the opinion that future computing systems will have to exploit self- \star properties in nearly all of their facets: self-configuration, self-tuning, self-healing etc. Whether you call it proactive computing or autonomic computing or — better and more comprehensive — “self-ware”, it is becoming rather clear that the intrinsic complexity and decentralization of today and future computing scenarios requires humans to be out of the loop. Any approach that requires software and complex network systems to be “manually” managed by human will soon become practically and economically unfeasible. In this context, most of current scientific and technological researches on “self- \star ” computing propose special-purpose and specially-tune solutions to specific kinds of problems. Most of these works are indeed interesting and are providing useful insights on the general problems.

However, we think that to effectively leverage self- \star approaches from a scientific curiosity to both a sound science and a practical engineering discipline we must also definitely look for general purpose approaches and solutions. Such general purpose approaches should provide a uniform set of abstractions and tools for deploying a variety of self- \star properties in a variety of heterogeneous computing scenarios that are emerging. These include: (i) micro-networks, i.e., networks of low-end computing devices typically distributed over a geographically small area (e.g., sensor networks and spray computers); (ii) ubiquitous networks, i.e., networks of medium-end devices, distributed over a geographically bounded area, and typically interacting via short/medium range wireless connections (pervasive computing systems, smart environments and cooperative robot teams); (iii) global networks, characterized by high-end computing systems interacting at a world-wide scale (the physical Internet, the Web, P2P networks and multiagent systems).

Starting from such a motivation, our current research work is aimed at identifying the role that can be possibly played by spatial abstractions and by their adoption as building blocks for a novel general-purpose “spatial computing” approach for distributed system development and management. A spatial computing approach — by providing application components with an explicit representation of their operational environment in terms of a space encoding some application-specific features, and by having application level activities expressed in terms of sensing the properties of space and navigating in it — can effectively deal with network dynamics in large scale systems, can facilitate the integration of variety of self- \star properties in distributed systems, and also suit those systems whose activities are intrinsically situated in an environment. More important: (i) a spatial computing approach, by providing application components with an abstract — high-level perspective of the operational environment, appears suitable

for a wide range of heterogeneous scenarios; (ii) most of current approaches to self-ware in distributed systems can be easily mapped into phenomena of spatial self-organization, thus making to model in spatial computing terms a variety of very diverse self- \star approaches. Our paper included in this book elaborate around spatial computing, its relations with self- \star approaches, and sketches some of our current research work in the area of “spatial computing middleware”.

QoS-Enabled Peer-to-Peer Systems

*Vana Kalogeraki, Fang Chen, Thomas Repantis and
Demetris Zeinalipour-Yazti*

Current efforts on P2P systems have focused on organizing the nodes in the network, improving resource usage, minimizing network latencies and reducing the volume of unnecessary traffic incurred in the P2P overlays. These have shown that P2P systems have been used successfully in the context of large scale file systems, resource sharing, multicast and information retrieval. Thus far, most of the work has concentrated in the sharing of “small” objects including MP3 music files, images, and audio.

Our position is that we can support distributed applications with Quality of Service (QoS) demands on Peer-to-Peer (P2P) systems. It is our belief that by exploring the self- \star properties such as self-organization, self-configuration and self-monitoring of the nodes of the P2P infrastructure and the necessary provisions, we will simplify the management and be able to support distributed applications that have QoS demands.

We believe that the decades of research in middleware technologies will help to achieve these goals. Examples include OMG’s Common Object Request Broker Architecture (CORBA), Microsoft’s Distributed Component Object Model (DCOM), Sun’s Java Remote Method Invocation (RMI) and the Simple Object Access Protocol (SOAP). These represent mature, extensive and portable infrastructures that simplify the management of the applications and enable them to interoperate independently of their computing platforms and networking protocols. This work should be fully considered when developing distributed applications in P2P systems.

However, there are significant challenges that must be addressed. These applications demand multiple end-to-end QoS guarantees, such as predictable latency and jitter, reliability, and scalability. Large-scale environments have unpredictable latencies and changing resource availability, thus require systems that are easy to manage and able to adapt to dynamic changes in the utilization or availability of the resources.

To achieve our goals, we need to explore mechanisms for managing local resources, prioritizing application requests and propagating resource and timing measurements system-wide, and adaptive self-organization algorithms that improve application latencies and balance the load across multiple peers to meet the application end-to-end soft real-time and QoS requirements. These will help us to achieve high levels of performance and scalability and truly create self-managing QoS-enabled P2P systems.