



Contract Number 034051

InterLink

International Cooperation Activities in Future and Emergine ICTs

Coordination Action

Priority 2 Information Society Technologies

Deliverable Number D3.1

State of the Art for the Engineering of Software-Intensive Systems

Lead contractor for deliverable: LMU

Author(s): Matthias Hölzl and Martin Wirsing

Date of preparation: 31 January, 2007

Revision: Final

Dissemination level: PU

Start date of the project: October 1, 2006

Duration: 30 months

Introduction

A growing range of products and services from all sectors of economic activity, our national infrastructure, our daily lives, they all depend on software-intensive systems. The growing productivity and the resulting increase in prosperity can to a large degree be attributed to developments in ICT [DGK03].

The ongoing decrease in size and cost of microprocessors and storage devices is leading to the development of increasingly distributed and decentralized systems. Systems are assembled as dynamic federations of autonomous and evolving components instead of monolithic applications, they perform tasks of staggering complexity with continuously changing requirements and in a permanently evolving environment. In the near future novel technologies will allow the construction of systems with millions of nodes, and systems will be likely to contain subsystems based on novel computing paradigms like quantum computing.

It is imperative that we develop the engineering techniques to reliably design, develop and deploy these novel systems. As a first step towards this goal we attempt to summarize the current state of the art in the area of engineering software-intensive systems. A forthcoming report will develop a research agenda.

Objectives

This report is principally concerned with summarizing current approaches for engineering software-intensive systems and clarifying the challenges that software-intensive systems face. We will briefly describe possible future directions for research in the field of software-intensive systems in Section 11; a more detailed assessment will be the topic of the forthcoming research agenda.

The area of “software-intensive systems” encompasses many large and active fields of industrial practice and academic research, ranging from systems and software engineering to network technologies and hardware. It is also closely connected to fields like control theory and cybernetics; some research directions for software-intensive systems are directly or indirectly inspired by chemistry, biology, the social sciences and many other areas. Given the enormous scope of the field it is obviously impossible to detail the state of the art of the whole field, or even to produce an exhaustive survey of all concerned areas. On the other hand, just focusing on a few areas that we consider important or interesting would not serve the purpose of a state-of-the-art report. We have therefore tried to reach a compromise by surveying all topics that are directly relevant to the engineering of software-intensive systems and giving enough pointers to the literature that the interested reader can explore the topics in more depth.

The “state of the art” presented in this report can roughly be divided into three categories:

- *State of the practice*: the processes, techniques and tools used to develop software-intensive systems.
- *Incremental research*: research about software-intensive systems in areas which are believed to be well-understood, where the direction of the research is relatively clear and most research contributions are of an incremental nature.
- *Research frontier*: research areas where we may have an idea of goal we are trying to reach but where we do not yet know how to reach that goal and whether the desired result is achievable.

We have addressed all three categories in this report although, because this report is meant to serve as a foundation for the forthcoming research agenda, we have focused most of our effort on the research frontier.

Related Work

One of the most important European publications in the state-of-the-art assessment of software-intensive systems is without doubt the ITEA Technology Roadmap for Software-Intensive Systems [Inf04]. This

report structures the field of software-intensive systems into four *software-technology clusters*, each with several *technology categories* and presents an overview of the state of the art, and short term, mid term and long term challenges for technologies in these categories.

Another important influence on this report was the work of WG6 of the IST-FET Coordinated Action “Beyond-the-Horizon.” The final report [WH07] identifies several important research directions and topics in the field of software-intensive systems.

One important resource for the state of the practice was the IEEE Software special issue about this topic [Gla03] which contains relevant articles about variations in software development practices, the different software development approaches adopted in different countries [CMKC03], the use of documentation [LSF03], requirements engineering [NL03], software reviews [CLB03], product-line engineering [BHJ⁺03], embedded software engineering [GLT03], “Internet speed” software development [BRL⁺03], and the relationship between state of the art and state of the practice [Rei03]. These reports are particularly valuable since most published literature is concerned with “best practice” and not the “actual practice.”

Acknowledgements

This work was partially funded by the IST Coordinated Action Interlink “International Coordination Activities in Future and Emerging ICTs” (034051). Without the input of the members of the Interlink WG1 “Software-Intensive Systems and New Computing Paradigms.” this report would not have been possible and we gratefully acknowledge their contribution.

Interlink WG1 Members

- Jean-Pierre Banâtre
- Gabriel Ciobanu
- José Fiadeiro
- Pascal Fradet
- Jean-Louis Giavitto
- Fausto Giunchiglia
- Seth Goldstein
- Teruo Higashino
- Matthias Hölzl
- Stephan Jähnichen
- Insup Lee
- Zhiming Liu
- Vincenzo Manca
- Oscar Nierstraß
- Mike Reed
- Wolfgang Reif
- Jefferey Sanders
- Heinrich Schmidt
- Graeme Smith
- Darko Stefanovic
- Carolyn Talcott
- Christof Teuscher
- Martin Wirsing

Structure of this Report

We have structured the topic of this report into two main parts: foundations and engineering. In the first part we present properties of systems and areas of research which are relevant to the engineering of software-intensive systems, in the second part we focus on the engineering process itself, in particular on software engineering processes, tools and techniques. The distinction between the two parts is not precise and does not represent a judgement on the importance of the individual areas. For example,

security might as well have been in the engineering part as security aspects pervade many areas of systems engineering, and clearly it is one of the most important properties for many systems.

In Chapter 11, we summarize the results of both parts and present some future research topics that we consider particularly important.

Part I

Foundations

The following topics cover important foundations for the engineering of software-intensive systems which have a direct influence on the engineering process. Progress in research or practice in one of these areas impacts the engineering of systems. We have structured Part I as follows:

- *Devices*: the individual nodes present in a software-intensive system.
- *Content*: data, information and knowledge which is acquired, stored and processed by the system.
- *Interoperability and Interaction* between systems as well as between humans and systems.
- *Adaptation* to new environments, new requirements or new regulations.
- *Assurance*: Quality of Service and Experience, Security, Dependability

These areas are not a complete partition of the foundations for software-intensive systems: on the one hand there is a certain amount of overlap between the individual areas, on the other hand there are many links between the areas and it is not always clear into which area an individual research challenge falls. Therefore the division should be understood as a presentational device, not as a proposal for a taxonomy of the field. Our division is slightly different from the one adopted in the ITEA report [Inf04], reflecting the facts that we are slightly less focused on embedded systems and that this report is planned as a precursor to a research agenda for long-term research, whereas ITEA has a stronger focus on short or mid-term research.

Furthermore we focus almost exclusively on technical matters which impact the development of software for systems; we do not address other topics that clearly play an important role in the engineering of systems, such as legal issues, the psycho-social context, or hardware design.

1 Devices

The development of software-intensive systems is in large parts driven by improvements in the available hardware for programmable controllers and computers: the dramatic decrease in cost, size, and energy consumption of integrated circuits and the availability of computational elements based on new principles enables engineers to control devices with software for which this was not economically feasible only a few years ago; the widespread integration of software in devices in turn enables novel features for many classes of devices, such as network connectivity for household appliances.

The use of software as control mechanism has many advantages for the development of systems, as software can easily implement complex control behaviors that would be difficult to realize in hardware. Furthermore, software is more malleable than hardware and can therefore better serve as the “glue” between individual components. On the other hand, the increasing use of software in systems poses some novel problems: the more complex behaviors of individual components may lead to more difficulties in determining and controlling the overall system behavior and to unintended interactions between components.

In the following sections we examine the state of the art of traditional micro-controllers and CPUs, of micro- and nano-scale devices, and of synthetic biology. We also give a very short overview of devices based on novel computing paradigms and sensors.

1.1 Miniaturized Devices

In the last 40 years the number of transistors that can be (inexpensively) placed on an integrated circuit has been growing exponentially [Moo65], and it is expected that this trend will continue for some more time to come [Sib00]. This has led to very inexpensive microprocessors and micro-controllers that enable designers to use flexible software-based control for devices which were previously purely mechanical and enabled many new devices, such as cellular phones or PDAs. In addition to having more devices with computer control, computers in embedded devices are becoming increasingly more powerful [Coa06]. A state-of-the-art assessment for integrated circuits is beyond the scope of this report but state-of-the-art reports for various application domains are readily available, e.g., [Ben06]. The increasing integration of computational elements with physical artifacts is sometimes called cyber-physical systems [Lee06], an overview of the state-of-the art and planned research is [NSF06].

The increasing number of transistors on integrated circuit also means that support for multiple parallel threads or processes is becoming more prevalent on desktop computers [URv03], techniques like hyper-threading or multiple cores are available on most current microprocessors for desktops and servers.

The trend towards more integration is taken even further in the area of miniaturized devices with the integration of whole systems or network into a single device, called Systems-on-Chip (SoC) or Networks-on-Chip (NoC) [BM06]. Micro-fabrication technology is sufficiently advanced that systems-on-chip can be built which consist of electronics, sensors and actuators on a single silicon substrate. In this case they are commonly called Micro-Electro-Mechanical Systems (MEMS) [Lys02, GEH01, All05, GEH06, Mem08]. MEMS are currently used in products as varied as accelerometers, gyroscopes, or the print-heads of ink-jet printers, to name only a few areas. Some of the research areas currently pursued in the MEMS sector are improvements in micro-fabrication technology, both for high- and low-volume production of MEMS, as well as packaging of MEMS.

The large number of chips deployed in systems means that the issue of power consumption becomes more important. Devices which consume less energy are not only cheaper and more environmentally friendly they also provide advantages from a purely technical point of view, as they generate less heat and require less battery power. Many power reduction techniques for microprocessors are currently employed and developed, [VF05] presents an overview of the state of the art in the area. The development of power sources for mobile and wireless systems is a task which is actively pursued both industry and research, partly by improvements to the power density of batteries and partly by the development of novel power sources such as fuel cells.

Further miniaturization of systems can be expected from the emerging field of nanotechnology and the corresponding area of Nano-Electro-Mechanical Systems (NEMS). In contrast to microelectronics there are, as yet, few commercial applications of nanotechnology outside of nanomaterials. Nanoelectronics can be used to build miniaturized replacements for components like FPGAs out of nano-material [RT07], but also to improve certain aspects of more traditional design, e.g., interconnects in integrated circuits [MN06]. As with microelectronics, a state-of-the-art assessment of nanotechnology is beyond the scope of this report.

There are several implications that increasing miniaturization has for the area of system design: the most obvious and visible aspect is that systems now generally contain several or even large numbers of controllers and programmable components, with widely differing computational power, see Section 7. Another trend is the inclusion of sensors in systems, see Section 2.1.

An important trend is the increasing adaptation of systems and components to their particular task and the flexible assembly of parts with limited functionality into more powerful ensembles. Examples are reconfigurable computing [CH02], micro-robotics, and claytronics [GM04, KAG⁺07, RGG07].

The increased number of system components and the inherent defect rate in the production of micro- and nano-scale systems mean that future systems will have to deal with a higher percentage of inoperative nodes, and more frequent failures of individual nodes in the course of their operation. Inoperative nodes in nano-devices can be identified using tests and then bypassed by post-fabrication configuration. To be feasible this will require changed design-flows for these devices [Tah06]. But increasing miniaturization

also leads to increased sensitivity to the environment and a higher rate of transient errors. Therefore increasing miniaturization requires additional design efforts to design dependable systems [PUBV07]. Current research in the area concerns, e.g., the design of fault-tolerant virtual reconfigurable circuits [Sek07], self-replicating hardware [TMM⁺07], or self-organizing SIMD architecture [PDL07].

1.2 Devices Based on Novel Computing Paradigms

There is ongoing research in the area of novel computing paradigms. We will address the software-development issues and opportunities of these approaches in Part II, in this section we are concerned with devices built out of non-traditional material, e.g. DNA. In general, these areas are not yet far enough developed to have industrial applications, but interesting results have been obtained in research labs. Research for building and manipulating objects at the nanoscale level in physics, chemistry and molecular biology reveals promising approaches to unconventional computing [MS03, SBC⁺06], as foreseen by R. Feynman [Fey60] and later popularised by E. Drexler [Dre81].

Microfluidics-based biochips provide new kinds of sensors for biochemical analysis; biomechatronics aims to integrate bio-sensors and -computers into the human body for rehabilitation and augmentation purposes. In both areas rapid progress is made in research and experimental trials [AHC⁺06, SC06], and commercial applications can be expected in the next few years. It is, at this point of time, not entirely clear how these developments will influence the design of software-intensive systems, but in particular military applications of biomechatronics seem likely.

Another important area of molecular computing is concerned with using DNA for computational purposes [Amo05]. While there are efficient DNA computers for certain specialized problems, molecular computing is still subject to the same theoretical limitations as classical computation, although it is possible to build computing devices with massive parallelism.

In the field of molecular biology, harnessing molecules to compute can be traced back at least to [Hea87] and research in this field explodes with the landmark experiment of L. Adleman [Adl94]. Molecules can be used for their physical interactions or their chemical reactions [HH98, Mac02, RPW03] or, in a biological context, using the gene regulation machinery of a cell to achieve some computations [BHS04, KKA⁺04, SSZ⁺06]. These computations can be designed and implemented directly “by hand” or using directed evolution [YWA02, FH04].

In this context, *synthetic biology* [WK00, WBH⁺03, Gib04] emerges as a new engineering discipline at the convergence of genetic engineering and computer science. It encompasses the design and the implementation of complex artificial biological systems for a variety of applications. The web site [Syn07] is a good introduction. The web site of iGEM, the international Genetically Engineered Machine competition, [GEM07] gives an outline of the envisioned applications. In this new area, the pace of the technological changes seems to be consistent with Moore’s law [Car03]. Computer science is relevant for their development at two levels. At the engineering level it addresses the topics of:

- how to specify and design a set of standard (biological) component with well defined characteristics and performances that can be used and reused in the building of (artificial) biological systems;
- how to hierarchize and compose these biological components;
- which design methodology and computer tools can help the development of these kind of systems;
- how to reverse engineer existing biological modules to optimize them and to adapt them to the needs of synthetic biology.

At the programming level, synthetic biology poses many of the challenges sketched in the previous section: a massive number (billions) of unreliable elementary entities that interact and cooperate dynamically and randomly. It is nevertheless necessary to ensure a global emergent behaviour, robust and persistent in time, that can serve as a foundation for computing.

The main question is how to “instruct” the enormous population of elementary entities to obtain a global and coherent behaviour from the local regulation (interaction, cooperation, development) of the elementary processes. It is crucial to note that this is similar to the problems encountered in large intensive software systems. Recognizing this fact, emerging new research fields like amorphous computing, spatial computing, autonomic computing, organic computing, etc., are investigating properties of artificial systems which are usually attributed to living systems like self-organization, self-healing, self-optimization, or sustainability.

Another important non-traditional computation paradigm is quantum computing, where quantum-mechanical phenomena are used as essential ingredients of computations. These computers may offer large theoretical complexity improvements on certain classes of problems, such as the factorization of large integers and discrete logarithm, and they offer provable complexity advantages for at least one other problem, quantum database search. The former would obviously influence the design of software-intensive systems if quantum computers became available commercially, since public-key methods are often based on one of these mechanisms [BEM⁺07]. For a more detailed description of the capabilities of quantum computers and their applicability to different areas we refer to the IST-FET publication [VdPK05], the ERA-Pilot Project [ERA08], and [MO06].

2 Content

In addition to inexpensive and miniaturized devices, content is one of the main drivers for the development and adaptation of new software-intensive systems. The main trend is to go from dealing with raw signals or data to information and knowledge. We adopt the ITEA definitions of data as “raw representation in binary format,” information as “representation of knowledge that can be understood by a user,” i.e., data that is structured, annotated with metadata and that possesses a well-defined semantics, and knowledge as “information that is of interest to users.” In general, according to this point of view, information and knowledge can only be distinguished by taking into account the point of view of the user of the data, not by their representation. See [Nau07] for a more in-depth treatment of this question. While we are aware that these definitions are in many ways problematic and the assignment of data into the various categories to a certain extent arbitrary, the concepts of data/information/knowledge nevertheless define an important conceptual distinction and it seems beneficial to use them in a way that is consistent with other existing documents.

We have structured this section similar to the corresponding technology clusters in the ITEA roadmap into sections about content acquisition and processing, content representation, and data and content management.

2.1 Content Acquisition and Processing

The acquisition of content can be divided into two distinct areas which nevertheless share many of the same issues: acquiring data from sensors and acquiring data that already exists in digital form in the system’s environment.

The improvements in the production of MEMS allow the wide deployment of sensors and sensor networks. Sensor networks are widely used in areas such as environmental monitoring, 3D shape recognition or target detection and tracking [YJS06, LD00, CPRS02, LHMW07, GCC06]. Fusion of sensor data from diverse sensors and sensor networks can be used to achieve functionality that surpasses that possible from single sensors. Mechanisms to integrate data from different sensor are already widely used in industrial and military applications, but data fusion is also a currently very active field of research [NLF07, RKW⁺06]

In many wireless sensor networks it is not easily possible to replace the batteries of the sensors. Therefore energy-efficiency is an important issue. Efficient use of the limited energy budget can be made by optimizing sensor placement [GCBL06], possibly taking into account irregularities in the environmental conditions [ZHKS06], or by improved power management [KHZS07, CSA06]. Reconfiguration

of the sensor network is another method to reduce power consumption which can additionally be used to increase performance [KKP⁺07], or provide a level of self-organization [KK07] to the network.

Closely related to the issue of power consumption are techniques for ensuring coverage of the sensor network's operational area [HTW07, LP06]. Most often designers of wireless sensor networks try to reduce superfluous redundancy as this negatively affects the energy budget and simultaneously increases network congestion [CGVC06].

When integrating the data of many sensor operating independently it becomes important to ensure that the individual data items are temporally [YVS07, FS06, Lan06] and spatially [JP06] correlated, and to monitor the correctness of the transmitted data [HSL⁺07]. Many sensor networks are deployed in hostile environments. Here, in addition to ensuring the correctness of the data, self-protection of the network becomes another important issue [WZL07].

Another important aspect for sensor data is the context sensitivity of capturing and interpretation. It is in general not practical or desirable to permanently capture the data from all available sensors. Therefore software-intensive systems have to be able to derive their current and future operating context and decide which data should be retained, which data should be processed and summarized, and which data should not be captured.

Given the large amount of data available both from sensors and other sources it becomes more important to automatically "understand" the data, i.e., to analyze trends, to make predictions, or to derive abstractions from data and information without or with little human interventions. There is significant scientific progress reported in this area [SH99, CC99, MWK⁺05, TMKW07, NAS08] but the application of these techniques to currently deployed systems seems to be limited to specialized areas. However, because of the large possible impact it is likely that the research will be adopted by more commercial products in the next years.

2.2 Content Representation

In the last decade many very successful standards for representing structured data and metadata were defined and widely adopted in industrial applications. The most significant is the XML standard [BPSM⁺06] and a series of related standards, like namespaces, inclusions (XInclude), the XML Information Set (Infoset), etc. See [W3C08a] for a complete list of the relevant standards. Similar standards for the description of resources, like the Resource Description Framework (RDF) [W3C08b] have been defined and adopted in industry, but they are not as ubiquitous as XML. A number of competing standards for the definition of ontologies and the semantic description of data have been defined, e.g. the Web Ontology Language (OWL) [W3C08c], but no clear consensus on a single format seems to be developing. Semantic methods are not yet commonly used in applications although some authors predict rapid adoption in the next few years [BDBY⁺08].

A more widespread use in semantic techniques will also necessitate tools to help users to understand and visualize ontologies [KHL⁺07], and theories, methods and tools to detect and resolve conflicts in structural generalization hierarchies [FM04]

It is a relatively novel trend in research to integrate "Web 2.0 techniques" like tagging or social matching [TM05] with semantic approaches like ontologies or automated reasoning [BDBY⁺08].

Other trends that are visible are the increasing integration of multimedia content and active content into systems, and the increasing separation of content from presentation. These trends are, of course, inspired by the availability of network bandwidth and the increasing number of different devices on which content is consumed. In both areas mature standards exist, e.g., GIF, PNG, JPEG, MP3, OGG or MPG for images, audio and video content, ECMAScript-enabled browsers with XHTML and CSS for active content with separated presentation. Proprietary solutions like Adobe Flash or Microsoft Silverlight will also play an important role in these areas.

2.3 Data and Content Management

The management of data and content is an area where the increasing distribution of systems poses great challenges with issues ranging from the identity of data, systems and people over digital rights/restrictions management and privacy to search and resource management.

Even centralized systems have issues with the identity of data items: databases often contain multiple entries for single items. This problem is compounded in federated information systems (FIS), where a single data item may be divided between different nodes of the system [WJH⁺04]. Furthermore, the integration of heterogeneous data with independently evolving schema representations is still an active area of research with no universally applicable solutions.

Consistency and integrity of data are more difficult to ensure in distributed systems, in particular in systems operating in open environments in which nodes may dynamically join and leave the network and where individual nodes may have competing goals. This topic is addressed further in Section 10.

The increasing variety of devices on which multimedia content can be displayed and the wide variety in their capabilities gives rise to the recoding of content, either to upgrade the content quality of stored data without changing its meaning, or to reduce the transmitted data for devices with limited capabilities or slow network connections. Various commercial and open source solutions exist for the latter application, but upgrading content quality remains a mostly manual process. Related to this issue is the certification of content: currently there exist reliable mechanisms for public exchange of keys and for signing files, but no adequate technologies to certify the content of a file independent of its encoding in a particular format.

Another content management issue that has been investigated for a long time is the protection of content and the management of digital rights and restrictions [Soh07]. While both commercial and academic research in this area have increased the capabilities of content protection technologies there are virtually no examples of software or content that has not been duplicated without the copyright holder's permission shortly after it was made public. The currently most successful approaches rely on either permanent or periodic connections to the Internet, or a hardware "dongle" which either contains data that is important for the execution of the application, or which executes important parts of the application logic. This situation is even more pronounced for content which is intended for consumption by people, such as music or videos. Here most approaches to enforce copyrights have been unsuccessful while imposing significant inconveniences on legitimate users, and currently the trend in the music business seems to be to provide unprotected content on disks and even online. A related issue is watermarking [ZLZS07] of digital content which enables the embedding of information of metadata into media files such that it cannot be easily removed.

The possibility to store and process previously unimaginable amounts of data also poses new challenges in the related areas of data mining, search, and privacy. On the one hand, the large amounts of cheaply available data and processing capability and the integration of previously unconnected databases enable many new forms of data mining technology, and many businesses routinely mine their databases for purposes ranging from customer satisfaction analysis to fraud recognition. However, this amount of data mining also raises serious privacy concerns which are currently only partially addressed by technical and regulatory measures.

Other issues arising from the increasingly large amounts of distributed data in modern software-intensive systems are resource management, including data backups, and retrieving data. Resource management concerns the traditional topics pertaining to data management in federated information systems, such as the distribution of data to the individual nodes, horizontal and vertical partitioning of data, but increasingly also issues such as retention of data according to various regulations and laws and the related topic of "distributed garbage collection," i.e., identifying data that is no longer needed and can be deleted. Pragmatic solutions for these problems exist, but they are often very labour intensive and not amenable to automation.

Currently mechanisms for retrieving data from databases or semi-structured data collections are either performed by queries in languages such as SQL or by syntactic keyword searches. Many research

efforts to improve the quality of searches exist, e.g., in the areas of adaptive information extraction [TAC06], association mining [CR06b], software retrieval services [MIRAG06], or indexing for search [ZM06], and in related areas such as autonomous hypertext authoring [TGA07], evaluation of search services [JBCF07], interestingness measures for data mining [GH06]. That research is currently not widely used in industrial applications [BDBY⁺08]. Similar observations can be made for the integration of different data sources [Zha07].

3 Interoperability and Interaction

An important aspect of software-intensive systems is the interoperation of distributed components and the interaction of users with the system. In the following sections we present the aspects of networking, system-system interaction and human-system interaction.

3.1 Networking

Today's network infrastructure seems to increasingly develop into the direction of "IP everywhere" with a host of different underlying infrastructures such as Ethernet and wireless LAN. The identification of devices can therefore often be performed by their IP address. The well-known shortage of IPV4 addresses has been addressed by the development of the IPV6 protocol, which is deployed on current network equipment and personal computers, and also increasingly on networked embedded devices. However adaption of IPV6 has been slow, most networks still rely on IPV4. An overview of the current issues in the development of networks can be found in [Car07].

One trend is the increasing importance of mobile and ad-hoc networks and their particular properties [BDS07], e.g., scalability of wireless networks [JMS07], inference analysis [QCS07], or delay and capacity trade-offs [SMS07].

Another current research area which is particularly relevant for wireless networks is the topology of networks, e.g., controlling the network topology [San05], or topology aggregation techniques [ULNB07]. Another topic is Quality of Service (QoS), for examples the scheduling for resources [PdV07], algorithms to ensure required levels of QoS [LORS06, MLC07].

The increasing number of personal devices and computers with network access has led to the growing importance of networks which are not structured according to the client/server paradigm. Peer-to-peer networks and data grids are the most prominent examples, a survey of peer-to-peer content distribution networks is [ATS04], a survey of data grids is [VBR06]. Since they no longer rely on central servers such networks present new challenges, for example some nodes in peer-to-peer networks may deliberately introduce mislabeled content, techniques to avoid this are, e.g., ranking of peers [WNET07]. Many nodes in current networks can perform multiple functions, as client, server, router, etc. The looser definition of the roles necessitates a more dynamic configuration of the network, e.g., by automated resource announcement and discovery, by auto-configuring and self-managing nodes.

An interesting aspect of software-intensive systems is the combination of network transparency and location awareness. Location awareness exists to a limited extent, e.g., by automated discovery of Bluetooth devices or printers. However, current systems generally have too little knowledge about the state and desires of their users to consistently take the right decisions about device selection; improvements in location awareness will therefore be closely related to improvements in human-computer interaction and semantics-based interactions, see Section 3.3.

The wide range of different devices and network connections varying several orders of magnitude in throughput and latency necessitate differences in the communication styles and in the data sent over the network. Currently this scaling is mostly done manually, either on the server side or even by clients having to select different resources according to their capabilities.

3.2 System-System Interaction

Interactions between nodes of software-intensive systems and between different systems are today commonly based on cross-platform integration techniques such as Web-services or Corba, with standards such as XML [BPSM⁺06] serving as data interchange format. However, integration of heterogeneous data from different systems generally has to be performed manually, e.g., by writing XSLT transformations between the different data representation. Checks for correctness and consistency of conversions are usually purely syntactic.

Coordination of different nodes or interacting systems is often programmed in coordination or orchestration languages, which are essentially programming languages augmented with some features for programming compositions and possibly integrated into a middleware. Service-level and quality-of-service agreements are mostly contracts in natural language which are not explicitly represented in the architecture or data of the system; their fulfillment is often monitored outside the usual system operation.

Current research in these areas is directed towards formal foundations and a more semantic understanding on the part of the systems so that declarative specifications of the desired system behaviour, service-level or quality of service can be used to automatically negotiate the necessary contracts with possible partners and to create an orchestration of the system that fulfills the requirements.

The changing network structure described in Section 3.1, with devices which are not always connected to a network, emphasizes the need to replicate data, which is of course also present for well-known other reasons [SSPvS04]. Often traditional database techniques are not particularly well-suited for the structure of modern software-intensive systems. For example, it is difficult to ensure ACID properties without relying on a central coordinator and in a network where partitioning is likely to occur. Therefore techniques like relaxed transactions or optimistic replication [SS05] or are often used instead of their traditional counterparts. To ensure reliability of the resulting systems additional techniques have to be used to compensate for the weaker guarantees provided by the infrastructure, e.g., rollback-recovery [EAWJ02], compensation [BMM05, BMM06, GMS87], or measures that change the system structure, e.g., by distributing the computation to fault-tolerant and transactional agents [PS04], or by provisioning backend databases reactively [SA06].

3.3 Human-System Interaction

The human-system interaction of current systems is often based on interaction with keyboard and mouse or touchscreen, color displays and the WIMP (window, icon, menu, pointing-device) paradigm. Current research and development efforts are directed to design simple, self-explaining user interfaces, often with support for multimedia components or gesture-based and multi-modal interaction [DFAB04, Can06a]. Research prototypes are exploring areas such as natural language interaction and “disappearing computers,” i.e., embedded devices which either have no visible user interface at all, or where the user-interface is integrated into a real-world item and the software is controlled by interaction with that item. This research also leads in the direction of systems with which the user can interact in natural language [McT02, OS04, ODC04, LG04], or which use sensors to react to the environment and the user [ZB05, HPHS05, LGHH08].

The inverse direction is also an active topic in the development of human-system interactions: interactive worlds, simulations of real environments and augmented reality. Currently, interactive multi-user environments are commercially available and used for tasks such as social networking, online conferences or training. One of the issues currently being addressed by the providers and users of these communities as well as by academic research is the management of virtual identities and the virtual representation of real-world items.

More generally, the increasing pervasiveness of software-intensive systems raises questions about better models for the acceptance of technology by users [CT07] and long-term human-computer relationships [BP05].

One issue related to human-computer interaction is managing the quality of the user experience and

providing context-aware user interfaces that adapt to the user's personality, emotional state, current activities and the social context [Cro06, MBB07, DM05, LG04, Edw05]. This is a challenging and active research area which will probably become more influential in the development of future products, in particular consumer devices, in the foreseeable future. Other aspects in the user experience are accessibility [SHM07, HB07] and the impact that delays, e.g., caused by network congestion, has on the user experience [JGH07].

One aspect of the user experience that is often overlooked is that the user interface ensures privacy and security of the user's data and that it clearly communicates the implication of user interactions for these areas.

The presentation of systems that consist of many independent, dynamically configured parts is another current challenge in the area of human-system interaction. Current technologies, such as portlets, provide relatively low-level integration, where each node can display its own data. The approaches to present a more unified user interface for dynamically orchestrated systems will probably depend on a semantic analysis of the composition and the available data.

4 Resilience, Adaptation and Emergence

We call resilience the ability of a system to recover rapidly from negative influences like component failure. Adaptation is the ability of a system to react in a useful manner to situations and environments that were not explicitly foreseen at the time of its development or deployment; emergent behaviour is behaviour that cannot be localized in one component of the system but that instead arises from the structure and interaction of several independent parts of the system. For all but the simplest systems non-functional properties like performance are emergent properties [Som07], however the term "emergent properties" is sometimes used to denote properties that negatively impact the work of a system in ways that its designers did not foresee.

Many researchers agree that adaptation and emergent behaviours will be key features of future software-intensive systems [Hol92, WH07] that pervade all aspects of their design and operation.

Our current understanding of the underlying concepts and theories for adaptivity and emergence, the design process for adaptive systems, and the mechanism for controlling and exploiting emergent behaviours are rudimentary at best. We cannot reliably predict emergent behaviours of systems, let alone design systems that exhibit desirable emergent behaviours; even state-of-the-art software possesses few properties that can be described as truly adaptive. The current state of the art are systems with several different configurations which can switch configurations based on external or internal criteria [HKMU06]. We have currently no means to predict or exploit the emergent behaviour in complex systems, and current engineering practices therefore try to eliminate the possibilities for emergent behaviour as far as possible.

Most research about adaptation is interdisciplinary with cooperation between biologists, economists, social scientists, physicists, mathematicians and computer scientists, and drawing on areas such as control theory, cybernetics, systems theory, complexity theory, catastrophe theory, etc.

Research directions in computer science that might impact the development of future adaptive systems are for example

- context-awareness,
- automated reasoning about ontologies,
- automated reasoning about real-world data and problems
- machine learning, user profiling and profiling of other systems, e.g., learning-based content management, learning- and profile-based user-interface selection
- using "ambient intelligence"
- self-* properties

Emergent behavior and adaptation also influences the design of the human-computer interaction as it will be necessary to provide users of the system enough information to allow them to monitor or controlling emergence [Can06b].

5 Assurance

We use the term *assurance* to denote binding commitments that a system makes about properties that it promises to maintain (or that the user reasonably expects the system to maintain).

5.1 Quality of Service/Experience

One area of assurance is the quality of service (QoS) and the quality of user experience that the system provides. This encompasses many possible properties, such as guaranteed availability, maximum response times for certain interactions, provision of certain multimedia content, etc. Most current QoS agreements are written in natural language and not available as precise specifications for systems. Researchers are currently developing methods to automatically negotiate QoS agreements, but current approaches are not yet suitable for most applications. Furthermore, this is an area which is intimately connected to legal and contractual requirements which are beyond the scope of this report.

Related to the issue of Quality of Service are compliance [All06, CB06] and Service-Level Agreements. In particular the formalization of Service-Level Agreements is an important current research topic [BM07].

The quality of experience that a system shows is closely connected to both its technical ability to provide the service that its users expect, and to its human-computer interface as discussed in Section 3.3.

5.2 Security and Trust

We use the following definitions from [AA01]: a trusted system or component is one whose failure can break the security policy; a trustworthy system is one that won't fail. There are several approaches to security engineering, both formal and informal. However experience with existing systems shows, that trusted systems fail with alarming regularity; the number of security breaches of Web-based system has almost doubled in 2007 compared to 2006. The increasing importance of networked systems in e-commerce and business transactions means that further research in the development of secure systems has to be a priority. However, many of the security breaches have relied on implementation errors rather than on conceptual weaknesses of the security systems [Sch00]. Therefore further improvements in the implementation of trusted systems and system components will be an important ingredient in the development of trustworthy software-intensive systems. Two of the leading researchers in computer security go so far to claim: "In the past decade, cryptography has done more to damage the security of digital systems than it has to enhance it. [...] the mathematics of cryptography is almost never the weakest link. The fundamentals of cryptography are important, but far more important is how those fundamentals are implemented and used." [FS03].

Increasing network connectivity increases both the perceived and actual risk for systems [Cym06, Gee06]. There is a lot of material available about network security threats, [KPS95] is a good overview of the state of the art. An introduction to secure web applications is [NN07]. A survey of network defense mechanisms is available as [PLR07], socio-technical aspects of security are addressed in [AA01, Sch00, KD07], among many other sources. An important aspect of security is the detection of and response to security threats [SQX⁺07].

The increasing distribution of systems leads to several security issues which are currently not sufficiently well understood. One such problem is the security of protected data in environments with untrusted components.

Key management [MDM07, RH03], access control [TAPH05, ZBKK05], and more generally identity management are important problems in distributed systems. Many current systems rely on user names

and passwords to authenticate their users. But as many users have to use dozens of different systems each day the management of passwords becomes unwieldy, leading many users to use the same password for all systems. Therefore a single compromised system can give an attacker access to user accounts on many other systems. Hardware-based solutions, such as tokens, smart cards or biometric systems exist and are often deployed to secure government or cooperate networks, however they are not commonly used for other purposes, such as authenticating Web sites. Single-sign-on solutions are technically feasible and have been deployed for some time, but they suffer from several problems: a security breach in a single-sign-on system can have dramatic consequences for the users since all their accounts are compromised, and the service providers relying on the single-sign-on provider have to entrust data about all their users to another company. For further information see [AA01, BBG07].

Another issue with single-sign-on is privacy: the authentication provider can observe all visits of users to protected sites and correlate this information. Similar privacy concerns exist in the area of online payment, as the of credit cards or payment services like PayPal is directly traceable to the person making the payment. Anonymous online cash is technically feasible, but not widely used. In general, approaches to issues like privacy, anonymity, pseudonymity, unobservability and unlinkability exist in research but are not widely deployed.

Many attack vectors exploit the interaction between different systems and rely on mismatches in the security boundaries between systems to subvert security mechanisms [Sch00]. One current research direction to overcome these problems is the use of bio-inspired models to improve the security of networks [Gee07]. As mentioned in Section 1.2, the availability of quantum computers would allow attacks on public-key cryptosystems and on protocols based on them. However it would also enable the use of quantum cryptography which is provably secure against certain kinds of attack [BEM⁺07].

One currently unsolved problem is to ensure that trustworthiness can be observed by the user of the system. In general users and administrators of today's systems have no mechanism to identify whether their data is transmitted to other systems, whether confidential data is retained, etc. Similarly the user interface has to be designed in a way that it respects the privacy of the user [BG05] without making the system inconvenient to use. Research in this area will be related to the research on context-aware user interfaces described in Section 3.3. There are some mechanisms to ensure that software has not been tampered with, e.g., by verifying secure hashes of the binaries and system data, however experience shows that these systems can often be bypassed, either because of implementation errors in the security system itself or because flaws in the system which are not directly related to the security components can be used to replace the program together with the authenticating information.

One important aspect of secure systems is the ability to recognize if security breaches have happened. Intrusion detection systems that attempt to monitor the whole system and recognize both intrusion attempts and successful intrusions into parts of the system are commercially available and widely deployed. Their use is currently mostly confined to larger networks and servers although intrusion detection systems for single PCs or workstations exist. We are not aware of intrusion detection systems for programmable embedded devices, except for devices powerful enough to run software for desktop computers or workstations.

From a practical point of view, many security problems remain in operating systems and applications, but recently some significant increases have been made regarding the security in widely deployed operating systems, for example the introduction of mandatory access controls (MACs) or address-space randomization in the Red Hat Linux distribution [LS01, Dre05]. Various studies have examined the security of open-source versus closed-source software [For07, HJ07].

To summarize, while there exist many areas in which future research is needed to ensure system security most observed security breaches seems to rely at least partly on implementation errors, either because the designer of the security system was not aware of the often subtle implications of design choices on the quality of the security system, or because the chosen implementation techniques were not adequate to build secure systems.

Part II

Engineering

The engineering of software-intensive systems poses many challenges, mostly because the complexity of software-intensive systems. We have structured this part of the report into several topics:

- *Engineering processes*: the various process models for system- and software engineering.
- *Distribution, Heterogeneity and Reuse*: approaches to deal with the distributed nature of and the heterogeneous components invariably found in systems, and reuse of existing components and system parts.
- *Separation of concerns*: approaches to structure the often multi-dimensional requirements.
- *Tool and language support*: The growing complexity of today's systems cannot be handled without adequate languages to express the requirements, design and implementation, and without support from tools.
- *Engineering for Resilience, Adaptivity and Emergence*: engineering techniques for systems that are resilient to failures, that can adapt to or be adapted to unforeseen circumstances, and techniques for controlling emergent behaviour.

6 Engineering Processes

In the process of software-intensive systems engineering one has to distinguish between two distinct activities: system engineering and software engineering. The distinction is that system engineering [Tha02, WAH⁺93] is concerned with development of the whole system, the result of systems engineering are documents that describe the system architecture and the distribution of the system functionality to individual components of the system. Various methods for systems engineering exist, a commonly used set of procedures for the system engineering process is codified in the IEEE standard 1220 [IEE98]. Reports on the state-of-practice of software engineering show, that many companies do not follow a defined systems engineering process [NL03].

Software engineering is concerned with the development of the software for system components; the result of software engineering are software artifacts. An overview of the state of the art for software engineering can be found in [Som07].

Both systems and software engineering have to identify often complex requirements. The current state of the art in this area is given in [NL03, GWBG06], information about requirements for security and trust can be found in [GMZ05]. The article [DB06] provides an experience report for modelling systems with high dependability requirements.

Unless they can reuse components from earlier projects or commercial off-the-shelf (COTS) components, developers of software-intensive systems are faced with the co-development of hard- and software. Because of the longer times required to develop and produce the hardware the co-development is often driven by the hardware requirements, although the situation is starting to change [NL03]. Often the teams responsible for software and hardware development are not well integrated, leading to misunderstandings and impedance mismatches in the developed artifacts. Techniques like model-driven systems development (MSDM) [CR05, CR06a] and simulation [Unr08] are used to address these issues. Important issues in the requirements process are determining and managing the interaction of different requirements [RPV03], and tracing the requirements throughout the software development process.

In the following paragraphs we will focus mostly on the software development aspect of system development.

There is an ongoing discussion about which software development process models are best suited for particular circumstances, and how tools and languages should support the process. Currently two

directions which might appear contradictory seem to be prevalent. One trend is the use of model based development, mostly based on the UML [BRJ05]. In industrial practice the models are often manually implemented, or development is manually continued after initial code has been generated from the models. A lot of research is currently undertaken towards approaches such as MDA which use transformational techniques to generate the complete code of the system directly from models. For model-based approaches the conformance and consistency of different models and different views is an important issue for which few practical solutions exist but which is under active research, e.g., [PBO07].

The opposing trend is the rise of agile approaches [Bec03, Coc06, Hig03, NB07] which stress the importance of incremental design and implementation over “up-front modelling.” These approaches have gained many supporters in recent years, and many companies have started to include agile methods into their development process. It is likely that both kinds of process models will be used in practice for the foreseeable future.

Many parts of systems can be seen as members of a “family of components” which is used in several systems but where customization is necessary for different uses. The software development side for such components is addressed by approaches like software product lines [CN01, CE00, BHJ⁺03] or software factories [GS04]. Product line engineering tries to build families of products as configurations or variations of a single model; applicable design and modelling techniques are, e.g., domain-driven design [Eva04] and feature modelling.

At a lower level of abstraction, design patterns are a common method to reuse modelling or implementation abstractions. General references are [GHJV95] for implementation-centric patterns, and [Fow02] for architectural design patterns. The “Pattern Languages of Program Design” (PLoP) conferences are conferences dedicated to design patterns. Patterns for particular domains like sensor networks are also widely published and used [GLC07].

Testing has always been an important part of the development process of software-intensive systems. An overview is given in [Bin00]. In industrial practice, the importance of early testing of individual components (unit testing) has been particularly stressed by agile approaches which rely on the coverage of their test suite to detect regressions during refactoring [Bec03]. Important testing aspects for systems are methods to compose test suites and cost-effective regression testing [REM⁺04].

Formal methods are increasingly being used in the development of software-intensive systems. Automated theorem provers like ACL2 [KMM00] have successfully been used to verify parts of microprocessors [RKSS05], similarly model checkers like Spin [Hol04, Spi08] are finding increasing application in particular in the area of finding defects in concurrent software designs. Model checkers have also recently been used to improve the performance of test suites, e.g., by eliminating redundant test cases [FW07a, FW07c, FW07b].

7 Distribution, Heterogeneity and Reuse

Software engineering has to adjust to the fact that typical software products are becoming more complex, more distributed and less tightly integrated. Component techniques and languages promise the re-use of architecture and components thereby increasing the level of abstraction at which the software developer works. Several techniques have been proposed to increase reconfigurability and facilitate fault localization in systems [RYC⁺07], to simplify or automate the configuration of distributed systems [PBK07] or for feature location [Erl05, ESW06]

To cope with the massive number of nodes in current and future systems new abstractions have been proposed that are inspired by biological or chemical metaphors [BCD⁺06].

Another approach to handle the increasing distribution of systems are Agents. Here current research includes protocols for multi-agent interaction [Pos07], and tropos to increase the variability [PPSM07], or security of agent-based systems [GMZ07]

In recent years services have become the dominant technology for distributed computing. [Erl05] gives an overview of the currently used technologies for Web services, [ACKM04] is a more conceptual

overview of the field. Active research on services is performed in many areas, such as negotiation [PTW07], service-level-agreements [BM07], choreography and orchestration [BGG⁺05], or verification of service-oriented systems [FUKM06, vBK06].

Two independent roadmaps for service-oriented computing have been developed independently: one by the NESSI technology platform [Pro07] and the other one by the International Conference on Service Oriented Computing. A number of European and international research programmes are currently investigating the field of service engineering, among them BIONETS, CASCADAS, ONTOGRID, SIMS, SODIUM, PLASTIC, SENSORIA, ONE, ASTRO, AOSD, MUSIC, WS-DIAMOND, SECSE, INFRAWEBS, DIP, AMIGO, WS2, ESFORS, S3MS, TRUSTCOM, ATHENA and DEDISYS. An introduction to many of the European projects and their goals can be found in the forthcoming [dNTSZ08].

8 Separation of Concerns

Experience shows that many requirements or system functionalities, for example logging or transactions, cannot easily be encapsulated in a single component but rather represent functionality that spans many different parts of a system. These functionalities are called “cross-cutting concerns” [KLM⁺97]. Object-oriented or functional software development approaches provide no direct support for encapsulating these cross-cutting concerns, therefore several extensions have been proposed under the name aspect-oriented programming (AOP). The common theme of different approaches to AOP is that different concerns of the program are specified separately and then combined in a modular way. Introductory articles about AOP are [EFB01, EAK⁺01].

The best-known approach to AOP is the AspectJ extension to Java [KHH⁺01]. AspectJ introduces several concepts that are not present in standard object-oriented languages: *join-points* are points in the execution of programs, *point-cuts* are collections of join-points, and *advice* are method-like constructs that can be attached to point-cuts. This allows the modular implementation of features like logging or transactions which would normally be distributed throughout the application. Other approaches to aspect-orientated programming are, e.g., based on adaptive methods [LOO01], composition filters [BA01, ABV92], or on multi-dimensional separation of concerns [OT01, TOHSMS99]. Aspect-orientation is currently also introduced into modelling languages such as the UML, see, e.g., [EOSW07].

Aspect-orientation is closely related to the more general topics of computational reflection [Smi84, Mae87] and meta-object protocols [KdRB91]. With the increasing need for dynamic adaptation and modification, aspect-oriented approaches based on general reflection and metaobject-protocols [Sul01a, Sul01b] seem to be promising areas of research.

A problem similar to cross-cutting concerns is context sensitive behaviour. Here relatively little research has been done, the most significant being ContextL [CH07, CH05] in the area of programming languages and modes for software architecture [HKMU06].

Most approaches that address separation of concerns perform non-local transformations of models or software. Therefore not all testing and verification techniques are applicable without modification. One recent research effort in the area of verification is incremental aspect model-checking [KF07].

9 Language and Tool Support

A trend toward variety can be observed in the area of programming languages. Whereas a decade ago most software was developed in C/C++ with scripting languages like Perl playing an important role in Web applications, today C/C++ are still prominent but no longer ubiquitous. Statically typed languages with more modern features such as garbage collection and introspection, the most common being Java and C#, are heavily used, particularly in the development of enterprise applications but also in other areas, and dynamically typed languages are currently seeing a significant rise in popularity, often integrated into the Java or .Net platforms. We also expect this trend to continue in the next years, with increasing effort expended on cross-language interoperability.

Domain-specific languages [MHS05] try to simplify the development of systems for particular areas by first defining a language in which the necessary concepts can be concisely expressed and then writing the application in this language.

In recent years modern languages have been more widely used in the development of real-time and embedded systems than previously. This was caused in part by the increasing computational power available on these devices, but also by research efforts to provide support for real-time systems in the run-time of modern languages, e.g., by providing real-time garbage collection [Bac07].

10 Engineering for Resilience, Adaptivity and Emergence

For many systems the notion of shutting them down to perform upgrades or maintenance is no longer feasible. This is not a completely new phenomenon, since for example, the power grid has to continue working, even if parts of its infrastructure are updated. However the increasing number of large-scale, distributed systems will make this situation more common. Most current tools are not well suited to develop for systems in which no distinction between development- and run-time exist, and not even for systems where individual parts can be replaced at run-time, although research and some industrial solutions exist [AVWW96].

The engineering of large distributed system poses another problem: controlling and dealing with emergent behaviour, i.e., behaviour which cannot be attribute from individual components in the system but rather to the system as a whole. Most system above a certain level of complexity exhibit emergent behaviour that was not initially planned by their designers. Currently we have no reliable methods to predict the type of emergent behaviour that a system will exhibit for different inputs and under different environmental situations. Analysis of emergent systems has delivered some insight into their behaviour, but predictions about long range effects is still beyond the current state of the art. Understanding emergence and developing tools to control and exploit emergent behaviour will be one of the great research challenges in the next years.

11 Summary and Future Research

We have surveyed the state of the art in various areas relevant to the engineering of software-intensive systems. The survey has been structured into two parts: foundations and engineering. In the first part we looked at technology fields and topics that shape the development of software-intensive systems: devices, content, interoperability and interaction, resilience, adaptation and emergence, and assurance. In the second part we addressed engineering processes; distribution, heterogeneity and reuse; separation of concerns; tool and language support; and engineering for resilience, adaptivity and emergence.

As mentioned in the introduction, this effort is by necessity incomplete. However the referenced literature should provide the reader with sufficiently many starting points to explore the areas we mentioned. Although a more detailed research agenda is the subject of a forthcoming report we sketch here some of the research topics that the members of the Interlink WG1 workshops considered particularly relevant.

In the next decades large numbers of software-intensive systems will be developed and deployed to fulfil vital functions. These systems will not only feature massive numbers of nodes per system, they will also have to operate in open, non-deterministic environments in which they interact with humans or other software-intensive systems and adapt to new requirements, technologies or environments without redeployment.

Massive Numbers of Nodes per System An increase in the number of nodes of future software-intensive systems will be one of the most visible features: the availability of cheap low-energy mobile processors and Moore's law ensure that we will see an increasing number of nodes with computational

capability in the next years, even for systems built out of traditional computing devices. Furthermore, new manufacturing methods like nanotechnology, synthetic biology, or MEMS will give rise to new kinds of ensembles, many of them with potentially millions of computational nodes.

Open and Non-Deterministic Environments Currently many personal computers, workstations and servers used in commercial environments are networked via local area networks, dedicated wide area networks, or globally via the Internet. We are currently seeing this trend for personal information devices and embedded systems as well. The increase in available devices in turn entices service providers to offer new services or to remove service offers that are no longer profitable or that have been superseded by newer offers. Therefore, software-intensive systems can no longer expect to operate in the environment that was current during their design time; they have to replace services that are no longer available with others that offer similar functionality; they should also take advantage of new services provided by the network environment that were not foreseen when the system was designed.

Adaptation The situation mentioned in the previous paragraph is one instance of a more general situation: future systems will often have to operate under conditions that differ significantly from the ones for which they were designed. They should not only be able to adapt to changes in their network environment, they should also be able to work reliably in the face of changes to their execution platform: even today reinstalling all necessary programs is a major burden when we switch to a new computer. Since future software-intensive systems will be more ubiquitous, more distributed, and will assimilate a large number of adaptations during their operation, the prospect of reinstalling them from scratch when switching to a new platform becomes infeasible. Therefore we need to develop systems that can adapt to different environments, to different users, etc.

We call this future generation of software-intensive systems ensembles. *Ensemble engineering* is the science and engineering discipline of complex, integrated ensembles of computing elements. The potentially huge impact—both positive and negative—of ensembles means that we need to understand ways to reliably and predictably model, design, and program them.

Research challenges for ensembles are

- Harnessing the stochastic behaviour and massive scale.
- Deducing a global specification from local rules, and finding local rules that produce a desired global behavior. Influencing the global behavior by making local changes.
- Replacing the notion of correctness with the one of “fitness for a purpose.”
- Social and emotional perception, quality of experience
- Resilience, adaptation, and controlled emergence
- Operating in open environments, recognizing and exploiting opportunities
- Assurance guarantees, certification, and formal verification of ensembles

Promising research topics are

- abstractions and models,
- programming languages and compiler technology for ensembles,
- testing and verification of ensembles,
- predicting emergent behaviour
- privacy, identity management, security, and trust

- mobility of code and data,
- integration of heterogeneous and federated data
- quality of service, quality of experience

The discussions of the Interlink WG1 have focused on three research areas:

- *Physical ensembles*, which are intimately connected to the physical world in space and time. They are equipped with sensors and actuators and have to take into account issues of locality and resource constraints. Examples are real-time embedded systems, claytronics, modular robots or programmable matter. These systems combine discrete and continuous, non-linear domains, and they exhibit complex interaction patterns between components. Coordination in space and time with limited resources is one of the major challenges faced by physical ensembles.
- *Organic software and systems engineering* addresses the challenge of designing software for ensembles that is reliable, predictable, with guarantees for security and trust, that acts autonomously and has self-* properties, and harnesses emergent behaviour. Approaches to organic software engineering may use bio-inspired and swarm algorithms and rely on nature-inspired programming paradigms, but they also include traditional software development techniques and formal methods.
- *Societal computing* addresses the problem of composing “living” and evolving societal software systems from parts that were not designed to be composed together and which may partially compete with each other while partially cooperating. These systems will require languages and environments which blur the distinction between compile-time and run-time and between design, implementation and deployment. Research in this area will have to investigate the dynamics of purposive interactions and the structure of evolving societal architectures: evolution of societal software systems has to be a long-term process that goes beyond single-run adaptation, and systems have to maintain societal coherence while supporting diversity and context awareness.

References

- [AA01] Ross J. Anderson and Ross Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems*. Wiley, January 2001.
- [ABV92] Mehmet Aksit, Lodewijk Bergmans, and Sinan Vural. An object-oriented language-database integration model: The composition-filters approach. In *ECOOOP '92: Proceedings of the European Conference on Object-Oriented Programming*, pages 372–395, London, UK, 1992. Springer-Verlag.
- [ACKM04] Gustavo Alonso, Fabio Casati, Harumi Kuno, and Vijay Machiraju. *Web Services: Concepts, Architectures and Applications*. Springer, 2004.
- [Adl94] Len Adleman. Molecular computation of solutions to combinatorial problems. *Science*, 266(5187):1021–4, November 1994.
- [AHC⁺06] R. Aaron, H. Herr, D. Ciombor, L. Hochberg, J. Donoghue, C. Briant, J. Morgan, and M. Ehrlich. Horizons in prosthesis development for the restoration of limb function. *Journal of the American Academy of Orthopaedic Surgeons*, 14(10):198–204, 2006.
- [All05] J. J. Allen. *Micro Electro Mechanical System Design*. CRC Press, Boca Raton, FL, 2005.
- [All06] Eric Allman. Complying with compliance. *Queue*, 4(7):18–21, 2006.
- [Amo05] Martyn Amos. *Theoretical and Experimental DNA Computation*. Springer, June 2005.

- [ATS04] Stephanos Androutsellis-Theotokis and Diomidis Spinellis. A survey of peer-to-peer content distribution technologies. *ACM Comput. Surv.*, 36(4):335–371, 2004.
- [AVWW96] J. Armstrong, R. Virding, C. Wikström, and M. Williams. *Concurrent Programming in Erlang*. Prentice Hall, 2nd edition, 1996.
- [BA01] Lodewijk Bergmans and Mehmet Aksit. Composing crosscutting concerns using composition filters. *Commun. ACM*, 44(10):51–57, 2001.
- [Bac07] David F. Bacon. Realtime garbage collection. *Queue*, 5(1):40–49, 2007.
- [BBG07] Rafae Bhatti, Elisa Bertino, and Arif Ghafoor. An integrated approach to federated identity and privilege management in open systems. *Commun. ACM*, 50(2):81–87, 2007.
- [BCD⁺06] Ozalp Babaoglu, Geoffrey Canright, Andreas Deutsch, Gianni A. Di Caro, Frederick Ducatelle, Luca M. Gambardella, Niloy Ganguly, Márk Jelasity, Roberto Montemanni, Alberto Montresor, and Tore Urnes. Design patterns from biology for distributed computing. *ACM Trans. Auton. Adapt. Syst.*, 1(1):26–66, 2006.
- [BDBY⁺08] V. Richard Benjamins, John Davies, Ricardo Baeza-Yates, Peter Mika, Hugo Zaragoza, Mark Greaves, Jose Manuel Gmez-Prez, Jess Contreras, John Domingue, and Dieter Fensel. Near-term prospects for semantic technologies. *IEEE Intelligent Systems*, 23(1):76–88, 2008.
- [BDS07] Bartosz Biskupski, Jim Dowling, and Jan Sacha. Properties and mechanisms of self-organizing manet and p2p systems. *ACM Trans. Auton. Adapt. Syst.*, 2(1):1, 2007.
- [Bec03] Kent Beck. *Extreme Programming Explained—Embracing Change*. Addison-Wesley Professional, 2003.
- [BEM⁺07] Dagmar Bruss, Gábor Erdélyi, Tim Meyer, Tobias Riege, and Jörg Rothe. Quantum cryptography: A survey. *ACM Comput. Surv.*, 39(2):6, 2007.
- [Ben06] Sandro Benetti. Intelligent co-operative systems in cars for road safety, July 2006.
- [BG05] Michael Boyle and Saul Greenberg. The language of privacy: Learning from video media space analysis and design. *ACM Trans. Comput.-Hum. Interact.*, 12(2):328–370, 2005.
- [BGG⁺05] Nadia Busi, Roberto Gorrieri, Claudio Guidi, Roberto Lucchi, and Gianluigi Zavattaro. Choreography and Orchestration: a synergic approach for system design. In *Proc. of 3rd International Conference on Service Oriented Computing (ICSOC'05)*, volume 3826 of *LNCS*, pages 228–240. Springer, 2005.
- [BHJ⁺03] Andreas Birk, Gerald Heller, Isabel John, Klaus Schmid, Thomas von der Maßen, and Klaus Müller. Product line engineering: The state of the practice. *IEEE Software*, 20(6):52–60, 2003.
- [BHS04] Asa Ben-Hur and Hava T Siegelmann. Computation in gene networks. *Chaos*, 14(1):145–151, March 2004.
- [Bin00] Robert Binder. *Testing Object-Oriented Systems: Models, Patterns and Tools*. Addison-Wesley Professional, 2000.
- [BM06] Tobias Bjerregaard and Shankar Mahadevan. A survey of research and practices of network-on-chip. *ACM Comput. Surv.*, 38(1):1, 2006.

- [BM07] Maria Grazia Buscemi and Ugo Montanari. Cc-pi: A constraint-based language for specifying service level agreements. In *Proc. ESOP'07*, volume to appear of *LNCS*, 2007.
- [BMM05] R. Bruni, H. Melgratti, and U. Montanari. Theoretical foundations for compensations in flow composition languages. In *Proc. of POPL'05*, pages 209–220. ACM Press, 2005.
- [BMM06] Roberto Bruni, Hernán Melgratti, and Ugo Montanari. Composing transactional services, 2006. Manuscript.
- [BP05] Timothy W. Bickmore and Rosalind W. Picard. Establishing and maintaining long-term human-computer relationships. *ACM Trans. Comput.-Hum. Interact.*, 12(2):293–327, 2005.
- [BPSM⁺06] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, François Yergeau, and John Cowan. Extensible markup language (xml) 1.1 (second edition). W3C Recommendation, August 2006. <http://www.w3.org/TR/2006/REC-xml11-20060816/>.
- [BRJ05] Grady Booch, James Rumbaugh, and Ivar Jacobson. *The Unified Modeling Language User Guide*. Addison-Wesley, second edition edition, 2005.
- [BRL⁺03] Richard Baskerville, Balasubramaniam Ramesh, Linda Levine, Jan Pries-Heje, and Sandra Slaughter. Is internet-speed software development different? *IEEE Software*, 20(6):70–77, 2003.
- [Can06a] John Canny. The future of human-computer interaction. *Queue*, 4(6):24–32, 2006.
- [Can06b] Bryan Cantrill. Hidden in plain sight. *Queue*, 4(1):26–36, 2006.
- [Car03] Robert Carlson. The pace and proliferation of biological technologies. *Biosecur Bioterror*, 1(3):203–214, 2003.
- [Car07] Brian Carpenter. Better, faster, more secure. *Queue*, 4(10):42–48, 2007.
- [CB06] J. C. Cannon and Marilee Byers. Compliance deconstructed. *Queue*, 4(7):30–37, 2006.
- [CC99] Carlo Combi and Luca Chittaro. Abstraction on clinical data sequences: an object-oriented data model and a query language based on the event calculus. *Artificial Intelligence in Medicine*, 17(3):271–301, November 1999.
- [CE00] Krystof Czarnecki and Ulrich Eisenecker. *Generative Programming: Methods, Tools and Applications*. Addison-Wesley Professional, 2000.
- [CF06] Deepayan Chakrabarti and Christos Faloutsos. Graph mining: Laws, generators, and algorithms. *ACM Comput. Surv.*, 38(1):2, 2006.
- [CGVC06] Bogdan Cărbunar, Ananth Grama, Jan Vitek, and Octavian Cărbunar. Redundancy and coverage detection in sensor networks. *ACM Trans. Sen. Netw.*, 2(1):94–128, 2006.
- [CH02] Katherine Compton and Scott Hauck. Reconfigurable computing: a survey of systems and software. *ACM Comput. Surv.*, 34(2):171–210, 2002.
- [CH05] Pascal Costanza and Robert Hirschfeld. Language constructs for context-oriented programming: an overview of contextl. In *DLS '05: Proceedings of the 2005 symposium on Dynamic languages*, pages 1–10, New York, NY, USA, 2005. ACM.
- [CH07] Pascal Costanza and Robert Hirschfeld. Reflective layer activation in contextl. In *SAC '07: Proceedings of the 2007 ACM symposium on Applied computing*, pages 1280–1285, New York, NY, USA, 2007. ACM.

- [CLB03] Marcus Ciolkowski, Oliver Laitenberger, and Stefan Biffel. Software reviews: The state of the practice. *IEEE Software*, 20(6):46–51, 2003.
- [CMKC03] Michael A. Cusumano, Alan MacCormack, Chris F. Kemerer, and Bill Crandall. Software development worldwide: The state of the practice. *IEEE Software*, 20(6):28–34, 2003.
- [CN01] Paul Clements and Linda Northrop. *Software Product Lines: Practices and Patterns*. The SEI Series in Software Engineering. Addison-Wesley Professional, August 2001.
- [Coa06] Terry Coatta. The (not so) hidden computer. *Queue*, 4(3):22–26, 2006.
- [Coc06] Alistair Cockburn. *Agile Software Development—The Cooperative Game*. Addison-Wesley, second edition edition, 2006.
- [CPRS02] T. Clouqueur, V. Phipatanasuphorn, P. Ramanathan, and K. Saluja. Sensor deployment strategy for target detection. In *The First ACM International Workshop on Wireless Sensor Networks and Applications (WSNA'02)*, September 2002.
- [CR05] Murray Cantor and Gene Roose. Hardware/software codevelopment using a model-driven systems development (mdsd) approach. <http://www.ibm.com/developerworks/rational/library/dec05/cantor/>, last accessed 2008-01-20, December 2005. IBM developerWorks.
- [CR06a] Murray Cantor and Gene Roose. Hardware/software codevelopment using a model-driven systems development (mdsd) approach—part ii: Illustrating the solution. <http://www.ibm.com/developerworks/rational/library/feb06/cantor-roose/>, last accessed 2008-01-20, February 2006. IBM developerWorks.
- [CR06b] Aaron Ceglar and John F. Roddick. Association mining. *ACM Comput. Surv.*, 38(2):5, 2006.
- [Cro06] James L. Crowley. Social perception. *Queue*, 4(6):34–43, 2006.
- [CSA06] Arnab Chakrabarti, Ashutosh Sabharwal, and Behnaam Aazhang. Communication power optimization in a sensor network with a path-constrained mobile observer. *ACM Trans. Sen. Netw.*, 2(3):297–324, 2006.
- [CT07] Hock Chuan Chan and Hock-Hai Teo. Evaluating the boundary conditions of the technology acceptance model: An exploratory investigation. *ACM Trans. Comput.-Hum. Interact.*, 14(2):9, 2007.
- [Cym06] Team Cymru. Cybercrime: an epidemic. *Queue*, 4(9):24–35, 2006.
- [DB06] Paolo Donzelli and Victor R. Basili. A practical framework for eliciting and modeling system dependability requirements: Experience from the nasa high dependability computing project. *Journal of Systems and Software*, 79(1):107–119, 2006.
- [DFAB04] Alan Dix, Janet Finlay, Gregory Abowd, and Russell Beale. *Human Computer Interaction*. Prentice Hall, third edition edition, 2004.
- [DGK03] Jason Dedrick, Vijay Gurbaxani, and Kenneth L. Kraemer. Information technology and economic performance: A critical review of the empirical evidence. *ACM Comput. Surv.*, 35(1):1–28, 2003.

- [DM05] Anind K. Dey and Jennifer Mankoff. Designing mediation for context-aware applications. *ACM Trans. Comput.-Hum. Interact.*, 12(1):53–80, 2005.
- [dNTSZ08] Elisabetta di Nitto, Paolo Traverso, Anne-Marie Sassen, and Arian Zwegers, editors. *At your service: An overview of results of projects in the field of service engineering of the IST programme*. MIT-Press, to appear, 2008.
- [Dre81] Kim E. Drexler. Molecular engineering: An approach to the development of general capabilities for molecular manipulation. *Proc. Nat. Acad. Sci. USA*, 78(9):5275–5278, 1981.
- [Dre05] Ulrich Drepper. Security enhancements in red hat enterprise linux (beside selinux). <http://people.redhat.com/drepper/nonselsec.pdf>, last accessed 2008-01-19, December 2005.
- [DSU04] Xavier Défago, André Schiper, and Péter Urbán. Total order broadcast and multicast algorithms: Taxonomy and survey. *ACM Comput. Surv.*, 36(4):372–421, 2004.
- [EAK⁺01] Tzilla Elrad, Mehmet Aksit, Gregor Kiczales, Karl Lieberherr, and Harold Ossher. Discussing aspects of aop. *Commun. ACM*, 44(10):33–38, 2001.
- [EAWJ02] E. N. (Mootaz) Elnozahy, Lorenzo Alvisi, Yi-Min Wang, and David B. Johnson. A survey of rollback-recovery protocols in message-passing systems. *ACM Comput. Surv.*, 34(3):375–408, 2002.
- [Edw05] W. Keith Edwards. Putting computing in context: An infrastructure to support extensible context-enhanced collaborative applications. *ACM Trans. Comput.-Hum. Interact.*, 12(4):446–474, 2005.
- [EFB01] Tzilla Elrad, Robert E. Filman, and Atef Bader. Aspect-oriented programming: Introduction. *Commun. ACM*, 44(10):29–32, 2001.
- [EOSW07] Gregor Engels, Bill Opdyke, Douglas C. Schmidt, and Frank Weil, editors. *Model Driven Engineering Languages and Systems, 10th International Conference, MoDELS 2007, Nashville, USA, September 30 - October 5, 2007, Proceedings*, volume 4735 of *Lecture Notes in Computer Science*. Springer, 2007.
- [ERA08] Web site of the ERA-Pilot QIST Coordinated Action. <http://www.qist-europe.net/>, last accessed 2008-01-07, 2008.
- [Erl05] Thomas Erl. *Service-Oriented Architecture—Concepts, Technology and Design*. Prentice Hall Service-Oriented Computing Series. Prentice Hall, 2005.
- [ESW06] Dennis Edwards, Sharon Simmons, and Norman Wilde. An approach to feature location in distributed systems. *Journal of Systems and Software*, 79(1):57–68, 2006.
- [Eva04] Eric Evans. *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Addison-Wesley Professional, 2004.
- [Fey60] Richard P. Feynman. There’s plenty of room at the bottom—an invitation to enter a new field of physics. *Engineering and Science*, February 1960.
- [FH04] Paul Franois and Vincent Hakim. Design of genetic networks with specified functions by evolution in silicon. *Proc Natl Acad Sci U S A*, 101(2):580–585, January 2004.
- [FM04] Anna Formica and Michele Missikoff. Inheritance processing and conflicts in structural generalization hierarchies. *ACM Comput. Surv.*, 36(3):263–290, 2004.

- [For07] Richard Ford. Open vs. closed: which source is more secure? *Queue*, 5(1):32–38, 2007.
- [Fow02] Martin Fowler. *Patterns of Enterprise Application Architecture*. Addison-Wesley Longman, Amsterdam, November 2002.
- [FS03] Niels Ferguson and Bruce Schneier. *Practical Cryptography*. Wiley Publishing, Inc., 2003.
- [FS06] Emerson Farrugia and Robert Simon. An efficient and secure protocol for sensor network time synchronization. *Journal of Systems and Software*, 79(2):147–162, 2006.
- [FUKM06] Howard Foster, Sebastian Uchitel, Jeff Kramer, and Jeff Magee. Ws-engineer: A tool for model-based verification of web service compositions and choreography. In *IEEE International Conference on Software Engineering (ICSE 2006), Shanghai, China, May 2006.*, 2006.
- [FW07a] Gordon Fraser and Franz Wotawa. Improving model-checkers for software testing. *qsic*, 0:25–31, 2007.
- [FW07b] Gordon Fraser and Franz Wotawa. Redundancy based test-suite reduction. In Matthew B. Dwyer and Antónia Lopes, editors, *FASE*, volume 4422 of *Lecture Notes in Computer Science*, pages 291–305. Springer, 2007.
- [FW07c] Gordon Fraser and Franz Wotawa. Using ltl rewriting to improve the performance of model-checker based test-case generation. In *A-MOST*, pages 64–74. ACM, 2007.
- [GCBL06] Deepak Ganesan, Răzvan Cristescu, and Baltasar Beferull-Lozano. Power-efficient sensor placement and transmission structure for data gathering under distortion constraints. *ACM Trans. Sen. Netw.*, 2(2):155–181, 2006.
- [GCC06] Franck Gechter, Vincent Chevrier, and François Charpillat. A reactive agent-based problem-solving model: Application to localization and tracking. *ACM Trans. Auton. Adapt. Syst.*, 1(2):189–222, 2006.
- [Gee06] Daniel E. Geer. Playing for keeps. *Queue*, 4(9):42–48, 2006.
- [Gee07] Daniel E. Geer. The evolution of security. *Queue*, 5(3):30–35, 2007.
- [GEH01] Mohamed Gad-El-Hak, editor. *The MEMS Handbook*. CRC Press, Boca Raton, FL, 2001.
- [GEH06] Mohamed Gad-El-Hak. *MEMS—Introduction and Fundamentals*. CRC Press, Boca Raton, FL, 2006.
- [GEM07] Web site of the international Genetically Engineered Machine competition, 2007. <http://parts.mit.edu/wiki>.
- [GH06] Liqiang Geng and Howard J. Hamilton. Interestingness measures for data mining: A survey. *ACM Comput. Surv.*, 38(3):9, 2006.
- [GHJV95] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Longman, Amsterdam, 1995.
- [Gib04] Wayt W. Gibbs. Synthetic life. *Scientific American*, 290(5):74–81, May 2004.
- [Gla03] Robert L. Glass. Guest editor’s introduction: The state of the practice of software engineering. *IEEE Software*, 20(6):20–21, 2003.

- [GLC07] David Gay, Philip Levis, and David Culler. Software design patterns for tinyos. *Trans. on Embedded Computing Sys.*, 6(4):22, 2007.
- [GLT03] Bas Graaf, Marco Lormans, and Hans Toetenel. Embedded software engineering: The state of the practice. *IEEE Software*, 20(6):61–69, 2003.
- [GM04] Seth Copen Goldstein and Todd C. Mowry. Claytronics: A scalable basis for future robots. In *RoboSphere 2004*, Moffett Field, CA, November 2004.
- [GMS87] Hector Garcia-Molina and Kenneth Salem. Sagas. In *SIGMOD '87: Proc. of ACM SIGMOD Int. Conf. on Management of Data*, pages 249–259, New York, 1987. ACM Press.
- [GMZ05] Paolo Giorgini, Fabio Massacci, and Nicola Zannone. Security and Trust Requirements Engineering. In *Foundations of Security Analysis and Design III - Tutorial Lectures*, volume 3655 of *LNCS*, pages 237–272. Springer-Verlag GmbH, 2005.
- [GMZ07] Paolo Giorgini, Haralambos Mouratidis, and Nicola Zannone. Modelling Security and Trust with Secure Tropos. In Haralambos Mouratidis and Paolo Giorgini, editors, *Integrating Security and Software Engineering: Advances and Future Vision*, chapter VIII. Idea Group, 2007.
- [GS04] Jack Greenfield and Keith Short. *Software Factories: Assembling Applications with Patterns, Frameworks, Models and Tools*. Wiley, September 2004.
- [GWBG06] Jr. George W. Beeler and Dana Gardner. A requirements primer. *Queue*, 4(7):22–26, 2006.
- [HB07] Simon Harper and Sean Bechhofer. Sadie: Structural semantics for accessibility and device independence. *ACM Trans. Comput.-Hum. Interact.*, 14(2):10, 2007.
- [Hea87] Tom Head. Formal language theory and dna: an analysis of the generative capacity of specific recombinant behaviors. *Bull. Math. Biology*, 49(6):737–759, 1987.
- [HH98] Tad Hogg and Bernardo A. Huberman. Controlling smart matter. *Smart Materials and Structures*, 7(R1), 1998.
- [Hig03] Jim Highsmith. *Agile Software Development Ecosystems*. Addison-Wesley, 2003.
- [HJ07] Jaap-Henk Hoepman and Bart Jacobs. Increased security through open source. *Commun. ACM*, 50(1):79–83, 2007.
- [HKMU06] Dan Hirsch, Jeff Kramer, Jeff Magee, and Sebastian Uchitel. Modes for software architectures. In *Proceedings of EWSA 2006, 3rd European Workshop on Software Architecture*, LNCS. Springer Verlag, 2006. To be published.
- [Hol92] John H. Holland. *Adaptation in Natural and Artificial Systems*. The MIT Press, second edition edition, 1992.
- [Hol04] Gerard J. Holzmann. *The Spin Model Checker: Primer and Reference Manual*. Addison-Wesley Professional, 2004.
- [PHPS05] Ken Hinckley, Jeff Pierce, Eric Horvitz, and Mike Sinclair. Foreground and background interaction with sensor-enhanced mobile devices. *ACM Trans. Comput.-Hum. Interact.*, 12(1):31–52, 2005.

- [HSL⁺07] Douglas Herbert, Vinaitheerthan Sundaram, Yung-Hsiang Lu, Saurabh Bagchi, and Zhiyuan Li. Adaptive correctness monitoring for wireless sensor networks using hierarchical distributed run-time invariant checking. *ACM Trans. Auton. Adapt. Syst.*, 2(3):8, 2007.
- [HTW07] Chi-Fu Huang, Yu-Chee Tseng, and Hsiao-Lu Wu. Distributed protocols for ensuring both coverage and connectivity of a wireless sensor network. *ACM Trans. Sen. Netw.*, 3(1):5, 2007.
- [IEE98] IEEE. *Std. 1220-1998. Standard for Application and Management of the System Engineering Process*. IEEE Press, Piscataway, N.J., 1998.
- [Inf04] Information Technology for European Advancement (ITEA) Office Association. ITEA technology roadmap for software-intensive systems, 2nd edition. <http://www.itea-office.org>, May 2004.
- [JBCF07] Eric C. Jensen, Steven M. Beitzel, Abdur Chowdhury, and Ophir Frieder. Repeatable evaluation of search services in dynamic environments. *ACM Trans. Inf. Syst.*, 26(1):1, 2007.
- [JGH07] Caroline Jay, Mashhuda Glencross, and Roger Hubbard. Modeling the effects of delayed haptic and visual feedback in a collaborative virtual environment. *ACM Trans. Comput.-Hum. Interact.*, 14(2):8, 2007.
- [JMS07] Predrag R. Jelenković, Petar Momčilović, and Mark S. Squillante. Scalability of wireless networks. *IEEE/ACM Trans. Netw.*, 15(2):295–308, 2007.
- [JP06] Apoorva Jindal and Konstantinos Psounis. Modeling spatially correlated data in sensor networks. *ACM Trans. Sen. Netw.*, 2(4):466–499, 2006.
- [KAG⁺07] Brian Kirby, Burak Aksak, Seth Copen Goldstein, James F. Hoburg, Todd C. Mowry, and Padmanabhan Pillai. A modular robotic system using magnetic force effectors. In *In Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS '07)*, October 2007.
- [KD07] Prakash Kolan and Ram Dantu. Socio-technical defense against voice spamming. *ACM Trans. Auton. Adapt. Syst.*, 2(1):2, 2007.
- [KdRB91] Gregor Kiczales, Jim des Rivières, and Daniel G. Bobrow. *The Art of the Metaobject Protocol*. MIT-Press, 1991.
- [KF07] Shriram Krishnamurthi and Kathi Fisler. Foundations of incremental aspect model-checking. *ACM Trans. Softw. Eng. Methodol.*, 16(2):7, 2007.
- [KHH⁺01] Gregor Kiczales, Erik Hilsdale, Jim Hugunin, Mik Kersten, Jeffrey Palm, and William G. Griswold. An overview of AspectJ. In Jørgen Lindskov Knudsen, editor, *ECOOP*, volume 2072 of *Lecture Notes in Computer Science*, pages 327–353. Springer, 2001.
- [KHL⁺07] Akrivi Katifori, Constantin Halatsis, George Lepouras, Costas Vassilakis, and Eugenia Giannopoulou. Ontology visualization methods—a survey. *ACM Comput. Surv.*, 39(4):10, 2007.
- [KHZS07] Aman Kansal, Jason Hsu, Sadaf Zahedi, and Mani B. Srivastava. Power management in energy harvesting sensor networks. *Trans. on Embedded Computing Sys.*, 6(4):32, 2007.
- [KK07] Aditya Karnik and Anurag Kumar. Distributed optimal self-organization in ad hoc wireless sensor networks. *IEEE/ACM Trans. Netw.*, 15(5):1035–1045, 2007.

- [KKA⁺04] Hideki Kobayashi, Mads Kaern, Michihiro Araki, Kristy Chung, Timothy S Gardner, Charles R Cantor, and James J Collins. Programmable cells: interfacing natural and engineered gene networks. *Proc Natl Acad Sci U S A*, 101(22):8414–8419, June 2004.
- [KKP⁺07] Aman Kansal, William Kaiser, Gregory Pottie, Mani Srivastava, and Gaurav Sukhatme. Reconfiguration methods for mobile sensor networks. *ACM Trans. Sen. Netw.*, 3(4):22, 2007.
- [KLM⁺97] Gregor Kiczales, John Lamping, Anurag Mendhekar, Chris Maeda, Cristina Videira Lopes, Jean-Marc Loingtier, and John Irwin. Aspect-oriented programming. In *ECOOP*, pages 220–242, 1997.
- [KMM00] Matt Kaufmann, Panagiotis Manolis, and J Strother Moore. *Computer-Aided Reasoning: An Approach*. Kluwer Academic Publishers, June 2000.
- [KPS95] Charlie Kaufman, Radia Perlman, and Mike Speciner. *Network security: private communication in a public world*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1995.
- [Lan06] Ivan Lanese. *Synchronization Strategies for Global Computing Models*. PhD thesis, Ph.D. school in Computer Science, University of Pisa, Pisa, Italy, 2006.
- [LD00] J. Lobo and J. Dias. Fusing of image and inertial sensing for camera calibration. In *Proceedings of the IEEE Conference on Multisensor Fusion and Integration for Intelligent Systems, MFI*, 2000.
- [Lee06] Edward A. Lee. Cyber-physical systems—are computing foundations adequate? In *NSF Workshop on Cyber-Physical Systems*, October 2006.
- [LG04] Oliver Lemon and Alexander Gruenstein. multithreaded context for robust conversational interfaces: Context-sensitive speech recognition and interpretation of corrective fragments. *ACM Trans. Comput.-Hum. Interact.*, 11(3):241–267, 2004.
- [LGHH08] Chunyuan Liao, François Guimbretière, Ken Hinckley, and Jim Hollan. Papiercraft: A gesture-based command system for interactive paper. *ACM Trans. Comput.-Hum. Interact.*, 14(4):1–27, 2008.
- [LHMW07] Damian M. Lyons, D. Frank Hsu, Qiang Ma, and Liang Wang. Combinatorial fusion criteria for robot mapping. In *AINA*, pages 847–852. IEEE Computer Society, 2007.
- [LOO01] Karl Lieberherr, Doug Orleans, and Johan Ovlinger. Aspect-oriented programming with adaptive methods. *Commun. ACM*, 44(10):39–41, 2001.
- [LORS06] Dean H. Lorenz, Ariel Orda, Danny Raz, and Yuval Shavitt. Efficient qos partition and routing of unicast and multicast. *IEEE/ACM Trans. Netw.*, 14(6):1336–1347, 2006.
- [LP06] Loukas Lazos and Radha Poovendran. Stochastic coverage in heterogeneous sensor networks. *ACM Trans. Sen. Netw.*, 2(3):325–358, 2006.
- [LS01] Peter Loscocco and Stephen Smalley. Integrating flexible support for security policies into the linux operating system. <http://www.nsa.gov/selinux/papers/slinux-abs.cfm>, last accessed 2008-01-19, February 2001.
- [LSF03] Timothy Lethbridge, Janice Singer, and Andrew Forward. How software engineers use documentation: The state of the practice. *IEEE Software*, 20(6):35–39, 2003.
- [Lys02] Sergey Edward Lyshevski. *MEMS and NEMS: Systems, Devices and Structures*. CRC Press, Boca Raton, FL, 2002.

- [Mac02] Bruce J. Maclennan. Replication, sharing, deletion, lists, and numerals: Progress on universally programmable intelligent matter, November 2002.
- [Mae87] Pattie Maes. Concepts and experiments in computational reflection. *SIGPLAN Not.*, 22(12):147–155, 1987.
- [MBB07] Joanna McGrenere, Ronald M. Baecker, and Kellogg S. Booth. A field evaluation of an adaptable two-interface design for feature-rich software. *ACM Trans. Comput.-Hum. Interact.*, 14(1):3, 2007.
- [McT02] Michael F. McTear. Spoken dialogue technology: enabling the conversational user interface. *ACM Comput. Surv.*, 34(1):90–169, 2002.
- [MDM07] Johann Van Der Merwe, Dawoud Dawoud, and Stephen McDonald. A survey on peer-to-peer key management for mobile ad hoc networks. *ACM Comput. Surv.*, 39(1):1, 2007.
- [Mem08] Web site of the memsnet.org project, 2008. <http://www.memsnet.org/>, last accessed 2008-01-10.
- [MHS05] Marjan Mernik, Jan Heering, and Anthony M. Sloane. When and how to develop domain-specific languages. *ACM Comput. Surv.*, 37(4):316–344, 2005.
- [MIRAG06] Eduardo Mena, Arantza Illarramendi, Jose A. Royo, and nI Alfredo Go' A software retrieval service based on adaptive knowledge-driven agents for wireless environments. *ACM Trans. Auton. Adapt. Syst.*, 1(1):67–90, 2006.
- [MLC07] Bernardo A. Movsichoff, Constantino M. Lagoa, and Hao Che. End-to-end optimal algorithms for integrated qos, traffic engineering, and failure recovery. *IEEE/ACM Trans. Netw.*, 15(4):813–823, 2007.
- [MN06] Yehia Massoud and Arthur Nieuwoudt. Modeling and design challenges and solutions for carbon nanotube-based interconnect in future high performance integrated circuits. *J. Emerg. Technol. Comput. Syst.*, 2(3):155–196, 2006.
- [MO06] Rodney Van Meter and Mark Oskin. Architectural implications of quantum computing technologies. *J. Emerg. Technol. Comput. Syst.*, 2(1):31–63, 2006.
- [Moo65] Gordon E. Moore. Cramming more components onto integrated circuits. *Electronics*, 38(8):114–117, 1965.
- [MS03] Robin Milner and Susan Stepney. Nanotechnology: Computer science opportunities and challenges, August 2003. Technical report, Submission by the UK Computing Research Committee to the Nanotechnology Working Group of the Royal Society and the Royal Academy of Engineering.
- [MWK⁺05] Cynthia Matuszek, Michael J. Witbrock, Robert C. Kahlert, John Cabral, David Schneider, Purvesh Shah, and Douglas B. Lenat. Searching for common sense: Populating cyc from the web. In Manuela M. Veloso and Subbarao Kambhampati, editors, *AAAI*, pages 1430–1435. AAAI Press / The MIT Press, 2005.
- [NAS08] NASA. Intelligent data understanding subproject of the intelligent systems project. <http://ti.arc.nasa.gov/is/IDU/index.html>, 2008.
- [Nau07] Peter Naur. Computing versus human thinking. *Commun. ACM*, 50(1):85–94, 2007.
- [NB07] Sridhar Nerur and VenuGopal Balijepally. Theoretical reflections on agile development methodologies. *Commun. ACM*, 50(3):79–83, 2007.

- [NL03] Colin J. Neill and Phillip A. Laplante. Requirements engineering: The state of the practice. *IEEE Software*, 20(6):40–45, 2003.
- [NLF07] Eduardo F. Nakamura, Antonio A. F. Loureiro, and Alejandro C. Frery. Information fusion for wireless sensor networks: Methods, models, and classifications. *ACM Comput. Surv.*, 39(3):9, 2007.
- [NN07] George V. Neville-Neil. Building secure web applications. *Queue*, 5(5):22–26, 2007.
- [NSF06] Web site of the nsf workshop on cyber-physical systems. <http://varma.ece.cmu.edu/cps/>, last accessed 2008-01-20, October 2006.
- [ODC04] Sharon Oviatt, Courtney Darves, and Rachel Coulston. Toward adaptive conversational interfaces: Modeling speech convergence with animated personas. *ACM Trans. Comput.-Hum. Interact.*, 11(3):300–328, 2004.
- [OS04] Sharon Oviatt and Stephanie Seneff. Introduction to mobile and adaptive conversational interfaces. *ACM Trans. Comput.-Hum. Interact.*, 11(3):237–240, 2004.
- [OT01] Harold Ossher and Peri Tarr. Using multidimensional separation of concerns to (re)shape evolving software. *Commun. ACM*, 44(10):43–50, 2001.
- [PBK07] Philip Papadopoulos, Greg Bruno, and Mason Katz. Beyond beowulf clusters. *Queue*, 5(3):36–43, 2007.
- [PBO07] Richard F. Paige, Phillip J. Brooke, and Jonathan S. Ostroff. Metamodel-based model conformance and multiview consistency checking. *ACM Trans. Softw. Eng. Methodol.*, 16(3):11, 2007.
- [PDL07] Jaidev Patwardhan, Chris Dwyer, and Alvin R. Lebeck. A self-organizing defect tolerant simd architecture. *J. Emerg. Technol. Comput. Syst.*, 3(2):10, 2007.
- [PdV07] Shailesh Patil and Gustavo de Veciana. Managing resources and quality of service in heterogeneous wireless systems exploiting opportunism. *IEEE/ACM Trans. Netw.*, 15(5):1046–1058, 2007.
- [PLR07] Tao Peng, Christopher Leckie, and Kotagiri Ramamohanarao. Survey of network-based defense mechanisms countering the dos and ddos problems. *ACM Comput. Surv.*, 39(1):3, 2007.
- [Pos07] Stefan Poslad. Specifying protocols for multi-agent systems interaction. *ACM Trans. Auton. Adapt. Syst.*, 2(4):15, 2007.
- [PPSM07] Loris Penserini, Anna Perini, Angelo Susi, and John Mylopoulos. High variability design for software agents: Extending tropos. *ACM Trans. Auton. Adapt. Syst.*, 2(4):16, 2007.
- [Pro07] NESSI Project. Web-site of the NESSI technology platform. www.nessi-europe.com, 2007.
- [PS04] Stefan Pleisch and André Schiper. Approaches to fault-tolerant and transactional mobile agent execution—an algorithmic view. *ACM Comput. Surv.*, 36(3):219–262, 2004.
- [PTW07] Shamimabi Paurobally, Valentina Tamma, and Michael Wooldridge. A framework for web service negotiation. *ACM Trans. Auton. Adapt. Syst.*, 2(4):14, 2007.
- [PUBV07] Lucian Prodan, Mihai Udrescu, Oana Boncalo, and Mircea Vladutiu. Design for dependability in emerging technologies. *J. Emerg. Technol. Comput. Syst.*, 3(2):6, 2007.

- [QCS07] Daji Qiao, Sunghyun Choi, and Kang G. Shin. Interference analysis and transmit power control in IEEE 802.11a/h wireless lans. *IEEE/ACM Trans. Netw.*, 15(5):1007–1020, 2007.
- [Rei03] Donald J. Reifer. Is the software engineering state of the practice getting closer to the state of the art? *IEEE Software*, 20(6):78–83, 2003.
- [REM⁺04] Gregg Rothermel, Sebastian Elbaum, Alexey G. Malishevsky, Praveen Kallakuri, and Xuemei Qiu. On test suite composition and cost-effective regression testing. *ACM Trans. Softw. Eng. Methodol.*, 13(3):277–331, 2004.
- [RGG07] Ramprasad Ravichandran, Geoffrey Gordon, and Seth Copen Goldstein. A scalable distributed algorithm for shape transformation in multi-robot systems. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems IROS '07*, October 2007.
- [RH03] Sandro Rafaeli and David Hutchison. A survey of key management for secure group communication. *ACM Comput. Surv.*, 35(3):309–329, 2003.
- [RKSS05] David Russinoff, Matt Kaufmann, Eric Smith, and Robert Summers. Formal verification of floating-point rtl at amd using the acl2 theorem prover. In Nikolai Simonov, editor, *Proceedings of the 17th IMACS World Congress on Scientific Computing*, July 2005.
- [RKW⁺06] Umakishore Ramachandran, Rajnish Kumar, Matthew Wolenetz, Brian Cooper, Bikash Agarwalla, Junsuk Shin, Phillip Hutto, and Arnab Paul. Dynamic data fusion for future sensor networks. *ACM Trans. Sen. Netw.*, 2(3):404–443, 2006.
- [RPV03] William N. Robinson, Suzanne D. Pawlowski, and Vecheslav Volkov. Requirements interaction management. *ACM Comput. Surv.*, 35(2):132–190, 2003.
- [RPW03] Paul W. K. Rothmund, Nick Papadakis, and Eric Winfree. Algorithmic self-assembly of DNA Sierpinski triangles. In *Preliminary Proceedings of DNA Computing, 9th international Workshop on DNA-Based Computers, DNA 2003*, page 125, June 2003. Madison, Wisconsin, USA 1-4.
- [RT07] Reza M.P. Rad and Mohammad Tehranipoor. Evaluating area and performance of hybrid fpgas with nanoscale clusters and cmos routing. *J. Emerg. Technol. Comput. Syst.*, 3(3):15, 2007.
- [RYC⁺07] Shangping Ren, Yue Yu, Nianen Chen, Jeffrey J.-P. Tsai, and Kevin Kwiat. The role of roles in supporting reconfigurability and fault localizations for open distributed and embedded systems. *ACM Trans. Auton. Adapt. Syst.*, 2(3):10, 2007.
- [SA06] Gokul Soundararajan and Cristiana Amza. Reactive provisioning of backend databases in shared dynamic content server clusters. *ACM Trans. Auton. Adapt. Syst.*, 1(2):151–188, 2006.
- [San05] Paolo Santi. Topology control in wireless ad hoc and sensor networks. *ACM Comput. Surv.*, 37(2):164–194, 2005.
- [SBC⁺06] Susan Stepney, Samuel L. Braunstein, John A. Clark, Andrew M. Tyrrell, Andrew Adamatzky, Robert E. Smith, Thomas R. Addis, Colin G. Johnson, Jonathan Timmis, Peter H. Welch, Robin Milner, and Derek Partridge. Journeys in non-classical computation II: initial journeys and waypoints. *Parallel Algorithms Appl*, 21(2):97–125, 2006.
- [SC06] Fei Su and Krishnendu Chakrabarty. Yield enhancement of reconfigurable microfluidics-based biochips using interstitial redundancy. *J. Emerg. Technol. Comput. Syst.*, 2(2):104–128, 2006.

- [Sch00] Bruce Schneier. *Secrets and Lies—Digital Security in a Networked World*. John Wiley & Sons, Inc., 2000.
- [Sek07] Lukáš Sekanina. Evolutionary functional recovery in virtual reconfigurable circuits. *J. Emerg. Technol. Comput. Syst.*, 3(2):8, 2007.
- [SH99] Apkar Salatian and Jim Hunter. Deriving trends in historical and real-time continuously sampled medical data. *J. Intell. Inf. Syst.*, 13(1-2):47–71, 1999.
- [SHM07] Andrew Sears, Vicki L. Hanson, and Brad Myers. Introduction to special issue on computers and accessibility. *ACM Trans. Comput.-Hum. Interact.*, 14(3):11, 2007.
- [Sib00] Sibilina R., editor. *Moore’s Law and its Implications for Information Warfare*, volume 3 of *International AOC Electronic Warfare Conference*, Alexandria, Virginia, USA, May 20-25 2000. Association of Old Crows (AOC).
- [Smi84] Brian Cantwell Smith. Reflection and semantics in lisp. In *POPL ’84: Proceedings of the 11th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*, pages 23–35, New York, NY, USA, 1984. ACM.
- [SMS07] Gaurav Sharma, Ravi Mazumdar, and Ness B. Shroff. Delay and capacity trade-offs in mobile ad hoc networks: a global perspective. *IEEE/ACM Trans. Netw.*, 15(5):981–992, 2007.
- [Soh07] David Sohn. Understanding drm. *Queue*, 5(7):32–39, 2007.
- [Som07] Ian Sommerville. *Software Engineering*. Addison-Wesley, eighth edition edition, 2007.
- [Spi08] SPIN Model Checker. www.spinroot.com, 2008. Last visited: 2008-01-24.
- [SQX⁺07] Mei-Ling Shyu, Thiago Quirino, Zongxing Xie, Shu-Ching Chen, and Liwu Chang. Network intrusion detection through adaptive sub-eigenspace modeling in multiagent systems. *ACM Trans. Auton. Adapt. Syst.*, 2(3):9, 2007.
- [SS05] Yasushi Saito and Marc Shapiro. Optimistic replication. *ACM Comput. Surv.*, 37(1):42–81, 2005.
- [SSPvS04] Swaminathan Sivasubramanian, Michal Szymaniak, Guillaume Pierre, and Maarten van Steen. Replication for web hosting systems. *ACM Comput. Surv.*, 36(3):291–334, 2004.
- [SSZ⁺06] Georg Seelig, David Soloveichik, David Yu Zhang, , and Erik Winfree. Enzyme-free nucleic acid logic circuits. *Science*, 314(5805):1585–1588, December 2006.
- [Sul01a] Gregory T. Sullivan. Aspect-oriented programming using reflection and metaobject protocols. *Commun. ACM*, 44(10):95–97, 2001.
- [Sul01b] Gregory T. Sullivan. Dynamic partial evaluation. In *PADO ’01: Proceedings of the Second Symposium on Programs as Data Objects*, pages 238–256, London, UK, 2001. Springer-Verlag.
- [Syn07] Syntheticbiology.org Web site, 2007. <http://syntheticbiology.org/>.
- [TAC06] Jordi Turmo, Alicia Ageno, and Neus Català. Adaptive information extraction. *ACM Comput. Surv.*, 38(2):4, 2006.
- [Tah06] Mehdi B. Tahoori. Application-independent defect tolerance of reconfigurable nanoarchitectures. *J. Emerg. Technol. Comput. Syst.*, 2(3):197–218, 2006.

- [TAPH05] William Tolone, Gail-Joon Ahn, Tanusree Pai, and Seng-Phil Hong. Access control in collaborative systems. *ACM Comput. Surv.*, 37(1):29–41, 2005.
- [TGA07] Mark Truran, James Goulding, and Helen Ashman. Autonomous authoring tools for hypertext. *ACM Comput. Surv.*, 39(3):8, 2007.
- [Tha02] Richard H. Thayer. Software system engineering: A tutorial. *IEEE Computer*, 35(4):68–73, 2002.
- [TM05] Loren Terveen and David W. McDonald. Social matching: A framework and research agenda. *ACM Trans. Comput.-Hum. Interact.*, 12(3):401–434, 2005.
- [TMKW07] Matthew E. Taylor, Cynthia Matuszek, Bryan Klimt, and Michael J. Witbrock. Autonomous classification of knowledge into an ontology. In David Wilson and Geoff Sutcliffe, editors, *FLAIRS Conference*, pages 140–145. AAAI Press, 2007.
- [TMM⁺07] Gianluca Tempesti, Daniel Mange, Pierre-Andre Mudry, Joël Rossier, and Andre Stauffer. Self-replicating hardware for reliability: The embryonics project. *J. Emerg. Technol. Comput. Syst.*, 3(2):9, 2007.
- [TOHSMS99] Peri Tarr, Harold Ossher, William Harrison, and Jr. Stanley M. Sutton. N degrees of separation: multi-dimensional separation of concerns. In *ICSE '99: Proceedings of the 21st international conference on Software engineering*, pages 107–119, Los Alamitos, CA, USA, 1999. IEEE Computer Society Press.
- [ULNB07] Suleyman Uludag, King-Shan Lui, Klara Nahrstedt, and Gregory Brewster. Analysis of topology aggregation techniques for qos routing. *ACM Comput. Surv.*, 39(3):7, 2007.
- [Unr08] Ronald C. Unrau. Development techniques for using simulation to remove risk in software/hardware integration. http://www.redhat.com/support/wpapers/cygnus/cygnus_risk/index.html#toc, last accessed 2008-01-20, 2008.
- [URv03] Theo Ungerer, Borut Robič, and Jurij Šilc. A survey of processors with explicit multi-threading. *ACM Comput. Surv.*, 35(1):29–63, 2003.
- [vBK06] F. van Breugel and M. Koshkina. Models and verification of BPEL, 2006. <http://www.cse.yorku.ca/~simfranck/research/drafts/tutorial.pdf>.
- [VBR06] Srikumar Venugopal, Rajkumar Buyya, and Kotagiri Ramamohanarao. A taxonomy of data grids for distributed data sharing, management, and processing. *ACM Comput. Surv.*, 38(1):3, 2006.
- [VdPK05] Thierry Van der Pyl and Antonella Karlson, editors. *Quantum Information Processing & Communications in Europe*. European Union, IST-FET, 2005.
- [VF05] Vasanth Venkatachalam and Michael Franz. Power reduction techniques for microprocessor systems. *ACM Comput. Surv.*, 37(3):195–237, 2005.
- [W3C08a] W3C Consortium. Extensible markup language (xml) web site, 2008. <http://www.w3.org/XML/>, last accessed: 2008-01-19.
- [W3C08b] W3C Consortium. Resource description framework (rdf) web site, 2008. <http://www.w3.org/RDF/>, last accessed: 2008-01-19.
- [W3C08c] W3C Consortium. W3C semantic web activity web site, 2008. <http://www.w3.org/2001/sw/>, last accessed: 2008-01-19.

- [WAH⁺93] Stephanie White, Mack W. Alford, Julian Holtzman, C. Stephen Kuehl, Brian McCay, David Oliver, David Owens, Colin Tully, and Allan Willey. Systems engineering of computer-based systems, state of practice working group. *IEEE Computer*, 26(11):54–65, 1993.
- [WBH⁺03] Ron Weiss, Subhayu Basu, Sara Hooshangi, Abigail Kalmbach, David Karig, Rishabh Mehreja, and Ilka Netravali. Genetic circuit building blocks for cellular computation, communications, and signal processing. *Natural Computing*, 2(1):43–84, 2003.
- [WH07] Martin Wirsing and Matthias Hölzl. Software-intensive systems. Report of the Beyond-the-Horizon WG6, 2007.
- [WJH⁺04] C. M. Wyss, A. James, W. Hasselbring, S. Conrad, and Hagen Höpfner. Report on the engineering federated information systems 2003 workshop (efis 2003). *SIGSOFT Softw. Eng. Notes*, 29(2):1–3, 2004.
- [WK00] Ron Weiss and Tom Knight. Engineered communications for microbial robotics. In *DNA: International Workshop on DNA-Based Computers*, LNCS, 2000.
- [WNET07] Kenichi Watanabe, Yoshio Nakajima, Tomoya Enokido, and Makoto Takizawa. Ranking factors in peer-to-peer overlay networks. *ACM Trans. Auton. Adapt. Syst.*, 2(3):11, 2007.
- [WZL07] Dan Wang, Qian Zhang, and Jiangchuan Liu. The self-protection problem in wireless sensor networks. *ACM Trans. Sen. Netw.*, 3(4):20, 2007.
- [YJS06] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking: A survey. *ACM Comput. Surv.*, 38(4):13, 2006.
- [YVS07] Suyoung Yoon, Chanchai Veerarittiphan, and Mihail L. Sichitiu. Tiny-sync: Tight time synchronization for wireless sensor networks. *ACM Trans. Sen. Netw.*, 3(2):8, 2007.
- [YWA02] Yohei Yokobayashi, Ron Weiss, and Frances H Arnold. Directed evolution of a genetic circuit. *Proc Natl Acad Sci U S A*, 99(26):16587–16591, December 2002.
- [ZB05] Shumin Zhai and Victoria Bellotti. Introduction to sensing-based interaction. *ACM Trans. Comput.-Hum. Interact.*, 12(1):1–2, 2005.
- [ZBKK05] Gefei Zhang, Hubert Baumeister, Nora Koch, and Alexander Knapp. Aspect-Oriented Modeling of Access Control in Web Applications. In *Proc. 6th Int. Wsh. Aspect Oriented Modeling (WAOM'05)*, Chicago, 2005.
- [Zha07] Huimin Zhao. Semantic matching across heterogeneous data sources. *Commun. ACM*, 50(1):45–50, 2007.
- [ZHKS06] Gang Zhou, Tian He, Sudha Krishnamurthy, and John A. Stankovic. Models and solutions for radio irregularity in wireless sensor networks. *ACM Trans. Sen. Netw.*, 2(2):221–262, 2006.
- [ZLZS07] Dong Zheng, Yan Liu, Jiyang Zhao, and Abdulmotaleb El Saddik. A survey of rst invariant image watermarking algorithms. *ACM Comput. Surv.*, 39(2):5, 2007.
- [ZM06] Justin Zobel and Alistair Moffat. Inverted files for text search engines. *ACM Comput. Surv.*, 38(2):6, 2006.