

Research Agenda for the Interlink WG1: Software-Intensive Systems and New Computing Paradigms

—DRAFT of October 14, 2008—

Matthias Hözl, Axel Rauschmayer, Martin Wirsing

October 14, 2008

Abstract

In the next decades large numbers of software-intensive systems will be developed and deployed to fulfill 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. Engineering these systems—called *ensembles*—will be the major challenge facing developers of future software-intensive systems. The members of the Interlink opening workshop have identified the following specific areas as promising for future research: Engineering Physical Ensembles, Organic Software and Systems Engineering, and Societal Computing.

Introduction

Software-intensive systems become increasingly important for a multitude of products and services from all sectors of the economy, our national and international infrastructure, and our daily lives. The gains in productivity and prosperity can to a large degree be attributed to developments in ICT (Information and Communication Technologies).

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 such as quantum computing.

We call this future generation of software-intensive systems *ensembles*. 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.

1 Research Challenges

Many of the numerous challenges that we expect to see in the development of software-intensive systems can be classified as belonging to one of the following areas: massive numbers of nodes per system, open and non-deterministic environments, and adaptation:

Massive Numbers of Nodes per System A massive increase in the number of nodes of future software-intensive systems will be one of the most visible features: the development of multi-core processors with tens to thousands of cores integrated on a single chip implies that even single-chip systems will have to be treated as consisting of large numbers of individual nodes, the availability of cheap, low-energy mobile devices ensures that we will see an increasing number of elements with computational capability in the next years. Both of these trends are true even for systems built out of traditional computing devices. Furthermore, new manufacturing methods such as 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 PDAs, mobile phones, 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 many other devices, such as embedded systems, or even RFID-equipped consumer goods, 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 huge impact—both positive and

negative—that ensembles will have on society means that we need to understand ways to reliably and predictably model, design, and program them.

Ensemble engineering is closely connected to, and incorporates ideas from, other research areas such as software engineering, artificial intelligence, complex adaptive systems and non-conventional programming paradigms. Ensembles can be categorized along several different axes, for example:

- **Physical—Virtual:** Is the system mainly concerned with sensing and acting in the real world, or is it purely virtual, or is it somewhere in between?
- **Homogenous—Inhomogenous:** Is the system made of (large numbers of) identical parts, is it made of (a large number of) different components taken from a few different types of component, or does it consist of a large number of parts, all of which are different.
- **Cooperative—Competitive:** Do the individual parts cooperate to achieve a certain task or do they compete with each other. There might also be systems which are in-between, such as a system where the components are cooperating but competing for a scarce resource to fulfill their individual tasks.

Further distinctions between systems are the topology of their communication links, the average computational power of the nodes, or the purpose of the system, etc. While some research topics are specific to a certain kind of ensemble or a certain application area, many are common to all ensembles.

1.1 Research Challenges for Ensemble Engineering

In the InterLink WG1 workshops, the following long-term research challenges have been identified. They can be classified as one of the categories “properties of ensembles”, “specification and design”, “assurances”, and “implementation and verification”.

1.1.1 Properties of Ensembles.

The increase in scale along a variety of dimensions leads to ensembles having either new unique properties or displaying properties of traditional systems, but on a much larger scale.

- *Harnessing the stochastic behavior and massive scale.* There are several reasons for stochastic behavior in ensembles: Large numbers of independent parallel devices can usually not be synchronized without creating inefficiencies; the resulting behavior can often only be described in a stochastic manner because of the huge number of combinatorial possibilities. Furthermore, massive numbers of components cause high probabilities for failures of individual components, even if the probability of failure for any individual component is relatively low. This effect is compounded by techniques such as nano-technology which inherently have high component defect rates.

With current development methods both stochastic behavior and massive scale pose problems for system development. This does not necessarily have to be the case, as can be seen in biological systems which harness

these effects in order to achieve resilience and adaptation. We need to find ways to do the same in software-intensive systems as well.

- *Resilience, adaptation, and controlled emergence.* Today's systems are overly brittle: a single failure in a typical computer program will cause it to "crash," failure of single components impede the functionality of the overall system, security breaches leave whole systems exposed to attackers. We have to develop methods to build resilient systems, that can cope with component failures or security breaches in individual components without degrading the overall system performance or security; we have to build systems that can adapt to changing situations or requirements without significant human intervention or redeployment; we have to control emergent behaviors and harness the positive aspects of emergence.
- *Operating in open environments, recognizing and exploiting opportunities.* This challenge repeats a point that has already been mentioned several times: dynamicity and openness of environments present significant obstacles to system performance and reliability. We need to find ways to reverse this situation and build systems that can recognize and exploit opportunities arising in these circumstances.
- *Designing and predicting emergent behavior.* Many of the behaviors of ensembles result not from actions of a single entity but rather from the combined activities of several independent nodes, either in a pre-planned or in a spontaneous manner. We need to better understand the behavior of concurrent systems, so that we can design and predict these emergent behaviors.
- *Mobility of code, data, and devices.* In some situations, mobility of code or data can be used to increase the responsiveness, performance or reliability of systems, or to work around system limitations, such as low network bandwidth. Mobility of devices leads to a dynamically changing network topology and the need for services to adapt to the continuous appearance and disappearance of nodes in a system.
- *Integration of heterogeneous and federated data.* With large systems, distributed across several organizational boundaries and integrating parts from different vendors, we can no longer expect the system to share a single, homogenous data model. Instead we have to find ways to integrate heterogeneous, federated, and mutually inconsistent data sources that invariably form part of large systems.

1.1.2 Specification and Design.

The emergent behavior and open-ended nature of ensembles poses new challenges for specification and design.

- *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.* It will often be no longer possible to design the complete system according to a certain specification; instead we will have to integrate services as parts into a larger environment. We will thus no

longer be able to use a top-down approach to specify the system behavior. Instead we will have to develop methods for influencing the global system behavior by affecting only the local environment of individual components, not the global system behavior.

- *Abstractions and models for massively parallel, open-ended systems.* We currently lack the necessary abstractions and foundations to describe, model, and design massively parallel systems. New theoretical foundations and abstractions are urgently needed.

1.1.3 Assurances.

Ensembles being both adaptive and being used in safety-critical systems makes it a necessity that one can give assurances regarding their performance.

- *Replacing the notion of correctness with the one of “fitness for a purpose.”* For systems operating in complex, dynamically changing, open-ended environments it is often no longer useful or even possible to specify a notion of correctness. Instead we need to design systems that are “fit for a particular purpose” and can achieve their desired results in varying and changing situations.
- *Social and emotional perception, quality of experience.* Current human-computer interactions are mostly influenced by the limitations of the computer: it is neither aware of the changing social context in which an interaction may take place, nor of the emotional or physical state of its users. As software-intensive systems pervade ever increasing parts of our professional and private lives, the interaction has to be focused more on the quality of experience for the human users.
- *Assurance guarantees, certification, and formal verification of ensembles.* Future systems, that adapt to their environment, exploit new opportunities, and take into account their user’s emotional state should still provide performance guarantees and limits (e.g., not harming people). We need to develop ways to certify these systems and to formally verify their properties.
- *Privacy, identity management, security, and trust.* As the importance of our online presence and identity have increased the problems of privacy, identity management, security and trust have become more apparent—although often only because of negative consequences that received media attention. For future systems we need to develop methods and tools to ensure these properties, and techniques to reliably communicate the security aspects of actions to users.
- *Quality of service, quality of experience.* To build reliable systems, components need to be able to negotiate quality of service criteria, preferably without manual intervention, and then be able to perform according to these criteria. On a higher level, many systems need to provide a certain “quality of experience” for their users.

1.1.4 Implementation and Verification.

If we are to ever build ensembles with the high standards we have set so far, we will need tools for doing so. These tools will be both evolutionary improvements of current tools, but will also need to incorporate revolutionary ideas to enable the above mentioned goals.

- *Programming languages, development environments, and compiler technology for ensembles.* Current programming languages, development environments and compilers are not adequate for the development of long-running, adaptive systems. For example, they lack features for monitoring, debugging, patching, or upgrading deployed systems from the development environment, or for upgrading systems while they are operating. They also offer no support for dealing with distributed systems which are only locally consistent, a situation which arises frequently in ensembles.
- *Testing and verification of ensembles.* Many of the complexities in the development of ensembles arise from the interactions of concurrently executing, (semi-)autonomous nodes. While test and verification methods have greatly improved in the last years, they are still not sufficient for current or future systems. In particular, verification of large or open systems are still unsolved problems.
- *Multi-paradigm programming.* In current software systems, several formal languages exist next to each other: General-purpose languages such as Java, database languages such as SQL, specification languages, etc. These languages adhere to different paradigms which means that development environments need to support *several* paradigms. Furthermore, diversity in programming languages will always exist (as there is too much knowledge encoded in legacy source code). Thus, one needs to support legacy paradigms *in addition to* new break-through paradigms.
- *Dynamism in programming languages.* Dynamism in programs appears in several facets: Boundaries between programs dissolve, as integration will be much more fine-grained. Formerly clearly separated software life-cycle stages (testing, deployment, updates) are becoming more tightly interlocked. Furthermore, self-* properties, especially where humans are concerned introduce additional levels of reactivity and dynamism. The challenge is to provide adequate tools and methods for dealing with this kind of dynamism: How can their design and implementation be supported? What kind of assurances can we give?

1.2 Research Areas

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 behavior. 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.

2 Conclusion

Software-intensive systems play an important role in almost every part of human society: Transportation systems, communication systems, energy networks, medical technology, etc. That is, they exist *today*, but they are very difficult to construct. This document gave an overview of how software-intensive systems are currently built. It then describes areas where software-intensive systems will face even greater challenges in the future: The number of nodes per system will increase, systems will be deployed in environments that are more and more open and non-deterministic, and adaptation in these environments will be essential. Software-intensive systems to which this characterization applies have been called *ensembles* by the InterLink WG1. This document described areas of research that might help with meeting the challenges of engineering ensembles. That ensembles will be crucial for social and economic competitiveness in the future is virtually certain, it is now upon the research community to help build, harness, and understand them.

Acknowledgements

This work was partially funded by the IST Coordinated Action InterLink “International Coordination Activities in Future and Emerging ICTs” (034051) and the IST Integrated Project SENSORIA, IST-2 005-016004. Without the input of the members of the InterLink WG1 “Software-Intensive Systems and New Computing Paradigms” this document would not have been possible and we gratefully acknowledge their contribution.