From Design to Intention: Signs of a Revolution

Franco Zambonelli
Dipartimento di Scienze e Metodi dell'Ingegneria
Università di Modena e Reggio Emilia
Via Allegri 13 – Reggio Emilia– ITALY
franco.zambonelli@unimo.it

H. Van Dyke Parunak
Altarum
3520 Green Ct, Suite 300,
Ann Arbor, MI 48105 USA
van.parunak@altarum.org

Categories and Subject Descriptors

D.2.2 [**Software Engineering**]: Design Tools and Techniques; I.2.11 [**Artificial Intelligence**]: Multiagent Systems; C.2.4 [**Computer-Communication Systems**]: Distributed Systems.

General Terms

Algorithms, Design, Theory

1 WHAT'S NEW?

The complexity raised in software systems by several emerging computing scenarios has moved beyond the capabilities of traditional approaches to computer science and software engineering. The scenario that will cause the next software crisis is rapidly forming under the eyes of everybody: computing systems are going to be everywhere, embedded in all our everyday environments, and they will be always connected and active [1].

The above scenarios do not simply quantitatively affect – in terms of number of components and effort required – the design and development of software systems. Instead, we argue that there will be a qualitative change in the characteristics of software systems, as well in the methodologies adopted to model and develop them. In particular, we argue that four main characteristics – in addition to the quantitative increase in interconnected computing systems – will distinguish future software systems from traditional ones:

i. *situatedness*: software components will execute in the context of an environment (physical or computational one) and will perceive such an environment in terms of environment-dependencies in computation [1]. Also, components can influence such environment and be influenced by it.
ii. *openness*: software systems will be subject to decentralized management and will dynamically change their structure: new components can be dynamically created or destroyed and, via mobility, will be able, to roam in and out the permeable boundaries of different software systems. Thus, the problem of openness is currently much broader than being simply a problem of interoperability;
iii. *locality in control*: the components of software systems will represent autonomous loci of control. In fact, most components of software systems will be active, and will have local control over their activities, although will be in need of coordinating these activities with other active components.
iv. *locality in interactions*: despite living in a fully connected world, software components interact with each other accordingly to local (geographical or logical) patterns. In other words, systems will have to be modeled around clusters of locally interacting components, and inter-cluster interactions will have to be modeled accordingly.

Any researcher who has worked in the area of agent-based computing will immediately recognize the above characteristics as the main distinguishing ones of agents and of multi-agent systems. These characteristics are going to be pervasive. In fact, to different extents and with different terminology, several research communities are recognizing their importance and are adapting their models and technologies to take them into account.

As an example, network of embedded and wireless sensors *(i)* are situated in a physical environment that they are devoted to monitor; *(ii)* live in an environment where new sensors can be added and low-power ones can die; *(iii)* are by definition autonomous; *(iv)* can sense and interact within a limited local range. Other pervasive computing scenarios, such as wearables, intelligent homes, and manufacturing control systems, share similar characteristics. As another example, Internet applications *(i)* execute in the computational Internet environment; *(ii)* can be joined by new applications and services at any time; *(iii)* have decentralized control; *(iv)* interact within logical (if not physical) localities. Similar consideration can be made with regard to business support systems, whether B2C or B2B, even if not Web-based. In fact, the dynamics of today's economy forces situatedness, decentralization, and locality in business process and, consequently, in the software system devoted to support them.

The integration of the above concepts and abstractions in software modeling and design does not simply represent a proper evolution of current models and methodologies. Instead, we claim that we are going to pass though a real revolution, changing the very way we conceive and model "software components" and "software systems", as well as the way we will build such systems.

2 CHANGING OUR ATTITUDES

Until now, software systems have been modeled by adopting a mechanical attitude, and engineered by adopting a design attitude. Computer scientists are both burdened and fascinated by the urge to define suitable formal theories of computation, while software engineers tend to design software architectures as reliable multi-component machines, capable of providing the required functionality in an efficient and predictable way. The new scenario may require models and methodology to be adopted that differ from the traditional ones and, even most important, will require a fundamental change of attitude.

2.1 Changes in Computer Science

Modeling and handling systems with very many components can be feasible if such components are not autonomous. However, when the activities of these components are autonomous, it is conceptually and computationally infeasible to describe the behavior of a system in terms of the behaviors of its components. All that can be done in these cases is to describe and model the system as a whole, in terms of macro-level observable characteristics, just as a chemist describes the characteristics of a gas in terms of properties like pressure and temperature. The above problem is exacerbated by the fact that components interact with each other. Accordingly, the overall global behavior of a system is more than the sum of the behavior of its components, but also includes results of their mutual interactions. These local interactions can have global effects that are very difficult to predict. As an additional problem, we must consider that software systems will execute in an open and dynamic scenario, where new components can be added and removed at any time, and where some of these components can autonomously move from one part of the system to another. Thus, it is difficult to predict and control precisely not only the global dynamic behavior of the system, but also how such behavior can be influenced by such openness. In fact, the effects of the interactions between autonomous active components strongly depend on the structure of the interaction graph [4]. A similar problem arises because of situatedness. Modern statistical mechanics and social science tell us that environmental forces can produce strange and large scale dynamic behaviors on situated physical, and social systems.

The above problems will force scientists to change their attitudes dramatically in modeling complex software systems. Only small portions of large software systems can be treated effectively with formalisms and logic-grounded approaches. The next challenge is to adopt a brand-new scientific background for the study of software systems, enabling us to study, predict, and control the properties of a system and its dynamic behavior despite the inability to control its individual components. Some signals of this trend can already be found in different areas of the research community. Recent study and monitoring activities on the Internet and on Web-access patterns have made it clear that unpredictable and large-scale behaviors are already here. Some approaches to model and describe software systems in terms of thermodynamic systems have already been proposed [2]. Similar approaches have been adopted in the area of massively parallel computing, where measuring specific global systems properties dynamically requires the introduction of synthetic indicators. In the area of multi-agent systems, it is being recognized that the behavior of a large scale software system is more similar to a human organization or to a society than to a logical or mechanical system. In the future, we expect theories from complex systems, statistical mechanics, as well as from biology, social science, and organizational science, to become the sine-qua-non background of the computer scientist.

2.2 How Will Software Engineering Change?

The change in the modeling and understanding of complex software systems will also definitely impact the way such systems are designed, maintained, and tested. Today, software systems are designed to exhibit a specific, predictable, and deterministic behavior at any level of the software system, from the level of single units up to the level of the whole systems. However, in the presence of interacting autonomous components in an open and dynamic environment, obtaining a predictable behavior of a multi-component system “by design” is not feasible. The next challenge is to build them so as to guarantee that systems as a whole will behave as desired despite the lack of exact knowledge about their micro-behavior. For instance, by adopting a "physical" attitude toward software design, a possible approach could be to build a system that, despite uncertainty on the initial conditions, is able to reach a given stable basin of attraction. By adopting a "teleological" attitude, the idea could be to build an ecosystem able to behave in an intentional way, and robust enough to direct its global activities toward the achievement of the required goal. Of course, the design of a system must also take into account the fact that the system will be immersed in an open and dynamic environment, and that the behavior of the system cannot be designed in isolation. Rather, the environment and its dynamics, as well as those software components that can enter and leave the systems, must become primary components of the design abstractions. First, the design could assume a defensive approach, by treating them as sources of uncertainty that can somehow be damaging to the global behavior of a system. Second, the design could assume an offensive approach, by considering openness and environmental dynamics as additional dimensions to be exploited with the possibility of improving the behavior of the system [3].

Again, it is possible to identify a few works that are already adopting such a novel perspective. Policies for the management of distributed resources are already being designed in terms of autonomous components able to guarantee the achievement of a global goal via local actions and local interactions, despite the dynamics of the environment. Ant colony optimization mimics nature to solve very complex problems. Cellular automata can evolve via genetic algorithms to produce useful behaviors.

In addition to the change in the way software is designed, the new scenario will also dramatically impact the way software is tested, maintained, and evaluated. When conceiving software systems in mechanical terms, testing mainly amounts to analyzing system state transitions. Such work is very hard for large systems, and it may become impossible when autonomy and dynamics of the environment produce a practically uncountable number of states. Thus, a large software system will no longer be tested with the goal of finding errors in it, but rather with regard to its ability to behave as needed as a whole, independently of the exact behavior of its components and of their initial conditions. Moreover, a software system that is likely to be immersed in an existing dynamic environment, where other systems are already executing and cannot be stopped, cannot be simply tested and evaluated in terms of its capability of achieving the required goal. Instead, the test must also evaluate the effect of the environment on the software system, as well as the effects of the software system on the environment.

3 CONCLUSIONS

Some researchers in the area of autonomous agents and multi-agent systems already adopt abstractions capturing the likely characteristics of future software systems. Still, their effective modeling and engineering requires a revolutionary re-structuring of our background, paving the way for a new set of conceptual tools and frameworks for the engineering of future software systems [5].

REFERENCES

[1] G. Cabri, L. Leonardi, F. Zambonelli, “Engineering Mobile Agent Applications via Context-Dependent Coordination”, *IEEE Trans. on Software Engineering*, 2002, to appear.

[2] V. Parunak, S. Bruekner, J. Sauter, "ERIM’s Approach to Fine-Grained Agents", *NASA/JPL Workshop on Radical Agent Concepts*, Greenbelt (MD), Jan. 2002, www.erim.org/~vparunak/WRAC2001.pdf

[3] D.Tennenhouse, "Proactive Computing", *CACM*, May 2000.

[4] D. Watts. *Small Worlds*, Princeton Univ. Press, 1999.

[5] F. Zambonelli, V. Parunak, “From Design to Intentions”, extended version, available at: www.dismi.unimo.it/Zambonelli