

Second International Workshop on Theory and Practice of Open Computational Systems (TAPOCS 2004)

Andrea Omicini Alessandro Ricci
DEIS, Università di Bologna
via Venezia 52, 47023 Cesena, Italy

Dina Goldin
Computer Science & Engineering
University of Connecticut, CT, USA

E-mail: {aomicini, aricci}@deis.unibo.it, dqg@engr.uconn.edu

Abstract

The 2nd International Workshop on Theory and Practice of Open Computational Systems (TAPOCS 2004) was one of the events of the 13th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE-2004). TAPOCS aims at exploring the various forms of openness for computational systems. We are interested in investigations of models and formal frameworks for open computation, as well as in issues related to the engineering and applications of open computational systems.

This report provides an overview of the scientific activity that took place at TAPOCS 2004. After an introduction to the scope and goals of the workshop, presentations are shortly summarized, and organized in a coherent conceptual framework along with the results from discussions. Finally, open issues are reported as well as some considerations about the future of TAPOCS.

1. Introduction

1.1. To Be Open

Openness is an elusive concept. Definitions of *open computational systems* (OCS) typically fail to capture all the possible meanings of this term, as it is used in computer science. For instance, a commonly shared definition describes OCS as “those systems whose composition is not completely known at design time” – where *composition* refers to the number and identity of system components, such as objects, processes, or agents [5]. This definition correctly notes that OCS may be dynamically composed at runtime. However, a definition of this sort does not account

for other interesting and relevant forms of system openness.

More generally, the notion of openness conveys the idea that systems are “open to change”. According to this idea, the whole process of complex system engineering is potentially a source of openness. Systems can be designed to be open, not merely in terms of the number and kind of components: the way in which components of any sort are configured could be only partially pre-determined at design time, and then change at runtime, when the systems can self re-configure, and their components re-organized. For instance, if the system’s mission is given, but its working conditions cannot be known a priori, the system architecture (structure) may change, or *evolve*, in order to adapt to the runtime conditions, even with no change in its composition. Furthermore, the mission itself might be not fixed at design time, and the system may be open to adopt new goals at run time.

Also, the development of system can be open. Systems that have to be permanently alive and cannot be switched off, might be anyway open to modifications by engineers. Critical portions of a system could be accessible and modifiable while the system keeps running, such as by editing values or adding entries in system tables. This results in a change to system behaviour, allowing for system re-design and incremental development without system halts.

Finally, other forms of system openness directly concern the runtime stage. Given the complexity of today’s software systems, empirical observations of their behavior may be critical for understanding the system, since its internal structure could be unknown or simply too intricate – in the same way as most physical or biological systems. So, complex systems should be open to observation, and be instrumented with the tools that provide the most suitable abstraction level over them, that is, the level that best allows for system modeling and understanding.

TAPOCS aims at exploring all these forms of openness,

both from a theoretical and practical points of view. From the theoretical perspective, we are interested in investigations of models and formal frameworks for open computation; from the practical perspective, we encourage investigations of issues related to the engineering and applications of OCS. These two points of view are closely connected; on the one side, by applying the results coming from the investigations of models and formal frameworks to engineer middleware and applications, on the other side, tuning the models with feedback coming from the engineering.

1.2. Issues for Open Systems

The notion of openness is not limited to computational systems; it also applies to complex systems other than computer-based ones. In physics, systems are *open* if they permit a flow of energy between the system and its environment. Similarly, OCS involve a flow of information, in the form of *input* and *output* between the system and the environment, as well as between the components of the system; in other words, OCS are inherently *interactive* [12]. It is this flow of information that allows OCS to adapt (evolve) during the computation.

The issues raised by our study of open systems are therefore relevant in many related fields. For instance, systemic dimensions such as collaboration, cooperation, coordination, competition, and organization are relevant in complex economical, social, or biological systems, too. The problem of design change and evolution of the system structures and processes is also not limited to computational systems.

On the other hand, some issues are more appropriate in the context of computational systems, for instance system *environment*. Even though the role of the environment in understanding and designing the behaviour of complex systems is quite general (see for instance modern evolutionary biology), it is also true that the notion of environment found in artificial systems such as OCS is quite a peculiar one. While physical environments are mostly to be taken as given, artificial (computational) environments are typically subject to engineering, and can (at least partially) be shaped as needed by the systems' requirements. This is typically the role of *middleware*, and of software infrastructures, which provide OCS with the required resources and services.

Important OCS issues include system online construction and observation, online change and evolution of organization and coordination strategies, and system self-organization and emergent behaviour. Software engineering methodologies play a fundamental role, both in supporting OCS engineering on top of infrastructures, and in promoting system observation and analysis. Mainstream methodologies, such as *object-oriented* or *service-oriented* ones, fall short in dealing with these issues at the proper level of

abstraction. *Agent-oriented* methodologies seem to be more appropriate, in particular those where the issue of openness is a main concern.

2. TAPOCS Overview

The Second International Workshop on Theory and Practice of Open Computational Systems (TAPOCS 2004) took place in Modena, Italy, from June 14 to 16, 2004, as one of the events of the 13th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE-2004). Organized and chaired by Andrea Omicini and Alessandro Ricci (Università di Bologna a Cesena, Italy), Rune Gustavsson and Martin Friedriksson (Blekinge Institute of Technology, Ronneby, Sweden), TAPOCS 2004 was the second event of the series started in Linz, Austria, in 2003 [5].

Among the several articles submitted, seven were accepted as full papers, and four as short papers; they are published in this volume. Nine of them were presented by the authors at the workshop, and are accounted for in this report. However, all contributions had full-time presentations during the workshop, and provided useful contributions to the general discussion. Presentations were organized into four sessions, as follows:

1. Models of Interaction in OCS (section 2.1)
2. Organization & Coordination in OCS (section 2.2)
3. Models & Infrastructure for OCS (section 2.3)
4. Applications of OCS (section 2.4)

Each speaker was asked to conclude his/her talk by pointing out some open issues that they considered important, both for their own research and for the general TAPOCS community. The final program and the presentations can be found at the TAPOCS web site [4].

Each session was concluded with a more general discussion of the related issue. As a consequence, many general paths were devised out, and some repeated patterns emerged from seemingly divergent research lines. Hereinafter, a short account of what happened at TAPOCS 2004, organized around the four sessions.

2.1. Models of Interaction in Open Systems

Interaction is today acknowledged to be a fundamental dimension of system modeling and engineering; this is particularly true for OCS, where the interaction of a multiplicity of possibly heterogeneous components determines the (often unpredictable) configuration and behaviour of the system as a whole.

TAPOCS speakers in this session were asked to consider some basic related questions: How to model interaction in OCS? Is there any kind of formalism that is more suitable for modeling OCS? What are the differences with respect to closed/traditional systems? And, when complexity strikes, are there any suitable models to observe interaction at the right level of abstraction, promoting empiric investigation of OCS?

First, Paolo Torroni [1] discussed a logic based approach to the design of interaction in open multi-agent systems (MAS), adopting a logic-based formalism to define the semantics of agent communication languages and interaction protocols. Torroni advocated for a general framework, promoting an objective perspective over agent interaction, and paving the way for formal verification methods. He also insisted on the notion of dissemination of technologies and methodologies as a key open issue.

Then, Dina Goldin [6] showed that *indirect (mediated) interaction* is fundamentally different from *direct (message-based) interaction*. Indirect interaction is interaction via persistent, observable changes to the environment; the participants in such interaction are the agents that make these changes, as well as the agents that observe them. While concurrency theory models interaction as direct, it was pointed out that indirect interaction exhibits several unique and useful properties that make it deserving of domain-independent modeling. As a fundamental example of a need for the formalization of indirect interaction, it was argued that decentralized coordination in self-organizing systems cannot be accomplished with direct interaction alone. However, questions concerning the conceptual approaches to the formalization of indirect interaction are still basically unanswered, and certainly call for further investigation.

2.2. Organization & Coordination in Open Systems

The dynamism required to an OCS in order to achieve its mission mandates for flexible and adaptable architectures. Systemic dimensions such as collaboration, cooperation, coordination, competition and organization play a fundamental role in defining the structure of open systems (not only computational ones) and its dynamics over time, as well as in ensuring the system ability to behave according to the designer's intentions despite the possibly changing conditions of the environment.

The main issues faced by the speakers of the second session were then: How to ensure that system structures can change and evolve during system lifetime, and possibly improve its efficiency / efficacy in achieving the system mission? Which models, theories, strategies best account for change, evolution and adaptation? What are the best ways to allow observability of system structures, and their change as well?

Alessandro Farinelli [10] presented a new approach to team coordination in complex mission execution. Role allocation was modeled as a distributed constraint optimization problem, and an algorithm was proposed to manage constrained roles. Discussion here focused on the apparent divergence between the claims of [6] and [10]; while in the former interactive computation was indicated as an orthogonal dimension with respect to algorithmic computation, in the latter algorithms were employed to constrain and control interaction.

The next talk introduced the issue of self-organization in complex computational systems. Marco Mamei [13] envisioned a future where clouds of microcomputers are sprayed in an environment where they spontaneously network with each other. The resulting emergent behaviors provide an endless range of futuristic applications. A design and development approach for these types of self-organizing systems was proposed and discussed at the different scales, where problems of different sorts arise.

Emergent behaviour was also a main topic in the last talk of the session. Jean Pierre Mano [7] showed how cooperation could be used as a local criterion of self-organization, to allow MAS to adapt to a changing environment. Again, an apparent dichotomy emerged when comparing this talk with the previous one; while Mamei [13] argued that self-organization calls for some form of control by engineers over system structure and evolution, Mano [7] promoted self-organization as a non-controlled property of OCS.

2.3. Models & Infrastructure for Open Systems

OCS are typically on-line systems, that is, systems whose life length is not limited and pre-determined a priori, and which are sometimes expected to change, adapt and evolve without stopping. An important issue here is how to support the engineering of such systems. This requires adequate methodologies for system design and development, as well as a suitable runtime support, usually in the form of an infrastructure. Along this line, the discussion tried to address some fundamental questions: Which are the models for infrastructures that better support OCS as on-line systems? Which sort of components do they assume? Which services should they provide? And, what sort of properties should they ensure to systems, and how?

Giovanni Rimassa [9] clarified the role of infrastructure for OCS, and focused on the relationship between technology transfer and middleware. Rimassa introduced the concept of *seamless* agent middleware, as a technology that allows to set the social/physical balance wherever it better suits MAS development. *Activity theory* was shown to provide a useful conceptual model to frame agent infrastructures, as well as an interesting inter-disciplinary link between organizational sciences and computational systems.

After that, Sara Manzoni [8] introduced the theme of *ubiquitous computing*, focusing on the relationship between computational systems and their environment. In particular, her talk presented a model for space as a first class concept, allowing both logical and physical space to be interpreted and framed in a coherent and uniform way.

Finally, Maria Rita Di Berardini [3] moved the attention of the TAPOCS audience toward the issue of code mobility in pervasive (ubiquitous) computing. In her talk, the minimal functionality for a platform supporting code mobility in pervasive computing scenarios are devised out, and a calculus is defined that could provide the basis for a formal framework for such a platform. As it was pointed out by the audience, this approach raised one of the most frequent issues in TAPOCS, that is the issue of formalization of OCS, and associated it to platform / infrastructure / middleware, as done by other TAPOCS papers such as [1].

2.4. Applications as Open Systems

OCS appear to be the right abstraction level for software engineering applications in complex domains. These domains typically involve a (structured) multitude of distributed entities interacting within an open and unpredictable environment toward the achievement of some kind of overall mission. Examples range from distributed workflow management to on-demand internet applications, and pervasive computing.

Consider for instance the case of Virtual Organization/Enterprise (VO). These are typical OCS, composed dynamically of heterogeneous and possibly anonymous entities/parties which participate in the same workflow processes, contributing their individual efforts and their skills in order to achieve the fulfillment of the VO business goals. The openness and unpredictability of the environment (e.g. the Internet) typically require dynamic organizational adaptation, or change of VO strategies and workflows.

In this context, primary issues concern experiences in engineering applications as OCS: What classes of applications can be suitably framed as OCS? What methodologies and engineering tools can be adopted? What has been the experience with middleware support for application engineering?

In this session Merelli [2] presented FarMAS, an agent-based system for the management of supply chains. FarMAS focuses in particular on data integration/elaboration and support of production quality. He discussed the adoption of FarMAS, focusing on integrating it with existing industrial systems.

This is a very important issue, concerning how to introduce research approaches in the context of existing legacy systems, extending them to provide better support for domain complexity. The talk underscored the need for suitable

methodologies to better support system organization and coordination and runtime management, in particular providing formal approaches for specifying/verifying consistency (integrity constraints) among roles, activities and workflow rules.

3. Conclusions & Final Remarks

Multi-agent systems (MAS) appear to be the dominant modeling and engineering paradigm for dealing with OCS, providing support for openness at a more appropriate level of abstraction than other mainstream paradigms, such as object-oriented or service-oriented ones (e.g. Web Services). An important issue then concerns the possibility of integrating MAS approaches with existing service-oriented infrastructures.

The workshop also focused on the key role of middleware. It provides explicit support for various systemic aspects:

- supporting different forms of interaction (in particular indirect interaction, differently from the mainstream MAS approaches such as FIPA, that are based on directly communication;
- providing mechanisms to enable forms of self-organization;
- providing high level social structures and rules for open MAS composed by logic-based agents
- providing to field-based mechanisms to enable (stigmery) coordination in mobile (typically reactive) agents.

The open questions identified by the authors and raised during the discussions pointed to several potential challenges for TAPOCS 2005. Most notably, much work on the foundation is still needed, devising better models for understanding and managing openness, correlating our findings with those in science and engineering fields, where openness has been already subject of investigation for a long time; a notable reference is [11]. Examples range from biology, ethology to cybernetics and social science. In particular there is a need to better clarify the dualism between openness and control of systems, between the engineering of self-organizing and coordinating systems which should at the same time be dynamically inspectable and controllable by system designers and engineers.

It became clear from the discussion that formal approaches are needed for the verification of properties concerning OCS dynamics, in particular concerning interaction. Here, infrastructures could play an important role: might the formalization of infrastructure components / services be useful for the specification and verification of OCS

using those services (in spite of the inherent global unformalisability due to openness)?

Finally, for the future of the workshop it will be extremely important to create a vibrant dialogue with the application community regarding applications engineered as open systems. Strong feedbacks from real-world case studies is needed in order to better address research on the theoretical perspective.

References

- [1] M. Alberti, M. Gavaneli, E. Lamma, F. Chesani, P. Mello, and T. Paolo. A logic based approach to interaction design in open multi-agent systems. In *this volume*, 2004.
- [2] D. Bonura, F. Corradini, E. Merelli, and G. Romiti. FARMAS: a MAS for extended quality workflow. In *this volume*, 2004.
- [3] F. Corradini, R. Culmone, and M. R. Di Berardini. Code mobility for pervasive computing. In *this volume*, 2004.
- [4] M. Fredriksson, R. Gustavsson, A. Omicini, and A. Ricci. *TAPOCS Home Page*.
<http://www.soclab.bth.se/workshops/tapocs2004/>.
- [5] M. Fredriksson, R. Gustavsson, A. Omicini, and A. Ricci. First International Workshop on Theory and Practice of Open Computational Systems. In *IEEE 12th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE 2003)*, pages 355–358, Linz, Austria, 9–11 June 2003. IEEE CS. Proceedings.
- [6] D. Goldin and D. Keil. Toward domain-independent formalization of indirect interaction. In *this volume*, 2004.
- [7] J. P. Mano and P. Glize. Property analysis of open networks of cooperative neuro-agents using AMAS theory. In *this volume*, 2004.
- [8] S. Manzoni, F. Nunnari, and G. Vizzari. Toward a model for ubiquitous and mobile computing. In *this volume*, 2004.
- [9] A. Omicini and G. Rimassa. Toward seamless agent middleware. In *this volume*, 2004.
- [10] P. Scerri, A. Farinelli, S. Okamoto, and M. Tambe. Token approach for role allocation in extreme teams: Analysis and experimental evaluation. In *this volume*, 2004.
- [11] H. Simon. *The Sciences of the Artificial, 2nd edition*. MIT Press, 1982.
- [12] P. Wegner. Why interaction is more powerful than algorithms. In *Comm. ACM*, May 1997.
- [13] F. Zambonelli, M.-P. Gleizes, M. Mamei, and R. Tolksdorf. Spray computers: Frontiers of self-organizations for pervasive computing. In *this volume*, 2004.