Specification, Analysis, and Prototyping of Mobile Systems

Cecilia Mascolo *
Department of Computer Science,
University of Bologna.
Mura Anteo Zamboni, 7 40127 Bologna, Italy.
phone:+39 051 354871
E-mail: {mascolo}@cs.unibo.it

ABSTRACT
Mobile code offers new strategies for the development of systems. I adopt a formal approach to study advantages, limitations, classification, and future trends of mobile code technologies.

Keywords
mobile code, formalization, verification, and prototyping.

1 INTRODUCTION AND MOTIVATION

Mobile code is a recent and complex topic on which application and theory oriented research is now focusing. Java based technologies and mobile agents are having a big influence in the development of new systems and their diffusion is faster than the growth of formal techniques able to deal with mobile code paradigms and characteristics. However, some informal classification [5], and some formal languages [1, 4, 8] have been used in order to study capabilities and differences among mobile code paradigms and systems.

The aim of my research is to formally reason about the already developed systems and about the future trends of mobile code using a large-grained approach in order to study the architectural aspects of systems containing mobile components, and a fine-grained approach to investigate the basic issues related with code mobility and the primitives operations for mobility. In the next two sections I will introduce my contribution and my research agenda.

2 ACHIEVED RESULTS

We have dealt with the formalization and the enhancement of a model (PoliS) based on hierarchically structured tuple spaces for the specification of different paradigms of mobility [2]. The mobility of data, code or agents carrying their state can be formalized using PoliS. In order to be able to analyze the different mobility solutions applied to the specified systems, we have devised a temporal logic (based on CTL, a branching temporal logic often used for model checking) for PoliS and we have built a model checker to obtain automatic proofs of interesting safety and liveness properties: that for example a mobile agent does not move forever under certain conditions, or that it will eventually accomplish its goal.

The coordination model (i.e. the tuple spaces model) of PoliS supports flexible moving of components and extensibility of the specification. Furthermore, the decoupling in space and time is a key issue of the language: mobile components can communicate asynchronously (but synchronization can be introduced) using tuples. Like in distributed systems, different tuple spaces can evolve concurrently in time. The separation of coordination from computation issues makes the specifications more readable and allows mobility mechanisms to be specified without worrying about computational details.

Using the PoliS model and the model checking tool we are able to perform compositional analysis. In studying complex systems, it may be convenient to first analyze components behavior, and then put together all the pieces and perform some kind of global analysis [6]. The study of components as isolated entities is useful when dealing with wide architectures where components are not elementary objects but they are composed of many parts. Research on reconfigurable systems often has to follow this kind of approach. This is, again, useful when the systems contain mobile components: for instance, to find out in which context a component is able to exploit a particular behavior; or which components could end up in the same environment and which kind of interaction they could obtain.

The coordination model allows the study of the behavior of single components, of their assumptions about the environment, and of the possible component interconnections. The structure of the model checker allows

*Part of this research is conducted while staying as a visiting researcher in the Dept. of Computer Science at Washington University in St. Louis, Missouri (USA).

the verification of properties on isolated parts of the system and the consequent verification of hypothetical configurations composed of multiple components [3].

As in PoliS "spaces" (i.e. agents) mobility is not a basic operation (also if it can be formalized), we have enhanced the model providing space mobility as a first class construct in the model [7].

In the next section, I illustrate my current work on other aspects of mobility. In order to treat most of the fundamental aspects of mobility in my thesis, I am now carefully inspecting the basic mobility mechanisms exploited by the new technologies. A more implementation-oriented language is used for this purpose.

3 FUTURE RESEARCH

The part of my research illustrated in Section 2 can be considered in general as "component-based study of mobility": the idea there is to reason on the interactions and the coordination among components mobility, and study the possible reconfiguration of the structure after the movements.

That part of research is pursuing a large-grained approach. To achieve a complete understanding of the future trends of mobile code technologies we are now researching on the basic issues of mobility: what is the basic unit of mobility? What are the primitive mobility operations? How should resources be handled in a mobile code setting? We are therefore devising a new model, more programming-oriented than PoliS, as we need to reason at a more fine-grained level about mobility of code.

The formal language we are using for this research is Mobile UNITY [8]. Mobile UNITY has been already used for the specification of physical and logical mobility based systems, and offers interesting features as a notion of location and a components variables sharing mechanism. It also has a well-founded proof-system and our aim is to find interesting properties to be proven on the different primitives and composition of primitives we can specify.

Most of the devised technologies for mobility offer fixed patterns of movement (agent mobility, or code mobility, or only data mobility). Usually the different mobility paradigms are encoded on the basic style embedded in the system. Also the way to access the resources is usually pre-defined and the user needs to adhere to these fixed paradigms. We are building a prototype based on the basic primitives we have isolated and on the semantics model we are building in order to offer a more flexible way of designing applications, and to let the designer choose the most appropriate level of mobility she thinks it is the most appropriate. Furthermore, in order to carefully evaluate our research, we will encode some of the actual technologies and models using our language and tool. Novel research ideas and ongoing technical developments will offer new mobility issues and paradigms. We are therefore devising our language and prototype to provide some scalability properties; as the model is so basic we think that new features and mobility styles could be added in order to be able to formalize and reason on forthcoming technologies.

ACKNOWLEDGMENTS

I would like to thank P. Ciancarini, and G.-C. Roman for their helpful suggestions and guidelines on my work. I am also grateful to G.P. Picco for the useful discussions and comments.

REFERENCES

[1] L. Cardelli and A. Gordon. Mobile Ambients. In M. Nivat, editor, *Proc. of Foundations of Software Science and Computation Structures (FoSSaCS), European Joint Conferences on Theory and Practice of Software (ETAPS'98)*, volume 1378 of *Lecture Notes in Computer Science*, pages 140–155, Lisbon, Portugal, 1998. Springer-Verlag, Berlin.

[2] P. Ciancarini, F. Franzé, and C. Mascolo. A Coordination Model to Specify Systems including Mobile Agents. In *Proc. 9th IEEE Int. Workshop on Software Specification and Design (IWSSD)*, pages 96–105, Japan, 1998.

[3] P. Ciancarini and C. Mascolo. Specification and analysis of component based software architectures. Proc. First IFIP Int. Working Conf. on Software Architecture. Position paper., Feb. 1999.

[4] R. DeNicola, G. Ferrari, and R. Pugliese. KLAIM: A Kernel Language for Agents Interaction and Mobility. *IEEE Transactions on Software Engineering*, 24(5):315–330, 1998.

[5] A. Fuggetta, G. Picco, and G. Vigna. Understanding Code Mobility. *IEEE Transactions on Software Engineering*, 24(5):342–361, 1998.

[6] P. Inverardi, A. Wolf, and D. Yankelevich. Checking assumptions in components dynamics at the architectural level. In D. Garlan and D. LeMetayer, editors, *Proc. 2nd Int. Conf. on Coordination Models and Languages*, volume 1282 of *Lecture Notes in Computer Science*, pages 46–63, Berlin, Germany, September 1997. Springer-Verlag, Berlin.

[7] C. Mascolo. MobiS: a Specification Language for Mobile Systems. In P. Ciancarini and A. Wolf, editors, *Third Int. Conf. on Coordination Languages and Models (COORDINATION)*, volume to appear of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, April 1999.

[8] G. Picco, G.-C. Roman, and P. McCann. Expressing Code Mobility in Mobile Unity. In M. Jazayeri and H. Schauer, editors, *Proc. 6th European Software Eng. Conf. (ESEC 97)*, volume 1301 of *Lecture Notes in Computer Science*, pages 500–518. Springer-Verlag, Berlin, 1997.