Understanding Code Mobility

Gian Pietro Picco
Dipartimento di Elettronica e Informazione—Politecnico di Milano
P.za Leonardo da Vinci 32, 20133 Milano, Italy
Tel.: +39-02-2399-3519, E-mail: picco@elet.polimi.it

ABSTRACT
The tutorial provides a conceptual framework for code mobility by illustrating a taxonomy of related technologies, architectural paradigms, and applications. As a final case study, the concepts developed in the taxonomy are then applied to a quantitative assessment of the benefits of mobile code technologies and architectures in the network management application domain.

1 INTRODUCTION

Code mobility can be defined as the capability of a distributed application to relocate its components at runtime. Component migration may involve the code of a software component (e.g., the code of a class) or even some combination of code and state, often referred to as a *mobile agent*. This possibility has been made popular by Java and by a myriad of other languages and systems. However, such systems differ in the way they provide support to code mobility, as they rely on different conceptual and terminological backgrounds, design choices, and abstractions. Even worse, the frequent use of the term "agent" to denote the application code being relocated adds further confusion to the area, suggesting links with disciplines like artificial intelligence which are actually not concerned directly with this research area.

Mobile code has gained great attention because, by freeing the behavior of distributed components from their location, it has the potential to change radically the way distributed applications are developed and deployed. Nevertheless, applications exploiting code mobility are still largely missing, or they are limited to minimal forms of mobility as in the popular example of the Java applets being downloaded into Web browsers. It is evident that this is an immature research area, and there is a strong need for a systematic approach to understanding the key characteristics of code mobility as well as for a careful analysis of the expected benefits.

2 TUTORIAL CONTENT

The aim of the tutorial is to provide a conceptual and terminological framework that can guide the understanding and assessment of code mobility. This framework comes under the shape of a taxonomy for mobile code technologies, architectural paradigms, and applications that, while covering the state of the art in the field, proposes a common lexicon and a precise characterization of the founding concepts of the research area.

The portion of the taxonomy related to technology is based on a reference model describing the basic abstractions into play, namely, executing units, resources, and sites. Following this reference model, the mechanisms provided by the existing technologies are classified, distinguishing among mechanisms dealing with mobility, security, translation and execution, and communication. Building on the classification of technology, a taxonomy of architectural paradigms is presented. Architectural paradigms abstract away from the underlying technology and provide the application designer with a more abstract view where to evaluate the opportunities for mobile code. These two dimensions prepare the context to discuss how code mobility can be exploited to build distributed applications in novel ways. A set of general advantages are discussed, together with a list of application domains for code mobility.

Finally, a case study in network management is the opportunity to illustrate how the taxonomy is useful in practice for understanding and evaluating mobile code. An informal evaluation is provided, followed by a quantitative model that analyzes the critical parameters of a network management task to assess the effectiveness of mobile code for reducing the overhead of management traffic in the network around the management station.

REFERENCES

[1] M. Baldi and G. P. Picco. Evaluating the Tradeoffs of Mobile Code Design Paradigms in Network Management Applications. In *Proc. of the 20th Int. Conf. on Software Engineering*, 1998.

[2] A. Fuggetta, G. P. Picco, and G. Vigna. Understanding Code Mobility. *IEEE Trans. on Software Engineering*, 24(5), 1998.