The Implementation of a Language Processor with Meta-level Architecture for Mobile Computing Environment

Noriki Amano
School of Information Science,
Japan Advanced Institute of Science and Technology

February 15, 1996

Keywords

Mobile Computing, Meta-level Architecture,
Generic Function, Meta Object Protocol

Abstract

We propose a language with meta-level architecture supporting development of applications on computing environments changing dynamically their system configurations such as mobile computing environments.

1 Background

The recent advances of small, low-cost, yet high-performance electronic devices, wide area networking infrastructures like the Internet, and high-speed wireless communication technologies have engendered a new paradigm of computing, called *mobile computing*. The users of portable computers have access to shared information infrastructure regardless of their physical location. This provides flexible and a novel way of communication between people. Mobile computing is expected to revolutionize the way of using computers. We call the computing environment in which mobile computers are connected in a distributed system through the network a *Mobile Computing Environment* (MCE). There are some immanent problems in MCE. Making true real MCE more difficult to be implemented. For example: limited battery capacity, network address (IP address) depending on physical locations, and wireless connections being of lower quality than wired connections making lower bandwidth, higher error rates, and more frequent spurious disconnections. Also in MCE, the configuration and the environment in a computer system change dynamically according to its use of style, place and purpose. In spite that the system configuration and environment on MCE change dynamically, the applications on MCE don't change

[0]Copyright ©1996 by Noriki AMANO

dynamically. The applications are almost static. So on MCE, one cannot expect that the applications execute efficiently as well as in normal conditions. There are some existing work on MCE providing system and network transparency by OS without changing the applications. But that is not enough. Because OS provides generic services to all applications, there is a limit to support individual applications behaviors, and OS is unable to realize complete system transparency. Applications on MCE with changing dynamically behaviors along their system configuration and environment are required.

2 Our Approach

It is difficult to develop applications that adapt flexibly to MCE with changing dynamically system configuration. Reflection and *Meta-level Architecture* are effective technologies to realize this. In this work, we propose a programming language and a processor for it with meta-level architecture that adapts to the dynamic changeable system environment like the MCE flexibility. Our approach is not to realize true system or network transparency but to let applications change its behavior positively accordingly to the system configuration and environment. The alterations on a system caused by dynamic change system configuration are absorbed by changing applications behaviors. But it doesn't mention that it has no need of system or network transparency. We apply this approach to application-level, which is upper than OS-level.

The language processor we proposed looks like a server providing an empty framework for applications. Some behaviors of applications are described in this empty framework, then this language processor works as a server that dispatches a behavior along the current system configuration to an application. So it supplies behaviors that adapt to the system environment like the MCE to individual applications.

3 The Programming Language LEAD

We propose the programming language LEAD with meta-level architecture that describes applications behavior adapting to the environment. The name of LEAD is an abbreviation, meaning that the language describes applications behavior adapting to the environment. The LEAD language processor is implemented as interpreter with meta-level architecture. The base-level is applications, and the meta-level is the LEAD interpreter itself. Accordingly to a system configuration and environment, the LEAD interpreter dispatches a method in many methods. The method is that a part of the applications depends on the system configuration and environment. We call a set of methods a *Generic Function*. Many methods are described into Meta-level by the LEAD language. By this mechanism, LEAD is able to select and execute a method that is applications' behavior that depends on the system configuration and environment. The mechanism that dispatches methods is supported as a kind of *Meta Object Protocol* (MOP). In short, it is implemented by the LEAD itself like an ordinary function on the LEAD, opened to the public for the users, and the users can modify or redefine it. By this modification, the users can implement a new dispatch policy on LEAD, and this new policy reflects the LEAD interpreter dynamically. The LEAD has a function to execute a program on a remote host computer. This remote execution maintains efficient execution of applications' because the high load method is executed on the remote host computer (more

high-performance computer), it can execute methods efficiently independently to the local host performance and system configuration. The LEAD interpreter looks as if a server that controls methods adapts to a dynamic changeable system. The inter process communication provide by LEAD is based on the Berkeley Socket. After all, applications made by an interpreter language with Berkeley Socket can communicate with the LEAD interpreter and add the function that supports applications' behavior accordingly to the dynamic changeable system environment like the MCE.

4 Contribution of this work

LEAD has many effects to the dynamic changeable system environment like the MCE.

- A part of program (method) that adapts to the system configuration and environment can separate from applications itself and describe as module independence of applications.
- It supplies the common mechanism (dispatch mechanism) which dispatches applications' behavior along the system configuration and environment to applications.
- Without changing applications and existing programming language, it can add the function that selects and executes applications' behavior on the dynamic changeable computing environment to applications made from existing programming languages.
- A part of program in generic function is not limited to specific programming language. Because generic function is a set of methods and methods are a part of the applications, it ought to include applications in itself, and it is executed on applications. So a part of program in generic function uses the language that describes applications.
- It can modify and redefine the dispatch policy flexibility along with system configuration and environment, because the policy that the method dispatch is opened for users as MOP.
- In LEAD interpreter, some functions that get a system configuration and environment can call the same normal function. So it can simulate dynamically an environment easily to define dummy functions.
- The LEAD interpreter has a function that executes method on remote high performance computers. This remote program execution maintains efficient execution of applications' independently of the system configuration and environment.