XMILE: An Incremental Code Mobility System based on XML Technologies

Cecilia Mascolo, Wolfgang Emmerich, and Anthony Finkelstein
Dept. of Computer Science University College London
Gower Street, London WC1E 6BT, UK
{C.Mascolo|W.Emmerich|A.Finkelstein}@cs.ucl.ac.uk
http://www.cs.ucl.ac.uk/staff/c.mascolo

Logical mobility ranges from simple data mobility, where information is transferred, through code mobility that allows the migration of executable code, to mobile agents, in which code and data move together. Several application domains need a more flexible approach to code mobility than the one that can be achieved with Java and with mobile agents in general. This flexibility can either be required as a result of low network bandwidth, scarce resources, and slow or expensive connectivity, like in mobile computing settings, or scalability requirements like in applications on several thousand clients that have to be kept in sync and be updated with new code fragments.

We show how to achieve more fine-grained mobility than in the approaches using mobile agents and Java class loading. We demonstrate that the unit of mobility can be decomposed from an agent or class level, if necessary, down to the level of individual statements. We can then support incremental insertion or substitution of, possibly small, code fragments and open new application areas for code mobility such as management of applications on mobile thin clients, for example wireless connected PDAs or mobile phones, or more in general distributed code update and management.

This work builds on the formal foundation for fine-grained code mobility that was established in [3]. That paper develops a theoretical model for fine-grained mobility at the level of single statements or variables and argues that the potential of code mobility is submerged by the capability of the most commonly used language for code mobility, i.e., Java. We focus on an implementation of fine-grained mobility using standardized and widely available technology.

It has been identified that mobile code is a design concept, independent of technology and can be embodied in various ways in different technologies. The eXtensible Markup Language (XML) [1] can be exploited to achieve code mobility at a very fine-grained level. XML has not been designed for code mobility, however it happens to have some interesting characteristics, mainly related to flexibility, that allow its use for code migration. In particular, we will exploit the tree structure of XML documents and then use XML related technologies, such as XPath and the Document Object Model (DOM) to modify programs dynamically. The availability of this technology considerably simplifies the construction of application-specific languages and their interpreters.

XML provides a flexible approach to describe data structures. We now show that XML can also be used to describe code. XML DTDs (i.e., Data Type Definition) are, in fact, very similar to attribute grammars. Each element of an XML DTD corresponds to a production of a grammar. The contents of the element define the right-hand side of the production. Contents can be declared as enumerations of further elements, element sequences or element alternatives. These give the same expressive power to DTDs as BNFs have for context free grammars. The markup tags, as well as the PCDATA that is included in unrefined DTD elements, define terminal symbols. Elements of XML DTDs can be attributed. These attributes can be used to store the value of identifiers, constants or static semantic information, such as symbol tables and static types. Thus, XML DTDs can be used to define the abstract syntax of programming languages. We refer to documents that are instances of such DTDs as XML programs. XML programs can be interpreted and in interpreters can be constructed using XML technologies. When XML programs are sent from one host to another we effectively achieve code mobility.

Unlike Java programs, which are sent in a compiled form, XML programs are transferred as source code and then interpreted on a remote host. Unlike Java, XML does not confine us to sending coarse-grained units of code; XML documents do not need to begin with the root of the DTD, they can also start with other symbols of the grammar. This enables us to specify sub-programs and even individual statements. We refer to such code fragments as XML program increments. Hence, we can specify complete programs as well as arbitrarily fine-grained increments in XML.

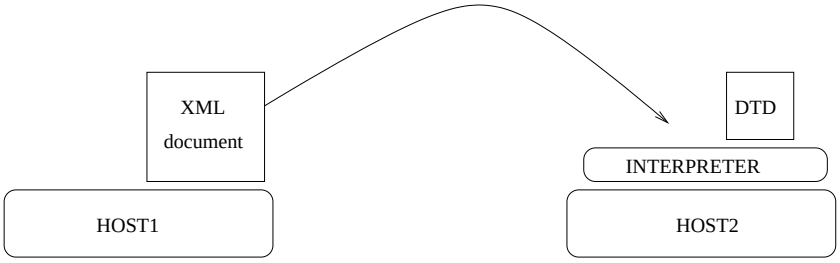

Figure 1: XML Program Migration to Remote Interpreter.

Figure 1 shows how we ship XML programs (every transport protocol can be used). It is possible to specify where to insert/substitute the code increment into the program through the use of XPath for addressing nodes of the tree model defined by the DOM on the XML program. Figure 2 shows the migration of the code increment and the addressing of the insertion point with XPath.

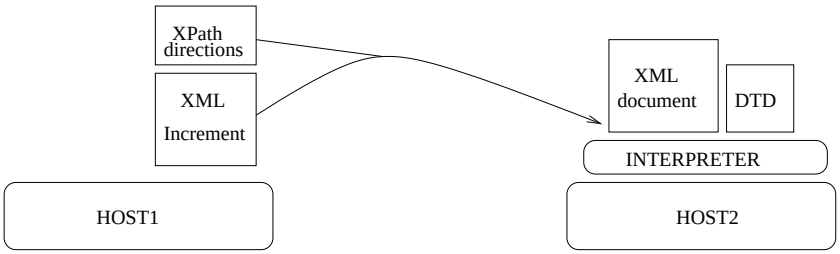

Figure 2: Increment Migration to Robot Site.

The model on which XMILE is based is presented in [2]. We are currently enriching the model and implementing a prototype. We introduced the ability to dynamically bind data and code; single variables as well as single lines of XML code can be moved and patched dynamically into a program, while the XMILE system provides dynamic binding. XML tags for pro-active mobility have been added to the prototype to allow the mobility of lines from the XML programs themselves.

The XMILE system also allows the update of the Java classes used by the XML interpreter (see Figure 1), in order to modify the interpretation of the XML code.

We are now working on mobile computing applications, in particular we used the Symbian EPOC RE5 emulator to run the prototype and we are currently evaluating the performance of XMILE in this setting. Java cards and PDAs are also in the list of devices that we consider for possible application of the approach.

Furthermore work on no-stop (no-reboot) applications can take advantage of XMILE as it is possible to exploit the tree structure to constrain the program execution temporarily to branches of the program while updating other parts (with no need of shutting down the application).

We are now investigating the use of XML schemas in order to allow the modification of the language that for the time being is constrained by the definition of the DTDs. Schemas allow much more flexible treatment of the domain specific languages defined and introduce yet another interesting potential evolution of the XMILE approach.

References

[1] T. Bray, J. Paoli, and C. M. Sperberg-McQueen. Extensible Markup Language. Recommendation http://www.w3.org/TR/1998/REC-xml-19980210, World Wide Web Consortium, March 1998.

[2] W. Emmerich, C. Mascolo, and A. Finkelstein. Implementing Incremental Code Migration with XML. In M. Jazayeri and A. Wolf, editors, *Proc. 22nd Int. Conf. on Software Engineering (ICSE2000)*, pages 397–406, Limerick, Ireland, June 2000. ACM Press.

[3] C. Mascolo, G.P. Picco, and G.-C. Roman. A Fine-Grained Model for Code Mobility . In O. Nierstrasz and M. Lemoine, editors, *Proc. 7th European Software Eng. Conf. (ESEC/FSE 99), Toulouse, France*, volume 1687 of *LNCS*, pages 39–56. Springer, 1999.