10th Workshop on Mobile Object Systems*

Ciarán Bryce[1] and Crzegorz Czajkowski[2]

[1] Object Systems Group,
University of Geneva, Switzerland
Ciaran.Bryce@unige.ch
[2] Sun Microsystem Laboratories,
Mountain View, California, USA
Grzegorz.Czajkowski@sun.com

Introduction

The ECOOP Workshop on Mobile Object Systems is now in its 10th year. Over the years, the workshop has dealt with topics related to the movement of code and data between application platforms, security, operating system support, application quality of service, and programming language paradigms. In many cases, the workshop has been a forum to discuss traditional object-oriented issues, since mobility influences such a broad spectrum of topics.

To celebrate the 10th anniversary of the workshop, we decided to accept papers dealing with all aspects of mobility. Eight presentations in total were made at the workshop, and some time was allocated to discussion of each. The tone of the discussions was informal. The workshop took place on Monday, June 14th, 2004, in Oslo University, just prior to the main ECOOP conference.

The Talks

The presentations made at the workshop were the following:

- *Portable CPU Accounting in Java* by Jarle Hulaas and Walter Binder.
- *Reducing the Overhead of Portable CPU Accounting in Java* by Jarle Hulaas and Walter Binder.
- *An Environment for Decentralized Adapted Services: A Blueprint* by Rüdiger Kapitza, Franz J. Hauck and Hans Reiser.
- *An Extensible Directory Service for Efficient Service Integration* by Walter Binder, Ion Constantinescu and Boi Faltings.
- *Mobile Code, Systems Biology and Metamorphic Programming* by Christian Tschudin.
- Perspectives on Mobile Code by Eric Jul.

* The title of this report should be referenced as "Report from the ECOOP 2004 10th Workshop on Mobile Object Systems".

J. Malenfant and B.M. Østvold (Eds.): ECOOP 2004, LNCS 3344, pp. 169–176, 2004.

– *Proactive Resource Manipulation for Agile Computing* by Niranjan Suri, Marco Carvalho and Jeffrey M. Bradshaw.
– *A Security Model for Autonomous Computing Systems* by Ciarán Bryce.

Jarle Hulaas from EPFL in Switzerland kicked-off the workshop by presenting joint work with Walter Binder entitled *Portable CPU Accounting in Java* for the first part of the talk, and *Reducing the Overhead of Portable CPU Accounting in Java* for the second part. Following an approach entirely based on portable bytecode transformation schemes, in order to allow ubiquitous support for CPU accounting, the presented research has yielded several significant improvements to J-RAF, a previous effort on portable CPU accounting in Java.

Accounting and controlling the resource consumption of applications and individual software components is crucial in server environments that host components on behalf of various clients, in order to protect the host from malicious or badly programmed code. Java and the Java Virtual Machine (JVM) are being increasingly used as the programming language and deployment platform for such servers (Java 2 Enterprise Edition, Servlets, Java Server Pages, Enterprise Java Beans). Moreover, accounting and limiting the resource consumption of applications is a prerequisite to prevent denial-of-service (DoS) attacks in mobile object systems, for which Java is the predominant programming language. However, currently the Java language and standard Java runtime systems lack mechanisms for resource management that could be used to limit the resource consumption of hosted components or to charge the clients for the resource consumption of their deployed components.

Prevailing approaches to providing resource control in Java-based platforms rely on a modified JVM, on native code libraries, or on program transformations. Resource control based on program transformations at the bytecode level offers an important advantage over the other approaches, because it is independent of a particular JVM and underlying operating system. It works with standard Java runtime systems and may be integrated into existing server and mobile object environments. Furthermore, this approach enables resource control within embedded systems based on modern Java processors, which provide a JVM implemented in hardware that cannot be easily modified.

CPU accounting in the initial version of J-RAF relied on a high-priority scheduling thread that executed periodically in order to aggregate the CPU consumption of individual threads and to adjust the running threads' priorities according to given scheduling policies. This approach hampered the claim that J-RAF enabled fully portable resource management in Java, because the scheduling of threads within the JVM is not well specified and the semantics of thread priorities in Java is not precisely defined. Hence, while some JVMs seem to provide preemptive scheduling ensuring that a thread with high-priority will execute whenever it is ready to run, other JVMs do not respect thread priorities at all. Therefore, scheduling code written for environments using J-RAF may not exhibit the same behaviour when executed on different JVM implementations.

To overcome this limitation, the new version J-RAF2 (the Java Resource Accounting Framework, Second Edition) comes with a new scheme for CPU

accounting. In J-RAF2 each thread accounts for its own CPU consumption, taking the number of executed JVM bytecode instructions as platform-independent measurement unit. Periodically, each thread aggregates the collected information concerning its CPU consumption within an account that is shared by all threads of a software component and executes scheduling code that may take actions in order to prevent the threads of a component from exceeding their granted CPU quota. In this way, the CPU accounting scheme of J-RAF2 does not rely on a dedicated scheduling thread, but the scheduling task is distributed among all threads in the system. Hence, the new approach does not rely on the underlying scheduling of the JVM.

During the second part of his presentation, Jarle Hulaas covered different optimization schemes, the goal of which is to lower the additional execution time needed by code transformed by the J-RAF2 tool. Such optimization techniques essentially consist in trying to minimize the number of inserted bytecode instructions, or adding the CPU account as last argument in all method profiles, including the JDK. In the end, the measurements showed an overhead dropping from 300% (for older JVMs) or 40% (for the most recent JVMs from Sun and IBM) down to around 27%. With certain approximation schemes designed to further reduce the amount of inserted accounting instructions, but at the expense of a loss of precision, the overhead would even be as low as 20%.

Rüdiger Kapitza then presented joint work with Franz J. Hauck and Hans Reiser from the Universities of Ulm and Erlangen, entitled *An Environment for Decentralized Adapted Services: A Blueprint.*

This talk described the design and implementation of an infrastructure for service provision that ranges from the traditional client-server to a peer-to-peer approach. This infrastructure is based on the Fragmented Object model. In effect, nodes can negotiate their entry into a service group. The aim of the system is to combine the advantages of both client-server and peer-to-peer.

In the last few years peer-to-peer systems have been one of the most evolving research areas. Apart form the great public demand, the use of peer resources is a major reason for this. In pure peer-to-peer systems, every peer has almost equal responsibilities and provides resources for the whole system. This concept has certain drawbacks in the context of a high number of nodes participating only for short periods of time or peers which try to attack the system. In these cases the overall system performance and service quality degrades or the system may even collapse. In contrast, traditional client-server applications can cope with these problems quite smoothly but offer no possibility to exploit client-side resources.

To fill the gap between the traditional client-sever model and a peer-to-peer based approach, the authors propose a model entitled *Decentralised Adaptive Services.* This concept enables the usage of client-side resources in a controlled, secure fashion and provides services in a scalable, fault-tolerant manner. Decentralised adaptive services consist of a fragmented object which is spread over a dynamic set of peers. Each part of the object might be replicated in the scope of the peer set for fault tolerance or load balancing reasons. Furthermore each

part of the service is mobile and can migrate on demand of the service within the scope of the peer set. Thus a decentralised, adaptive service can be seen as a distributed mobile agent. The peer set dynamically expands or shrinks depending on the participating peers. In this way, decentralised adaptive services are comparable to peer-to-peer systems. In contrast to peer-to-peer systems, where each new peer does not have to support the system. First a peer has to signal the willingness to support the service and provide information about the offered resources. Second the decentralised adaptive service has to decide if the offering is accepted and in what way the provided resources can be used in the context of the service. In fact a decentralised adaptive service can dynamically change the internal service model from a client-sever scenario where client-side resources are simply not used to a peer-to-peer based approach by accepting only peers which offer resources and give each peer the same responsibilities for service provision.

The aim of the Environment for Decentralised Adaptive Services (EDAS) is to provide the basic concepts and mechanisms for the development and the operation of decentralised adaptive services. This includes mechanisms for group membership, resource monitoring and management, mobility, and mechanisms for the use of resources of partially trusted or untrustworthy peers.

Walter Binder presented joint work with Ion Constantinescu and Boi Faltings from EPFL, Switzerland entitled *An Extensible Directory Service for Efficient Service Integration*. The talk presented two techniques for improving the quality of service of an Internet directory service. The first is the addition of sessions, the goal of which is to ensure that an interacting client sees the same view of the directory despite changes that might occur in the services registered. The second modification is a framework for the deployment of ranking functions in the form of mobile code. The code is verified on deployment to ensure safety. An API for these extensions was also presented.

In a future service-oriented Internet, service discovery, integration, and orchestration will be major building blocks. One particularly important issue is the efficient interaction between directory services and service composition engines. In previous work, the authors designed special index structures for the efficient discovery of services based on their input/output behaviour. Each service is characterized by the set of its required input parameters (name and type) and the set of the output parameters provided by the service. The indexing technique is based on the Generalised Search Tree (GiST), proposed as a unifying framework by Hellerstein. Moreover, the authors have also developed service composition algorithms that incrementally retrieve service descriptions from a directory service during the service integration process. The composition problem is specified by a set of available inputs and a set of required outputs. The service integration algorithm creates a workflow that describes the sequence of service invocation and the passing of parameters in order to obtain the required results.

The authors present two recent extensions to their directory, service integration sessions and user-defined ranking functions. The session concept provides a consistent long-term view of the directory data which is necessary to solve complex service integration problems. It is implemented in a way to support a

large number of concurrent sessions. Custom ranking-functions allow to execute user-defined application-specific heuristics directly within the directory, close to the data, in order to transfer the best results for a query first. As a consequence, the tranfer of a large number of unnecessary results can be avoided, thus saving network bandwidth. As different service integration algorithms require different ranking heuristics (e.g., forward chaining, backward chaining, etc.) it is important to have a flexible way to dynamically install new ranking functions inside the directory. As custom ranking functions may be abused for attacks, the directory imposes severe restrictions on the code of these functions.

Christian Tschudin from the University of Basel in Switzerland presented *Mobile Code, Systems Biology and Metamorphic Programming*. The talk discussed some commonalities between computer science and life sciences. Christian Tschudin argued that the lack of acceptance of mobile code is mainly due to the poor general understanding we have of large systems, and that this weakness also is hampering advances in life sciences. Suggestions are made to build methodologies that might benefit both sciences.

The interpretation offered is that mobile code, if one wants to focus on its failure so far, hit the systems wall right from the beginning. Only its very restricted forms (Applets, update deployment) have been successful, while the fully autonomous versions of mobile code were rejected, not so much because of the psychological problem of losing control, but because of the inablity of its proponents to predict the resulting behavior at the system level. Both sides, computer science and system biology, lack engineering know-how about the assembly of mobile code fragments and molecular pathways.

A first avenue presented by the speaker is to cut a slice through the design space such that at least some form of processing can be engineered and tailored for specific requirements. Taking the hypothetical task of creating an artificial cell from scratch, this would mean to first come up with basic building blocks, transistor like, and to build a hierarchy of complexity abstractions up to the point where one can program such a cell. The analogy in the networking area would be the GRID where the next service level is achieved by adding a coordination layer to an existing infrastructure. A basic belief and hope of this approach is that all additional levels of complexity can still be mastered, even for molecular or for otherwise massive computing systems.

The other approach is basically bottom up and is a propagation of the methods found in molecular biology: It consists in turning the basic exploratory instruments into a generative tool by combinatorial means and blind mutation. Translated into computer science this means that one explores the design space in a methodological way instead of cutting out a slice. By generating and evaluating a huge number of program candidates, new solutions can be extracted from the program space that an engineer would never dream of. This second approach is specifically promising for mobile code.

The Fraglet system starts with a simple and executable model that blends code and data: Whether some mobile item is named a data packet or a mobile code fragment becomes a (human) interpretation which is irrelevant. Writing

Fraglet code resembles very much the programming with monadified functions a la metamorphic computing and is tedious at best (for humans). Although it is conceivable to use Fraglets as a target for a BioAmbient compiler, one sees a more attractive use in exploring algorithmic solutions outside the range induced or imposed by a modeling environment. The belief in this case is that search strategies for exploring program space can remain simple while producing high quality results in reasonable time.

In conclusion, complex (man made) systems are a recent challenge both in computer science and biology. The author argues that advances in mobile code research will be linked to progress in this area. There is a natural affinity between mobile code and the corresponding level of autonomous molecules that has become visible in the research literature and that was anticipated in the metaphorical speech of early mobile code research. In another ten years we will better understand their relationship.

Eric Jul from the University of Copenhagen then gave an invited talk on object mobility. The talk compared issues and challenges facing mobile object system designers today, with those faced by the Emerald system designers over 20 years ago. Interestingly, there are close similarities, despite the fact that Emerald was initially designed for a local area network of machines and current object systems aim for the Internet or pervasive computing environments.

One of the issues that all application designers face is when to migrate an object. Moving an object A can also require moving objects that closely communicate with A, since otherwise, the advantage of migrating A is offset by the overheads of the resulting communication between A and these objects. The Emerald system dealt with this issue using the attach command, which allowed the application to specify the set of objects to move with an object. Another mobility issue that designers still face is to locate objects. In Emerald, the fix command allowed an application to bind an object to a specific machine, so that mobility can be controlled to a certain extent.

So what are the new challenges for object mobility today? Perhaps the answer lies in how mobility will be exploited in emerging applications, notably in the pervasive computing domain.

Afterwards, **Niranjan Suri** and **Marco Carvalho** from the Institute for Human and Machine Cognition (USA) and Lancaster University (UK) presented joint work with Jeffrey M. Bradshaw entitled *Proactive Resource Manipulation for Agile Computing*.

This talk's subject deals with the exploitation and active manipulation of any available computing resources in order to get a job done. Resource manipulation may even include physical displacement of the computer. The approach also generalizes a number of domains, including peer-to-peer, active networking and mobile agents.

Exploitation of a resource discovery involves assigning a task to the node that contains the resource. The request to use the resource is handled by the middleware operating on the node. One of the key concepts in the approach is the use of mobile code in order to provide complete flexibility in how the resource

is used. For example, suppose node A contains spare CPU capacity that the agile computing middleware plans to exploit in order to carry out a computation. In order to use that capacity, the middleware will dispatch code that embeds the computation to node A, which in turn will instantiate and execute the code. Similarly, in order to use a node's excess storage capacity, code that is capable of receiving, holding, and serving data would be dispatched. In the same way, in order to use network bandwidth, code that is responsible for relaying data would be dispatched. Mobile code plays a key enabling role in the realization of agile computing without which the system could not be opportunistic. If a new, previously unexpected node with resources is discovered in the military settings we have been investigating, the probability that this node already contains the code necessary to service the request at hand is minimal. Dynamically pushing code allows any resource that is discovered to be put to use as needed.

Policies play an important role in regulating the autonomous behavior of agile computing middleware. If we look at agile computings role as "exploiting the wiggle room" through taking advantage of underutilized system resources, the role of policy is to define and bound the wiggle room so that people are comfortable with allowing their resources to be manipulated and used by the agile computing middleware. The middleware relies on the existence of policies previously defined by humans that constrain the manipulation of resources. For example, policies may be used to govern the movement of the robotic platforms. Policies are defined in the KAoS framework and may range from basic constraints (a ground robot may not venture into water) to safety constraints (a robot must stay a minimum of three feet away from a human) to operational constraints (a robot must not be positioned in the middle of a road and thereby blocking traffic). Policies may also govern various aspects of information feeds (e.g., latency, resolution, access control) from sensor resources. At a more abstract level of consideration, policies may also constrain the adjustment of autonomy itself in an agile computing context.

The FlexFeed middleware realizes the goal of agile computing for distributed sensor networks. FlexFeed addresses several challenges including providing efficient sensor data feeds, hierarchical distribution of data, and policy enforcement. FlexFeed relies on a coordinator that computes resource utilization costs in order to determine the best possible data distribution approach.

Ciarán Bryce from the University of Geneva then presented *A Security Model for Autonomous Systems*. This talk described results from the European project entitled SECURE (IST-2001-32486), a collaboration between the Trinity College Dublin, the University of Cambridge (U.K.), the University of Strathcylde (U.K.), the University of Aarhus (Denmark) and the University of Geneva (Switzerland).

The goal of the SECURE project is to develop a trust-based security model for autonomous systems. Given the very large number of entities in these systems, a model of security was designed that is closer to the way that humans establish trust than to traditional computer models. The heart of the SECURE project is the development of a computational model of trust that provides the

formal basis for reasoning about trust and for the deployment of verifiable security policies. The trust model alone is not sufficient to allow us to deliver a feasible security mechanism for the global computing infrastructure. It is equally important that we understand how trust is formed, evolves and is exploited in a system, i.e., the trust lifecycle; how security policy can be expressed in terms of trust and access control implemented to reflect policy; and how algorithms for trust management can be implemented feasibly for a range of different applications. Further activities address these issues based on an understanding of trust derived from the formal model but also contributing to the understanding of trust as a feasible basis for making security decisions to be embodied in the model.

The result has been the development of a software framework encompassing algorithms for trust management including algorithms to handle trust formation, trust evolution and trust propagation. The framework was used in the development of a SPAM filter application that classifies mail messages as SPAM based on the trust established in the message sender, either via observation of his behaviour or from recommendations received for him. This application is being used to help us to evaluate the SECURE approach. More information about SECURE can be found at the web site:

`http://secure.dsg.cs.tcd.ie.`

Conclusions

The consensus at the end of the workshop was that the workshop was a success, and that the theme of object mobility is present in applications, in some form or another.

One of the issues discussed at the end of the day was whether the workshop series should continue to be held in the context of the ECOOP conference. On the one hand, the number of attendees has been falling over the years as the vision of objects roaming the Internet becomes less attractive. Research into mobility has shown that security is still a major challenge and that programming abstractions for building mobile object applications are missing.

It was suggested that the pervasive computing field might be closer to the mobile object domain today than is standard object-orientation. Many of the issues that are treated in mobile objects occur in pervasive computing. In the former, mobility occurs through objects and threads changing their site of execution. In the later, mobility occurs by physically carrying the device – and its objects and threads – from one network environment to another. A key suggestion in the future is to involve people from the pervasive computing domain in the workshop series.

All papers and presentations of this year's workshop and of all previous editions are available at our website:

`http://cui.unige.ch/~ecoopws`