

# Position Summary: Separating Mobility from Mobile Agents

Kåre J. Lauvset , Kjetil Jacobsen and Dag Johansen  
Dept. of Computer Science, University of Tromsø, Tromsø, Norway

Keith Marzullo  
Dept. of Computer Science, University of California San Diego, La Jolla, USA.

*Mobile agents*, like *processes*, are separate units of concurrent execution. They differ in how they view the processor upon which they run. For processes, the processor is abstracted away: each process can consider itself to be running on an independent virtual machine. For mobile agents, the processor upon which they run is not abstracted away: it is a first-class entity that is under program control. A mobile agent can move from one processor to another in order to profit from the details - such as fast access to local data, use of computational resources and I/O devices, and so on - of the new processor.

The reasons for using mobile agents are well-known: moving computation to data to avoid transferring large amounts of data; supporting disconnected operation by, for example, moving a computation to a network that has better connectivity; supporting autonomous distributed computation by, for example, deploying a personalized filter near a real-time data source. Many mobile agent systems have been constructed and are in the public domain. But, despite these well-known advantages and widely available software, mobile agents are not yet being used as a common programming abstraction.

We have been working since 1993, under the name of TACOMA, on operating system support and application of mobile agents. We have addressed issues including fault-tolerance, security, efficiency, and runtime structures and services. We have built a series of mobile agent middleware systems and evaluated them by building realistic and deployed applications. We have found that mobile agents are especially useful for large-scale systems configuration and deployment, system and service extensibility, and distributed application self-management.

The programming model TACOMA supports has changed over these years to reflect our experience with writing real applications. Like other mobile agent systems, TACOMA started with a programming model that resembled the characterization given above of mobile agents being processes with explicit control over where they execute. We call this the *traditional model* of mobile agents. Using the traditional model leads to several problems including: the complexity of code that contains an explicit and dynamic trajectory; the overhead of implicit

state capture; and the temptation to support only a single programming language (which is typically Java, whose use presents yet other problems).

More fundamentally, we have found that mobile agents are best thought of as one of several tools used together to build distributed applications. A distributed application has, in addition to its *function*, nonfunctional aspects such as its deployment, monitoring, adaptation to a changing runtime environment, and termination. We call this the *factored* model of distributed applications. This model separates the functional aspect of the application from its *mobility* and *management* aspects. Mobile agent platforms can be used to implement the mobility aspect, and the mobile agents themselves to implement the management aspects.

Legacy and COTS software constitute a significant portion of the function of many real-world distributed applications. The mobility aspect provides the mechanisms and structures necessary for deploying the function and for its adaptive reconfiguration. The management aspect manages both function and mobility at a higher level. More specifically, it implements policies for *when*, *where*, and *how* to execute the function. Examples of management policies of applications include fault-tolerance, server cloning to accommodate increased demand, and invoking security countermeasures in response to intrusion detection alarms.

We have redesigned TACOMA to only directly provide the mobility aspect of distributed applications. This version, called vTOS, provides less than full-fledged mobile agent systems and more than remote execution facilities such as `ssh`, `rsh` and `rexec`. It is rather small: it consists of approximately 90 lines of Python code. Despite its diminutive size, it can be used to implement itinerant mobile agents that move over encrypted network channels.

We have used vTOS to construct some simple but realistic distributed applications such as a parallel image renderer based on COTS components. We are now building a personal computational grid called *Open Grid*. Doing so is making concrete issues of deployment, security, fault-tolerance, and adaptation.

Further details on the vTOS project can be found at <http://tacoma.cs.uit.no>.

---

This work was supported by NSF (Norway) grants No. 112578/431 and 126107/431 (Norges Forskningsråd, DITS program).