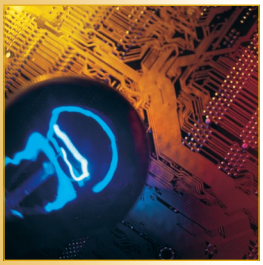

Deploying FIPA-Compliant Systems on Handheld Devices

LEAP is a runtime environment for deploying agents on a network of Java-enabled devices. It complies with FIPA international standards for multiagent systems.

Federico Bergenti and Agostino Poggi
University of Parma

Bernard Burg
Hewlett-Packard Laboratories

Giovanni Caire
Telecom Italia Lab

The market for portable devices is stimulating the migration of technologies originally developed for PCs to the realm of handhelds and wireless networks. Agent technology is following this downsizing trend, and many projects are under way to enable multiagent systems on handheld devices. The possibility of enabling agent technology in the telecommunications world, with its base of several hundred million users, calls for particular attention to an infrastructure that can readily interoperate with third-party software. At the moment, only the Foundation for Intelligent Physical Agents, or FIPA, is producing industrial-strength specifications for developing such an infrastructure (see the sidebar, "FIPA: Open Standards for Agent Systems"). FIPA specifications address not only communication issues, but also general-purpose services like naming and a standard way to manage agent life cycles.

The Lightweight Extensive Agent Platform project is the first attempt to implement a FIPA agent platform that runs seamlessly on both mobile and fixed devices over both wireless and wired networks. It is funded by the European Commission and involves research centers in France (Motorola), Germany (Siemens and Allgemeiner Deutscher Automobil-Club), Ireland (Broadcom), England (British Telecom), and Italy (Telecom Italia LAB and University of Parma).

A Phase 1 version of LEAP has been released to the technical community and is available to registered users for testing purposes from the Web site at http://leap.crm-paris. com/; a Phase 2 version is planned for release in open source by the end of this year.In this article, we begin by clarifying the agent technology underlying the LEAP project, followed by a description of the platform itself and plans for future work.

 1089-7801/01/$10.00 ©2001 IEEE

FIPA: Open Standards for Agent Systems

The worldwide information-technology infrastructure established by the Internet and by international application-layer standards, such as those developed by the World Wide Web Consortium (W3C) and Object Management Group (OMG), lays a foundation for deploying agent technologies in real-world applications.

The Foundation for Intelligent Physical Agents (FIPA) was established in 1996 to specify the interfaces to components in this environment with which an agent can interact. With a worldwide membership and a tight schedule for producing, validating, and testing specifications, we believe that FIPA has a good chance to set the standards for open-agent systems, especially in the telecommunications industry.

FIPA does not specify an agent's internal architecture, which is left to the developer. Specifications released in October 1997 and October 1998 cover the following aspects of agent technology:

- *Normative specifications* that mandate an agent's external behavior and ensure interoperability with other FIPA-specified subsystems; they include specifications for agent management, agent communication languages, agent and legacy software integration, human-agent interaction, agent security, agent mobility, and ontology services.
- *Informative specifications* for use in real-world industry applications; these include personal travel assistance, personal assistants, audiovisual entertainment and broadcasting, and network management and provisioning.

The FIPA 2000 specifications, which supercede the 1997 and 1998 specifications, include an abstract architecture to guide the creation of open multiagent systems.

Aspects of FIPA 2000 relevant to the LEAP project are

- *Agent management and agent communication*: support for a variety of encoding and naming schemes, including schemes that allow agents to operate in mobile contexts;
- *Nomadic computing*: agent management and communication frameworks geared toward support of mobile communication networks.

The FIPA 2000 specifications can be downloaded from the FIPA Web site at http://www.fipa.org/.

FIPA Platforms and LEAP Locations

A LEAP application entails the creation of a community of software agents that perform a distributed computation. LEAP is agnostic about the internal architecture of these agents. It relies instead on what are called *typed-message agents*,[1] defined solely in terms of a communication model and a simple notion of autonomy. Typed-message agents work together by sharing messages expressed in an agent communication language, or ACL. The ACL defines some semantics for the community prior to runtime. Such agents can send messages that are not in direct response to a query. In other words, typed-message agents can initiate a conversation autonomously.

This definition of agent does not depend on a measure of intelligence or on a more general notion of autonomy. Nor does it rely on "working on behalf of the user," which entails a direct interaction between the agent and user. The typed-message agent model is simple, but it does not limit the use of LEAP to support richer agent models (for example, we have used LEAP to implement interoperable goal-oriented agents with planning and reasoning capabilities[2]).

Agents need resources to operate and to communicate. In the FIPA abstract architecture (see the FIPA sidebar), such resources are provided by an *agent platform* that provides the basic services for life-cycle management and communication:

- a means for sending and receiving messages, and
- a means for discovering other agents—that is, a yellow pages service and a naming convention.

Agents run only in the scope of a platform.

FIPA provides a standard ACL to support interoperable message exchange. Agents are addressed using unique identifiers. Thus, messages can reach agents within the same platform or on other platforms. This enables open societies where agents running on different platforms can join and leave the system dynamically.

Deploying agents across platforms is one way to distribute a system in the network. From the beginning, FIPA specifications have promoted this kind of distribution, using CORBA as the standard interface between platforms. FIPA is in the process of standardizing more protocols, but it still falls short in two areas:

- handheld devices, which have limited resources and where the choice of the communication protocol affects performance and memory usage; and
- new wireless networks available on the market, such as Bluetooth.

Related Work in Multiagent Platforms

For surveys of multiagent platforms, including comparisons of their characteristics, see Nowostawski et al.[1] or Ricordel and Demazeau.[2]

Besides the JADE platform (http://jade.cselt.it) used in LEAP, other FIPA-compliant platforms for multiagent system development include

- **Bee-gent.** Bee-gent (http://www2.toshiba.co.jp/beegent/) is a software framework developed by Toshiba.[3] It provides two types of agents: wrapper agents used to "agentify" existing applications, and mediator agents to support the coordination of wrappers. Mediator agents are mobile, and agents communicate through XML/ACL messages. Bee-gent also offers a tool to describe agents through state-transition diagrams; naming/directory facilities; ontology-matching facilities to translate synonyms; and security facilities based on digital fingerprint authentication and key encryption.
- **FIPA-OS.** FIPA-OS (http://fipa-os.sourceforge.net) is a software framework developed by Nortel Networks.[4] The framework provides the mandatory components of a FIPA agent platform – that is, AMS, DF, and an internal platform message transport system. FIPA-OS also includes an agent shell and an agent template.
- **Zeus**. Zeus (http://193.113.209.147/projects/agents.htm) supports the rapid development of Java agent systems through a library of agent components and a visual environment for capturing user specifications.[5] The agent-building environment includes an automatic code generator and a collection of classes that form the building blocks of individual agents.

  Agents are composed in five layers: an *API layer* permits interaction with the non-agentized world; a *definition layer* manages agent tasks; an *organizational layer* manages knowledge concerning other agents; a *coordination layer* manages negotiation with other agents; and a *communication layer* enables the communication with other agents.

References

1. M. Nowostawski et al., "Platforms for Agent-Oriented Software Engineering," *Proc. Seventh Asian-Pacific Software Engineering Conf.*, IEEE CS Press, Los Alamitos, Calif., 2000, pp. 480-488.
2. P.M. Ricordel and Y. Demazeau, "From Analysis to Deployment: A Multi-Agent Platform Survey," in *Engineering Societies in the Agents World*, A. Omicini, R. Tolksdorf, and F. Zambonelli, eds., Springer-Verlag, Berlin, 2000, pp. 93-105.
3. T. Kawamura et al., "Bee-gent: Bonding and Encapsulation Enhancement Agent Framework for Development of Distributed Systems," *Proc. Seventh Asian-Pacific Software Engineering Conf.*, IEEE CS Press, Los Alamitos, Calif., 2000, 260-267.
4. S. Poslad, P. Buckle, and R. Hadingham. "The FIPA-OS Agent Platform: Open Source for Open Standards," *Proc. 5th Int'l Conf. and Exhibition on the Practical Application of Intelligent Agents and Multiagents* (PAAM 2000), Practical Application Co., Blackpool, UK, 2000, pp. 355-368; also available at http://fipa-os.sourceforge.net/.
5. H.S. Nwana, D.T. Ndumu, and L.C. Lee, "ZEUS: An Advanced Toolkit for Engineering Distributed Multi-Agent Systems," *Proc. Third Int'l Conf. and Exhibition on PAAM* (PAAM 98), Practical Application Co., Blackpool, UK, 1998, pp. 377-391.

To overcome these problems while preserving FIPA compliancy, LEAP distributes a single platform across the network in terms of *locations*. LEAP can allocate different locations of a single platform to different network nodes; it includes internal mechanisms that hide the distribution from agents. The platform is seen as a whole even if it is distributed because of a *front-end location*, which acts as a dispatcher for messages coming from and going to other platforms. This technique for distributing agents allows the use of proprietary protocols for the communication between locations, thus solving the problems faced with FIPA protocols.

All active locations must be able to reach the front-end location, which maintains platform-wide information and provides platform-wide services, including those that are mandatory for FIPA compliance:

- the agent management service (AMS), which acts as a white-pages service and represents the authority of the platform;
- the directory facilitator (DF), which acts as a yellow-pages service for the platform.

LEAP is thus composed of a single front-end location and a set of peripheral locations, allowing high modularity by allocating lightweight locations to network nodes with restricted resources.

Because our agents run on mobile devices, they face disconnections from the network. We delegate responsibility for handling such events to the agent platform. In particular, the platform hides any connectivity loss by buffering incoming and outgoing messages.

The LEAP Project

The LEAP project's primary goal is to develop a FIPA-compliant agent platform that is

- sufficiently lightweight to execute on a mobile device with limited resources, such as a cellular phone, and

- sufficiently open and flexible to be a first-class choice for devices with no severe limitations on resources, such as an enterprise server.

LEAP is deployed according to a set of profiles identifying the functionality available on a particular device. The *basic profile* supports only the functionality required by FIPA compliancy and suits the smallest device that the platform supports, namely, a mobile phone. The *full-featured profile* provides the functionality available on a platform designed to run on PCs and suits any device with sufficient memory and processing power. All profiles are instantiations of the FIPA abstract architecture, and agents running on platforms configured with different profiles can interoperate.

We based the LEAP implementation on an existing open-source FIPA platform, the Java Agent Development Framework,[3] called Jade for short. Jade is a modular software framework that provides an agent platform organized in terms of interacting containers; a main container provides FIPA's mandatory AMS and DF services. Jade supports the concept of profiles, and it offers runtime and agent programming libraries, as well as tools to manage platform execution and to monitor and debug agent communities. Such tools are implemented as FIPA agents and they require no special support from the platform. LEAP essentially provides a new kernel for Jade.

While there are several FIPA platforms available (see the sidebar, "Related Work in Multiagent Platforms"), Jade is popular among agent builders and familiar to us. Because LEAP shares the basic design principles of Jade, it preserves the APIs and permits agents developed for Jade to run on LEAP without any modification. Thus any agent developed with Jade can also run on a wireless device that has sufficient resources and power to support it.

Nevertheless, the implementation of LEAP is basically different from the Jade implementation. Specifically, it involved redesigning the Jade communication modules and addressing several deployment issues related to the constraints of the LEAP scenarios. We address these issues in more detail below.

Communication Mechanisms

LEAP is naturally distributed across locations that offer agents the runtime resources they need. Following Jade nomenclature, we call the locations *agent containers*, or simply containers. LEAP imposes no restrictions on the way containers are deployed across network nodes. Even though the best way of deploying LEAP is to have one container running in one Java virtual machine for every node, it is not mandatory. You could concentrate the whole platform into a single container.

For example, if an application consists simply of a personal assistant supporting the user in managing information on a PDA, the best deployment is probably a single-container platform running on the PDA. However, the resources required by FIPA's AMS and DF in their current implementations suggest that the main container — that is, the front-end location of the platform — should run on a device with no particular restrictions on processing power and memory.

FIPA specifies both a communication protocol set for sending messages to agents and a gateway for matching the individual protocols to allow platforms using different protocols to interoperate. The 1997 and 1998 FIPA specifications required all FIPA agents to be addressed through a CORBA interface, so the first FIPA platforms relied on CORBA only. Where applicable, this choice is still adopted.

In LEAP, the main container is supposed to run on a full-featured computer, and we can reasonably expect to find a CORBA implementation in that environment. This assumption is not valid, however, for peripheral containers running on mobile devices, and it may not be the best solution for peripheral containers running on PCs. LEAP therefore implements the communication between the main container and the rest of the platform through a set of protocols that we call *internal transport protocols*.

The current implementation of LEAP provides three ITPs: IIOP, Java RMI, and a proprietary protocol that we developed over TCP/IP for wired and wireless connections. However, for a resource-limited device with a wireless connection, only the proprietary protocol is a reasonable solution. It provides simple remote method invocations with object-by-values, and it fits the restriction of Java 2 Micro Edition (J2ME).

We have implemented the protocol with TCP/IP over GMS (the Groupe Spécial Mobile pan-European public land mobile system) and IEEE 802.11 wireless LAN. FIPA's AMS acts as the authority governing the agent platform, and agents must refer to it to undertake life-cycle transitions — that is, to change their state from active to suspended or to move from one platform to another. FIPA does not specify how agents should interact with the AMS for life-cycle transitions. These activities are private to the platform, and each platform can use its own optimized techniques.

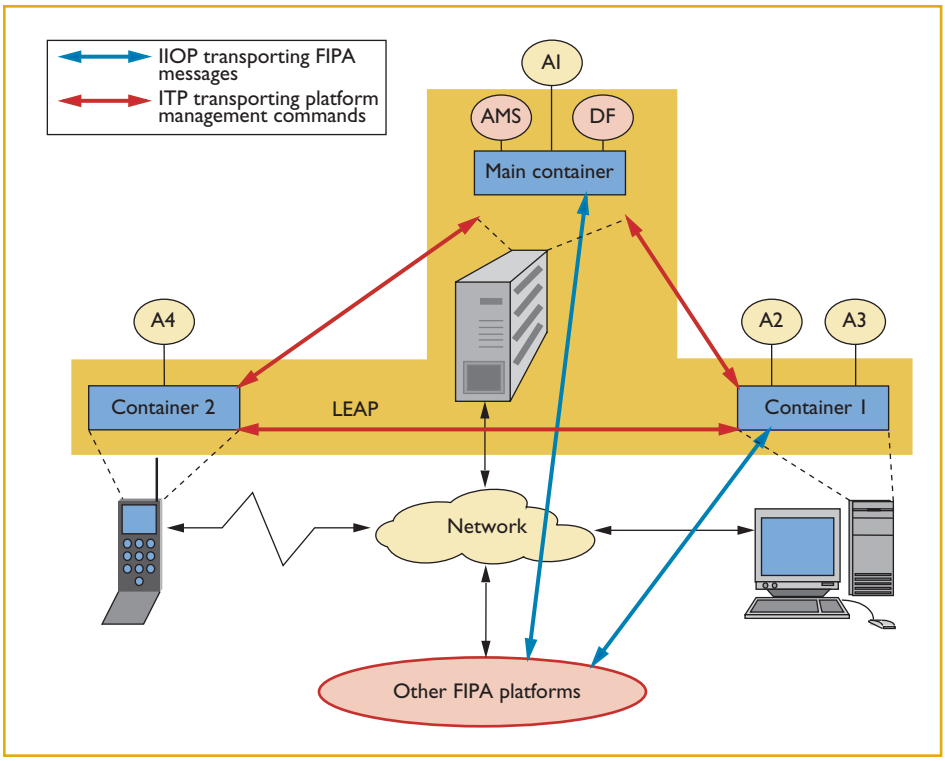

Figure 1. LEAP transport mechanisms. Only the main container and container 1 can support the IIOP, which lets them connect directly to other third-party FIPA platforms. Container 2 must use an internal transport protocol (ITP) suitable for a handheld device and connect to other FIPA platforms through a container that supports IIOP.

Figure 1 shows a possible LEAP deployment, emphasizing the communication mechanisms. It shows three containers:

- the main container runs on an enterprise server connected to the Internet,
- container 1 runs on a PC, and
- container 2 runs on a Java-enabled mobile phone.

Containers are connected between each other through a feasible ITP. If there is no feasible ITP, then the main container acts as a dispatcher for messages. Only the containers supporting IIOP, that is, the main container and container 1, can connect directly to other FIPA platforms. Messages to and from container 2 are routed through the main container.

LEAP exploits the availability of the three ITPs to support internal communication with the AMS.

Agents send and receive messages exploiting the containers that host them. As far as these activities are concerned, a container can perform the following functions:

- send and accept ACL messages through at least one ITP,
- buffer incoming messages directed to one of its agents,
- forward to the main container all ACL messages directed to an agent running in another container, and
- send management messages to and accept them from the main container.

For profiles with no severe constraints on resources, containers can also

- forward ACL messages directed to an agent running in another container through direct connection with the peer container, and
- cache the addresses of an agent and the associated containers where it runs.

Deployment Issues

The target devices for LEAP can vary widely in terms of memory, computational power, display, pointing devices, and connectivity capabilities, as shown in Table 1. The resources available to Java applications may differ significantly from those in Table 1, mainly in the implementation of the available virtual machine. For example, the KVM available for PalmOS reserves at most 200 kilobytes of memory for Java applications even if the device has 8 megabytes.

To adapt to target devices while maintaining a minimal footprint, the LEAP architecture is split into modules that are categorized as follows:

- mandatory and device independent, like the AMS module responsible for managing agent life cycles;
- mandatory and dependent on device capabilities, like the communication module;
- optional and device independent, like the security plug-in for the communication module; and
- optional and device dependent, like all GUIs.

High-level platform functionality can be spread across different modules. For example, the agent communication channel that FIPA mandates to support interoperable messaging between third-party

platforms is split over the communication modules of different containers of the same platform.

LEAP development is based on the Java 2 platform, which provides a common layer of platform-independent functionality available on mobile phones, PDAs, and PCs running all sorts of operating systems. Although Java provides a solid foundation, small devices do not provide a full-featured Java 2 platform. To meet the platform goals of being sufficiently lightweight for a mobile device while remaining a first-class choice for an enterprise server, LEAP implements an adaptation layer that matches the classes available on the most restrictive Java 2 platform — namely the connected, limited device configuration (CLDC) of the J2ME — to the classes available in standard Java. We used the Abstract Factory[4] pattern to match the classes available on different Java 2 profiles and configurations, and we employed the Half-Object Plus Protocol[5] pattern to make LEAP independent from communication protocols.

Table 1. Target devices for LEAP.

Category	Memory	Connectivity
Cellular phone	< 16 Mbytes	1. TCP/IP over GSM
		2. SMS
PDA	16–32 Mbytes	1. TCP/IP over GSM
		2. SMS
		3. IEEE 802.11
PC	64–256 Mbytes	1. TCP/IP

Conclusion

LEAP is being applied to agent services that support dynamic enterprises and mobile teams. In Phase 1 of the project, a first implementation of LEAP software has been realized. In Phase 2, field trials are being conducted for three services that anticipate a user's knowledge needs on the basis of skill and location, then provide access to collective knowledge assets and support for coordinated collective activities. The trials target mobile engineers working the telecommunications and roadside assistance industries. The results are intended to clarify practical issues related to industrial deployment and management of agent services on mobile devices.

For more information, see the project Web site at http://leap.crm-paris.com.

Acknowledgments

We would like to thank the other partners of the LEAP project: Allgemeiner Deutscher Automobil-Club, Broadcom, British Telecom, Motorola, and Siemens. The European Commission funds this work through the grant IST-1999-10211.

References

1. C. Petrie, "Agent-Based Engineering, the Web, and Intelligence," *IEEE Expert*, vol. 11, no. 6, Nov./Dec. 1996, pp. 24-29.
2. F. Bergenti and A. Poggi, "A Development Toolkit to Realize Autonomous and Interoperable Agents," *Proc. Autonomous Agents 2001*, ACM Press, New York, 2001, pp. 632-639.
3. F. Bellifemine, A. Poggi, and G. Rimassa, "Developing Multi-agent Systems with a FIPA-Compliant Agent Framework," *Software–Practice and Experience*, vol. 31, 2001, pp. 103-128.
4. E. Gamma et al., *Design Patterns*, Addison-Wesley, Reading, Mass., 1994.
5. G. Meszaros, "Pattern: Half-Object + Protocol," *Pattern Languages of Program Design*, J.O. Coplien and D.C. Schmidt, eds., Addison-Wesley, Reading, Mass., 1995.

Federico Bergenti is a PhD candidate in computer engineering at the Information Engineering Dept., University of Parma. His research interests include intelligent agents, ontologies, and software engineering. His current work focuses on methodologies for agent-oriented software engineering and the application of agents to CSCW.

Bernard Burg is a department manager for Hewlett-Packard Laboratories, Palo Alto, California. He coordinated the LEAP project up to the end of 2000. His research interests include agents, machine learning, and ontologies. Burg received a PhD in computer science from Paris XI University. He is director of the FIPA board and chair of its technical committee on service-level agreements.

Giovanni Caire is a researcher at the Telecom Italia Lab. His current interest is in intelligent agent technology, including the LEAP project and the implementation of Jade. He received a degree in engineering from Politecnico di Torino in 1992.

Agostino Poggi is associate professor of computer engineering at the University of Parma, where he coordinates the Agent and Object Technology group. He received a Laurea degree cum laude in electronics engineering in 1987 and a PhD in computers and electronics engineering in 1992, both from the University of Genoa. He is member of AAAI, AI*IA, the IEEE, and the editorial board of *IEEE Internet Computing*.

Readers may contact the authors via e-mail at {Bergenti,Poggi}@ce.unipr.it, Bernard_Burg@hp.com, and Giovanni.Caire@tilab.com.