Programmable Coordination Infrastructures for Mobility

Franco Zambonelli, Giacomo Cabri, Letizia Leonardi
Dipartimento di Scienze dell'Ingegneria – Università di Modena e Reggio Emilia
Via Vignolese 905 – 41100 Modena – ITALY
Phone: +39-059-2056133 – Fax: +39-059-2056126
{franco.zambonelli, giacomo.cabri, letizia.leonardi}@unimo.it

Abstract

Mobile application components can be modeled in terms of autonomous agents situated in given interaction contexts. A framework based on the definition of programmable interaction contexts promotes an engineered approach to application design and, if supported by a proper infrastructure, can make applications more modular and easy to maintain. The paper sketches the proposed framework and analyses the main issues related to the implementation of a programmable coordination infrastructure.

1 Introduction

Mobility may appear in different flavors in today's Internet computing environments:

- *Virtual mobility*. The wideness and the openness of the Internet scenario makes it suitable to design applications in terms of components that are aware of the distributed nature of the target and explicitly locate and access resources, services, and agents. From a different perspective, components "navigate" the Internet and *virtually move* across its resources.
- *Actual mobility*. This refer to the components' capability of moving across Internet sites while executing by dynamically and autonomously transferring their code, data and state, toward the resources they need to access.
- *Physical mobility*. Mobile devices accessing the Internet – such as palmtop, cellular phones, etc. – will be more and more present in application scenarios, and they will have to be properly handled and modeled.

To limit complexity of application design and development, suitable models and infrastructures are needed to handle all the above kinds of mobility in a natural and uniform way.

A promising approach to deal with mobility and associated issues is to model application components, as well as physical mobile devices, in terms of *autonomous agents*. In general terms, agents are autonomous entities capable both of reacting to changes in the environment and of executing in a proactive way, and are typically implemented by active objects integrating event-handling and exception-handling capabilities. This implies that:

- agents are provided local control over their activities, thus enabling dealing in a natural and decentralized way with the openness, dynamicity, and unpredictability of the Internet and, more generally, of dynamic and decentralized networked scenarios;
- agents are explicitly designed to be situated in an environment. This naturally invites thinking mobile application components in terms of agents that situate in different environments during their lives.
- physical mobility, unlike virtual and actual ones, cannot be controlled at the application level, and a useful way of modeling it is in terms of an additional dimension of autonomy of application components.

In the following, we detail how such a perspective can drive the definition both of a suitable conceptual framework for the design of applications and of the corresponding infrastructure.

2 The Conceptual Framework

2.1 Local Interaction Context

Handling mobility requires facing different issues at different levels, also depending on the type of mobility to be handled. However, as far as the high-level issues related to the modeling, design and development of complex multi-component applications are concerned, handling mobility is basically a problem of handling the coordination activities of application agents. These may include accessing the local resources of an environment and communicating and synchronizing with executing agents, whether belonging to the same application or foreign Internet agents.

In our approach, we model mobility of agents across the Internet as movements across *local interaction contexts*. A local interaction context defines the agents' perceivable world, which changes depending on the agent position, and which represent the logical place in which agents' coordination activities occur.

We disregard the modeling of the *execution contexts* intended as the places in which the agents actually execute. In this way, our model can fully disregard the specific issues related to the type of mobility exhibited by application agents (i.e., virtual, actual, or physical).

2.2 Local Coordination Laws

Movements across interaction contexts may impact on agents' coordination activities. In fact, in the open Internet scenario, one cannot conceive that agents' coordination activities can be totally free and unregulated. Instead, coordination activities in the Internet are likely to be strictly ruled by proper security and resource control policies, which may be different from site to site, i.e., from a local interaction context to another. Moreover, each local interaction context is likely to adopt peculiar local choices for the representation of local resources, and it is likely to host the coordination activities of different agents, and of making available different services to agents.

In such as scenario, the local interaction context cannot be simply considered as the place in which coordination activities occur, but it is also an active context, capable of enacting specific *local coordination laws* to rule and support the agents' coordination activities.

2.3 Application-Specific Laws

The above is not the full picture. In fact, despite the fact that mobility makes agents interact within different interaction contexts during their lives, agents are not necessarily stand-alone entities. Instead, they may be part of a cooperative multi-agent applications, and move in the Internet to cooperatively achieve, according to specific protocols and patterns, specific application sub-goals. In other words, agents may logically belong to an application-specific interaction context, despite the fact that they actually situate in different, distributed, local interaction contexts.

Given that, it is clear that agents' coordination activities within a multi-agent application may not be fully deregulated but, again, may be required to occur accordingly to specific laws that rule the global application and ensure the proper achievement of the global application goal.

2.4 Designing Applications around Programmable Interaction Contexts

The above analysis suggests modeling and designing applications in terms of agents interacting via *active interaction contexts*. Contexts are no longer merely the place in which agents coordination activities occur, but they become the place where both local and application-specific coordination laws reside and are enacted, via programmability of the behavior of the interaction spaces.

The adoption of such a conceptual framework – that we have defined *context-dependent coordination* [2] – can have a very positive impact on the engineering of mobile agent applications. From the point of view of application designers, the framework naturally invites in designing an application by clearly separating the intra-agent aspects and inter-agent ones. The formers define the internal behavior of agents and its observable behavior. The latters define the application-specific coordination laws according to which agents should interact with each other and with external entities for the global application goal to be coherently achieved. These lead to the identification of the coordination laws that agents should spread on the visited interaction context. This separation of concerns is likely to reduce the complexity of application design and can make it more modular and easy to be maintained (design-for-change perspective).

Independent of the role of the application designers and is the role of site administrators. When new kinds of application agents are going to be deployed on the Internet, the administrator of one site can analyze which local coordination laws that (s)he may find it necessary to locally enforce. These can be used both to facilitate the execution of the agents on a site and to protect it from improper exploitation of the local interaction context. These site-specific laws will work together with the application-specific coordination laws, to be possibly spread by application agents.

3 Infrastructures

The separation of concerns promoted by context-dependent coordination during analysis and design can be preserved during the development and maintenance phases too if a proper coordination infrastructure is available that somehow reflects the concepts and the abstractions of the context-dependent coordination. In that case, the code of the agent can be clearly separated from the code implementing the coordination laws (whether local or application-specific ones). Thus, agents and coordination laws can be coded, changed, and re-used, independently of each other.

A coordination infrastructure for context-dependent coordination must be based on an architecture implementing the abstraction of programmable local interaction contexts abstraction, i.e., *programmable coordination media*, intended as those software systems mediating and ruling all coordination activities of application agents within a locality.

There are several issues to be handled in the definition of a programmable coordination infrastructure for the handling of agents' coordination activities. Such issues include:

1. defining an architecture based on a multiplicity of independent and independently programmable coordination media, each associate to a locality scope, and providing for dynamically bounding an agent to a coordination medium accordingly to the agents' movements;
2. defining the interaction model to be actually supported by the coordination media, e.g., message-based, or tuple-based, or event-based;
3. enabling a dynamic programming of its behavior both by the local administrators and by application agents' themselves;
4. identifying a suitable model (and the associated language) for the programming of the coordination media behavior;

In the following, we mainly focus on issues 1 and 3 (due to page limitations) by distinguishing the case in which a fixed network infrastructure is available from the one in which it is not.

3.1 Static Network Infrastructures

When agents execute and interact via the support of a fixed network infrastructure, the most natural choice is to conceive coordination media as allocated on the fixed network architecture.

With regard to the architecture of coordination media, several choices can be adopted.

On the one hand, coordination media can be associated to a single Internet node, to act both as the place via which to access to the local resources of that node and as "agora" for a set of interacting agents. On the other hand, a set of Internet nodes can share a single coordination medium, to be exploited for all the agents interacting in the context of that domain. That coordination medium can then be associated to a single node, or it can be implemented in a distributed across a multiplicity of nodes in the domain.

It is also worth noting that nothing prevents from conceiving and implementing a single coordination medium to serve a geographically distributed domain of nodes, and thus realizing a sort of virtual locality abstraction. However, in these cases, implementation problems arise in guaranteeing a correct and consistent execution of programmed reactions. In fact, the behavior of a coordination medium can rely on the access to a possibly large status of the medium itself. Thus, maintaining the consistency of the state of a geographically distributed coordination media can incur in high overhead and made the behavior of the medium itself unreliable. These problems may be affordable when the interaction model defined by a coordination medium and the specific behaviors programmed in it are likely to exploit only loosely the state of the media, as it can be the case of a message-based coordination medium. However, in the case of interaction models in which the status of the medium plays a central role, as it can be the case of tuple spaces, the cost of maintaining the consistency on large status may be simply unaffordable. Thus, in general, we can see a federation of geographically distributed nodes not as a set of nodes sharing a single coordination medium, but rather as a set of nodes each implementing its own independent medium, and sharing only the coordination laws enacted in their media. The above perspective is the static, administrator-centered, counterpart of the perspective enforced by giving application agents of dynamically programming the visited coordination media with application-specific coordination laws. In that cases, all agents of an application spread the same coordination laws over all visited sites to share the same application-specific coordination laws (in addition to the locally enforced ones).

Enabling dynamic programmability is basically an issue of *code mobility*. In general, coordination laws represent a computational behavior to be assumed by the coordination media in response to interaction events. Then, as agents interact with different coordination media allocated on different sites during their lives, and they may be in need of programming such media, coordination laws must be necessarily expressed by using mobile code technology [3]. However, it is interesting to show how the specific type of agent mobility may impact on the specific paradigm of code mobility to be adopted for enabling agent to program the accessed coordination media via application-specific coordination laws.

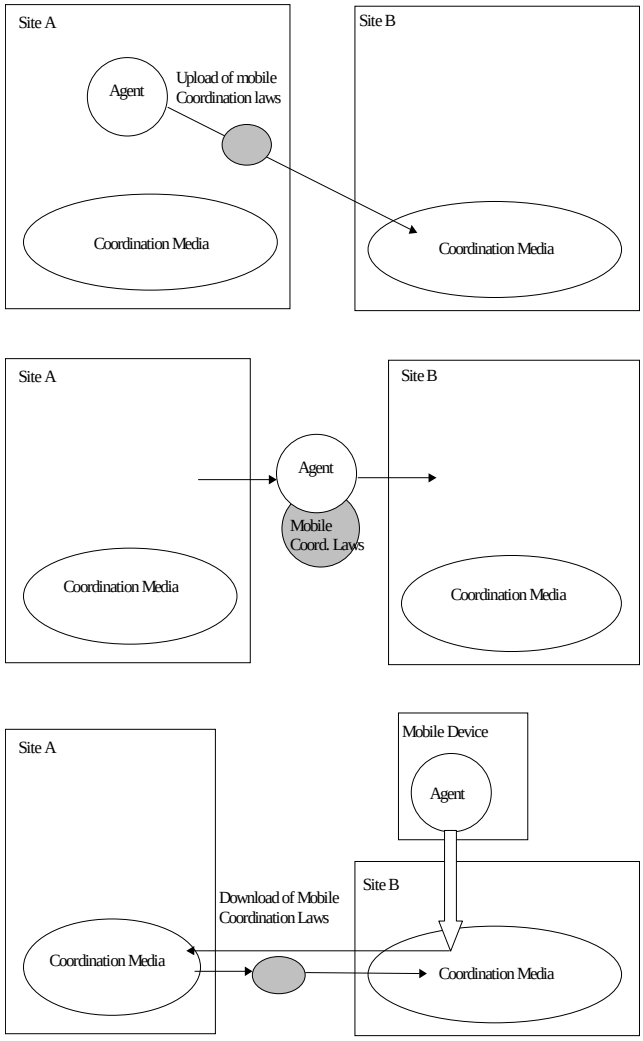

Figure 1a (up) a virtually mobile agent uploading coordination laws. Figure 1b (middle): an actually mobile agent carrying on the coordination laws. Figure 1c (bottom): a physically mobile agent makes the coordination medium downloads the coordination laws.

In the case of a virtually mobile agent (Figure 1a), the agent can interact with remote coordination media without actually transferring itself. However, if the agent wants to impose

some specific coordination laws on a remote medium, it has to upload to the remote medium the mobile code implementing the coordination laws. That defines a model of "remote evaluation", in that the code is transferred to a remote site and there execute in response to the agent's coordination activities.

In the case of an actually mobile agents (Figure 1b), the agent transfers across the network its code and state, and carries on also the mobile code needed to implement the application-specific coordination laws.

In the case of a physical mobile agents (Figure 1c), one can think at having the agent itself (i.e., the physical device it represents) carry the code needed to implement the coordination laws, and install it in the coordination media it connects to during its itinerant life. However, in most of the cases, mobile devices are likely to be resource-limited devices, for which it is often desirable to limit the amount of memory exploited by application as well as the amount of bandwidth. Therefore, a more suitable way for physical mobile agents to program coordination media is to let the coordination media, upon the access of a physical mobile device, download from a specific "home site" the required code. This defines a model of "code on demand", in which the code is transferred to a coordination medium upon a local request generated from the coordination medium itself.

In any case, we emphasize that, whatever the type of mobility of application agents and whatever the consequent type of code mobility, they can coexists in a coordination media. In fact, the coordination media, upon notification of the arrival of an agents, can discover whether has virtually, actually, or physically arrive, and can act accordingly for properly installing the coordination laws. Thus, all types of mobility can be handled transparently from the point of view of application designers.

3.2 Mobile Ad-Hoc Networks

More challenging from the implementation point of view is the case in which agents have to interact in the absence of any fixed infrastructure, i.e., the case of mobile ad-hoc networks (MANET). We believe that the cases in which a set of mobile devices will have to compulsory interact in the absence of any fixed infrastructure are very limited (e.g., satellite communications will be ubiquitous). Nevertheless, other reasons (such as the need of limiting energy consumption or the high costs possibly implied in interacting with a fixed infrastructure) may suggest minimizing in any case the exploitation of the services of the fixed infrastructure. The problem, in that case, is that the absence of the fixed infrastructure makes it hard to find a place where to actually allocate coordination media.

The basic model will not substantially different from the application developers' point of view. However, due to the lacking a physical network infrastructure, the coordination media via which agents have to interact and in which coordination laws reside becomes only a *virtual medium*, implemented in a distributed way and enforcing coordination laws too in a distributed way. The idea is to have a group of agents interacting in the context of a mobile ad-hoc network being somehow "attached" to a software substrate capable of influencing their coordination activities and, thus, of enacting specific coordination laws. When an agent arrives within a MANET, i.e., get connected to a new group and to the corresponding virtual coordination medium (see Figure 2), the code implementing to the coordination laws and the software substrate needed to enact them is dynamically uploaded locally to the agent and attached to him to filter all its coordination activities. Such a substrate, again, can exploit mobile code technology to transfer coordination laws from an agent to another one. The substrate, in itself, can be implemented and attached to the agents using different techniques, such as aspect-oriented or reflective programming, or it can be an autonomous software component in its turn, acting as a mediator between the agent and the network.

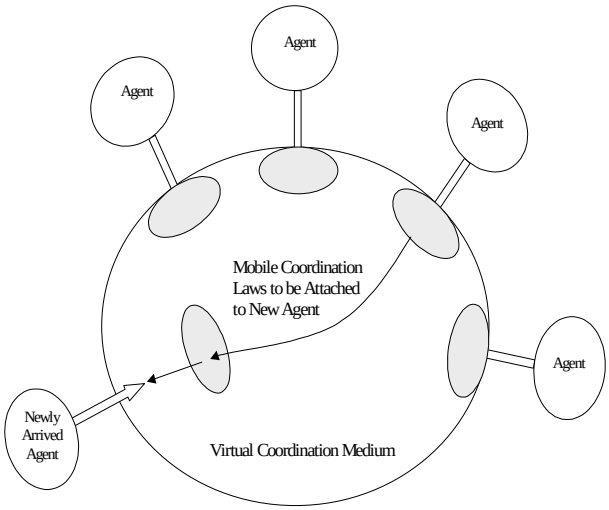

Figure 2. Programmable Coordination Infrastructure for a MANET

Again, the type of interaction model that one may wish to adopt for the virtual interaction media may strongly influence the complexity and the efficiency of the implementation. A peer-to-peer interaction model it is likely to limit the state size of the virtual interaction medium, and this it is like to reduce the problem of maintaining the consistency of the state of the medium and of properly enacting the coordination laws. However, as MANET typically defines a very dynamic scenario, in which agent can connect and disconnect very frequently from a group, relying on direct peer-to-peer agent interactions may be problematic. Conversely, a data-oriented model such as a tuple-based one, by fully uncoupling interacting entities, alleviates the problem related to the dynamicity of agents' connection and disconnection. However, it introduces the problem of maintaining the consistency of a possibly very

large state size of the coordination medium. In between, the adoption of an event-based model, exploiting a global data status of limited size, may achieve a good trade-off between the two issues of uncoupling and state consistency.

An additional issue that arises in MANET is the one related to the no longer sharp distinction between "local coordination laws" and "applications-specific coordination laws". However, the analysis of this issue would require investigating the possible application scenarios of MANET, and it is thus outside the scope of this short position paper.

4 Related Works

Although there is no room to discuss in details all the systems and model that somehow has relationships with our work, it is at least mentioning a few relevant systems and models.

MARS [1] is the coordination infrastructure that we have implemented within our research group and it the one whose most closely maps the concepts we have presented. Although adopting different implementation choices, similar considerations can apply to the TuCSoN model [6], developed in the context of an affiliation research project.

LIME [8] defines an interesting and peculiar tuple-based architecture for handling in a uniform way both physical and actual agent mobility, also in the context of MANETs. It integrates useful forms of reactivity that, however, does not reach the full programmability required for context-dependent coordination.

LGI [5] defines a model for controlling the interaction in a group of agents interacting in a peer-to-peer way. The programming model closely resembles our approach, but it does not explicitly take into account mobility.

Other architectures such as Jedi, Jini, and T Spaces, defines suitable middleware to handle interactions in the presence of mobility that, to different extents, integrate concepts and adopt approaches related to context-dependent coordination.

As a final note, it is worth noting that concepts somehow related to the ones of context-dependent coordination and of programmable coordination infrastructure can be found in completely different research areas, such as swarm-based computing [7] and organizational computing [4].

5 Towards a General Model

Context-dependent coordination promotes a shift of focus from "engineering components" to "engineering interaction contexts". As a consequence, for the approach to be effective and practically usable there is need for models of programmable interaction contexts, enabling a suitable and engineered approach to the definition and verification of coordination laws, prior to their actual coding in a specific coordination infrastructure.

Unlike other computational models, the model should abstract away from internal details about the behavior of agents, e.g., of the entities that are intended to be coordinated by the model. Instead, it only has to focus on the *observable behavior* of components: no matter what agents internally do, what matters is that coordination laws can be effectively modeled and their properties verified. In another work [2], we have shown several simple example of dynamic and harmless composition of coordination laws in our MARS tuple-based programmable coordination infrastructure. However, in general, programming the behavior of the interaction space via complex compositions of coordination laws may be very difficult to be handled. This makes the definition of a formal model and of the associated operational semantics of basic importance.

We are currently completing the definition of a general model for programmable interaction context and of its operational semantics. The model relies on an *event-based kernel*, which models interactions in terms of generation of events, subscriptions to events, and reaction to events. Programmability of the kernel is modeled in terms of programmable reactions to events and to subscriptions. In reactions, the kernel can access to and modify a distributed state space. At a *higher-level*, the basic kernel functionality can be used to realize and needed interaction model, e.g., message-based or tuple-based, by providing a specific interface to agent and a specific implementation of the kernel state.

We intend to use it to model and analyze the properties of different coordination infrastructures, there included MARS, TuCSoN, as well as different proof-of-concepts infrastructures. In addition, we will model different possible use-cases coordination laws enacting specific widely used coordination patterns or security policies. Finally, we intend to use the model as a solid foundation toward an event-based extension of the MARS coordination infrastructure in the context of MANETs.

Selected References

1. G. Cabri, L. Leonardi, F. Zambonelli, "MARS: a Programmable Coordinaiton Architecture for Mobile Agents", IEEE IC, 4(4), 2000.
2. G. Cabri, L. Leonardi, F. Zambonelli, "Engineering Mobile Agent Applications via Context-Dependent Coordination", 23rd International Conference on Software Engineering, May 2001.
3. A. Fuggetta, G. Picco, G. Vigna, "Understanding Code Mobility", IEEE TSE, 24(5), May 1998.
4. N. R. Jennings, "On Agent-Based Software Engineering", Art. Int., 117(2), 2000.
5. N.H. Minsky, V. Ungureanu, "Law-Governed Interaction: A Coordination & Control Mechanism for Heterogeneous Distributed Systems", ACM TOSEM, 9(3), 2000.
6. A. Omicini, F. Zambonelli, "Coordination for Internet Application Development", JAAMAS, 2(3), Sept. 1999.
7. V. Parunak, et al., "Distinguishing Environmental and Agent Dynamics", 1st ESAW Workshop, LNCS, No. 1972, 2000.
8. G.P. Picco, A.M. Murphy, G.-C. Roman, "LIME: Linda Meets Mobility", 21st International Conference on Software Engineering, May 1999.