A Coordination Model for Ubiquitous Computing

AMINE TAFAT-BOUZID, MICHELE COURANT, BEAT HIRSBRUNNER
Department of Informatics
University of Fribourg
Chemin du Musée 3 CH 1700 Fribourg
SWITZERLAND

Abstract: -This paper presents a coordination model called XCM, intended to conceptualise coordination within ubiquitous computing, and one instantiation of this model called UCM. The constraints brought up to coordination by ubiquitous computing are the need to face immersion within physical environments, a very strong heterogeneity of software and hardware components, and a very high dynamicity and context-sensitivity of applications. XCM is an generic model organized around a few abstract concepts (*entity*, *environment*, *social law* and *port*) addressing coordination basically in terms of proactive and reactive contextual behaviours. The UCM instantiation allows us to show how XCM concepts can be applied, and to discuss the expression power gained through XCM high level of abstraction.

Key-Words: Distributed computing, Ubiquitous systems, Coordination, Service, Composition.

1 Introduction

The ubiquitous computing field denotes a new dimension of computing induced by the constant miniaturization of electronic components, entailing their massive spreading in everyday objects, combined with the generalization of mobile communication over professional and personal environments.

For software design and management, the characteristics of this domain are: (1) immersion of computing resources in the physical world and mobility (2) high heterogeneity of devices and software components, and (3) very high dynamicity and context-sensitiveness of applications. From a coordination perspective, this moves the focus on to the context-sensitiveness of the manipulated entities, both from reactive and proactive points of view, and on to the mobility management. For tackling this problem, which marks a complexity increase in interoperability management regarding classical computing, our proposal is to make a step forward towards genericity and abstraction in software component coordination.

The here presented work is then a generic coordination model called XCM[1] built around a few abstract concepts allowing a homogeneous management of heterogeneous context-sensitive and potentially mobile interacting entities. Its contribution consists namely in an explicit management of the environment, and a very flexible approach of communication.

The present paper is organized as follows. Section 2, describes the abstract model XCM, while section 3 illustrates the introduced concepts through a paradigmatic example. Section 4 presents an instantiation of XCM called UCM, which is part of a Ubiquitous Computing middleware. This allows us to detail how abstract concepts may be interpreted within a given framework, and which expression power is made accessible by the high level of abstraction of XCM. The conclusion summarizes the main features of the model, and the main arguments in favour of the solution proposed for encompassing ubiquitous computing requirements within coordination models.

[1] XCM for X Coordination Model

2 X Coordination Model (XCM)

XCM is a coordination model intended to support the specificity of a ubiquitous application. Its essential characteristics are:

- Genericity, which is obtained through a high level of abstraction based on the notion of *entity* –and *agent*–;
- A capacity to handle the dynamics of ubiquitous execution environments –either they are physical or virtual–, and the context-sensitivity of applications, thanks to the explicit notion of *environment;*
- A homogeneous management of the contextual dynamics of components by the unique formalism of *social law* attached to the notion of environment, and a mechanism of *port* allowing entities to interact both very flexibly and powerfully.

As a coordination model, XCM comes within P. Ciancarini's approach [1], and the vision of coordination proposed by T. Malone [2], while prolonging an experience of coordination platform development we had previously carried on [3]. Within this approach, it however adds on a theoretical component inspired by « autopoiesis » i.e. the modelling of living systems elaborated by F. Varela and H. Maturana [4]. The interest of this heritage is double. First, it allows profiting from the specificity of the physical space for modelling mechanisms like the construction and the maintenance of organism frontiers. Second, it introduces a fundamental distinction between organisation (domain of control expression) and structure (domain of entity existence).

2.1 Entity and agent

Everything is an entity in XCM. An entity e_i is defined by its *structure*, which is expressed as a recursive composition of entities $e_{i1}…e_{in}$ –called components of e_i– and by its *organisation*.

An entity, whose structure can not be decomposed, is called *atomic;* it denotes a pre-constructed element of the system. Oppositely, the highest-level entity recursively containing all the other entities of the system, is called the *universe* of the system.

The *organisation* of an entity e_i specifies the rules, which are governing the assembling of components in the structure, and their dynamics. It then characterizes the domain of the interactions, which are applying to e_i. It is expressed as a set of rules called by extension[1] the *social laws* of e_i.

2.2 Environment and social laws

At a given moment of the existence of the system, every entity e_i –except the universe– therefore exists as a component of another entity e. This entity, which contains it is called its *environment*.

Thanks to its social laws, the environment e prescribes the structure and the dynamics of e_i. These ones determine in particular the interactions between e_i and e, as well as between e_i and the e_j – i.e. they rule out the assembling and disassembling of e_i with the other components of e–. These laws also govern the input of e_i into e, and the output of e_i from e. Let us for example consider the case of an antenna: its environment is its coverage area, and the entering (respectively leaving) of mobile devices into (respectively from) it is controlled by its social laws.

When its social laws confer to an entity the capacity to initiate operations modifying its own structure (internal autonomy) or its relations with its environment (external autonomy), this entity is commonly called an *agent*[2].

An entity can be tight to several environments. However, due to the enrooting of «ubiquitous» entities in the physical space, the Pauli's principle applies, this means it can be active at the most in one environment. Apart from this environment, it can be at the most «virtually» or «sensorially» present in the other environments (cf. § 2.3).

The notion of environment then encompasses within a single concept all the semantic diversity of the ubiquitous application components: a social semantics, inherited from coordination in general, and a physical semantics of entities, which becomes essential as soon as the entities are evolving onto –for example mobile– devices subjected to the laws of the physical space.

---

[1] i.e. independently from the fact that they are governing a physical, or a virtual space.

[2] We do not develop here in formal terms the notion of autonomy, which is relying behind this terminological distinction. Note only that "agent" refers to a certain behavioural indeterminism of the entity when it is perceived by another entity. The term "agent" is then used here as a shortcut, intended to reveal the heterogeneity supported by XCM. For more details on the theoretical underlying questions, see our previous article [5].

By social semantics, we mean for example the capacity of an entity to belong to a social structure, such as a group of entities it is interacting with (typically a person belongs to a group of persons with which it is presently in meeting). The model supports a multiple organisational linking of an entity, the equivalent of multiple heritage in object systems (typically a person is linked to the football club it is member of, but also and mainly to the company in the name of which it is predominantly acting during the meeting).
By physical semantics, we namely mean the impossibility for an agent to act in two environments at the same time, or to be «teleported» from one environment to another[1].
An entity can however remain « aware of » another environment than the one in which it is active. As we will see further, it can open some specific communication channels in this environment, thus implementing a remote perception mechanism. It can also create dedicated entities and « send » them to others environments for achieving certain tasks. These entities may act as « avatars » of itself, thus providing a mean for an entity to be « virtually» present out of its basic environment. However, entity and avatar remain distinct entities.
Finally, the same way the «autopoietic dynamics» rules in Varela & Maturana include meta-control and self-control, social laws may also govern the structure and the organisation visibility along the entangled entity and environment hierarchies.

2.3 Ports

A *port* is a special type of entity dedicated to communication between entities. A port p has the specificity to be generally active while being coupled to an agent a_i, which is the port's master. The coupling between a_i and p is obtained through a special type of composition called *interface*, which is therefore specified by social laws (of p and a, and of their common environment). These ones define how the port is assembled to its master, for example *maintained* versus *not maintained by master's movement*, linked by *ownership*, or by *usage*, etc. They may also define the modalities of using the port (in terms of communication protocol, of bandwidth, etc). For answering ubiquitous computing needs, we also distinguish *removable* and *irremovable* ports.

> *Example: For a human agent, a mobile phone is a removable port, whereas an audio-prosthesis is irremovable. A pair of glasses is somewhere in between, obeying to coupling laws, that are stronger than the phone's ones, but looser than the prosthesis ones.*

An agent a may be coupled to several ports. It can acquire ports, and dissociate itself from ports dynamically. The agent-port assembling and disassembling procedures are triggered either explicitly, by an initiative of a, or implicitly by the entrance of a into a new environment**,** or by the environment dynamics, which may for example welcome new ports, which are automatically coupled to a.
The notion of port is then a fundamental mechanism, which confers to XCM the ability to coordinate context-sensitive entities. This context-awareness is the central characteristics of application components in ubiquitous computing.

3 Ubiquitous Coordination Model (UCM)

While the previous section has introduced XCM, a generic coordination model oriented towards ubiquitous computing, we will now describe an instantiation of this model called Ubiquitous Coordination Model (UCM). UCM is designed as a coordination layer working over a service-oriented layer for forming a ubiquitous computing middleware called UbiDev [6][8].

3.1 UbiDev Middleware

UbiDev is a lightweight middleware aiming at ubiquitous computing scenario handling, and providing at application level a host independent interface of the underlying service-based system

[1] In addition the already cited Pauli's principle, the usual laws of physics are supposed valid, quantum relativist physics being excluded. The Heisenberg's uncertainty principle (present/not present) hence is eliminated from the model, as well as the possibility of instantaneous transfer from one environment to another (teleportation), which would violate the connexity constraints between environments dictated by the space topology.

[6][7][8]. It takes place in classical layer architecture, organized in four levels:
The *physical layer* manages the resources available in the application environment, and provides a uniform access protocol to these resources.
The *service layer* is responsible for the service management within the application environment. It provides service descriptions called *capsules* to the coordination layer, in order to hide heterogeneity of components at the underlying level. A capsule is an instantiated service, which is wrapped in a dedicated execution environment. It is an action, which takes a resource as an input, and produces a resource as an output.
The *coordination layer* is in charge of coordinating services according to context-dependent rules governing their composition and interactions, i.e. their coupling, mobility, and communication.
At the *application layer*, we then find the whole structure in which homogeneous entities encapsulating services are coordinated.

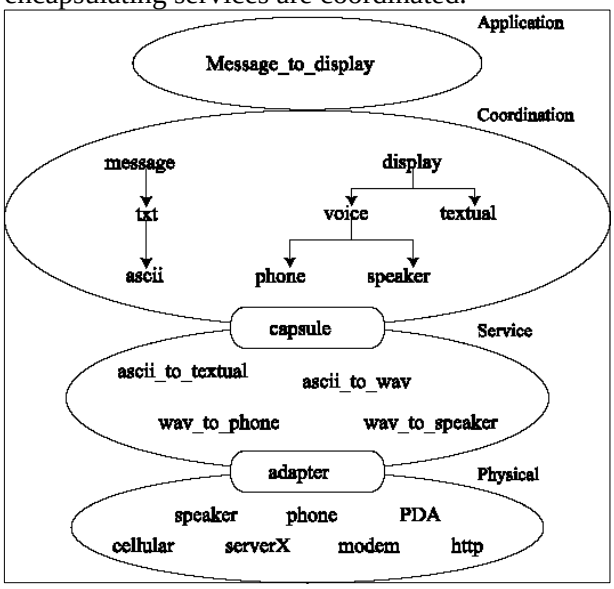

Fig.2: UbiDev description of a ubiquitous communicator

3.2 A UbiDev example

In order to get more familiar with the UbiDev context, we will now consider a quite trivial example of «ubiquitous communicator» defined as a broadcast messaging service tool in UbiDev (see figure 2). At the service layer, UbiDev provides basic services like ascii_to_wav, ascii_to_textual, wav_to_phone, wav_to_speaker (see *Service layer* in figure2). Suppose now that in the framework of a broadcasting application, a user wishes to execute the message_to_display service (see *Application layer* in figure 2). The execution of this request implies to coordinate the available basic services. In this case, coordination consists in a service composition, which aims to find out an execution path for the requested service, starting from the basic services. Such a composition is governed by the social rules attached to the user's environment.

3.3 UCM

The UCM model has been designed as an instantiation of XCM for the UbiDev middleware. This is how the generic concepts of XCM may be reinterpreted in UCM:

- Entity / agent: these universal concepts in XCM naturally correspond to the basic notion of capsule provided as interface by the service layer in UbiDev. Capsules are XCM atomic entities.
- Environment: it is an execution environment, like the atomic entity's one corresponding to capsules, or the user context.
- Port: it is a capsule input/output port in UbiDev.
- Social laws: they are matching rules specifying how to combine capsules in UbiDev, in order to form higher level structures through service composition.

For the «ubiquitous communicator» example considered in section 4.1, this would then give us three UCM entities: ascii_to_wav, wav_to_voice, voice_to_phone, with the ports: ascii_in, wav_in and out_wav, voice_in and out_voice, and out_phone. The environment of these entities is the user's context, and its social rules specify that entity composition is obtained through coupling of same type in and out ports (cf figure 3).

3.4 Implementation

The UCM instantiation has been developed within the UbiDev platform for validating the XCM model trought WSDL, SOAP, UDDI. It allowed us to test the expression power of the XCM generic concepts, and especially how the explicit management of the environment can be used for service composition.
In addition to the methodological benefits brought up by a higher level of abstraction regarding

system design, the interest of XCM –and of having derived UCM from XCM– could however not be shown through the trivial example previously selected for didactic reasons. This interest is actually a significant increase in interoperability obtained through software composition and component mobility management, allowing to face the heterogeneous environments, the variable conditions, and the mobility imposed by ubiquitous computing.

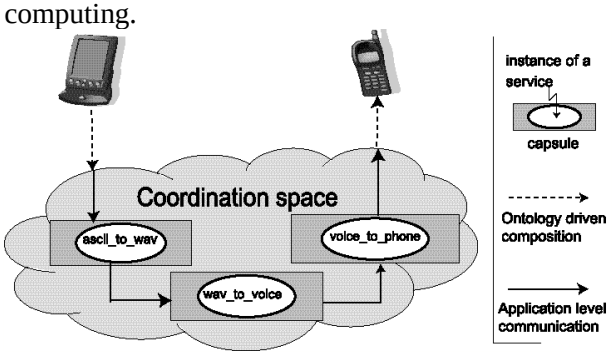

Fig.3: UCM, coordination in UbiDev.

4 Conclusion

A model of generic coordination oriented towards ubiquitous computing called XCM has been developed, and then instantiated within the ubiquitous computing dedicated environment UbiDev.

Through some abstract concepts *–entity*, *environment*, *social laws* and *port–* XCM takes place in a layer architecture allowing to apprehend in a conceptually simple and homogeneous way the diversity and the dynamics of ubiquitous application universes. It integrates in particular the immersion of the application components within the physical universe, and the context-sensitiveness required by the ubiquitous applications. The abstract model for which a coordination methodology can now be developed constitutes a generic middleware suitable for relieving the applications from the coordination tasks resulting from their ubiquitous character, while remaining open downwards (towards the service level and the physical level) and upwards (towards the application level). So doing, it provides interoperability regarding devices, software components, and platforms, together with full control and context-awareness at the application level

References

[1] P. Ciancarini, F. Arbab and C. Hankin: Coordination languages for parallel programming. Parallel Computing, 24 (7):989-1004, 1998.

[2] T.W. Malone and K. Crowston: The interdisciplinary Study of Coordination. ACM Computing Surveys, 26 (1): 87-119, March 1994.

[3] M. Schumacher: Objective Coordination in Multi Agent Systems Engineering. Springer Verlag. LNAI 2039, 2001. (also published as PhD Thesis, Dept of Informatics, University of Fribourg, Suisse).

[4] F. Varela and H. Maturana. Autopoiesis and Cognition: The realization of the Living. Boston Studies in the Philosophy of Science in Cohen, Robert S., and Marx W. Wartofsky (eds.) , Vol. 42, Dordecht (Holland): D. Reidel Publishing Co., 1980.

[5] M. Courant, B. Hirsbrunner and K. Stoffel: Managing Entities for an Autonomous Behaviour. In Nadia Magnenat Thalmann and Daniel Thalmann (eds): “Artificial Life and Virtual Reality”, John Wiley & Sons, 1994, 83-95.

[6] S. Maffioletti and B. Hirsbrunner. UbiDev: an homogeneous environment for ubiquitous interactive devices. In Short Paper in Pervasive 2002 - International Conference on Pervasive Computing, Zurich, Switzerland, August 2002.

[7] S. Schubiger. Automatic Software Configuration, PhD Thesis, Dept of Informatics, Univ. of Fribourg, Switzerland.

[8] S.Maffioletti, B. Hirsbrunner. Towards a Homogeneous Coordination Space for Ubiquitous Interacting Entities. Submitted to Smart Object Conference (SOC’03), Grenoble, France, May 15-17, 2003.