

Logical versus physical programming for ubiquitous applications

Julien Pauty¹, Michel Banâtre², and Paul Couderc²

¹IRISA, Université de Rennes 1
Rennes, France
Julien.Pauty@irisa.fr

²INRIA
Rennes, France
{Michel.Banatre,Paul.Couderc}@irisa.fr

Abstract — *Ubiquitous computing provides services to users, according to their current situation. Interactions with such programs are as implicit as possible. We find among the applications the electronic supermarkets or the virtual guides for museums' visitors.*

The majority of ubiquitous applications use a digital model of the physical world to compute the delivered services. With our approach the computations are done directly in the physical space. There's no need to define a digital model of the physical environment. This note presents a physical programming environment dedicated to ubiquitous applications design. It also identifies a problem with the data addressing mode that can't take into account the relative position of the moving entities.

1 Introduction

The goal of ubiquitous computing is to deliver services to users according to their context with minimal interactions [1]. User's context defines information, which is related the physical world, needed by the applications to take decisions [2]. It can be the user's position, the time or the weather. The virtual guides for museums' visitors are one application of ubiquitous computing. For this kind of applications, the context is usually limited to the user's position and eventually his orientation. The application provides information about what the visitor is looking, according to his position.

We can identify two approaches for ubiquitous applications implementation: the logical one and the physical one. To compare them we can make an analogy with the “interpreted versus compiled” solution for programming languages. For example with Java, an interpreted program can be executed on every computer that provides a Java interpreter, whereas a compiled one can be executed only on the targeted architecture but with a greater speed. The main requirement to implement an application with the logical approach is to have a model of the physical environment where the application is executed.

We can always build such model, so the logical approach can be used for every application but it leads to complicated programs for complex situations. On the other hand, the physical one is not fitted for all applications but gives simpler programs.

Section 2 discusses the two approaches to program ubiquitous applications. Section 3 presents spatial programming. We propose an enhanced addressing mode for spatial programming in section 4. Section 5 is dedicated to the related works.

2 From the logical to the physical approach

We present two approaches for ubiquitous applications programming. An ubiquitous application is linked to the physical world. The physical entities implied in the application can be places, physical objects or people. We call them physical objects, or more simply objects.

Logical approach The logical approach relies on a representation of the physical world, called digital model. It is managed by a machine that we call services platform. The digital model is used to deliver services to users according to their context. If we take the example of the virtual museum guide (see Figure 1(a)), this model can be a database where the information is associated to physical positions [3]. The user's coordinates are sent to the database to get data according to the user's position.

Scalability and complexity are the main drawbacks of this approach. Firstly, the model must be up to date, so the more dynamic a system is, the more numerous model updates are. The services platform is a potential bottleneck if it must deliver services to all users, and if the model updates are numerous. Modeling the physical environment is always possible, nevertheless some models lead to extensive computations, like images processing.

Physical approach The physical approach doesn't rely on a digital model of the physical world. The service delivery is computed where the user is. This is done by spreading data and wireless computing devices directly in the physical environment, on the physical objects implied in the application. Each device manages and stores the data of the corresponding object. In this way, data is physically linked to objects. For example, with this approach, there is no need to update a positions database when physical objects move since the data *physically* moves with them.

With the physical approach computations are done by the embedded devices. The devices interact when they are connected. The interactions' goal is to deliver the service to the user. The user context is represented by the set of objects connected to his device. If we come back to the museum example, the data is directly embedded on the paintings. When the visitor's guide is connected to the paintings' device, it receives the information and display it (see Figure 1(b)).

With the logical approach, the application can be seen as an outer observer who sees all the objects. With the physical approach the application is distributed over the objects. Since the network interfaces have a limited range, each object has a partial view of the global situation. In this way an object can interact only with the proximate objects, applications that must see all the objects are not programmable with this approach.

Figure 1 illustrates both approaches with the museum example. For the logical approach, the model of the physical environment is an array that associates paintings’ coordinates to artists’ names. With the physical approach the names are directly embedded on the paintings. The user receives the name from the painting connected to his guide.

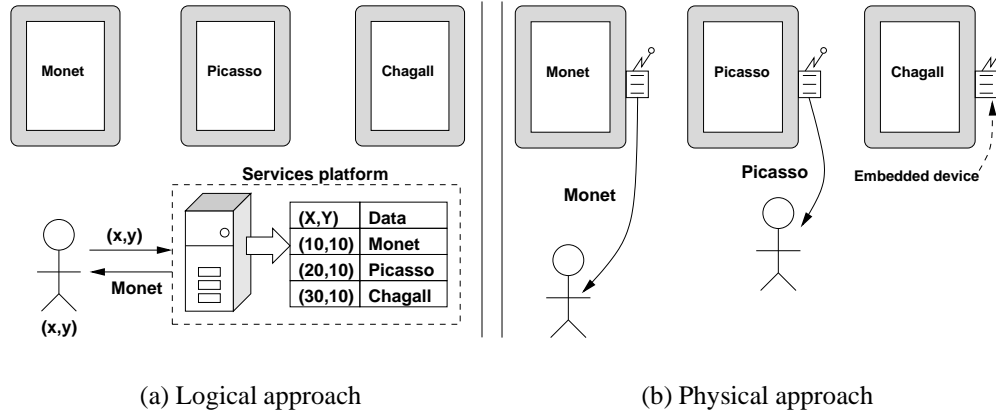


Figure 1: The logical versus the physical approach

3 Spatial programming

Spatial application We call spatial application an application whose data is related to the physical place. Application’s execution flow is driven by spatial conditions ; in the program we find statements such as “if the user is in front of a painting then display the artist’s biography” or “if the user enters his office play a welcome message”. Many ubiquitous applications are spatial applications.

Spatial applications can be programmed with either the physical or the logical approach. Now, we present an environment to build spatial applications with the physical approach.

Spatial and physical programming Spread [5] is a programming environment dedicated to spatial programming with the physical approach. With Spread, the data is physically attached to the objects of an application and moves with them. Each data has a physical shape that defines the area where it is accessible. A process can access to a data if it’s inside the data’s area.

Each implied object executes at least one process of the application. The data manipulated by a process is addressed and accessed in the physical space. Processes’ memory is the physical space. A process can read a value from the memory and/or add a new data inside the memory. A process can delete only its data. The visible data space for a process changes when it moves (see Figure 2). The read access is a blocking operation, processes stay blocked until they are in the area of the expected data. Data is “recognized” by its type that can be a simple type like a C integer, or composed type like a C structure. The blocking aspect of the read accesses is used to synchronize the execution of the processes on the position of the corresponding objects.

Implementation Spread embeds on the objects of the application a computing device with a wireless interface. The wireless cards have a spherical range. For simplicity reasons, Spread uses this sphere for the data's shape, and doesn't propose any other. The detection of a data is based on the network connectivity. If two processes can connect each other, each is inside the area of the other. In this way, the rule for data access becomes: a process can access to its data and the data of the connected processes. On the Figure 2 we have a moving process and three static processes. The grey boxes contain the visible data for the moving process at two different positions.

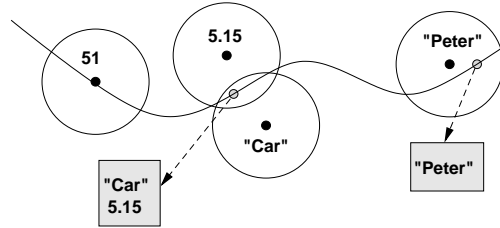


Figure 2: Accessible data by a moving process

When an process A accesses a data of the process B , we just know that A is inside B 's area and that B is inside A 's area. We can't know what is the relative position of B compared to A . For example, A can't know if B is on the left or the right.

Currently the shape of the data is fixed. For example, with WiFi interfaces, the shape is a 100 meters sphere. This is unfitted for many applications, like the virtual guides. If the guide displays all the information of the connected processes, it will display information on works that the visitor doesn't see. For this application, the data should occupy a rectangular area in front of the paintings.

In the next section we propose a finer addressing mode for spatial programming that solves these two problems.

4 Extension of the spatial programming model

Targeted applications We present here an application that illustrates the requirements of the new addressing mode. We want to embed traffic lights directly inside the vehicle's cockpit. This can be useful in protected environments like an airport where there is no pedestrian. We can embed traffic lights on the vehicles that transport the luggage.

For this application, we need to define rectangular areas which overlap the road. These areas are the shape of the data that represents the light's color. The application also defines a data that overlaps the entire crossroad. The cars indicate their presence by a spherical data. The Figure 3 illustrates the application with a four ways crossroad with two cars.

The application is divided into processes that are distributed on the cars and the crossroad. The cars detect the crossroad with an instruction like `read ("crossroad", front)`. The crossroad sets the lights' color, according to the cars' positions, with an instruction like `write ("green", way4)`.

Finer addressing mode With the new addressing mode, the applications defines the data's shape when it is added inside the spatial memory (the physical space). The defined

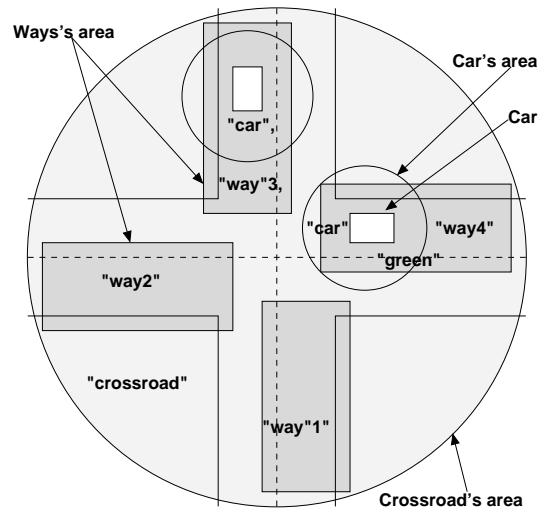


Figure 3: The crossroad example

data can have any shape. This shape is unrelated to the wireless interface. In the case of the crossroad, we have rectangular and circular data. This shape is defined relatively to the process position. For example, with this system a process could define a data that occupy a cube located ten meters on its left.

Currently, when an processes requests a data, the only information it has about the data position is that the returned data is inside its communication range. This is insufficient for the targeted applications. Now, the processes define precisely the area where the retrieved data must be. These areas are defined relatively to the processes' position. If we have a pending request on a moving process, the area associated with the request moves with it.

The main requirement to implement this addressing mode is to have a simple way to get the relative positions of the processes.

5 Related works

Spatial database [6] and the Geographic Information System (GIS) are two components used to program spatial applications with the logical approach. Spatial databases organize data with spatial properties. These spatial properties are used in the requests to retrieve the corresponding data. The logical approach for the museum guide can use a spatial database to store its data. The Geographic Information System (GIS) links data of any type to physical locations. For example, a GIS, which records all the places where you can eat in a town, can be queried to get the nearest Chinese restaurant.

Our work is related to all ubiquitous applications in general, like Cooltown [7] or Cyberguide [8]. The founder applications of ubiquitous computing like ParcTab [1] come from the Xerox PARC. This tabs are palm sized devices that were used for several applications like mail checking, users mapping or electronic notes.

To our knowledge, our approach, based on spatial programming, is innovative. Almost all ubiquitous applications deal with spatial data but they are not programmed with a physical programming model. Lime [9] is close to our work in the implementation point

of view, nevertheless Lime doesn't rely on any spatial notion, it is agents based.

6 Conclusion and future works

Via applications that we have already implemented, Spread has shown its advantages for programs with local interactions. One of them is an application to help partially blind people to take the public transports.

With an application like the traffic lights manager we have identified new requirements for the Spread's addressing mode. The new addressing mode permits to access the memory by specifying the area where the requested data must be. Moreover, each data has its own physical shape. Starting from this point, we can consider that the data's type is composed of this shape plus the classical type (int, float...).

Our future works will be dedicated to further study the geometric aspects of the data addressing. Location systems and geometric calculations will get special attentions. Then, we will start the development of a prototype that will be tested with real applications.

References

- [1] M. Weiser. Some Computer Science Issues in Ubiquitous Computing. *Communications of the ACM, Back to the Real World, Special issue on Computer Augmented Environments*, 36(7):75–84, 1993.
- [2] G. Chen and D. Kotz. A Survey of Context-Aware Mobile Computing Research. Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College, November 2000.
- [3] I. MacColl, D. Millard, C. Randell, and A. Steed. Shared Visiting in EQUATOR City. In *Proceedings of the 4th international conference on Collaborative virtual environments*, pages 88–94. ACM Press, 2002.
- [4] G. Kollios, D. Gunopulos, and V. J. Tsotras. On Indexing Mobile Objects. In *Proceedings of the Eighteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 31 - June 2, 1999, Philadelphia, Pennsylvania*, pages 261–272. ACM Press, 1999.
- [5] P. Couderc and M. Banâtre. Ambient computing applications: an experience with the SPREAD approach. In *36th Annual Hawaii International Conference on System Sciences (HICSS'03)*, 2003.
- [6] R. Hartmut Güting. An introduction to spatial database systems. *The VLDB Journal - The International Journal on Very Large Data Bases*, 3(4):357–399, 1994.
- [7] T. Kindberg, J. Barton, J. Morgan, G. Becker, D. Caswell, P. Debaty, G. Gopal, M. Frid, V. Krishnan, H. Morris, J. Schettino, B. Serra, and M. Spasojevic. People, Places, Things: Web Presence for the Real World, 2000.
- [8] G. Abowd, C. Atkeson, J. Hong, S. Long, R. Kooper, and M. Pinkerton. Cyberguide: A mobile context-aware tour guide, 1997.
- [9] A. Murphy, G. Picco, and G.-C. Roman. Lime: A Middleware for Physical and Logical Mobility. In *International Conference on Distributed Computing System*, pages 524–536, 2001.