Mobile Agents for Mobile Games

Eduardo Mena
Dept. of Computer Science & System Engineering
University of Zaragoza
Spain
emena@unizar.es

Carlos Bobed
Dept. of Computer Science & System Engineering
University of Zaragoza
Spain
cbobed@unizar.es

ABSTRACT

Mobile agents have been with us for a quite long time, showing their usefulness when it comes to developing distributed information systems. In particular, their dynamism, mobility, and adaptivity have been shown to be especially well-suited to deal with different problems that arise in mobile computing scenarios: the need for efficient access to heterogeneous and distributed data sources, dynamic load balancing, robustness in unstable connections and communication failures, etc. Indeed, developing mobile games using mobile agents technology would benefit from these characteristics in terms of infrastructures and on-demand deployment, but this is not the only benefit they could obtain from it.

In this paper, we advocate using the paradigm of software mobile agents to enhance mobile games design. By exploiting their mobility, context adaptivity, autonomy, and, last but not least, their intuitive design paradigm, game developers are provided with an alternative approach to design games that otherwise might be impossible to implement in an efficient manner. We illustrate this potential using a sample mobile game.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems, Intelligent agents*

General Terms

Novel game types

Keywords

Mobile agents, architectural design for next generation mobile games, interaction among devices in the vicinity

1. INTRODUCTION

Since mid 1990's we have the possibility of designing autonomous software modules which can *move* across computers. We are talking about the mobile agent technology. Many different mobile agent platforms (the software that supports agents and hides low level communication details) were created during the next years as this technology became a new design paradigm for distributed information systems: when remote data must be accessed there exists a second choice beyond installing a remote server and making a call from a client (i.e., the client/server approach); the new choice is to send a mobile agent to the remote computer to process the data locally. Mobile agent technology showed its importance in the arising area of mobile computing, as the autonomy of mobile agents deals perfectly with the unstable wireless communications. However the potential of this technology has not been exploited in other areas.

In this work we advocate the use of mobile agent technology for mobile games of different kind. Mobile agents may play a key role as 1) they autonomously move the processing tasks wherever they are needed, 2) they are able to track the relevant data and computers/devices involved in a certain task, 3) they adapt themselves to changes in their execution environment, and 4) they provide an easy way to bring new behaviors to game arenas without the need of global updates.

In the following, we summarize in Section 2 the mobile agent technology. In Section 3 we present the benefits of using mobile agents in mobile games, and we introduce the application of mobile agent paradigm to a sample video game. Section 4 includes some related work. Finally, in Section 5 we present some conclusions and future work.

2. MOBILE AGENTS

Mobile agents [4] are programs that execute in contexts called *places* and can autonomously pause their execution, travel from a place to another place on another device, and resume their execution there. Thus they are not bounded to the device where they were created. Those execution places where agents *live* are provided by a certain *mobile agent platform* [10], which is a software that has to be present on devices where agents should execute. Mobile agent platforms provide mobile agents with different services (such as a transportation/mobility service, a communication service, and a security service) that facilitate the development of distributed applications. One particularly useful service is *location transparency*, which allow mobile agents to easily communicate with other agents independently of the devices where they are executing at that moment [12]. Thus we can design multiagent systems where several agents interact and cooperate, both in a centralized or decentralized way.

MobiGames'15, May 19, 2015, Florence, Italy.

http://dx.doi.org/10.1145/2751496.2751500.

Most of the mobile agent platforms are programmed in Java due to its portable nature. Thus the execution of agents is independent of the underlying device which, in the context of game development, makes cross-device gaming possible.

As pointed out in [4], mobile agents offer many interesting benefits, such as:

1. *They encapsulate tasks.* The mobile agent technology has proved to be useful in the design phases, as it encapsulates many low-level details (communication, synchronization, etc). Thinking in terms of mobile agents eases the complexity of establishing the different tasks and distributing them.

2. *They ease application deployment.* Their ability to travel to remote devices provides a great flexibility to develop and deploy applications in distributed environments. Just one service has to be run on each device, the mobile agent platform.

3. *They reduce the network load and latency.* When it comes to processing data, a mobile agent can travel where the data are and access locally, which optimizes the response time and reduces heavily the amount of not needed data sent over the network. Moreover, the network is used only while the mobile agent moves to another device.

4. *They are asynchronous and autonomous.* A mobile agent does not need to keep contact with its source device while performing its tasks. This is particularly important in mobile environments to save battery consumption in wireless connections.

5. *They adapt dynamically to their environment.* They are able to sense the environment and can be programmed in order to react autonomously and adapt themselves to changes. For example, they could travel to another device when the current device is overloaded, for load balancing.

From a resource consuming point of view, different works [3, 5] show that moving an agent across devices is as efficient as using remote procedure calls (client/server approach). Hence, using mobile agents is as easy as installing a mobile agent platform on every user computer or device: This single service allows user devices to create and host different agents of different nature (even belonging to different games or applications[1]). Without the use of mobile agents every user device must install all the possible software and services that could be needed in different games and applications, which increases the use of resources of user devices.

From a programming point of view, defining a mobile agent is as easy as including some libraries and then a method *moveTo(targetDevice)* will be available to be used in agent coding; the mobile agent platform will make real the *magic*: software programs that pause their execution, move across devices, and resume their execution once there. For example, in Figure 1, we show the code of a simple mobile agent that travels across a list of devices harvesting the users logged on each of those machines. Note how, despite the fact that in the example the agent is provided with the list of devices to visit, the movement decision could be made in each of the visited devices, depending on the context and available resources. Implementing this example without mobile agents would involve to have a dedicated server process in all the possible devices to be visited, offering a service to remote clients that returns the logged on users. If the requested information from the remote devices change, all the servers must be updated. On the contrary, using a mobile agent approach would only require to change the code of the mobile agent (e.g., the line below the comment *Harvest information* in Figure 1). Thus, mobile agents bring computation wherever is needed.

```
public class Traveler extends MobileAgent {
   Device HOME;
   Hashtable <Device,UserList> users;

   public Traveler () {
      this.HOME = getLocalDevice();
      this.users = new Hashtable<Device,UserList> ();
   }

   public void goRecollect (Stack<Device> devices) {
      // Harvest information
      users.put(getLocalContext(),getUsers());
      if (!devices.isEmpty()){
         Device nextDevice = devices.pop();
         moveTo(nextDevice,goRecollect(devices);
      }
      else
         moveTo(HOME,showResults());
   }
   ...
}

public class Launcher {
   public static void main() {
      Traveler t = new Traveler();
      // Prepare a list of devices to visit
      Stack<Device> devices = ...
      t.goRecollect(devices);
   }
}
```

Figure 1: Code excerpt from a sample mobile agent

When talking about agents we usually think of a multiagent system: We can easily create a network of collaborating mobile agents that follow some common strategy or goal, and cooperate among them to achieve it in an efficient manner. Furthermore, a mobile agent can also create in run-time other mobile agents (its own network of helping agents) to help it to achieve its goals.

Summing up, mobile agent technology helps developing more flexible, dynamic, and adaptive systems and can be very useful in distributed system design.

Mobile Agents on Mobile Devices

The interest of making agent technology available for mobile devices is not new: We can find works [8] that measure the feasibility and performance of mobile agents in laptop computers and PDA's[2]. An analysis of the most popular mobile agent platforms and mobile devices identified the requirements and desired features to be used in scenarios with mobile devices [9].

[1]A mobile agent platform may limit the type of agents that creates and receives, for example, to reject unknown agents.

[2]Personal Digital Assistants, precursors of current smartphones and tablets.

In these last years, with those requirements in mind and the increasing processing power and connection capabilities of current mobile devices, some mobile agent platforms (like SPRINGS [12]) have been ported to Android. In the preliminary tests that we have carried out, we have used six well-known Android-based devices[3], with about 40 agents per device continually jumping from one device to another, making remote calls to other agents at different devices, and performing different tasks regarding the processing of distributed mobile queries, and the provision of location-based services. New mobile devices released lately are much more powerful and efficient than those used in the test so it is easy to understand that, despite their lower capabilities with respect to their desktop counterparts, today's mobile devices are very capable to run multiagent systems without compromising their general performance.

3. MOBILE AGENT BENEFITS FOR MOBILE GAMES

In the case of games that execute on a mobile device, mobile agents offer specific benefits apart from the general ones pointed out in the previous section:

- Regarding multi-device games (which can be multi or single-user), they provide an easy-to-use way to extend the game across mobile devices in the vicinity: A mobile agent can travel between two devices by means of a simple sentence *moveTo(targetDevice)* in its code.
- Location-based mobile games can also benefit from this technology as mobile agents could jump from device to device until reaching a certain geographic area to perform some task inside it.
- Game experience can be differentiated and enhanced for each user separately, even allowing advanced users to customize the behavior of non-playable characters (NPC) of the game (always under the strict control of the game platform to avoid cheating). Technical game designers can take advantage of this feature for a myriad of distributed/mobile games.
- Even with single-user and single-device games, the ease of deployment of mobile agents makes it possible to update and extend different parts of the implemented games (e.g., new game elements with modified behavior and rules) without having to update all the game clients.

Thus, mobile agent technology eases the development and deployment of mobile games, especially multi-user/device mobile games. In the following we illustrate these benefits with an example of how the mobile agent technology could be applied to a mobile Real-Time Strategy (RTS) video game.

3.1 Sample scenario

Let us detail the gameplay of a location-based RTS we could want to design for mobile users:

1. *Context*: Players are organized in two or more armies that fight against each other (death matches are also possible). Each player controls a Space Base (SB) and a certain number of mobile space combat units to defend his area or attack enemy units. It is a location-based game as, whenever the player moves, his space base jumps into hyperspace (10m in real world→1 parsec in hyperspace). A space base is able to create a wormhole between it and other space bases within 5–10 parsecs, depending on radiation levels between them (in real world, about 50–100m between players, depending on their P2P connectivity). So players can detect and attack other players in the vicinity, sending their combat units to enemy space areas while both space bases are in range. When a space base jumps into hyperspace all the (friend or enemy) units inside its space area will travel with it.
2. *Units*: Each player can create and deploy a set of mobile combat units of different nature, in our example, destroyer ships (DS), mother ships (MS), fighters (F), and missiles (M), each one with different features (endurance, speed limit, weapons, etc.). Some combat units can launch other combat units (mother ships can launch fighters; any ship or fighter can launch missiles).
3. *Unit control*: Using a GUI, each player can control manually his units both in his space area or in other space areas connected by wormholes. However, any unit is able to behave autonomously if it gets out of range of its player. Other units (e.g., missiles) are controled automatically only.
4. *Unit deployment*: The base of each user resides on his device although some of his mobile units can be deployed to other devices. Units may move to other device, whenever the target device is in range (considering GPS locations of both devices).
5. *Battles*: User devices act also as battlefields where different units (some allied, some enemy) engage in a fight. Units can be damaged or destroyed by means of missile impacts or even crashes with other units. Damaged units can be repaired at allied bases to avoid their destruction.

Thus a player has to physically get near others in order to attack them; in the same way, he can run away from players to avoid attacks... although enemy units already residing on his device will remain there.

3.2 Technical game design using mobile agents

Here we detail how mobile agents provide game designers with a new paradigm to design a game like the described previously. Specifically, mobile agents allow us to implement certain game features just like in the real world. We revisit gameplay items from the previous section to show how it can be implemented following a mobile agent approach.

3.2.1 Context

Each mobile device has a place where different agents can execute, representing the local space area around the player's space base. A static agent representing the player's space base resides on his device; it scans mobile devices in the vicinity to retrieve information from them; the player decides when to attack or physically move to get out of range of dangerous opponents.

[3]Galaxy Nexus, Galaxy Tab 1 & 2, and Nexus 4, 7, & 10.

In Figure 2.a we can see the real location and agents on devices of players Ed and Tim while engaged in a battle; player Jane, ally of Tim, is too far to detect/be detected by the others and her army is undeployed. In Figure 2.(b–d) we show the kind of GUI of each player; in the center we see the current (virtual) location of player's base and his own and enemy units[4]. Players in range are shown by labels (in red for enemies, green for allies) with a blue bar representing their distances to the user. Clicking on another player's label will show the location of each unit at that space area. We can move units to other players' space area while in range.

3.2.2 *Units*

Each unit is implemented as a mobile agent, so it has an autonomous behavior and tries to achieve its goal asynchronously, including movements across user devices (space areas). In Figure 3 we show the different agent classes, along with some attributes and methods, for the sample video game.

The act of "launching" a fighter or a missile is easily implemented: for example, the mobile agent that represents a mother ship creates a new mobile agent of class Fighter or Missile, which since its creation will run autonomously; we so distribute game control across independent mobile agents, in a way very similar to what would happen in real life.

3.2.3 *Unit control*

Here the mobile agent technology provides us with a very important feature: mobile agents representing combat units, although can be controlled manually by the human player using a GUI, include a basic code to control each unit according to their nature (tracking targets or avoiding fights against too powerful armies). Advanced players could *override* this *control* method to customize their units... without having to update the software residing on devices of the rest of players. In other words, we can create customized combat units that behave differently than those from enemy players. For instance, we may create new (more intelligent) subclasses of missile or fighter classes and make them available to be created by our mother ship or destroyer units. Now enemy units will be surprised with our customized and improved combat units. The behavior of ships to face enemy attacks can be customized in the same way. Just think about the open possibilities of this idea easily implemented using mobile agents technology. Game developers could also use this feature to offer new units to players as (free or pay-to-use) upgrades.

To enjoy this very interesting possibility, game programmers must make use of two programming techniques from Java and mobile agent technology: 1) the client program used by the player must be designed using the *Java reflection pattern* to obtain in run-time the kind of combat units (mobile agents) available at each time (the native Java dynamic class loading feature will make the rest of the magic), and 2) the mobile agent technology will make possible to *export* this new functionality (new specialized mobile agents implementing new combat units) to other devices in the middle of an already started game and without the need of updating the software of all the players.

[4]Screenshots and icons in Figures 2 is just a schema of a possible GUI, not a real one. Units (and their agents) are subindexed with its player's name and a unit number.

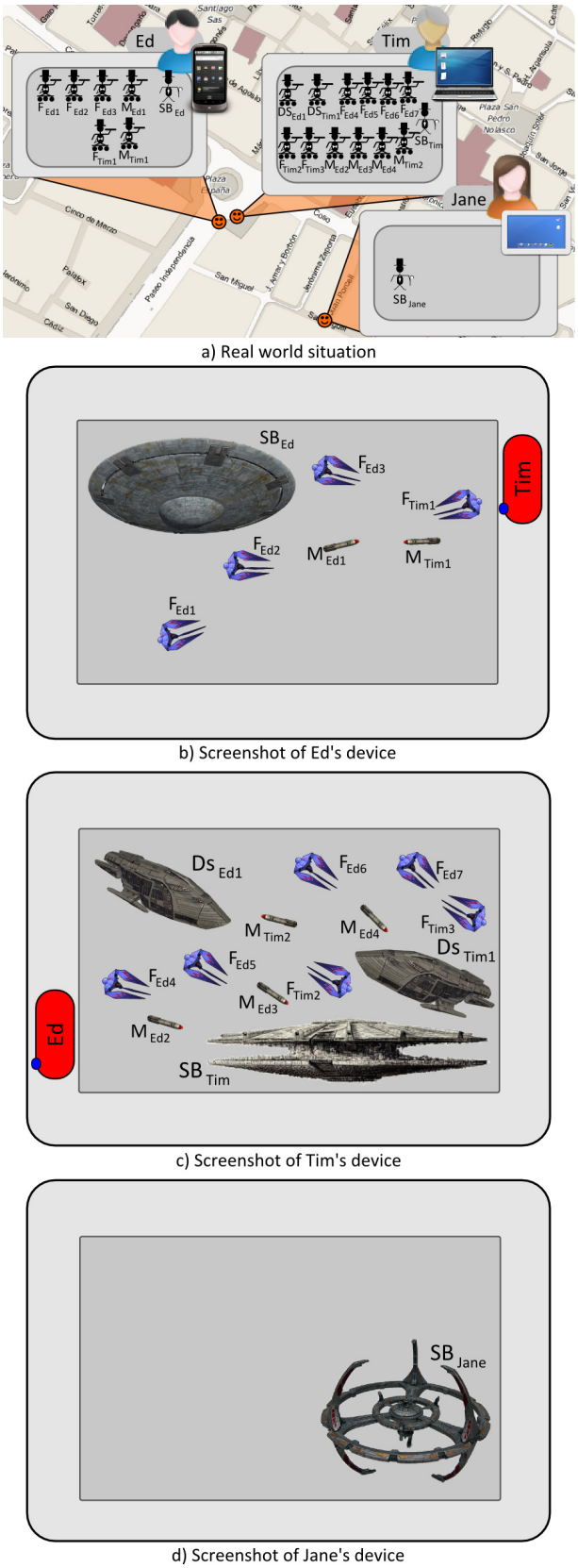

Figure 2: Game battle: Real world vs. users' screens

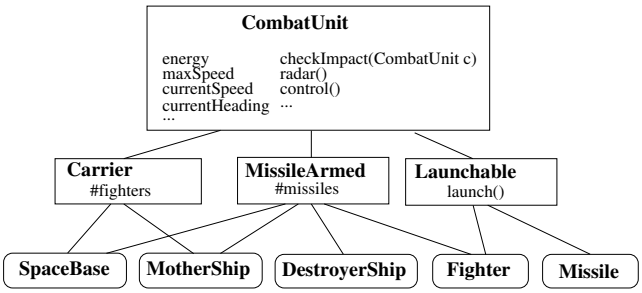

Figure 3: Different classes of combat units

For example, we could create a new class of Missile (e.g., SmartMissile, see Figure 4) which approaches to its target not directly but in a wavy and more unpredictable trajectory to avoid being destroyed. This new class will be compiled and stored in a class repository managed by the mobile agent platform (and so available for all the mobiles agents no matter where they execute). When a running Fighter looks for available weapons to fire (using reflection pattern), the new SmartMissile will be one of the possibilities.

To avoid cheating, the basic unit classes to be extended have private fields and methods that are not seen by the advanced users and that enforce the rules of the video game. As we can see in Figure 4, we have defined a new mobile agent *SmartMissile* which overrides the by-default *control* method. Notice that cheating is not possible as: 1) The only way to change the orientation angle and speed is by means of four methods that cannot be overridden[5], 2) The main loop of the missile (*run* method), which controls the fuel consumption, the update of missile coordinates, and the movement between devices, cannot either be overridden.

3.2.4 Unit deployment

Moving a combat unit to a new space area is directly implemented by the capability of mobile agents of moving across different devices. The number of combat units running on a given device could be limited by the game to avoid overloading player devices.

3.2.5 Battles

They happen between units residing on the same device[6]. Here the mobile feature of mobile agents is not needed but their autonomy: Each unit run asynchronously and tries to reach, destroy, or avoid enemy units following its own (possibly customized) behavior. Any unit can obtain public info about units in range (method *radar*) and takes those data in consideration in different manners. Thus, units alert each other their interactions, which are double checked by both sides to avoid cheating behaviors.

For example, when a missile detects that has impacted against a destroyer ship, it calls the *checkImpact* method of such a ship, which will check that, according to the missile's *radar* method and its own position, such a missile impact is correct. In that case, the destroyer ship is accordingly damaged or even destroyed. If the information is not verified the impact alert is just ignored.

[5]In Java, final methods cannot be overridden and private fields are not directly visible by subclasses.

[6]We could also implement battles involving different devices in range.

```
class Missile extends MobileAgent {
   Target target;
   private float x, y, angle, currentSpeed, fuel;
   ...
   final run() {
      while (!checkImpact(target)) {
         control();  // it could change angle, speed
         fuel=fuel-currentSpeed*k;
         if (fuel<=0) { fuel=0; currentSpeed=0 }
         x=x+currentSpeed*cos(angle);
         y=y+currentSpeed*sin(angle);
         moveTo(deviceAt(x,y), run());
            //moveTo(localhost) does nothing
      }// a general GUI method will repaint
       //this object at (x, y) coordinates
   }

   void control() {
      // consulting target.x, target.y, x, y
      // it invokes methods to change
      // orientation angle and speed as needed
      // to go directly towards the target
   }

   final void turnRight() { ... }
      // decreases angle in 10 degrees
   final void turnLeft() { ... }
      // increases angle in 10 degrees
   final void incrSpeed(){ ... }
      // increases current speed in 10 units
   final void decrSpeed() { ... }
      // decreases current speed in 10 units
}

class SmartMissile extends Missile {
   void control ()  {
      // we override the by-default control method
      // We can program a different way to approach
      // to the target, for example, following a wavy
      // trajectory to avoid being destroyed
      // as easily as when approaching in a straight line
   }
}
```

Figure 4: Code excerpt from SmartMissile agent

Remember, as afore mentioned, that the most basic methods implementing the basic and inviolable game rules (e.g., *radar*, *control*, *checkImpact*, ...) are part of the default behaviors, which cannot be overridden by the customized classes implemented by advanced users.

3.2.6 Summary

We have seen how a peer to peer video game of a certain complexity can be designed and implemented easily and intuitively thanks to the benefits provided by mobile agents technology. Just think about how to implement the above features without the help of mobile agents: for example, new behavior of units should be updated in *all* the player devices to prevent the chance of facing such customized units.

We would like to stress that, although we have shown the advantages of applying mobile agents to a multi-player RTS video game (where players control several units), and that mobile agents seem to be specially well suited for location-based games (such as the presented one) or even for pervasive games [6], such a technology can be applied to many other videogames, specially when multiple devices are involved or when we want to make users able to contribute

using their own versions of different game elements. Finally, whenever distributed deployment or analysis of scattered data is needed for a mobile game, mobile agents can help.

4. RELATED WORK

Mobile agent technology has been thoroughly used and evaluated in the context of general information systems, exhibiting good performance when compared with traditional client/server approaches [3, 5]. Moreover, as emphasized in [1], they provide a nice programming abstraction for many application scenarios, easing the development of distributed applications.

While in other contexts, such as location-based services, the mobility of mobile agents has been exploited [11], mobile agent technology has not been exploited massively in gaming. In [2], the authors propose a mobile agent-based framework to enhance the performance of network games where agents are used to achieve adaptive game execution and delivery from an architectural point of view, instead of being used to model in-game elements as we do. In the context of pervasive gaming, in [7] they use mobile agents to develop their games, but they do not exploit the capabilities of mobile agents as we propose (e.g., 1 ship, 1 mobile agent).

To the best of our knowledge, our proposal is the first one to bring and adapt the flexibility of mobile agent technology to mobile gaming context, with the feature of promoting user involvement and collaboration.

5. CONCLUSIONS

In this paper we have proposed the use of mobile agent technology to design mobile games where moving software across player devices makes senses or it is just the main goal. The main contributions of our work are:

- Using mobile agents is easy and efficient, even in mobile devices, and provides mobile game developers with possibilities very difficult to achieve without mobile agent technology.
- Some game actions like unit deployment or launching units are implemented in a way very intuitive and similar to real life. Game units physically *move* across the game area (among player devices).
- New specialized non-playable characters, whose behavior is programmed by advanced players, can be created and deployed in already started games. The potential cheating is avoided by construction.

As future work we plan to implement a sample full game based on mobile agents that people can download and play it; the code will be publicly available to show the small coding impact of using mobile agents. We are also working on improving the mobile agent platform SPRINGS to become more efficient on mobile devices.

Acknowledgments

This work has been supported by the CICYT project TIN2013-46238-C4-4-R and DGA-FSE.

6. REFERENCES

[1] C. Bobed et al. Distributed mobile computing: Development of distributed applications using mobile agents. In *Proc. of PDPTA'10, Las Vegas (Nevada, USA)*, pages 562–568, July 2010.

[2] C. Hull et al. FRAGED: A framework for adaptive game execution and delivery to improve the quality of experience in network aware games. In *Proc. of PGNET'14, Liverpool (UK)*, June 2014.

[3] C. Spyrou et al. Mobile agents for wireless computing: the convergence of wireless computational models with mobile-agent technologies. *Mobile Networks and Applications*, 9(5):517–528, 2004.

[4] D. Milojicic et al. *Mobility: processes, computers, and agents*. ACM Press/Addison-Wesley, 1999.

[5] E. Mena et al. Adaptable software retrieval service for wireless environments based on mobile agents. In *Proc. of ICWN'02, Las Vegas (Nevada, USA)*, pages 116–124, June 2002.

[6] I. Hwang et al. Toward a mobile platform for pervasive games. In *Proc. of MobiGames'12, Helsinki (Finland)*, pages 19–24, August 2012.

[7] L. Görgü et al. Freegaming: mobile, collaborative, adaptive and augmented exergaming. *Mobile Information Systems*, 8(4):287–301, 2012.

[8] O. Urra et al. Testing mobile agent platforms over the air. In *Proc. of DS2ME'08, Cancun (Mexico)*, pages 152–159, April 2008.

[9] O. Urra et al. Mobile agents and mobile devices: Friendship or difficult relationship? *Journal of Physical Agents*, 3(2):27–37, May 2009.

[10] R. Trillo et al. Comparison and performance evaluation of mobile agent platforms. In *Proc. of ICAS'07, Athens (Greece)*, June 2007.

[11] S. Ilarri et al. Location-dependent queries in mobile contexts: Distributed processing using mobile agents. *IEEE Transactions on Mobile Computing (TMC)*, 5(8):1029–1043, August 2006.

[12] S. Ilarri et al. SPRINGS: A scalable platform for highly mobile agents in distributed computing environments. In *Proc. of WoWMoM'06, Buffalo (NY, USA)*, pages 633–637, June 2006.