An Overview of the AMUSE Social Gaming Platform

Federico Bergenti
Dipartimento di Matematica e Informatica
Università degli Studi di Parma
Parco Area delle Scienze 53/A, 43124 Parma, Italy
Email: federico.bergenti@unipr.it

Giovanni Caire and Danilo Gotta
Telecom Italia S.p.A.
Via Reiss Romoli 274, 10148 Torino, Italy
Email: {giovanni.caire, danilo.gotta}@telecomitalia.it

***Abstract*—This paper presents an overview of the novel platform *AMUSE* (*Agent-based Multi-User Social Environment*), an agent-based social gaming platform that leverages the power of industrial-strength agent technologies. The core need that motivated the initial work on AMUSE was to provide game developers with a solid tool targeting common horizontal issues in social gaming, like user management and game state management, for games with synchronous and asynchronous interactions. AMUSE fulfills such a need by means of industrial-strength agent technology. Actually, AMUSE is not only a development framework that can be effectively used to implement prototypes and small-scale games with just a few concurrent players. Rather, it is thought as a *PaaS* (*Platform as a Service*) tool that enables service provides, like game portals and community portals, to relief game factories from the burden of implementing horizontal functionality that are common to a large set of games. This paper is a first presentation of the work on AMUSE and it starts framing AMUSE into the scope of social gaming. Then, the paper describes the architecture of the multi-agent system that represents the core of AMUSE and it relates the presented agent types with the functionality that AMUSE provides. Finally, the paper outlines some directions of future development.**

I. Introduction

This paper describes a recent work in the important industrial sector of online social games: an agent-based innovative platform that leverages the power of industrial-strength agent technologies to provide game developers with horizontal features that are common to most, if not all, online social games. Such a platform, namely *AMUSE* (*Agent-based Multi-User Social Environment*), gives developers a set of functionality that free them from the burden of implementing, and possibly reimplementing over and over again, common features like user management and game state management. The approach that AMUSE fosters lets developers concentrating their effort on game-specific features, it ensures solidity, and it ultimately reduces time-to-market and increases product quality.

AMUSE is designed to meet the requirements of large-scale service providers and it is intended for a *PaaS* (*Platform as a Service*) usage in large-scale scenarios. This, combined with the expected scalability of underlying agent technology, makes AMUSE an ideal tool for experimental prototypes intended can scale up to large-scale services. In fact, AMUSE is developed on top of *WADE* (*Workflows and Agents Development Environment*) [3], [24], the popular open-source platform for *agent-based BPM* (*Business Process Management*). One of the key characteristics of WADE is that it can be easily deployed on commodity computers and networks, and it can also be smoothly scaled up to huge services. We use WADE every day in our laboratories, and it is worth noting that the same software has been in daily use over the last 5 years [23] for large-scale network and service management in Telecom Italia for more than 8.95 million broadband connections for retail and business customers over a network of 114 million km of copper lines and 5.7 million km of optical fibers [22]. This is the reason why we say that the choice of implementing AMUSE on top of WADE ensure low-budget game development, giving the possibility to deploy the platform, and then smoothly scale up the service to a large number of users and to hosted deployment, if needed.

WADE is essentially the main evolution of *JADE* (*Java Agent and DEvelopment framework*) [4], [5], [7], [8], [16], the open-source framework that facilitates the development of interoperable multi-agent systems. JADE has been used in many research and industrial systems at an international scale since its initial development back in 1998 and today it is a reference for industrial-strength agent technology. WADE mainly adds to JADE the support for the execution of tasks defined according to the *workflow metaphor*, and it also provides a number of mechanisms that help managing the inherent complexity of a (distributed) multi-agent system both in terms of administration and fault tolerance.

While we measure a decline in investments in social gaming [1], social gaming, and mobile gaming in particular, is still on the rise from a game count perspective, with the industry seeing a 105% [18] increase in the number of mobile and social games on the market since 2000. Moreover, the industry experienced its biggest boom just last year, in 2012, when total games reached from 90 million to more than 211 million total [21]. This is sufficient to justify a research investment in this industrial sector as it is one of the driving forces of IT today.

This paper is not only intended to provide an overview of AMUSE, rather it is also meant to frame the work on AMUSE in the broad scope of social gaming in order to motivate, identify and justify the core design decisions. In the following section we frame AMUSE into the broad research on social gaming by giving essential definitions and terminology. Then, in Section III we outline the coarse-grained architecture of AMUSE and we detail the roles of single agents and their responsibilities. Finally, we conclude the paper with a brief summary of the work and with some insight on future developments of AMUSE.

II. What is Social Gaming?

The industry of video games, and *online video games* in particular, plays a significant role in our society that has been recently boosted by the pervasive diffusion of games for mobile appliances. Actually, the peculiar combination of novel game dynamics with the functionality of modern mobile appliances, like undisrupted connectivity, advanced graphics and sound capabilities, and on-board sensors, ensures a prolific and long lasting synergy between the industries of mobile appliances and video games.

A. Basic Terminology

Scientists and philosophers from diverse background have been discussing the notion of *play* and *game* for a long time, and they have already established agreed results and terminology.

One of the most cited attempts to characterize the play activity dates back to mid-50's to Huizinga's *Play Theory*. In his seminal book, best known as *Homo Ludens*, Huizinga characterizes the play activity as:

> ... a free activity standing quite consciously outside "ordinary" life as being "not serious" but at the same time absorbing the player intensely and utterly. It is an activity connected with no material interest, and no profit can be gained by it. It proceeds within its own proper boundaries of time and space according to fixed rules and in an orderly manner. It promotes the formation of social groupings that tend to surround themselves with secrecy and to stress the difference from the common world by disguise or other means. [14]

Even if this characterization of the play activity has undergone very reasonable critiques over the years, it is worth noting that the act of playing is always associated with a social nature, and we can broadly say that the play activity is social *per se*.

Unfortunately a characterization of the play activity is not enough because we generally *differentiate games from play*. The Huizinga's characterization includes rules in the play activity, but such rules are always flexible and subject to change, with no real need for rules to be agreed or adopted beforehand. On the contrary, games are based on rules that are, often implicitly, adopted and that are not subject to frequent or unjustified change. Rules structure games, and make them repeatable.

One of the most cited definitions of game, which has been recently developed in the scope of video games by Juul, defines a game as a:

> ... rule-based formal system with a variable and quantifiable outcome, where different outcomes are assigned different values, the player exerts effort in order to influence the outcome, the player feels attached to the outcome, and the consequences of the activity are optional and negotiable. [17]

Games inherit much from the play activity and all games are social in some sense, if nothing else, because players often retell their experiences.

Finally, to better understand the landscape of social gaming, we should remember that *gameplay* is the specific way in which players interact with a game and, in particular, with a video game. We can adopt one of the many available definitions of gameplay as follows:

> Gameplay is the formalized interaction that occurs when players follow the rules of a game and experience its system though play. [20]

Once we are happy with the fact that the play activity and games are social in nature, we need to discuss how so called *social online games*, or *social games* for short, differentiates from other forms of games. This discussion has lead us to identifying salient characteristics of social games that any social gaming platform like AMUSE is demanded to provide.

B. A Characterization of Social Gaming

A primitive approach is to take the platform perspective and mark as social any game that use a *social network platform*. These are the so called *social network games* and the pervasiveness of online social networks in our society makes them one of the most important examples of social games. Any game delivered via, e.g., Facebook, is a social network game but unfortunately this is by far not enough to allow us to descend any salient characteristic of a social gaming platform.

A less primitive approach leads to a notable body of literature that identifies many dimensions of social gaming. Here we restrict to a threefold characterization that relates to the *timing of social interactions* and to the *type of social relationship* [19]. Together, these characteristics encapsulate the social interactions of most online games, including the two extremes of the range, namely, *MMOs* (*Massive Multi-player Online games*), that group hardcore players in large and long-lasting games, and *casual games*, targeted at and used by a mass audience of casual players for short burst. A real-world social gaming platform should be able to provide support for the whole, or at least for a large part, of this spectrum, and the scalable design of AMUSE ensures this.

In summary, the three characteristics of social gaming that we consider here, and that we detail below, are [19]:

- *Synchronous vs. asynchronous player interaction.* Do interactions occur simultaneously in real time or at different times as in a turn-based game?
- *Symmetrical vs. asymmetrical relationship formation.* Does forming a relationship require input from both parties or can they be formed unilaterally by a single party?
- *Strong tie vs. loose tie relationship evolution.* Do relationships tend to become deep and long lasting or are they more likely to be light and transitory?

The remaining of this section is devoted to analyzing such characteristics and providing example of how they are concretely adopted in social games.

Synchronous vs. Asynchronous Interaction. The superficial understanding is that MMO games feature synchronous, real-time play while casual social games are asynchronous with interaction occurring at disconnected times. However, all MMOs

also feature important asynchronous features like in-game messages, and some Facebook game employs synchronous features such as a chat. Rather than an absolute position, current social games tend to offer a mix of synchronous and asynchronous interactions. Some games may highlight one or the other, but there are many that use both to establish a richer layer of engagement and retention.

The idea of synchronous gameplay is intuitively easy: players interact in real time rather than taking turns. Examples of synchronous social interactions include text chat, voice chat, video chat, and game elements like battles. Synchronous interactions can scale from two players to large groups.

The term *asynchronous game* might at first remind images of something slower and less intriguing, but asynchronous games can be just as engaging as synchronous ones, e.g., think of playing chess with a remote friend. Asynchronous social games come in different basic flavors, with some of the more common being:

- *Turn-based shared games.* They work well socially because each move is a mini game and there is social pressure to come back and complete the next turn. Moreover, bite-sized gameplay is easy to fit into schedules and players can play multiple games at once.
- *Turn-based challenge games.* Essentially, one of such games quickly becomes a set of two separate matches. Player 1 challenges and then player 2 responds; aggregate score determines the winner. They work socially because of the social pressure to return challenge and there is less waiting than shared turn-based since each player can complete his/her entire game independently.
- *Score-based challenge games.* These are the traditional *beat my high score* format. These games work socially because of the social pressure to return challenge and there is less waiting than turn-based options since players can try for their high scores anytime. Obviously, these types of games can be less interactive than other types.
- *Open-world asynchronous games.* In many ways, this is the most common Facebook game model. It works socially because the model supports a variety of game modes, including single-player and multi-player. It can variably approximate MMO experience without incurring into the technical issues of real-time play. Moreover, it still offers convenience of more casual games, i.e., players can play at different times and for short bursts.

Having said this, it is worth noting that chats are a powerful synchronous tool for player engagement and retention in both casual games and MMOs that deserves special attention, especially from the platform point of view. The chat, as a part of the game experience, always has a similar effect: boosting player engagement and facilitating long-term retention. When there is a real, vibrant support community present, players come back to a game more often and are less likely in search for other games.

Symmetrical vs. Asymmetrical Relationship. Perhaps the clearer example for understanding this characteristic of social gaming is the formation of social connections in Facebook versus Twitter. Facebook social relationships are symmetric: a member of the community asks to be a *friend* of another member and the latter must agree in turn for the relationship to exist. This approach as the advantage of making the acknowledgment mutual between parties and thus allowing for deeper sharing. On the contrary, the interaction is often limited to confirmed friends and friend relationships require (sometimes complex) management tools.

Examples of symmetric social interactions in online gaming include friending, gifting, trading, and private chatting on an individual scale, and parties, alliances, and manual multiplayer matchmaking on a group scale.

Twitter social relationships are asymmetric: a member of the community can *follow* anyone, without their reciprocation. This approach enables a widespread broadcasting and facilitates rapid dissemination of information. On the contrary, it requires less investment in social relationship and can be more prone to unsolicited interactions because communication filters are necessarily less sophisticated.

Examples of asymmetric social interactions include following, broadcasting, tweeting, and blogging on an individual scale, and public quests, factions, and random matchmaking on a group scale.

Although Facebook games are less known for such symmetric relationships, they do exist in many games. The *neighbor* approach prevalent in many games is a symmetrical social relationship. Even players who are in the same game and that are already Facebook friends still need to become *neighbors*.

Facebook games also feature numerous asymmetric social interactions. Instead of the neighbor approach, many games simply add a player's Facebook social graph directly to his/her game without requiring the permission of friends. These types of relationships are shallower than the ones originating from the neighbor approach but, because they are so broad, they lower the barriers to interaction and they create a high-density of ties among players.

Asymmetric relationships also exist in MMOs. A great example of this is the *public quest*. For example, if a public enemy is attacking the area, anyone who comes within a certain range is automatically considered to be participating in the public quest to capture him/her. Players can quickly and easily get a taste of group play and then go their own ways afterwards. The low social barrier allows for more frequent cooperation.

Strong Tie vs. Loose Tie. The former symmetry characteristic describes how relationships form but it does not necessarily dictate how they evolve. Ultimately, the relationship depends on what happens after the relationship itself has just been established. This characteristic is a simple measures of the evolution of a social relationship with a focus on the depth of interaction.

Examples of strong-tie gaming relationships range from the smallest scale, i.e., two players co-operatively play, to the group scale. Examples of loose-tie relationships also range from the smallest scale, e.g., game neighbors, to the group scale.

III. The AMUSE Platform

AMUSE (*Agent-based Multi-User Social Environment*) is an open-source development platform that can be downloaded from JADE Web site [16] and that is intended to tackle specific issues of social games, as discussed previously in this paper. The platform can be easily deployed in an *in house* setting, but it is designed to give service providers a tool to implement a PaaS with specific features of social games.

A. The Architecture of AMUSE

The characterization of social games sketched in Section II does not provide any means to concretely implement the desired features in a platform. In other words, we can classify a part of a game as employing symmetrical or asymmetrical relationships, but we have no best-practice tool to offer to developers to implement either symmetrical of asymmetrical relationships. *Gameplay design patterns* [10] are a pattern language that summarizes best practices in video game development and they are good reference for a list of features that a social gaming platform should provide.

Not all the over 700 gameplay design patterns identified in the literature and collected by the *Gameplay Design Pattern Project* [12] are interesting from the point of view of AMUSE. Some pattern is not related to the social aspect of games, while others are intended to provide specific features to games and they do not identify platform-level abstractions. We restrict here only to the best known gameplay design pattern that can contribute to the identification of the features that a social gaming platform like AMUSE should provide.

Before going into the details of the gameplay design pattern that AMUSE adopted, we need to clarify some underlying design decisions. First, we always assume the availability of a lower-level infrastructure for managing social relationships between users. This can be either a third-party infrastructure, like Facebook, providing a rich user profile and counting a large number of relationships between users; or it can be a private infrastructure accessible only from within AMUSE games, and normally providing a restricted user profile and a restricted set of relationships. AMUSE provides a generic interface that hides to the developers whether the infrastructure is third-party or private, thus ensuring scalability and allowing for the right infrastructure to be adopted for each game.

Another early decision that was taken is that AMUSE should provide a very flexible and highly scalable environment capable of scaling up with the success of a game. This enables early prototypes and low-cost experiments that can scale up to huge phenomena. *WADE* (*Workflows and Agents Development Environment*) [3], [24], the open-source platform for *agent-based BPM* (*Business Process Management*), is the ideal tool for ensuring such a high level of scalability and flexibility because one of its key characteristics is that it can be easily deployed on commodity computers and networks, and it can also be smoothly scaled up to huge services like nationwide network and service management [22].

Having said this, we simply decided to follow the best practices of WADE development and identified a set of *back-end agents* running on the server-side platform that communicate with *front-end agents* in charge of managing the actual interaction with the user.

Back-end and front-end agents implement the social gaming features of the platform depending on the centralization required by each and every single feature. For example, involving players in a mobile game does not always require a centralized authority: the agent on the user's mobile device contacts the respective agents of other users by means the users' profile stored locally in the device. AMUSE provides such a feature by assigning specific responsibilities to front-end agents. On the contrary, the management of a *table* in a *room* to let players engage in synchronous card game implies some centralized management of tables and rooms, as AMUSE actually provides.

In summary, the design of AMUSE comprises the following types of back-end agents that cooperatively deliver the functionality of the platform together with front-end agents:

- *UMA* (*User Manager Agent*). A socially inclined evolution of user manager agents of many other agent-based systems that manages the profile of single users and his/her relationships with other users. It relies on the underlying social network infrastructure for the concrete storage and discovery of profiles and relationships.
- *GRA* (*Games Room Agent*). It is an agent in charge of managing the shared game space in games with synchronous interaction. It can be effectively used to deliver asynchronous interaction if a back-end support is needed.
- *AMA* (*Application Manager Agent*). Taking the PaaS perspective, it is the agent in charge of managing the provided games and their lifecycle.
- *MTA* (*Match Tracer Agent*). It serves the needs of games that require a persistent game state and that needs restart options.

Finally, AMUSE provides a generic *MMA* (*Match Manager Agent*) in charge of interfacing the client application with back-end agents and to deliver the features that do not require a back-end support. For the current design, it is the only front-end agent that AMUSE provides.

B. Adopted Gameplay Design Patterns

After this brief sketch of the architecture of the multi-agent system that implements AMUSE functionality, we can briefly enumerate the gameplay design pattern that AMUSE uses internally, and that implement the features for game developers. Not all the following gameplay design pattern are currently implemented in AMUSE, but they are all important to grasp where AMUSE intends to go in the long term.

Some of the gameplay design patterns adopted in the design of AMUSE deal with the state of the game and with its evolution over time. These are implemented by the *GRA*, and the *GRA* itself provides an abstract view of the game state:

- *Private game spaces.* The game has parts of the game world that only a single player can manipulate directly.
- *Massively single-player online games.* These are games that make use of other players' games instances to provide input to the game state.

Together with the *MTA*, the *GRA* also implement the *persistent game world* design pattern, intended to make the game state independent from individual players' game and play session. The game state is available continuously and in continuous evolution.

Other gameplay design patterns that the *GRA* implements regard the management of how time is correlated with the evolution of the game, as follows:

- *Tick-based games.* The game progresses according to real-time, but in discrete steps.
- *Events timed to real world.* Gameplay events are initiated by specific real-time events occurring.

The *UMA* implements with no specific cooperation with other agents the *extra-game events broadcasting*, that allows game events to be broadcast in a medium where others can perceive them. This pattern is implemented by the *UMA* because it is the only agent that can interact with the underlying online social network infrastructure and its notification services.

Moreover, the *UMA* together with the *GRA* implement the following design patterns:

- *Drop-in/Drop-out.* Players entering and leaving ongoing game sessions are welcome.
- *Public player statistics.* The platform provides a means to publicize the player statistics inside and outside the game to users. The underlying online social network infrastructure is used to access the users' relationships and to publicize the statistics.
- *Visits.* Under the application of *UMA* policies, the *GRA* can provide temporary access to other players' private game spaces.
- *Invites.* Under the application of *UMA* policies, and taking into account the logics of the underlying social network infrastructure, the *GRA* allows inviting new players to a game as game actions.

Finally, only the *non-player help* design pattern request the cooperation of the *UMA*, the *GRA* and the *MMA* to ensure that players can receive help in the games by actions from those not playing.

C. The Implementation of AMUSE So Far

The current implementation of AMUSE does not yet provides all features described in the previous section, even if a clear plan is drawn to achieve full functionality briefly.

At the time of writing AMUSE provides a *UMA* with restricted functionality that manages a private online social network infrastructure. It is also able to manage various possible game involvement schemes and it is the final responsible for stored user profiles, which also include public game statistics.

Prototype *GRA* and *MMA* are provided to implement games with synchronous or asynchronous interactions. For the moment, the type of interaction that characterize the game is statically assigned and a game cannot have diverse parts with diverse types of interactions, nor it can dynamically change the type of interaction.

The available *MMA* is also in charge of providing a text channel that players can use for social interactions parallel to the actual play activity.

The current *GRA* prototype provides a game state representation that has proven valid for a wide range of games and it is based on the abstractions of *rooms* and *tables*, that players share and that provide the principal means of interaction during the game.

Taking the PaaS perspective, AMUSE now provides a prototype *AMA* that manages the interaction with the underlying WADE platform and that enables developers to choose between *decentralized* (or *client-only*) and *server-based* types of deployment. For the case of server-based deployment, the AMA already provides the features needed to leverage the flexible deployment schemes of WADE, which ensures that the server would not become a system bottleneck.

Moreover, the available *AMA* provides all needed administration services to quickly set up a private gaming infrastructure that, thanks to WADE, can scale up far over the initial deployment.

Finally, AMUSE includes a prototype *MTA* that is in charge of using WADE persistency support to implement persistent and restartable game state.

Front-end agents are now restricted to Android agents or desktop agents because, at the moment, no porting of JADE is available for other platforms. All in all, the flexible and somehow standardized agent communication protocol makes the interface between back-end and front-end agents fully device agnostic, and we see no problems is accommodating new and unexpected user devices in the architecture.

It is worth noting that current AMUSE prototype already includes Web games because we assume that such games can be structured into a lightweight client module connected to a heavyweight server module. We can already have server-side agents, running inside JADE/WADE containers, that communicate with the lightweight-client user interface via one of the available Web communication protocols, e.g., WebSockets.

IV. Conclusions

This paper presents the basic ideas that guided the development of AMUSE, a novel agent-based social gaming platform. The initial motivation for this work is that we felt the urge for a sharable tool capable of providing horizontal features of social gaming and we thought that agent technology, and WADE in particular, would have been ideal for this. Agent technology has already been applied to foster collaboration (see, e.g., [9]) and, more recently, it was used to address large-scale social networks (see, e.g., [6]), thus providing a solid base for the coordination of large communities. The initial vague idea that stimulated this work eventually turned into a complex architecture that encompasses back-end agents and front-end agents cooperatively providing social gaming features to developers.

AMUSE leverages the power of WADE to provide game developers and social gaming service providers with a scalable architecture with applicability ranging from initial prototypes to large-scale deployment. We think that this is a very important feature of AMUSE because it restricts the time-to-market

and it extends the range of possible AMUSE developers to the open-source community, which is provided with fully open-source tools.

At present AMUSE uses WADE only for its proven features of flexibility and scalability in deployment. It does not really take advantage of the other major feature of WADE, namely its workflow-based development approach. This ensures that no game developers needs to understand and appreciate the flexibility of workflow-based development, but it also allows advanced developers to make an effective use of workflows to implement very dynamic games where parts of the game can be visually programmed, possibly by players.

At the moment we are investigating the possibility of providing a generic lightweight Web client using the *GWT* (*Google Web Toolkit*) [15] to give Web developers a minimalistic JADE implementation to adopt for their games. All in all, this is like using the Web browser as a single-agent JADE/WADE container using a proprietary protocol that is bridged to the common agent communication protocols by the Web server. A very similar approach has been available in JADE for more than a decade under the name of *split container*.

At the time of writing, AMUSE has been tested and validated by means of *four* mobile games and at least two of them will be released open-source together with AMUSE. Such games have been chosen to put in practical usage most of the functionality that AMUSE provides and to testbed the usability of AMUSE to implement fully fledged mobile games.

The first game that we developed, codename *Numblers*, is a number board game with asynchronous interactions that closely follows the lessons learned from largely appreciated games like *Ruzzle*. The dynamics of game engagement and gameplay does not need the support of back-end servers, and this game can be ideally deployed with no server support.

The second game, codename *TwentyOne*, is a variant of Numblers based a different game challenge, and it was chosen to try different ways for delivering similar functionality.

The third game developed so far, codename *BattleSpheres*, is a synchronous, real-time game that has been developed with the help of *AndEngine* [2]. In BattleSpheres, player A challenges player B by throwing virtual balls towards him/her and B is expected to block such balls before they reach the bottom of his/her screen. This game uses the experimental real-time features of AMUSE.

Finally, the fourth game is *Wadeoku*, a synchronous variant of Sudoku puzzles intended to have a group of players synchronously sharing a Sudoku board and gaining points for every good assignment of a number to an empty cell. This game does need a significant support from back-end agents and it is close to the real use of AMUSE that we foresee in the near future.

The development experience gathered with such four games can be considered positive and the early experimentation on concrete examples provided significant feedback on core platform-level decisions. In addition, early experimentation allowed us to identify interesting best practices in the utilization of AMUSE that were not initially envisaged.

AMUSE is open-source and it can be downloaded from JADE Web site [16].

REFERENCES

[1] A. J. Agnello. *Investment in social gaming drops by $1 billion in 2012*. Available at http://www.digitaltrends.com

[2] AndEngine Web site. http://www.andengine.org

[3] M. Banzi, G. Caire, D. Gotta WADE: A software platform to develop mission critical, applications exploiting agents and workflows. Procs. Int'l Conf. *Autonomous Agents and Multi-Agent Systems*, 2008.

[4] F. Bellifemine, G. Caire, D. Greenwood. *Developing multi-agent systems with JADE*. Wiley Series in Agent Technology, 2007.

[5] F. Bellifemine, A. Poggi, G. Rimassa. Developing multi-agent systems with a FIPA-compliant agent framework. *Software: Practice & Experience*, 31:103–128, 2001.

[6] F. Bergenti, E. Franchi, A. Poggi. Selected models for agent-based simulation of social networks. Procs. *Symposium on Social Networks and Multiagent Systems*, 2011.

[7] F. Bergenti, A. Poggi. Ubiquitous Information Agents. *Int'l J. Cooperative Information Systems*, 11(3–4):231–244, 2002.

[8] F. Bergenti, A. Poggi, B. Burg, G. Caire. Deploying FIPA-compliant systems on handheld devices. *IEEE Internet Computing*, 5(4):20–25, 2001.

[9] F. Bergenti, A. Poggi, M. Somacher. A collaborative platform for fixed and mobile networks. *Communications of the ACM*, 45(11):39–44, 2002.

[10] S. Björk, S. Lundgren, J. Holopainen. Game Design Patterns. Procs. *Digital Games Research Conference*, 2003.

[11] G. Caire, E. Quarantotto, M. Porta, G. Sacchi. WOLF - An Eclipse Plug-in for WADE Procs. IEEE Int'l Workshops *Enabling Technologies: Infrastructures for Collaborative Enterprises*, 2008.

[12] Gameplay Design Pattern Project Web site. http://gdp2.tii.se

[13] T. Grant. *The Art of Videogames*. Wiley-Blackwell, 2009.

[14] J. Huizinga. *Homo Ludens: A Study of the Play Element in Culture*. Beacon Press, 1955.

[15] GWT (Google Web Toolkit) Web site. http://www.gwtproject.org

[16] JADE (Java Agent DEvelopment framework) Web site. http://jade.tilab.com

[17] J. Juul. *Half-Real: Video Games between Real Rules and Fictional Worlds*. The MIT Press, 2005.

[18] S. Mlot. *Infographic: How Do You Get Your Mobile Gaming Fix?* Available at http://www.pcmag.com

[19] M. Ricchetti. *What Makes Social Games Social?* Available at http://www.gamasutra.com

[20] K. Salen, E. Zimmerman. *Rules of Play: Game Design Fundamentals*. The MIT Press, 2004.

[21] E. Swallow. *What Makes a Good Social Game?* Available at http://www.forbes.com

[22] Telecom Italia S.p.A. Relazione Finanziaria Annuale 2012. Available at http://www.telecomitalia.com

[23] L. Trione, D. Long, D. Gotta, G. Sacchi. Wizard, WeMash, WADE: Unleash the power of collective intelligence. Procs. Int'l Conf. *Autonomous Agents and Multiagent Systems*, 2009.

[24] WADE (Workflows and Agents Development Environment) Web site. http://jade.tilab.com/wade