Agent-Based Social Gaming with AMUSE

Federico Bergenti\textsuperscript{a,}, Giovanni Caire\textsuperscript{b}, Danilo Gotta\textsuperscript{b}

\textsuperscript{a}Università degli Studi di Parma, Parco Area delle Scienze 53/A, 43124 Parma, Italy
\textsuperscript{b}Telecom Italia S.p.A., Via G. Reiss Romoli 274, 10148 Torino, Italy

Abstract

This paper describes the core features and the multi-agent architecture of AMUSE (Agent-based Multi-User Social Environment), a novel agent-based platform for social gaming. AMUSE is designed to offer game developers readymade solutions to many issues that are common to social games like, e.g., advanced management of matches, turns and players. AMUSE is developed on top of WADE to leverage the scalable and solid agent-based deployment environment and the PaaS approach that it provides. This paper first outlines some of the motivations that originated the development of AMUSE. Then, it presents the multi-agent architecture of AMUSE and it enumerates the major applicative features that the platform provides. Finally, the paper briefly outlines initial experiments on game prototyping using the platform.

1. Introduction

After more than a decade from the popularization of the term agent platform to denote a horizontal tool intended to support the implementation and deployment of multi-agent systems, we witness today a significant number of agent platforms like JADE\textsuperscript{1,2,3,4,5}, WADE\textsuperscript{6,7}, and many others\textsuperscript{8} that are characterized by specific features, but that maintain their horizontal essence. This ensures that such platforms are widely applicable and that they can support developers across diverse application scenarios, but it also limits the functionality that they provide to so called application-agnostic features like robust message passing and scalable deployment.

We believe that such an application-agnostic approach is no longer sufficient to enable the full utilization of the characteristics of agents as software components\textsuperscript{9} because the developer is left alone against a number of issues of the application domain that, for their application-specific nature, are out of the scope of agent platforms. We think that a significant step forward in the direction of improving the way people use agents is to develop vertical agent platforms (or domain agent platform) targeting specific, yet wide, application domains.

AMUSE (Agent-based Multi-User Social Environment)\textsuperscript{10,11} is a novel vertical agent platform that targets the specific, yet extremely wide and important, application domain of social gaming\textsuperscript{12}. 

---

\textsuperscript{a} Corresponding author. Tel.: +39-(0)521-906929 ; fax: +39-(0)521-906950.
E-mail address: federico.bergenti@unipr.it (Federico Bergenti).
While we measure a decline in investments\(^1\), social gaming and mobile social gaming are still on the rise from a game count perspective, with the industry seeing a 105\%\(^2\) increase in the number of mobile and social games on the market since 2000. Moreover, the industry experienced its biggest boom just recently, in 2012, when total games reached from 90 million to more than 211 million total\(^3\). This is sufficient to make the industry of mobile social gaming crucial for ICT and an significant opportunity for agent technologies.

AMUSE leverages agent technologies to give to developers a set of functionality that free them from the burden of implementing, and possibly reimplementing over and over again, common features like advanced management of matches, turns and players. Such features characterize social gaming and they are always needed, in possibly different forms, by all social games, even if they are considered too application-specific to find a convenient accommodation in traditional agent platforms.

The advocated approach of vertical agent platforms couples well with the decennial work on agent platforms because the former benefits from the latter, and vice versa. This is the reason why AMUSE is built on top of W ADE: to benefit from the scalability and robustness that characterize W ADE\(^6\). In fact, W ADE can be easily deployed on commodity computers and local networks, but it can also be smoothly scaled up to huge services like nationwide networks and services\(^16\). AMUSE, through W ADE, provides game developers with a scalable architecture with applicability ranging from initial prototypes to large-scale deployment and we think that this is a crucial feature that extends the range of possible AMUSE developers to the open-source community.

The design of a vertical agent platform on top of a traditional agent platform does not force the former to fully adopt the development approach of the latter. At present AMUSE uses W ADE only for its proven features of robustness and scalability. It does not really take advantage of the other major feature of W ADE, namely its workflow-based development approach. This ensures that game developers are not demanded to understand and appreciate the flexibility of workflow-based development, and 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.

This paper is organized as follows. Next section presents the multi-agent architecture of AMUSE, and it also enumerates and discusses the major features of the platform that are concretely implemented by means of specific agents. Then, a brief outline of current experiments on game prototyping present the class of games that can be already implemented using AMUSE.

2. Architecture and Features of AMUSE

AMUSE is intended to provide game developers with a rich set of effective features designed to simplify the implementation of social games, to gain quality and to reduce time-to-market. At the time of writing the list of envisioned features for the platform is still not exhaustive and new features are continuously added as the project evolves. AMUSE does not try to introduce radically new features; rather it tries to benefit from agents taking into account best practices of game development like game design patterns\(^17,18\).

2.1. Architecture of AMUSE

The features that AMUSE offers to developers are all implemented using the coarse-grained architecture sketched in Figure 1, where we can see:

- **Application clients**, that are in charge of embedding all needed user interfaces and game-specific components into an executable application. Figure 1 refers to the case of mobile application clients, but AMUSE is not restricted to mobile social games and application clients can be hosted on a variety of devices including smartphones, tablets and desktop PCs.
- **AMUSE (client) library**, that provides application clients with the services of AMUSE by participating into a distributed W ADE platform. For the specific example in Figure 1 where shown application clients are mobile, JADE-Android\(^19\) is used to join the W ADE platform.
- **Application server components**, that are server-side, game-specific components that interact with respective application clients by means of the robust and scalable mechanisms that W ADE offers.
• **AMUSE server components**, that are confined in the server premises and that are deployed as WADE agents within a WADE distributed platform to implement game-agnostic services. Such services are intended to manage major AMUSE elements like matches, turns and players.

• **AMUSE administration interface**, that allows managing AMUSE deployment sites using the PaaS approach that WADE natively supports.

By means of this architecture, AMUSE offers game developers the possibility of implementing two kinds of games: *(i)* **client-only games**, that lack application server components and that are fully implemented by means of application clients; and *(ii)* **server-based games** that delegate to application server components part of the game logics. AMUSE ensures equal support to both kinds of games and the developer is free to choose one or the other on the basis of the characteristics of the specific game.

2.2. Agents in AMUSE

The coarse-grained architecture of AMUSE as depicted in Figure 1 is based on best practices of WADE development and it implements most of the shown elements by means of WADE agents. Game developers are not demanded to know and appreciate WADE agents to use AMUSE because most of the services that agents provide are accessible by means of dedicated APIs. Deep or unforeseen customizations of the platform still require directly working with agents and understanding agent lifecycle management and agent communication mechanisms.

AMUSE agents are grouped into: *(i)* **back-end agents** running on the server-side of the platform; and *(ii)* **front-end agents** in charge of managing the actual interactions with users at the users’ terminals. Front-end agents may or may not require back-end agents to implement a specific social gaming feature of the platform depending on the level of centralization required by the specific feature. For example, the act of involving players in a mobile game does not necessarily require a back-end support: the agent on the game initiator’s mobile device may contact directly other players’ agents. On the contrary, the management of a table in a room to let players engage in a synchronous board game demands centralized management of tables and rooms, as AMUSE actually provides.
In its current implementation, AMUSE provides a generic MMA (Match Manager Agent) in charge of interfacing an application client with back-end agents and to deliver the features that do not require a back-end support. For the current design, the MMA is the only front-end agent that AMUSE provides and the developer interact with MMA by means of the AMUSE client library.

Besides the single front-end agent, AMUSE comprises four types of back-end agents as follows:

- **UMA (User Manager Agent).** A socially inclined evolution of user manager agents of many other multi-agent systems that manages the profile of single users and his/her relationships with other users.
- **GRA (Games Room Agent).** An agent in charge of managing the shared game space in games with synchronous interactions.
- **AMA (Application Manager Agent).** Taking the PaaS perspective, it is the agent in charge of managing the games and their lifecycle.
- **MTA (Match Tracer Agent).** An agent that serves the needs of games that require a persistent game state and that need restart options.

### 2.3. Features of AMUSE

The current implementation of AMUSE provides features specifically designed to support the management of the lifecycle of matches as follows.

- **Peer-to-peer match management.** This feature supports the organization of, possibly persistent, one-to-one matches. Match organization is based on the invitations: the match organizer invites the opponent explicitly specifying his/her nickname or asking AMUSE to randomly select one suitable opponent. The opponent is notified about the invitation and he/she can accept or reject. If the opponent accepts the invitation the match starts, otherwise the match is implicitly terminated. Games based on this approach do not require game-specific logics running on the server-side of the platform to manage the lifecycle of matches and therefore they can be often implemented as client-only games.

- **Centralized match management.** This feature supports the organization of matches involving two or more players possibly joining and leaving the match while it is running. Match organization is based on tables: the match organizer creates a table that players proactively join. The match starts as soon as the minimum number of players at the table is reached. Tables are organized in rooms that players enter before joining tables. Games based on this approach most often require game-specific logics running on the server-side of the platform, at least to manage the dynamics of players entering and leaving tables. Therefore such games can only be implemented as server-based games.

The current implementation of AMUSE provides a set of core features not strictly related to the social gaming domain but that are essential to develop games, as follows.

- **User management.** This feature concerns the registration and authentication of users as well as a minimal support for managing user profiles in terms of sets of game-specific name/value pairs. In its current implementation AMUSE allows searching the user base by means of user profiles and it also allows picking random users matching specific criteria.

- **Application management.** This feature concerns the possibility of registering, installing, configuring and activating games using the PaaS approach. Such operations are made available through the AMUSE administration interface. In the current implementation such an interface is available in the forms of: (i) a CLI (Command Line Interface) for experienced site administrators and large-volume sites; and (ii) a user-friendly Web application for small-volume sites.

- **Clock synchronization service.** This feature provides a basic best-effort synchronization mechanism that allows application clients running on different terminals to perform synchronized actions. This is particularly important, for instance, to support interactive games where a match starts synchronously for all players and a synchronization failure may result in a loss for a player.
Peer-to-peer pipe service. This feature concerns the possibility of establishing a direct communication channel, a pipe, between the application clients of different users to support the exchange of game-specific data.

Text message exchange service. This feature concerns the possibility of sending and receiving text messages to/from other players within a game. This service supports multicasting and buffering: in case one receiver is not available at the time a message is sent, the message is automatically stored at the server-side of the platform and it is delivered as soon as the receiver becomes available.

3. Initial Experiments

The current implementation of AMUSE is under constant validation and, besides its recent first public release, researchers and students have already manifested their interest in the project. At the time of writing AMUSE has been validated by means of prototype implementations of four games, as follows.

The first game that we prototyped, codename Numblers, is a number board game with asynchronous interactions that closely follows the lessons learned from largely appreciated games like Ruzzle. Gameplay and the dynamics of game engagement do not demand a support of back-end agents, and this game is deployed as a client-only game.

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 the same functionality that Numblers delivers.

The third game developed so far 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 valid assignment of a number to an empty cell. This game does need a significant support from back-end agents.

Finally, the fourth game, codename BattleSpheres, is a synchronous, real-time game that has been developed with the help of AndEngine. 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 clock synchronization features of AMUSE and it is currently the most interesting experiment because it faces users with appreciated challenges and with a nice mobile user interface.

Figures 2 show game screenshots. Figure 2(a) shows the initial countdown that ensures the two players would start playing synchronously. Both the countdown screen and the game initiation mechanism use the AMUSE clock synchronization feature, which is also used to keep shared, yet not centralized, game duration countdown, as shown in Figure 2(b) and Figure 2(c). Besides the game duration countdown, Figure 2(b) also shows the game playground as seen by player A; the (green) balls in figure were previously placed by player A as a defensive line. Figure 2(c) shows a new (red) ball thrown by player B entering player A’s view of the playground.
4. Conclusions

This paper presents an outline of AMUSE, a novel agent-based social gaming platform. The initial motivation for the development of AMUSE originates in the urge for a sharable tool capable of providing vertical 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 and, more recently, it has been used to address large-scale social networks, thus it represents a solid base for the coordination of the large communities of popular social games.

AMUSE leverages the power of WADE to provide game developers and service providers with a scalable architecture that is applicable from initial prototypes to large-scale deployments. 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.

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

AMUSE is fully open source and it has been recently released in version 0.9b. Initial documentation and the source code of the platform and of examples is available at the AMUSE official Web site: http://jade.tilab.com/amuse.

References