# ASAP: A Framework for Designing Gamified Models of Complex Systems

David Solís-Martín, Juan Galán-Páez, Joaquín Borrego-Díaz, Fernando Sancho-Caparrini
Dpto. Ciencias de la Computación e Inteligencia Artificial - Universidad de Sevilla

*Abstract* — **This paper introduces ASAP, a framework for the development of mobile gamification of problems where Complex Systems play a fundamental role. This framework allows adding a layer of gamification concepts over a set of multi-agent systems for the raw modeling of the problem. Over it, a layer of data acquisition can be added in order to apply data analysis to the interactions and evolution of games. Some simple running examples are provided.**

***Keywords - gamification; data crowdsourcing; mobile computing; complex systems***

## I. INTRODUCTION AND BACKGROUND

Complex Systems research constitutes one of the big challenges of science in the XXI century due to several reasons. On the one hand, they are still far away from the capabilities of current theoretical science modeling (mainly using equational mathematical tools), on the other hand they are everywhere when one tries to understand the interesting processes occurring in real world [6]. Understanding these systems and having tools able to predict and control their behaviors will be crucial, not only in scientific contexts, but also in other aspects of real life such as business, education, health or financial services among others.

Following Gartner [2], *Gamification is the concept of applying game mechanics and design techniques to engage and motivate people to achieve their goals*. As this company publishes [4], a gamified service for consumer goods marketing and customer retention will become as important as Facebook, eBay, or Amazon, and more than 70% of Global 2000 organizations will have at least one gamified application.

The relationship between Gamification and Complex Systems can be summarized as follows [12]:

- Complex Adaptive Systems Theory [5, 6] (dynamic systems able to adapt *in* and evolve *with* a changing environment) should be adopted into Gamification.
- Emergence (the existence or formation of collective behaviors) should be part of the gamification design process.
- Playing and designing games rewires people's brains in order to better manage complexity in the real world.
- Gamification will better enable us to cope with Complex Systems management in enterprise.

The great difference between creating games in general and the concept of gamification is that the later requires a deeper understanding of the world in order to obtain an experience that the user could describe as real [22, 23]. In this sense, taking into account that most interesting real processes are related to complexity, it is mandatory to have a good knowledge about how to include these features in games but without complexification of the creation process.

Even though several authors have been interested mainly in the importance of Complex Systems for game creation [1, 21], we are also deeply interested in two more facets: the role that complexity can play on the players as a way for improving their knowledge about the world, and the possibility to learn, from real players, in order to understand the different ways the complexity can be controlled.

At this point, and as part of our experimental stage to manage complexity in bigger projects, we are not interested in reproducing big real complex systems (in order to create games with real-feeling environments), but in providing a framework for rapid development of games, allowing to easily implement systems sharing some real complex properties, specifically, a set of *simple* individuals (with very limited capabilities) interacting locally and massively with each other in the limits of a controlled environment.

In this way, one promising goal is to capture users' interactions with the artificial environment of the gamified system in order to both, explore the *solutions* a human intuitively finds for the evolution of the system, and create new artificial skilled opponents by learning from the behaviors of the players. In this way, the use of collective intelligence [11] helps finding solutions good enough for problems where no optimal classical solutions are known (in some cases, indeed the situation is worse and it is known that no optimal solution can be found at all).

Summarizing these large range goals, the framework we present here is the first step towards a tool that, at least in games contexts, extracts tacit knowledge on the problem (game) from the implicit knowledge of the collectivity (a huge number of players). With this aim, the system provides data (game logs) obtained from gamified models based on the features and behaviors of Complex Systems, in order to gain some insights about the dynamics of their evolution and to provide robust ways for controlling it. The data acquisition capabilities that the framework includes are only a first step forward in this direction, allowing researchers to apply external data analysis tools to obtain knowledge from players' behavior and the system dynamical evolution. Also, in order to have a good penetration in the market and gather as much users as possible, the first version of the framework has been developed for Android systems, as being one of the most active, popular and open mobile OS.

The structure of the paper is as follows. The next section provides an overview of the framework for rapid development of gamification of complex processes, its main goals and some notes about the architecture of the framework (named ASAP). In section III, some examples on Complex Systems Gamification using ASAP framework are presented. The last section is devoted to conclusions and future work.

## II. ASAP: A Framework for Easy Development

Several challenges must be faced in the ASAP framework:

- *The transformation of (experimental, scientific, mathematical) models into gamified models*: as primary solution, ASAP provides a framework for a smooth transformation of a number of modelizations on Complex Systems topics into a mobile platform, enabling the transformed models to be collectively used, explored and (hopefully) solved.
- *Exploiting collective intelligence to study complex behavior*: as secondary challenge, ASAP framework provides easy ways to record player interactions with the developed games in order to apply Data Analysis methodologies and obtain deeper insights on the real complex system from players' collective behavior.

The next subsections provide a more detailed overview of the framework architecture, and propose a global methodology for real Complex Systems' gamification.

### A. Architecture

The framework consists of four main modules (Java packages) encapsulating the functions needed for the different gamification tasks necessary to create a complete game (see Figure 1):

- **MAS (multi-agent system)**: implements the basic structures necessary to model a complex system in a similar way NetLogo does, a popular multi-agent programmable environment [14, 20]. This module comprises the following main classes:
  - *AbstractModel*: represents a raw model of the complex system. In this first version, we have reduced the options in a way that each model is composed of a grid of motionless agents (called Patches in some contexts). In future versions the framework will include the possibility of working with mobile agents.
  - *Patches*: represents a grid of agents with some specific width and height. In the current version, only regular grids with 8 neighbors for every patch (Moore's neighborhood) are allowed. In future versions, other topologies and neighborhoods will be considered in order to extend modelization possibilities.
  - *Patch*: is a single motionless agent in the grid. A list of properties can be defined for it, for example, storing a color or a list of sprites for visual representations, as well some other feature, the data structure will need to model the system.
- **Drawing**: contains the drawing classes to render the current state of the model(s) on the screen. It consist of one main class (it can be extended for different render options in the future):
  - *OpenGLDrawer*: draws on the screen the current state of the model(s) by using OpenGL [13]. When working with sprites, this class will use SpriteBatcher to preload them on memory and speed up execution time.
- **Gamification**: implements the classes that allow extending the model with playable concepts. Managing achievements, leader-boards, or synchronizations in the case of a multiplayer games can be represented in these

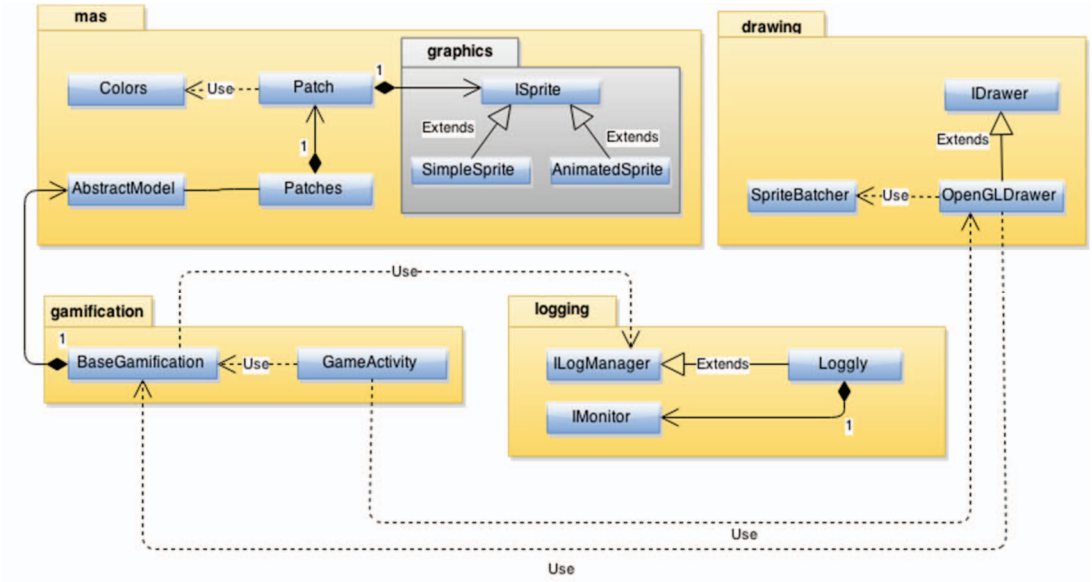


Figure 1. Architecture of ASAP Platform

classes. It includes only one main class:

  - *BaseGamification*: used by the drawer to access the model(s). This class must be extended by adding the gamification logic (synchronization, achievement, etc.) into it.

- **Logging**: performs the recording of the information and interactions the developer chooses for being analyzed.

### B. *ASAP Methodology for (mobile-based) Gamification*

The proposed methodology to build a game from a complex system model by means of ASAP framework comprises a number of activities, covering both levels of features (see Fig. 2). Next, we detail the phases followed in the different cases carried out, climbing from the features corresponding to the underlying complex system, to the features associated with the gamification concept, that is to say, the interaction between players (humans) and the first level model:

- **Agent-Based Modeling (ABM) interpretation of the problem**. In this phase, the designers have to interpret the complex system by identifying which elements must be considered agents and which interactions between them should be modeled. Also, it is frequent to consider some kind of environment where agents' dynamics are developed. This activity is critical and it is possible to re-use existing ABM models from other platforms. Those implemented in well documented Models Library of other platforms like NetLogo [19]. In this way, it is possible to provide a (first level) model tightly linked to the real system. Therefore, the main question to be considered by designers is: Which are the main features of the original model to be maintained?

- **Gamification of the model: meta-agents**. In this phase, the designers define player capabilities on the model. The player can be seen as a meta-agent with the ability to change certain instances within the previous model (both agents and environment) by means of some allowed interactions with the model (influencing their dynamics and evolution) or with another player(s). From a theoretical point of view, in this phase we must decide, within a Perceptions – Actions – Goals – Environment (PAGE) specification [10], the high-level player actions. In this stage, the main question the designers must have in mind is: Which are the best gamification decisions for studying the main features of the original model?

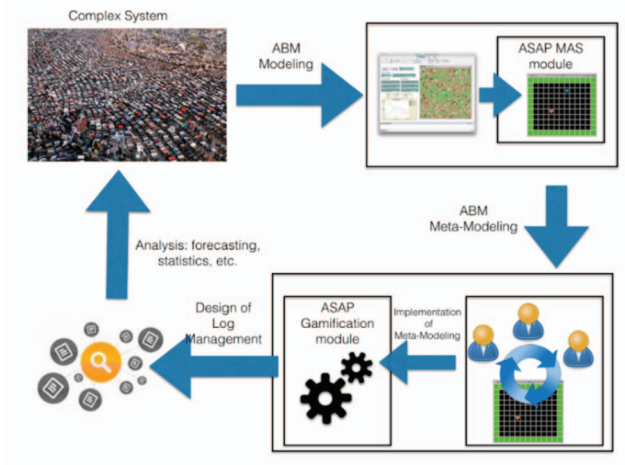


Figure 2. ASAP Methodology for ASAP Platform

- **Design of the meta-environment**. In this phase of the methodology, the ASAP architecture plays a key role. ASAP framework abstracts the communication system that allows the development of multiplayer games (as sending and receiving messages and error handling). In this way, developers can focus on the communication protocols in their games, avoiding managing low level details. Fig. 3 shows how simple is sending and receiving data from the opponent(s), by means of the abstract method *processDataReceived* from the class *BaseGamification*.

- **Design of the log management for Data Analytics**. In this phase, the data model to represent game logs information must be selected in order to exploit such data and get insights on the game itself, the behavior of the players, and the complex system source model. ASAP provides a raw storage system, although connections with other log management platforms (e.g. [18]) can be used.

- **Analysis and interpretation**. In this phase, interpretations of the data collected from different model executions and player interactions, in terms of the real complex system, must be given. Currently, this activity must be performed outside ASAP framework, and the designer can use one of the multiple software/services available for this purpose[1]. The implementation of these functionalities is not contemplated on future iterations of the framework.

In the last phases of the methodology we must have in mind the following fundamental question: How can one learn from player interactions in order to get deeper insights on the model?

```
private void processDataReceived(int type, bytes[] data) throws IOException {

   switch(type) {
      case PATCHES:
        processOponentPatches(data);

      ...
      ...
      ...
   }
}

private void sendMyPatches() throws IOException {

   RealTimeMultiplayer.sendData(PATCHES, patches.encodeBytes() );
}

private void processOponentPatches(bytes[] data) throws IOException {

   Patches opponentPatches = Patches.decodeFromBytes(data);
   ...
   ...
   ...
}
```

Figure 3. Code snippet for data interchange between players

[1] See, for example, RStudio (www.rstudio.com) for R based tools, or PyData (pydata.org) for Python based tools.

## III. SOME EXAMPLES

In this section, some basic examples about how to use ASAP framework for the different stages of the previous methodology are presented. As starting point, one of the simplest and most popular Complex Systems model has been chosen, and only a direct implementation will be provided, as preliminary case study. Further, one possible adaptation for multiplayer gamification of this raw model is presented. And finally, the last subsection introduces one simple model showing log management capabilities of the framework.

### *A. A basic example to describe the system: Game of Life*

Roughly speaking, any modelization tool for studying Complex Systems follows the steps described in the previous section. To illustrate how this process has been implemented in ASAP, a classic complex system, Conway's Game of Life (GoL), has been selected [3].

This model consists of patch agents on a square grid with only two possible states: alive or dead. In the original version, GoL is a zero-player game, meaning that its evolution is determined by its initial state, requiring no further input. In this case, the only permitted interaction with the game consist in creating an initial configuration and observing how it evolves.

Formally, GoL is a cellular automata with the following basic rules driving its dynamics:

- **Survival**. Every agent with 2 or 3 alive neighbors survives in the next time step.
- **Death**. Each agent with 4 or more alive neighbors dies. Every agent with 1 or zero neighbors dies by isolation.
- **Birth**. Each empty cell adjacent to exactly 3 alive neighbors - no more, no less - is a birth, and an agent will be placed on it at the next time step.

Fig. 4 shows the code snippet implementing these rules for GoL in the framework that will be executed in each simulation step (*generation*).

```
public synchronized void process_tick() {
    for (Patch p : patches) {
        //count the living neighbors
        int living_neighbors = countLivingNeighbors(p)
        if (!p.getBoolean(IS_LIVING)) {
            if (living_neighbors == 3) {
                //next state will be alive
                p.setProperty(WILL_LIVE, true);
                //select a color from the neighborhood
                p.setColor(getNeighborColor(p));
            }
            else {
                p.setColor(deathColor);
            }
        } else {
            if (living_neighbors == 2 || living_neighbors == 3) {
                //next state will be alive
                p.setProperty(WILL_LIVE, true);
            }
        }
    }

    for (Patch p : patches) {  //update next state
        p.setProperty(IS_LIVING, p.getBoolean(WILL_LIVE));
        p.setProperty(WILL_LIVE, false);
    }
}
```

Figure 4. Code snippet for GoL rules

This model is simple enough to be implemented in a direct way, and presents interesting dynamics where, for different initial conditions, it is not possible to decide how the system will evolve.

In the implementation made for ASAP, the user can add new agents in real time (during the evolution of the system), by interacting directly with the screen, starting with an empty world (see Figure 5). It provides a first sample of a basic interaction game, with well-defined goals (survival), where it is possible to measure the capabilities of the player predicting and generating good patterns for population survival.

### *B. A Two-Player Game of Life*

To illustrate the potential of ASAP as a (mobile) gamification tool, a two-player version of GoL [8], called *p2life*, has been implemented (see Fig. 6). Some variants of this game have been previously developed for mobile gaming[2].

Among other interesting features, two-player versions of this kind of models are very useful to explore heuristics for studying (and controlling) system's dynamics. In this case, implementing a competition-based version in a mobile environment aims to maximize the attractiveness of the game for the user and, consequently, to increase the amount of data collected from him/her.

The rules of the gamified model are described in [8], where the asymptotic behavior of the system is analyzed. The first level model runs without human interplay, that is to say, the system starts from an initial population and its evolution continues without interactions.

In order to recognize patterns useful to predict and control the evolution of the system, the model allows the interaction between two players (each one controls a different species, recognized by shapes and colors) trying to overcome the opponent's population. In [8], authors claim to devise *p2life* into a game where players are allowed to make moves between generations, which change the configuration of the tokens, and devise strategies to overpower or live side-by-side with their opponent.

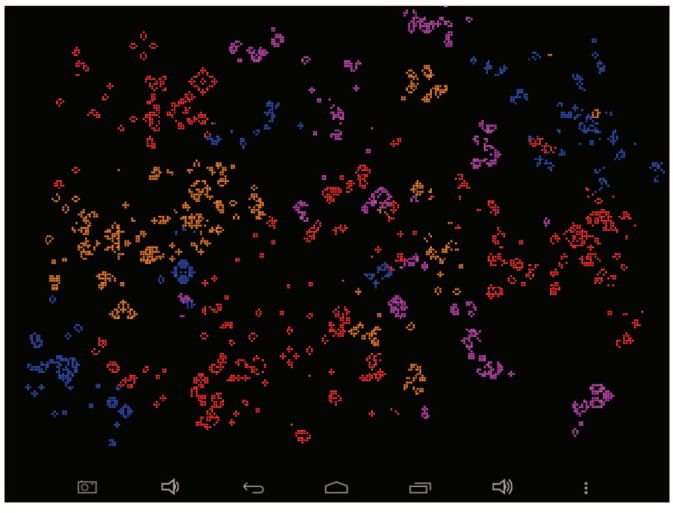

Figure 5. Screen of Game of Life developed on ASAP

[2] See e.g. http://behaviorgames.com/games/colonize

The game starts from an initial configuration where each player's model sets randomly his agents in the world (coincident agents are removed from both groups). Game dynamics are driven by the rules from [8], which are an adaptation of classical GoL. From the point of view of player A (they are symmetrical for the other player) these rules are:

- **Birth**. If a cell is empty, and it has exactly 3 alive agents of A, and the number of neighbors from B is different from 3, then an A agent is born.
- **Survival**. If a cell is occupied by A, two cases can be considered: If the difference between the number of A agents and B agents is 2 or 3, then the A agent survives. If this difference is 1 and the A neighbors are at least 3, then the A agent survives.
- **Death**: If a cell is occupied by A, the difference between the number of A agents and B agents is 1 and the number of A neighbors is not equal to 3, then the A agent dies.

During the game, each player has a limited number of allowed interactions with the world. For each interaction each player can add new agents to the world in order to increment its own population or to decrement opponent's population during the future evolution of the game.

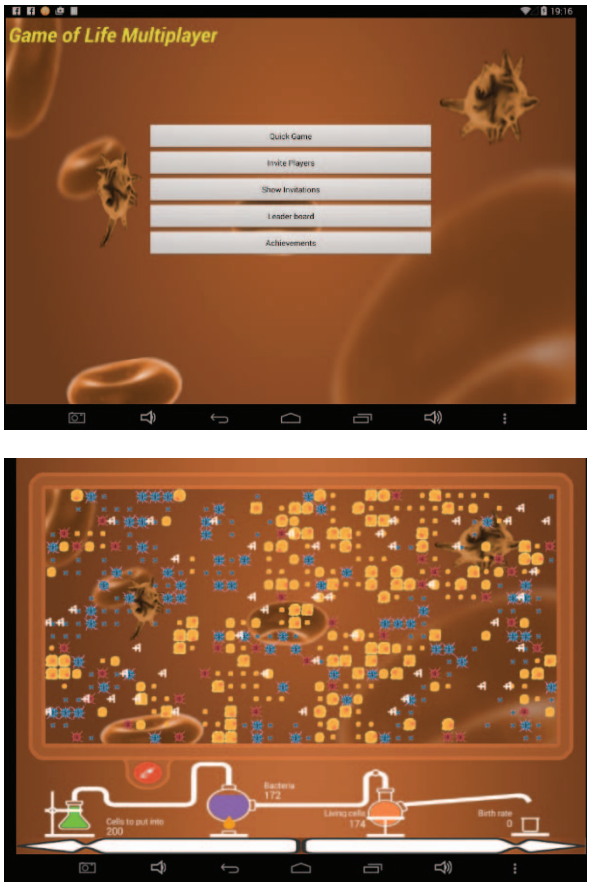


Figure 6. Up: Screen with the main menu of the game (invite opponent, look for random opponent, show invitations, show leaderboard and achievements). Down: Gamified version of GOL (p2life) using sprites and graphical scoreboard.

The game considers mainly two measures to calculate the score of the players: the growth rate of their population and the number of killed opponent agents.

### C. *Fire Spreading: A Data Acquisition Example*

This last example focuses on the data acquisition tools that ASAP framework provides to developers. In this case, the idea consists in gamifying a model that simulates the spreading of fire on a forest with a very simple representation of the world by using just a grid of patches. Green patches represent alive trees, red patches represent burning trees, and black patches represent areas with no trees (see Fig. 7).

Once a fire has started in a tree (a green patch) it will spread to every neighboring patch containing a tree (according to von Neumann neighborhood, i.e. north, east, south and west directions). The gamification of this model will allow the user to choose the starting point of the fire and he/she will score depending on how much proportion of the forest has been burnt and how fast the fire has spread. The goal is to measure how the human intuition is able to predict the evolution of complex behaviors from initial conditions, as well as to serve as training model to improve these predictions.

Since the gamification stage is very similar to those from the previous samples, this section focuses only on showing how to set the data acquisition up for the model. The choice of the information that will be recorded from the games for the subsequent data analysis is made by using a simple XML file where it is possible to define the record containers (logger) and each one of the data items to be saved (it is necessary to specify *when* it must be done and if any *preprocessing* over the data is needed). Fig. 8 shows a valid data acquisition file for this model.

Later, ASAP will use this XML file to log all this data from the model in execution time.

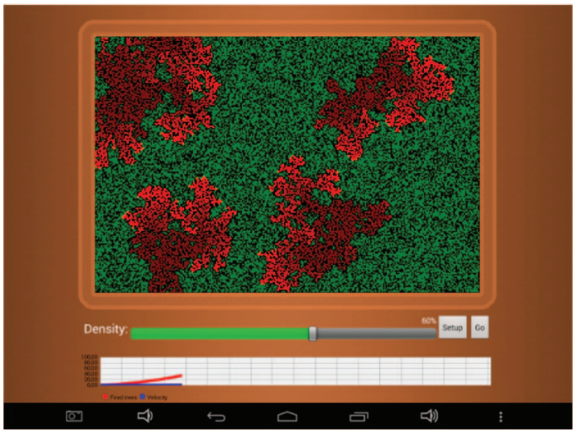


Figure 7. Screenshot of Fire Spread gamification on ASAP

```
<logging>

   <logger>loggly</logger>

   <monitor what="datetime" when="start"></monitor>
   <monitor what="patches" when="after_model_initialization" how="base64Encoder"></monitor>
   <monitor what="tick" when="each_tick"></monitor>
   <monitor what="firedTrees" when="each_tick"></monitor>
   <monitor what="velocity" when="each_tick"></monitor>

</logging>
```

Figure 8. Data acquisition file for Fire Spreading Model

## IV. Conclusions

In this paper ASAP, a framework for rapid prototyping and development of mobile gamifications of Complex Systems, has been presented. The main goal in the creation of this framework has been to provide an environment where the process from the original model of a specific complex system to a final gamified version is as soft as possible. In this way, the developer can focus on some fundamental questions to be answered: Which are the main features of the original model to be maintained? Which are the best gamification decisions for that? How can one learn players' interactions to get deeper insights on the model? How can we use this information to create artificial players (opponents)?

From the beginning of the development of ASAP, one of the main concerns was about taking advantage of collective intelligence to provide solutions good enough to problems related with the dynamics of Complex Systems, and trying to predict some specific features of their evolution. The data acquisition capabilities that ASAP includes are a first step forward in this direction, allowing the researchers to apply data analysis tools in order to extract knowledge from the behavior of players and the evolution of the system.

The classes of the framework in charge of the Complex Systems modeling are based on those from several popular Multi-Agent Systems Modelers ([15, 16, 17, 20]), facilitating the access to a huge amount of interesting models that now can be easily gamified [19].

In this paper three simple examples have been presented to show the capabilities of the current framework version. Nevertheless, in order to meet standard Multi-Agent Systems modelers, new features are currently being added to the next version: mobile agents, link or connection agents for complex network modeling, and different world topologies (3D, hexagonal, triangular, etc.).

Finally, it is worthy to mention two non-exclusive lines of ongoing work. On the one hand the aim pass through producing a number of games from complex systems of different nature in order to evaluate the soundness of ASAP as development platform. On the other hand, Data Science methodologies will be applied to analyze logs of concrete models in order to discover (rational) heuristics to understand real complex system dynamics.

## Acknowledgment

This work has been partially supported by TIC-6064 Excellence Project of the Junta de Andalucía and TIN2013-41086-P from Spanish Ministry of Economy and Competitiveness (co-financed with FEDER funds).

## References

[1] A. Bilsen, G. Bekebrede, I. Mayer. Understanding Complex Adaptive Systems by Playing Games. Informatics in Education, 2010, Vol. 9, No. 1, 1–18.

[2] B. Burke, Gartner Redefines Gamification. http://blogs.gartner.com/brian_burke/2014/04/04/gartner-redefines-gamification.

[3] M. Gardner. Mathematical games: The fantastic combinations of John Conway's new solitaire game "life". Scientific American 223, October 1970, pp. 120–123.

[4] L. Goasduff, C. Pettey, Gartner Says By 2015, More Than 50 Percent of Organizations That Manage Innovation Processes Will Gamify Those Processes. http://www.gartner.com/newsroom/id/1629214

[5] J. H. Holland. Complex Adaptive Systems. Daedalus, Vol. 121, No. 1. A New Era in Computation, 1992, pp. 17-30.

[6] J. H. Holland. Hidden Order; How Adaptation Builds Complexity, Addison-Wesley, 1995.

[7] K. M. Kapp. The Gamification of Learning and Instruction: Game-Based Methods and Strategies for Training and Education. Pfeiffer \& Co., 2012.

[8] M. Levene, G. Roussos. A two-player game of life. International Journal of Modern Physics C (IJMPC), 14, 2003, pp. 195–202.

[9] M. Mitchell. Complexity: A Guided Tour. Oxford University Press. 2011.

[10] S. J. Russell, P. Norvig, Artificial Intelligence: A Modern Approach (3 ed.). Pearson Education, 2010.

[11] T. Segaran. Programming Collective Intelligence: Building Smart Web 2.0 Applications. O'Reilly Media, 2007.

[12] E. Sheely. Systems Based Gamification Volumen II: Complex Systems, 2014. http://www.slideshare.net/EugeneSheely/systems-based-gamification-complex-systems.

[13] The Industry's Foundation for High Performance Graphics. https://www.opengl.org.

[14] U. Wilensky, W. Rand. An introduction to agent-based modeling: Modeling natural, social and engineered complex systems with NetLogo. Cambridge, MA: MIT Press, 2015.

[15] MASON Multi-agent Simulation Toolkit. https://cs.gmu.edu/~eclab/projects/mason.

[16] Swarm Multiagent System. http://www.swarm.org.

[17] Gama Platform. https://code.google.com/p/gama-platform.

[18] Loggly service. https://www.loggly.com.

[19] NetLogo Models Library, http://ccl.northwestern.edu/netlogo/models.

[20] NetLogo, http://ccl.northwestern.edu/netlogo.

[21] M. Sysi-Aho, A Game Perspective to Complex Adaptive Systems, PhD. Thesis, Helsinki University of Technology (Espoo, Finland), 2005. http://lib.tkk.fi/Diss/2005/isbn9512277328/

[22] G. Zichermann. Game-Based Marketing, Wiley Publishing, 2010.

[23] G. Zichermann. Gamification by Design, O'Reilly, 2011.