LEGO, PICTIONARY AND POKER: LESSONS LEARNT FROM GAMIFICATION FOR INFORMATION SYSTEM TEACHING AT UNIVERSITY

Sophie Dupuy-Chessa, Eric Céret, Gaëlle Blanco-Lainé

LEGO, PICTIONARY AND POKER: LESSONS LEARNT FROM GAMIFICATION FOR INFORMATION SYSTEM TEACHING AT UNIVERSITY

Sophie Dupuy-Chessa[1], Eric Céret[1,] Gaëlle Blanco-Lainé[1]

[1]Univ. Grenoble Alpes, IUT2, F-38000 Grenoble, France
Sophie.Dupuy@imag.fr, Eric.Ceret@imag.fr, Gaelle.Blanco-Laine@iut2.upmf-grenoble.fr

Abstract

Our context is the teaching of information systems at the license level of a technical institute. It requires technical knowledge as well as managerial one. In this context, an important part of the teaching concerns modelling of data that will be manipulated by the system and projects management for building systems. To facilitate learning and improving students' results, we investigated the Elaboration phase of the BSCS 5E Instructional Model as well as the Concrete Experience stage of Kolb experiential learning cycle and we experimented gamification in some of our teaching activities. This paper presents four case studies in gamification for information systems teaching: one of them is related to the learning of data modelling with a personalized version of the Pictionary, the two others concern project management with Legos and Poker. These case studies clearly show the usefulness of gamification in learning.

Keywords: Technical institute, information systems, games, engagement.

1 INTRODUCTION

We teach computer science at the license level of a technical institute. In this context, some of our teaching modules concern Information Systems (IS). Information systems gather together management and informatics techniques to efficiently identify, store and share information inside a company. They thus require technical knowledge as well as managerial one. In this context, an important part of the teaching concerns modelling of data that will be manipulated by the system and projects management for building information systems.

License students often feel that these notions are very difficult to learn because they require dealing with abstract concepts such as models and resources management. To facilitate learning and improving students' results, we investigated the Elaboration phase of the BSCS 5E Instructional Model [1] as well as the Concrete Experience stage of Kolb experiential learning cycle [2] and we experimented gamification in some of our teaching activities. In this case study, the main objective was to increase students' engagement in learning by adapting games, which, in their original version, already promote collaboration and emulation. Contrary to several existing experiments, we chose to use traditional games that are well known by our students. We did not use computer games because our students are already highly involved in activities with computers for their technical training: in this perspective, computers are professional tools but not supporting tools. Our guess was that students needed alternative ways to access information.

This paper presents three case studies in gamification for information systems teaching. It details the methods used with the gamification elements and specifies the lessons we learnt for each case study.

The second section of the paper presents a first case study, which uses the Pictionary game for teaching data modelling. The third section describes two case studies for project management. Finally, we discuss the lessons learnt from all these case studies before concluding and presenting perspectives.

2 PICTIONARY FOR DATA MODELING

2.1 Context and goals

The first case study takes place during the first year of license. Each gaming session concerns a group of 13 computer science students. It takes part of a teaching module about data modelling. Data modelling is a process of software engineering, which aims at identifying and describing data that need to be stored and manipulated by the software. For instance, a software managing students will deal with data such as the student's name, number, date of birth, degree. These data are represented graphically in diagrams such as the one presented in Fig. 1. Students are represented as a class (i.e. a structured piece of information) with a number as characteristic (attribute). Student inherit name and date of birth from the class Person. Student is linked through an association to the Degree class; the association carries the number of links between the two notions: for instance, the 1..* on the Student to Degree association means that a student can be registered in several degrees, i.e. one per year. Each relation between a student and a degree also carries information about the student's registration in this degree: this is represented by the class Registration: this kind of class related to an association is named an association class. A degree is composed of modules: this is represented by a special association named composition. This simple example illustrates the complexity of modelling with notions such as inheritance (between person and student), composition (between degree and module) or association class (e.g. registration).

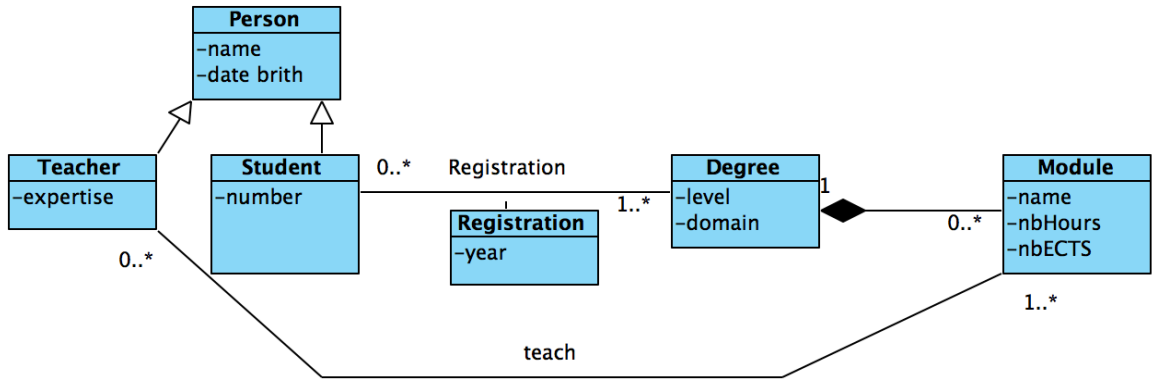

Fig. 1- Simple example of data model

Several approaches can be used for data modelling, particularly the Unified Modeling Language (UML) [3], which is a standard. It proposes models for representing software functions, data, architecture… The learning of UML require 1) to make abstractions so that to identify the main concepts in the "universe" the software is dealing with; 2) to describe these abstractions graphically, following the rules of the UML language, 3) to understand the technical problems in order to describe the corresponding solutions. So modelling is a very difficult task, particularly while considering subtle notions like inheritance or composition. Moreover as UML models are not "runnable", most students do not appreciate it. Their emotion varies from boredom to indifference even if some of them show interest.

We introduced a game to make modelling more engaging and to encourage students in using it. It takes place before the exam in the application phase of learning for checking whether students have acquired some automatic reflex in modelling. We experimented the game approximately 10 times.

2.2 Game description

The case study relies on a personalized version of the Pictionary[1], a game originally promoting speed and collaboration. In Pictionary, a team has to identify the word drawn by a teammate within a minute, sometimes challenging other teams.

[1] http://people.irisa.fr/Francois.Schwarzentruber/mit2_cvfp_2012/uml_pictionary_cards.zip

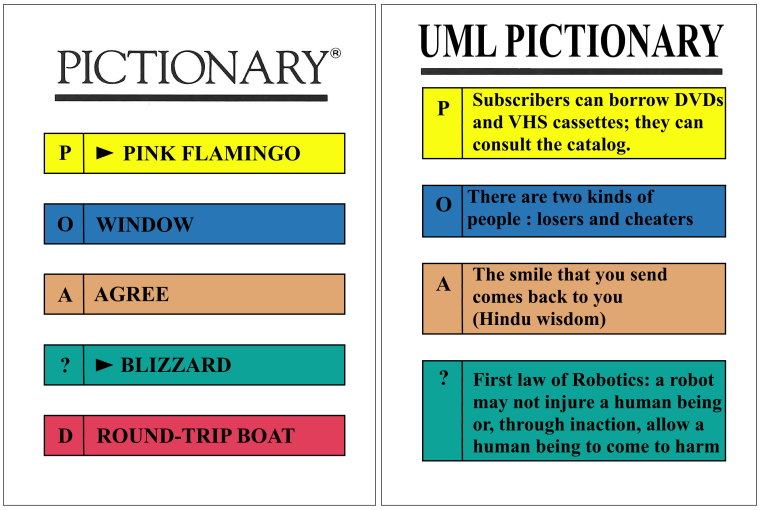

Fig. 2 – Examples of (a) a classical Pictionary card (on the left) and (b) an UML Pictionary card (on the right)

In our modelling context, students have to create models instead of drawings. But the game rules are identical. There are 4 teams of 3 or 4 players. At each turn, a team takes a card; one player plays the role of drawer and has to make a UML model corresponding to the sentence written in the card. In the traditional Pictionary, there are 4 kinds of subjects: person/place/animal, object, action, difficult and "all play" (see figure 2a). In the UML version, the subjects are related to the different kinds of models. There are class diagram, use case diagram and sequence diagrams, as shown on figure 2b. We also keep the "all play" subject.

The model, as shown on figure 3, is realized on a paperboard so that all players can see it. If one other member of the team says the expected sentence, the team wins and can go some steps forward. There are some specific squares on the game board for challenges where all the teams model at the same time. In this case, each team uses its own sheet of paper to draw its model. The fastest one wins the turn. The winner of the game is the team that arrives at the final square.

During the game, at the end of a turn, the teacher can correct very quickly some models or explain some good solutions. However to keep the game spirit, students are not supposed to take notes or corrections.

The gamification elements we used here are 1) small challenges with time-pressure and competition so as to progress and to arrive first at the end of the game and 2) a reward i.e. a packet of sweets for the winners.

2.3 Lessons learnt

First, lessons learnt concern how to practice with UML Pictionary. The main difficulty is to find the right balance between gaming and learning. On the one hand, to encourage gaming and engagement, teachers must limit their interventions to the minimum; teacher's interventions can indeed lead

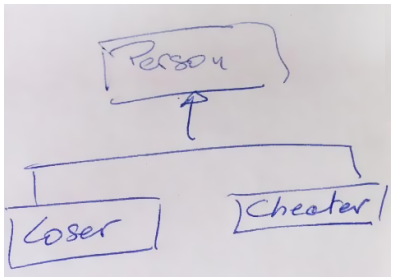

Fig. 3 – Example of answer to the question #2 on fig. 2.

students to wait for a given solution. On the other hand, for encouraging learning, they should correct models and let students take notes about the discussions and correction. So during the game, we decide to quickly correct models, if necessary, just to give an idea of one or another solution. Students are not asked to take notes. They must listen and understand the correction.

Secondly, we have learnt lessons about the pedagogical aspect of the game. The first lesson is that students appreciate the game and really play it. One of them even cheated to win! So gaming really improves engagement.

It is a pleasant way of reviewing concepts for the exam. As a matter of fact, the game is appropriate for players who know UML. The idea is to apply the notions quickly and to get to the key point. It allows students measuring their level of acquisition in UML and confronting their knowledge. For instance, sometimes students discussed about the choice of one notion rather than another (e.g. an association versus a composition), each one explaining to the other one what he/she has understood about the notion. The teacher is here to (eventually) correct and complete explanations.

The last benefit was that students understood how crucial it is to make clear and understandable models. Using the wrong notation, making an error in the choice of model component directly leads the teammates to misunderstand the model and makes the team lose. Therefore, all students were highly involved in making "good models", leading them to refine their understanding of the differences between the notions.

3 GAME FOR PROJECT MANAGEMENT

This section presents two case studies for learning software project management. One of them used Legos as a mean to practice for knowledge acquisition. Legos are used for introducing development life cycles. In the other game, Legos are used for resources management, especially for estimating workloads and risks inside projects. In the same manner, Poker game is used for helping to understand resources management. The design of the session was largely based on the work proposed by Alexey Krivitsky [5]. Each case study is described in a following subsection.

3.1 Legos for development life cycles

3.1.1 Context and goals

The first case study takes place at the second year of license. It concerns a group of 17 students with background in computer science and statistics. It takes part in a teaching module about web development. In this module, students will have to build a website in pair. In order to help them organizing their project, the module starts with an introduction of development life cycles.

A life cycle is a framework of processes and activities that may be organized into stages. It also acts as a common reference for communication and understanding of the project. Many different life cycles are used in computer science. To make students understand their differences, we use two opposite cycles:

- the Waterfall model [4], which is linear (Fig. 4a). One stage follows another one and there is no feedback to a previous stage. It is based on the creation of documents from requirements analysis to design before starting activities like implementation or testing.
- An agile cycle, which is iterative and incremental (Fig. 4b). The Agile Manifesto [6] defines twelve fundamental principles such as "*customer satisfaction through early and continuous delivery of valuable software*" or "*welcome changing requirements, even late in development*". Here we use the SCRUM cycle [7]. SCRUM focuses on user needs and suggests a short iterative development cycle that can be adapted according to changes occurring in the customer needs. Once a global list of features is created (Product Backlog), one or several of them are realized during an iteration. The complete software is produced thanks to several iterations. Changes can be made at each iteration.

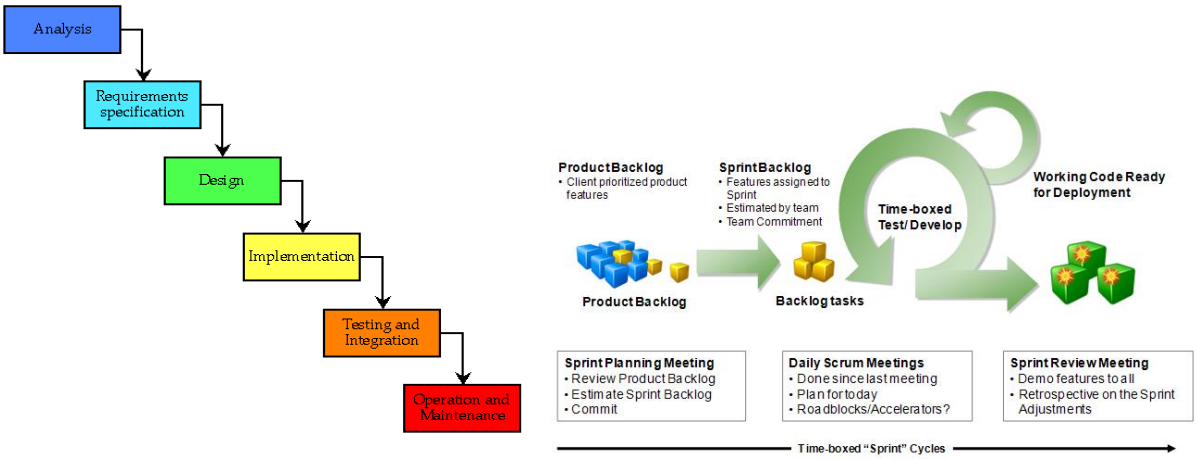

Fig. 4 – (a) Waterfall cycle (on the left) and (b) SCRUM agile cycle (on the right)

The learning of life cycles usually requires a strong experience in projects that students do not have. It is very difficult for them to imagine how a software (and more globally a product) can be built by several persons in a given delay. It makes them feeling confused.

We introduced a game for helping students acquire some experience in project building and to make them understand notions that are not understandable without some practice. We experimented the game with two groups of 17 students.

3.1.2 Game description

After a short introduction to life cycles goals and some vocabulary, we proposed students to play two games with Legos in groups of 4 to 6. The goal was the creation of a set of buildings, constituting a whole infrastructure, in Legos. There were two sessions of 90 minutes, one with the Waterfall cycle and one with the SCRUM cycle. The groups could not change between the two sessions.

At the beginning of each session, a short presentation of the cycle to apply and of the goal of the game were given. Then groups had to follow instructions to build their complex. They were three phases of 15 minutes.

The first session concerned the Waterfall cycle. Students were asked to build a holiday village. During the first phase, they had to identify the list of buildings and some construction constraints. During the second phase, they had to specify with drawings or schemes their village. During the last phase, they built the village with Legos. Finally, they presented their work to the other groups.

The second session used the SCRUM cycle. Students had to build the ideal university campus. During the first phase, they had to identify requirements. During the two other phases, they had to build the campus. As changes are welcome in Agile cycles, the teacher made comments during all phases for pointing out problems or refining requirements.

After the two sessions, a 2 hours debriefing session allowed students to discuss about their experiences and to identify the drawbacks and advantages of each cycle.

The gamification element we used here is mainly small challenges with time-pressure. They also enjoy seeing other groups' constructions.

3.1.3 Lesson learnt

This case study was realized at the very beginning of the second year of license. It constituted the first lectures in computer science for this year. So they were very important for the relationship between teacher and students. It allowed all of us to start the year in a pleasant manner. Students really appreciated it. They were curious and enlightened by this new way of teaching. They were engaged in the sessions and tried to capitalize on the Lego experiences in order to have a critical analysis about life cycles during the debriefing. The debriefing session was a place of exchange and sharing, which brought unexpected discussions about concepts.

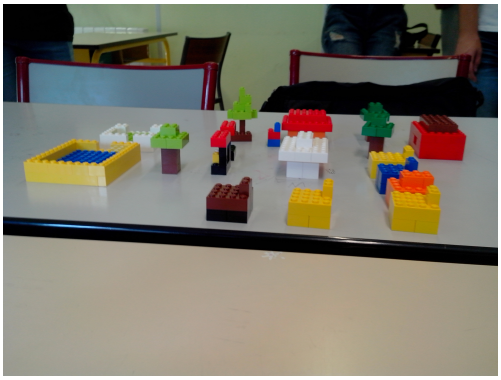

Fig. 5 – (a) Unfinished holiday village (on the left) and (b) Bad global architecture without space between buildings (on the right)

The Lego sessions helped students to understand notions that are not understandable without practice. First, students understand in deeper details each cycle. For instance, while applying the Waterfall cycle, a group wanted to add a new building at the last phase i.e. during the building of the holiday village. But adding a new requirement at this stage is not possible in a Waterfall cycle. In the SRUM session, one group has built an amusement park inside the campus without asking the client if he/she wants it. This is not possible in an agile cycle where stakeholders must be part of any decision and give their opinion before developing new features. The experience showed students what is possible and not possible in each development cycle by doing the parallel between building Lego villages and software.

It also allowed them to identify some drawbacks or limits of cycles. For instance, one group did not succeed in finishing the building of their holyday village whereas the whole plan was ready (Fig. 5a). The lack of time at the end of the project is a well-known problem in the Waterfall cycle, which was clearly identified by students. In the same manner, students succeeded in identifying advantages of each cycle. For instance, the fact of iterating on a construction is clearly seen as an advantage of agile cycle, but they also identified that it can lead to a bad global architecture if this is not considered at the beginning of the project (Fig. 5b). In conclusion, the sessions really helped student in understanding the cycle principles and their differences. This cannot be learnt during a theoretical lecture.

3.2 Legos and poker for project management

3.2.1 Context and goals

The first year of computer science training includes a module about software project management. This module aims at teaching students how to work in an efficient manner in the context of software projects. Students should acquire the basic skills of a software project leader: they must be able of organising and planning successful projects as well as taking into account and monitoring all the resources and constraints of the project, in accordance with costs consistency, deadlines and quality.

The module organization is classic: it includes the presentation of the concept of project, of the various stakeholders, of the activities related to project management including the choice of software development life cycles and workloads estimation. We will call a development method the process that gathers the use of a life cycle with resource planning such as workloads estimation.

In this context, two activities based on gamification are used: poker for estimating workloads and Legos for organizing and realizing the sequences of stages. In the previous case study with Legos, gamification were used for understanding stages sequencing (i.e. development life cycles), not for the resource planning as it is done in this experiment. Gamification replaces a more traditional lecture on principles and techniques about these project management activities.

As for the previous case study with Legos, the main difficulty of this module is related to the students' lack of business experience. The goal of gamification was to make practices and challenges of project management more real, going beyond the experience feedbacks delivered by professionals (in videos,

for example). The use of games, here the Legos and poker, illustrates the concepts in order to show the application of project management principles and their limitations.

As in the previous experiment with Legos, small challenges with time-pressure are the elements of gamification.

3.2.2 Game description

The game took place over a 2 hours session. It followed:

- a lecture about software development life cycles in general,
- a lecture on agile development methods and especially the presentation of SCRUM, starting from Henrik Kniberg video realized in 2012, "Product Agile ownership in a nutshell"[2].

The teacher, in a SCRUM logic, took the role of product owner (i.e. the customer), who is typically a project key stakeholder. The Users Stories i.e. the expression of requirements with scenarii were simple, and based on the classical format of the construction of a city. The requirements were used for workload estimation using the poker, and for the organization and realization of stages sequences (names sprints in SCRUM) using Legos.

Poker, already used in the context of agile development methods, allowed highlighting the importance of communication while estimating workloads. This method, described by James Grenning [8] is based on the comparison of estimations by different persons who argue their motivations. As the method claims that the bigger the task to realize the fuzzier the estimation, it uses the Fibonacci sequence for evaluations: evaluators have to choose a number of working days within the possible values defined by the Fibonacci sequence (i.e. 1, 2, 3, 5, 8, 13, 21, 34, 55, 89).

The teaching session lasted 2 hours and was structured in two phases:

- Task definition, planning and sequencing: during this phase, Poker was used for load estimate;
- Iterative construction: here, Legos allowed students to build and refine their city. At each iteration, they had to decide which buidings must be built or changed during the iteration. Some of them chose to start the city by public infrastructure like hospitals and schools while others preferred residential area. At the end, they are supposed to propose a complete city. But it was not always the case due to time pressure.

The first phase, in a predictive sense, aimed at defining and delimitating the project, as well as defining the work organization. The second phase addressed the realization of the defined project. We have imposed to follow an iterative and incremental process (Fig. 6).

In the sprints reviews (at the end of each iteration) and at the end of the session, a debriefing was done.

3.2.3 Lesson learnt

From the use of Poker for workload estimation and Legos for project management, several observations can be done. The introduction of a game session caused a break in the teaching pace and methods and kept the students' attention and curiosity aroused.

[2] please, see *www.youtube.com/watch?v=502ILHjX9E*

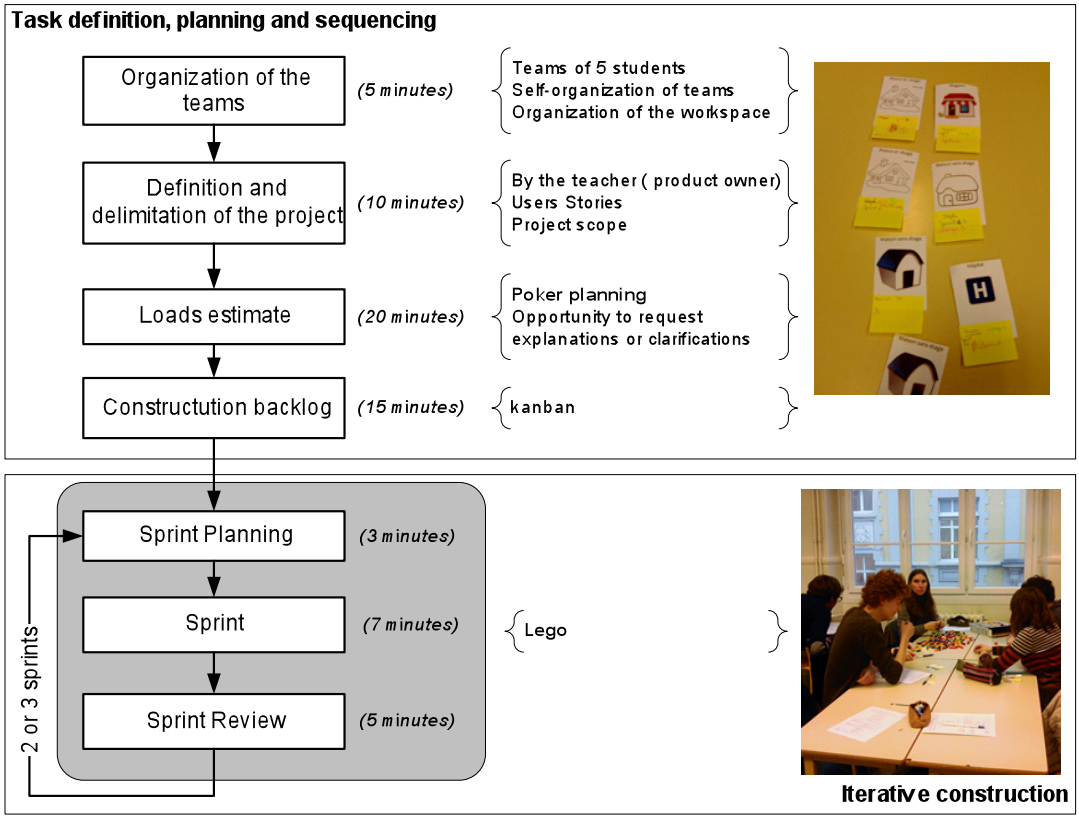

Fig. 6 – Game session organization

This gaming session succeeded in:

- Establishing links between previously presented concepts, like workloads estimation, development cycle or team organization;
- Giving meaning to some definitions related to development methods, such as the concepts of increment and iteration;
- Highlighting the importance of communication in the context of software project, not only between product owner and project team but also inside the project team.

It was also used to introduce, in the stages sequence, the concepts that were discussed in more traditional ways, especially the theme of analysis and risk management.

Students enjoyed this way of practicing, they were more engaged in learning and they interacted more with the teacher even after the game. However, as students of different ages were involved, we noticed that mature people are less prone to play for learning.

4 CONCLUSION AND PERSPECTIVES

Contrary to new practices, which use computer games for learning, we used traditional games for teaching computer science. Our experiences clearly show the usefulness of gamification in learning. Students are now able to understand and manipulate difficult and abstract concepts introduced in games. Playing and working as a team led them to share knowledge. We also noticed a very good students' engagement in most cases.

Including gamification for helping students to learning subtle notions has been efficient: it first increased students curiosity and engagement and, thanks to concrete experience, they could make a first idea of the notions, that facilitated the acquisition of theorical concepts.

One key point is that both parts have to "play the game", especially the teacher, who needs to be aware that the game spirit is important and has to go beyond his casual behaviour. For students, engagement is easy to obtain most of the time. However they can call the game into questions if they

have a too traditional vision of teaching. It is especially a problem for elder students. They have to accept that playing is also a way of learning.

We would like to continue our experience in traditional games for teaching computer science. First of all, we plan to modify the Pictionary questions for adapting the questions to our specific teaching program and for introducing levels of difficulties in questions. We also think about adding new gaming sessions in our practices. Especially, we think that games would be useful for introducing modelling process. By increasing the gaming frequency, we can also test either students get used or if they keep their engagement.

5 ACKNOLEDGEMENTS

This project has been supported by the French government's “Investment for future” program IDEFI (ANR-11-IDFI-0031) (initiative for excellence in innovative training).

REFERENCES

[1] Bybee, R. W., Taylor, J. A., Gardner, A., Van Scotter, P., Carlson, P. J., Westbrook, A., Landes, N. (2006). The BSCS 5E instructional model: Origins and effectiveness. *Colorado Springs, CO: BSCS, 80 p.*

[2] KOLB, A. Y. (2005). The Kolb learning style inventory - version 3.1: 2005 technical specifications. *Boston, MA: Hay Resource Direct* 200, 72 p.

[3] Object Management Group (2013) OMG Unified Modeling Language (OMG UML) v2.5, 831 p., http://www.omg.org/spec/UML/2.5/Beta2/PDF

[4] Royce, W.W. (1970). Managing the development of large software systems: concepts and techniques. Proc. IEEE WESTCON, Reprinted in the Proceedings of the Ninth International Conference on Software Engineering, March 1987, p. 328–338.

[5] Krivitsky, A (2009), Lego for extended Scrum simulation,

[6] Fowler M., Highsmith J. (2001), The Agile Manifesto, Software Development, Miller Freeman, Inc., p. 28–32.

[7] Schwaber, K. (1995). SCRUM Development Process. In Proceedings of the 10th Annual ACM Conference on Object Oriented Programming Systems, Languages, and Applications (OOPSLA). p. 117–134.

[8] Grenning, J., 2002. Planning Poker or How to avoid analysis paralysis while release planning, available online at http://sewiki.iai.uni-bonn.de/_media/teaching/labs/xp/2005a/doc.planningpoker-v1.pdf