Louvain School
of Management

Working paper 08/02
e-SPM : An online software project management game

Yves Wautelet, Manuel Kolp & Nicolas Neysen

Academic Year 2007 - 2008

Louvain School of Management

e-SPM: An Online Software Project Management Game

Yves Wautelet[*], Manuel Kolp[*], Nicolas Neysen[+],

[*] *IAG – Institut d'Administration et de Gestion,*
ISYS – Unité de Systèmes d'Information,
Université Catholique de Louvain, 1 Place des Doyens, Belgium
Email: {wautelet, kolp@isys.ucl.ac.be}

[+] *IAG – Institut d'Administration et de Gestion,*
POGE – Politique Générale et Economie d'Entreprise,
Université Catholique de Louvain, 1 Place des Doyens, Belgium
Email: {neysen@fusl.ac.be}

Abstract. Today's software development has become a very complex task and no one has the required skills or time to resolve a sophisticated problem on his own. Software development phases need the implication of lots of people having to use concepts and ideas for which they share a common understanding. Into such a context, several software development methodologies appeared in the last thirty years. Those methodologies use different development life cycles, one of the most famous being the iterative one used in the Unified Process (UP). To learn project managers and computer science students to deal with such development processes few methods exist. Indeed, one can learn from ex cathedra courses, books or directly into "real life" projects but no software project simulation games exists. That is why we are developing an online multi-user game simulating the job of a project manager facing user requirements, development planning, human resources allocation, budget constraints, risk and quality management onto a UP/UML software project case study.

1 Introduction

Computing science and techniques are deeply linked to human activities. Unfortunately, software development methodologies are nowadays traditionally inspired by programming concepts and not organizational and enterprise ones. This leads to ontological and semantic gaps between the systems and their environment. Moreover, software development is becoming increasingly complex. Stakeholders' expectations are growing higher while the time to market has to be as low as possible. In order to be competitive in such markets, analysts, project leaders, software developers need adequate methodologies to model the organization, capture requirements and build efficient and flexible software systems. Those methodologies have to cover the whole project life cycle while reducing risk as much as possible. For user-intensive software applications the objective will be better achieved using a Spiral (or Iterative) System Development Life Cycle [Boehm88]. Indeed, the later is able to deal with an environment facing difficulties to capture rapidly evolving requirements in an efficient way.

The aim of this research is to build and validate an original iterative software development game for object-oriented development using UML use cases and UP as a development methodology. This games takes a series of engineering concepts as fundamentals: disciplines that have to be repeated iteratively during one of the four phases. The disciplines, the activities performed during them, the models taken as input or output to these activities, etc. are part of the gamer dimensions i.e. the aspects he has to deal with. The project management framework is consequently central into this game. Indeed, an iterative methodology cannot simply advice users to proceed iteratively but has to offer such a framework to support the project development. The gamer is the main actor using this framework, allocating scope use cases for each iteration, planning human resources, dealing with risks, etc. Risks must be identified and evaluated continuously so that the project can be planned and the planning can evolve dynamically. To be more realistic we point to the application the theoretical framework of the game developed in this paper on a real life case study: the game applied onto the production management information system for a coking plant is currently under development.

The paper is structured as follows: Section 2 describes the context of the problem, presenting iterative development and other UP development methodology. Section 3 defines the gamer dimensions while section 4 presents a meta-model of the game framework. Finally, section 5 concludes the paper.

2 Problem Statement

2.1 Iterative Development

A System Development Life Cycle (SDLC) is a conceptual model used in software project management [Royce98] to describe the stages involved in a system development project; from an initial feasibility study through maintenance of the completed application. Several SDLC such as the Sequential/Waterfall Model [Royce70], the V-Model [Forsberg97], the Incremental Model [Dorfman97], the Evolutionary Model or the Iterative Model [Boehm88, Boehm94] have been proposed over the years. Depending on the project, some SDLC appear to be more adequate than others. For the development of huge and complex user-intensive enterprise information systems the Iterative SDLC is more appropriate due to its iterative nature than the use of a traditional Waterfall SDLC. Indeed the former is much less risky than the later, especially in the last stages of the project.

The iterative SDLC uses a spiral process model which is *"a **risk**-driven **process model generator** used to guide multi-stakeholder concurrent engineering of software intensive systems. It has two main distinguishing features. One is a **iterative cyclic** approach for incrementally growing a system's degree of definition and implementation while decreasing its degree of risk. The other is a set of **anchor point milestones** for ensuring stakeholder commitment to feasible and mutually satisfactory system solutions"*. [Boehm00]

The assumptions for an optimal Waterfall SDLC (requirements knowable in advance, requirements with no high risk implications, etc.) are seldom met. Causes are:

- requirements can rarely be known and defined in advance of implementation, especially for modern user-intensive dynamic systems;
- defining requirements and freezing them early on in the project is a very risky task. Early defined requirements can meet users' expectations but often analysts discover later in the project that their practical achievement is (too) constraining;
- requirements often evolve during the software development process.

These remarks highlight the need for a SDLC that includes continuous requirements acquisition and modeling. Using an Iterative SDLC implies that risk is considered during the early stages of the project not at the late ones as in a process fully driven by sequential activities. Such development process requires a strong project management discipline for process application and optimization.

2.2 UML with UP lifecycle

The Unified Process (UP) (see [Jacobson99, Jacobson00, Kruchten03, RUP]) is an iterative software development process using the Unified Modeling Language (UML) (see [UML, Rumbaugh99, Booch99]) as modelling language. This methodology is intended to develop object-oriented systems. Models defined using UML are used as artefacts in the UP.

UML is a modelling language to specify, visualize, construct and document software-intensive systems. UML has been normalized by the Object Management Group (OMG) [OMG] in order to furnish a universal communication support because it is independent from application domain and programming languages.

UML provides multiple system views as well as multiple layers of abstraction. These views are represented by different types of diagrams:

- Structural diagrams: the class diagram, the object diagram, the component diagram and the deployment diagram.
- Behavioral diagrams: the use-case diagram, the sequence diagram, the activity diagram, the collaboration diagram and the state chart.

The most important diagram in the context of this game is the use-case diagram. Use-cases model users requirements by characterizing the interactions between the actors and the system. The actors are the different types of users that will use the system. The actors can visualize the information proposed by the system. They can modify the system state, in the case the system has to furnish an answer to this modification. This view is abstract, it allows to identify systems required functionalities so that users can fulfill their tasks. Use cases allow understanding the way the overall system operates.

The iterative software development process UP allows producing quality information systems better meeting user requirements. UP is said to be *Use-Case Driven* which means that the process employs use cases to drive the development process from the beginning of the project to its end.

3 Gamer Dimensions

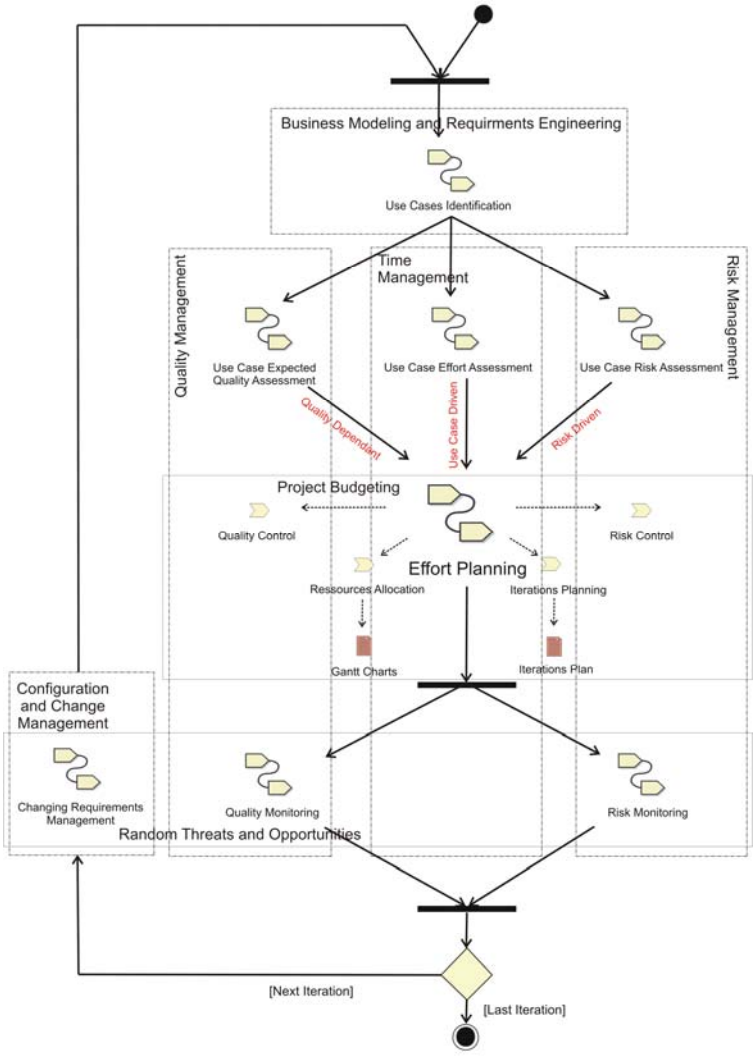

Figure 1. Gamer Dimensions.

Figure 1 gives an overview of the dimensions offered by the software to the gamer using the formalism defined by the Software Process Engineering Metamodel (see [SPEM]). His tasks are concretely divided onto the following disciplines:

- Business Modeling and Requirements Engineering: the gamer receives a set of use cases specific to a predefined case study. On the basis of a redundancy analysis he selects the use cases he has to realize i.e. he has to plan for

practical achievement in terms of design, implementation and test. This first step is already strategically important since effort optimization is not compatible with redundant developments.

- Quality Management: predefined expected quality levels following defined benchmarks are associated with use cases. As a consequence, resources allocation has to be sufficient onto them to meet the expected level.
- Time Management: on the basis of UML activity diagrams further describing the use cases, the gamer has to evaluate the effort (the number of resources having to be allocated) required to realize them in terms of design, implementation and test.
- Risk Management: on the basis of some predefined threats, each use case overall risk must be evaluated by the gamer. He then has to prioritize use case realization on the basis of their overall risk.
- Project Planning and Budgeting: using the quality, effort and risk factors he has evaluated, the gamer has to plan a first overall project planning and expected cost. Once determined the gamer runs a one iteration simulation.
- Random Threats and Opportunities: during simulation positive and negative factors can intervene. After iteration simulation the gamer gets acquainted with the factors supervening so that he can react in terms of planning, budgeting for the following iteration(s).
- Configuration and Change Management: random events can also subsequently modify use cases specification during the game. Those events are discovered during testing. When those events occur, the gamer has to take them into account for the next iterations planning.

Gamer performance is evaluated on the basis of:

- Ability to respect planning: if too little HR are hired on the project, software developments will last longer than originally planned.
- Ability to respect budget: the gamer has a budget constraints he has to respect. Hiring and even firing staff has different costs in function of HR role; the gamer has to optimize resources allocation.
- Ability to meet users' requirements and expected quality levels: this benchmarks lowers if the project manager omits some non redundant use case realization, if he does not take into account changing requirements and if quality levels are below what was expected.

4 Game Meta-Model

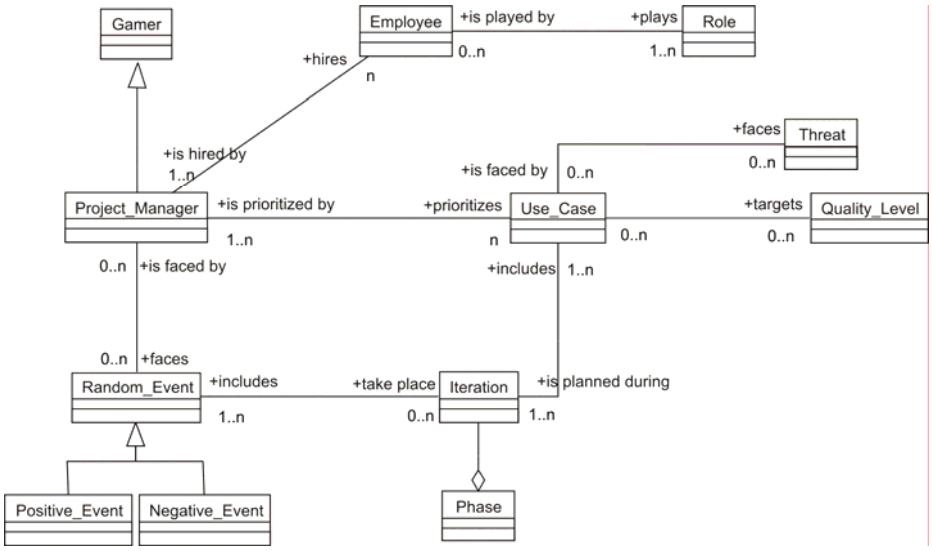

Figure 2. The Project Management Game Meta-Model.

In this section, we will bring the game to further formalization. For this purpose, a game meta-model using a UML-like class diagram is represented into Figure 2 and formal definitions will be used. The game we are describing is a multi-user game where the gamer plays the role of a project manager. His core business is driving the software development process by managing the iterative development life cycle.

Definition 1 *A tuple* $(\{(act_i, q_{mfi}^{a}), \ldots, (act_{i+m}, q_{mfi+m}^{a})\}, Ga^{a})$ *is called a gamer a, where* act_i *is a project management activity. An activity is an issue the gamer has to deal with when playing the game i.e. managing the project, for example evaluating a threat, hiring human resources, etc. The gamer owns the ability to manage those activities at cost* q_{mfi}^{a}.

Definition 2 $\langle a, uc_n, it_m \rangle$ *associating a use case* uc_n *to an iteration* it_m *by a gamer a is a Project Manager* a_i^{PM}.

The primary user function is to plan use cases achievement through an iteration plan. To achieve such a goal, the gamer has to perform a certain number of project management activities. Those activities include the evaluation of factors such as risk, quality and effort. Risk management is materialized into the game through the presence of Threats. In function of the use case complexity, enough human resources must be allocated for the use case realization. An expected quality level in terms of correctness, reliability, efficiency,… should be allocated to each use case so that resources allocation sometimes has to be increased to fulfil the expected levels.

Definition 3 *An activity act_i is $\langle act_i^{pre}, \tau_i, act_i^{post} \rangle$ where act_i^{pre} describes the activity precondition, τ_i is a specification of how the gamer executes the activity and act_i^{post} describes the postconditions i.e. the state of the gamer's environment after the activity is executed.*

Definition 4 *$\langle uc_i^t, uc_i^q, uc_i^w, ucState_j \rangle$ is a use-case uc_j, where uc_j^t represents the threats faced by the use-case, uc_j^q represents the quality levels required for the use-case and uc_j^w represents the use-case weight i.e. the number of resources required for use-case fulfilment. Finally, $ucState_j : uc_j \rightarrow \{act_i^{pre}\} \cup \{act_i^{post}\}$ represents the activities precondition i.e. $\{act_i^{pre}\}$ and postconditions i.e. $\{act_i^{post}\}$ for the use case fulfilment.*

Finally, the gamer score is computed on the basis of factors such as the time required to fulfil the project, the total cost of the project and the quality level of the "realized" use-cases.

Definition 5 *$\langle score_i^t, score_i^c, score_i^q, scoreAmount_j \rangle$ is a the gamer score $score_j$, where $score_j^t$ represents the time required by the gamer to complete the whole project, $score_j^c$ represents the total cost of the project fulfilment by the gamer and $score_j^q$ represents the use-cases effective quality i.e. the quality effectively measured in function of defined benchmarks, the ability to meet user requirements is included. Finally, $scoreAmount_j$ is a function returning the overall gamer score on the basis of the parameters $score_i^t$, $score_i^c$, $score_i^q$.*

5 Conclusion

Nowadays developing software involves hundred of individuals for the most sophisticated software projects. Consequently it is no more just a matter of analysts, designers and developers but requires people able to manage the whole software process adequately. Learning those skills is usually done on an empirical bases rather than in a uniform manner at university and other high schools that is why we formalized a game allowing students to learn the basic dimensions of software project management.

This paper presented the state of achievement of a research aimed to build a game for iterative software development using UML/UP. The literature review pointed out that many object-oriented software development methodologies event advising to develop iteratively, a unified software project management discipline to support this type of development is seldom provided. That's why we developed a framework to evaluate whether the dimensions a software project manager has to deal with.

The theoretical bases have been given in this paper, they can be used as a starting point for a basic online project management game. The implementation placing the gamer onto the project manager role managing a team for developing a real life case study (the production planning of a coking plant) is currently under development.

References

[Boehm88] B. Boehm, “A Spiral Model of Software Development and Enhancement”, Computer, May 1988, pp. 61-72.

[Boehm94] B. Boehm, “A Collaborative Spiral Software Process Model Based on Theory W”, IEEE CS Press, Los Alamitos, Calif., 1994, pp. 59-68.

[Boehm00] B. Boehm edited by Wilfred J. Hansen, “Spiral Development : Experience, Principles, and Refinements”, Spiral Development Workshop February 9, 2000, July 2000.

[Booch99] G. Booch, J. Rumbaugh, and I. Jacobson, "The Unified Modeling Language User Guide", Addison-Wesley Object Technology Series, 1999.

[Dorfman97] M. Dorfman, “Requirements Engineering”, In R. H. Thayer and M. Dorfman (eds) “Software Requirements Engineering, Second Edition”. IEEE Computer Society Press, 1997, pp 7-22.

[Forsberg97] K. Forsberg and H. Mooz “System engineering overview” In R. H. Thayer and M. Dorfman (eds.) “Software Requirements Engineering, Second Edition”, IEEE Computer Society Press, 1997, pp.44-72.

[Jacobson99] I. Jacobson, G. Booch, J. Rumbaugh, “The Unified Software Development Process”, Addision-Wesley, 1999.

[Jacobson00] I. Jacobson, S. Bylund, “The Road to the Unified Software Development Process”, Cambridge University Press, 2000.

[Kruchten03] P. Kruchten, “The Rational Unified Process : An Introduction”, Longman (Wokingham), Addison-Wesley, December 2003.

[OMG] Object Management Group, http://www.omg.org/.

[Royce70] W. Royce, “Managing the Development of Large Software Systems”, Proceedings of the IEEE WESCON, August 1970, pp. 1-9.

[Royce98] W. Royce, “Software Project Management. A Unified Framework”, Addison-Wesley, 1998.

[Rumbaugh99] J. Rumbaugh, G. Booch, and I. Jacobson, "The Unified Modeling Language Reference Manual", Addison-Wesley Object Technology Series, 1999.

[RUP] IBM, “The Rational Unified Process. Version 2003.06.00.65”, Rational Software Corporation, 2003.

[SPEM] OMG, “The Software Process Engineering Metamodel Specification. Version 1.1”, January 2005.

[UML] OMG, “OMG Unified Modeling Language Specification. Version 2.1.1”, 2007.