EXPERIMENTATION OF A GAME DESIGN METHODOLOGY FOR MOBILE PHONES GAMES

Viviane Gal, Alexandre Topol
Centre d'Etudes et de Recherche en Informatique (CEDRIC)
Conservatoire National des Arts & Métiers (CNAM)
292 rue St-Martin, 75003 Paris, France
{viviane, topol}@cnam.fr

ABSTRACT

Today, the mobile games tend to be democratized. This is mainly because mobiles are getting cheap and they integrate technologies allowing the use of new networks and new tools. The mobile game development is not done like the development of a PC or console game even if certain aspects are common. Several criteria have to be taken into account such as management, technical and technological criteria. It is thus necessary to find a good compromise between the originality and the feasibility of the game. This must be done by knowing the hardware and software constraints/forces of the mobile phones.

The process of mobile games development is similar to the Web application development. We thought that software engineering techniques could also be very useful in our case. We chose the 2TUP technique associated with UML (Unified Modeling Language) to observe if it’s well suited for this type of projects. Our approach is to find a non-constraining and fast methodology that can improve the game production while preserving its creativity.

Key Words

Methodology, 2TUP, mobile phone games

1 INTRODUCTION

The design and the proposal of an adapted game design for the mobile games are similar to the development of applications and, more particularly, Web applications. The development’s cycle of such applications is iterative corresponds to the evolution of the game design document during the progress of the project.

The information consigned in the game design document consists of the main aspects of the game (context, main features, and principles of the gameplay, objects classes and so on). It is the result of a long team work and falls under a process which is similar to an intuitive planning process.

Today, no methodology is given to make this document. The only supports available are rather anecdotic and informal recommendations or texts. Some researches are carried out and propose some semi-formal methods [CHU99], [KRE03], [DES04]. These methods don’t replace the game design document but come in addition in the design of the game development process as well in formalization than in validation [VEG04].

We thought that the use of iterative and incremental methods in video games and, more particularly, in the mobile games project, seemed the most accurate method. This paper presents a process to find a suited methodology to the mobile phone game design. Section 2 is devoted to methodology: state of the art, carried methodology. Section 3 presents our process: preliminaries, the left track of the Y-shaped cycle, the right track and the middle one. Section 4 shows the way of future investigations.

2 METHODOLOGY

In the software engineering domain, there are several approaches for development. It is important to distinguish the modeling techniques from the management methods. The application modeling techniques are based on models like "entity-relation", "flow", "object". New technologies, among which our project is, often use object models. The standard in term of analysis and modeling is UML. The management or development methods on which rest the project’s management is based on four elements:

- a model,
- a language,
- a process,
- tools.

2.1 STATE OF THE ART

2.1.1 Usual methodologies

According to the range and needs of a project, one or more methods can be considered. For example, some methods are:

- MERISE is a French cascading method of analysis and structural design which was invented in 1978. MERISE consists of three ‘cycles’, the decision cycle, the life cycle and the abstraction cycle. The abstraction cycle is the most important. In this cycle, both data and processes are first viewed at the conceptual level, then the logical or organizational level and finally at the physical or operational level.

- SSADM (Structured System Analysis and Design Methods) belongs to the structured methodologies, adopts a cascading model. SSADM is a methodology used in the analysis and design stages of systems development.

- SADT (Structure Analysis and Design Techniques) is a conception model based on diagrams organized in a tree structure.
- OMT (Technical Object Modeling) (Rumbaugh J.) uses data flow diagrams, hybrid E-R diagrams, and statecharts to model software requirements using object-oriented concepts. The OMT notations are only partially formal (statecharts have been formalized), and methods for analysis and design with the diagrams proposed are only described informally at a high level.
- RAD (Rapid Application Development), which marked a revival in methodologies by knocking into the traditional modes of the software life cycles management, has six core elements: prototyping, iterative development, time boxing, team members, management approach, and RAD tools.
- Agile methods: the rapid development took part in the emergence of approaches privileging the flexibility and the adaptability. These approaches are declined under the name of “agile” methods and are presented hereafter.

2.1.2 Agile methods

Agile software development is a conceptual framework for undertaking software engineering projects. There are a various agile software development methodologies, such as those validated by the Agile Alliance, a non-profit organization. The agile methods [PRI05] answer to a set of assertions, rules and behaviors.

In 2001, 17 prominent figures in the field of agile development came together to discuss the unifying theme of their methodologies. The agile method concept is the result of the Agile Manifesto in which we find four fundamental values, twelve principles defined as good practice rules. The fundamental values are as follows:

- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan.

Some agile methods are the following:

- DSDM (Dynamic System Development Management) provides a framework for an iterative and incremental approach to the development of information systems. DSDM “focuses on the priorities of the business and delivers what can safely be delivered within the time and cost constraints of the project, in priority order determined by the business needs and the objectives of the project”.
- SCRUM is an agile method for project management. It is designed to be ultra-productive, where working prototypes are delivered in thirty day "sprints." What the team will accomplish during the sprint is predefined : no work may be added during a sprint. ”Scrum” is a term borrowed from the rugby world and means “mixed”.
- UP (Unified Process) is an iterative and incremental software development methodology controlled by the UML use cases. The Engineering Requirements present in the Unified Process are mostly technology centric and has only recently focused on user centered design.
- XP (eXtreme Programming) is a method or approach to software engineering. It is the most popular of all agile software development methodologies. It is founded on a mechanism for social change, a style of development, a path to improvement, an attempt to reconcile humanity and productivity and a software development discipline. It is controlled by the user's needs. XP can be regarded as a variation of UP.

2.2 SELECTED METHODOLOGY

The project development process can be considered with several approaches. The project leader, the editor, the computer scientist, the manager... don’t have the same needs nor the same objectives. However, they have a same goal which is the success of the project.

The game design constitutes the process of design and specification of a video game. It is the main document of the development process. The presentation of this document will be different according to the interlocutors to which it is addressed. Adams and Rollins had notified it very well in their book "on Game Design" [ROL00], [ROL03]. Indeed, the game design document, according to its level design, forms a good marketing tool as well as a reference book for the various jobs: graphic designers, scriptwriters, programmers...

The meta-modeling techniques, promoted by the Object Management Group (OMG), bring brief replies to these problems on formalism definition, on sharing concepts and on information conversion.

Many works have been made these last years and a convergence towards the concept of process appears whatever the apprehended domain is.

In the software engineering field, works on SPEM (Software Process Meta-model Engineering) [OMG05] were retained. SPEM defines a formalism for the description of the software development processes [BEL05].

Thus, our choice was made on the 2TUP method, “2 Tracks Unified Process”. It is based upon a SPEM modeling architecture in order to conceive elegant and adapted solutions but also to take advantage of the new techniques and technologies.

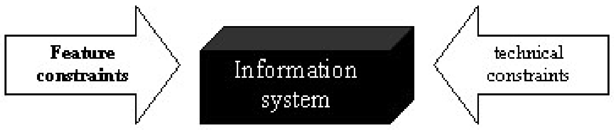
Our goal is firstly to propose a rapid way to develop a game design for mobile phones games. Secondly, we wanted to move along at the same pace the functional and the technical parts. Finally, we wished some good graphical representations (Y-shaped process, UML diagrams) and different understanding levels (abstract idea, coding, …). The 2TUP method answers to all these requirements.

2.2.1 2TUP

2TUP is a unified process (i.e. a software development process) built on the UML modeling language [ROQ04]. Every process answers the following main characteristics:

- It is an incremental process, allowing a better technical and functional risk management and thus constituting the deadlines and the costs control.
- It is an iterative process. The degrees of abstraction are increasingly precise at each iteration.
- It is component oriented, offering flexibility to the model and supporting the re-use.
- It is user oriented because built from their expectations.

The 2TUP process answers to the constraints of change of the information systems subjected themselves to two types of constraints: functional constraints and technical constraints as shows it the following diagram:



In concrete terms, the process is modeled by two branches (tracks):

- A functional track (capitalization of knowledge trade)
- A technical track (re-use of a technical know-how).

Then these two tracks amalgamate for the realization of the system. This is why this process is still called Y-shaped process.

With this development process, a model is essential in order to anticipate the results. A model can be used with each step of the development with an increasing detailed manner. The industrial standard of object modeling, UML, was selected as the development tool. It appeared very difficult to consider the 2TUP process without using UML and, more particularly UML 2.0 which support the oriented design component.

2.2.2 Implementation

2.2.2.1 Introduction

The first phase of the implementation starts with the preliminary study which introduces the project. The specifications, initial document of the functional and technical needs, are partly the result of this work. It is supplemented by the modeling of the total system context. Not everything is to be described at this stage but simply to identify the external entities interacting (actors), to list the interactions (messages) and to represent this unit on a model (context).

2.2.2.2 Feature track

The functional branch makes an inventory of the functional needs and analyzes it. This phase formalizes and specifies the elements of the preliminary study. The applied use case technique translates the whole interactions between the system and the actors. The obtained use cases are then organized (treated on a hierarchical basis, generalized, specialized...). They make it possible to identify the classes and they permit the oriented object modeling generated in the analysis part.

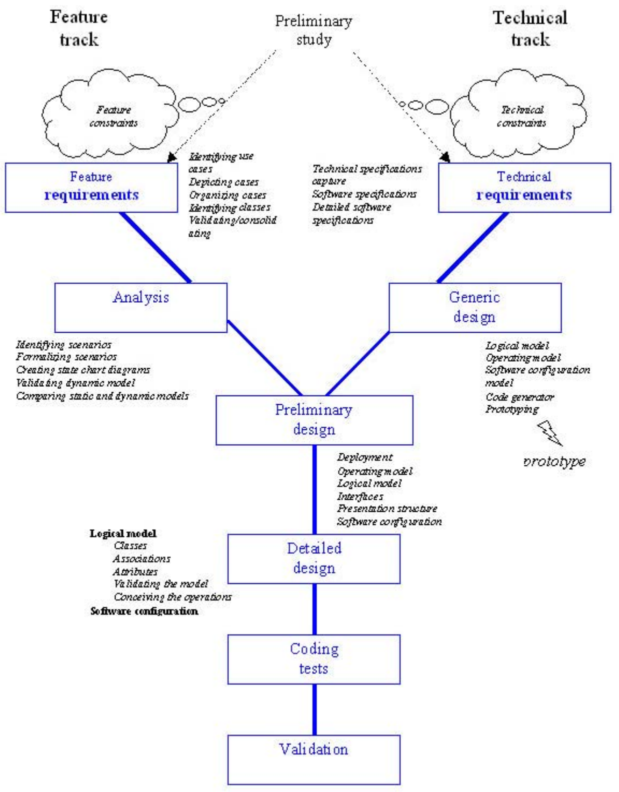
2.2.2.3 Technical track

The technical branch lists the technical needs and proposes a generic design validated by a prototype. The pre-necessary techniques revealed in the preliminary study, showing the operational needs and the strategic choices of development, lead to the development of the construction process. To do this, several stages are necessary:

- The inventory of technical specifications related to the hardware,
- The inventory of the software specifications.

2.2.2.4 Middle track

The medium branch supports the preliminary design, the detailed design, coding, the tests and the validation. The preliminary design is one of the most sensitive steps of the 2TUP process. It represents the fusion of the functional and technical tracks. It finishes when the deployment model (working stations, architectures), the operating model (components, applications), the logical model (classes diagrams, classes), interfaces (users and components) and the software configuration model are defined.



The detailed design conceives and documents very exactly the code that will be generated. It is largely founded on UML representations and implements, in an iterative manner, a system construction process to obtain a "model ready to code".

The various components carried out in the detailed design are coded. The code units obtained are tested as they are written and produced. The validation, called “recipe” step, consists of the homologation of the developed system functions.

3 OUR PROCESS

3.1 Preliminaries

While sticking to the 2TUP method, we initially enumerate the various choices and needs necessary for the study. The starting point is the specification which stipulates that the objective is to conceive a game and to propose a game design adapted to mobile phones. The game design document is used as a reference and as a link between all the team members. It evolves unceasingly from the beginning until the game is complete by various iterations and additions.

The conception and the development of the game design should thus make it possible to distinguish and specify the essential generic characteristics of mobile games. To end up with a general development process, it is necessary to take into account several things:

- The basic aspects of all game designs (mechanisms of the game, interactivity, history, context...)
- The specific aspects when dealing with mobile phones like physical and technological characteristics
- The ergonomic and sociological aspects

It is essential to determine what is awaited for the game to be delivered. Indeed, it forces us to consider all elements essential for the production of the game. For example, when it is told that the game is a multi-player game, it implies that transactions will be propagated through the network between the various actors. The type of network, the modes of interactions will have to be defined clearly. With mobile phones, the data transfer over the network generates a cost which should also be evaluated precisely.

The following list gives some key technical choices:

- UML, RUP
- Internet
- I-mode
- Different types of mobile devices
- Programming languages (Java, C#, …)
- SGBDR, …
- GSM, GPRS, UMTS
- WAP, MMS
- Bluetooth, infrared, …
- Client/server
- P2P

The collection of functional needs consists of:

- Internet provider (invoicing, maintenance, …)
- Content provider (services, games, …)
- I-mode provider (services, …)
- Telephony provider/operator

The collection of the operational needs is:

- Identification of the user (at connection time)
- Access to his/her data
    - The need to have an administrator (for the system and the network)
- Data processing (in real time or in batch mode)

As for the actors of the system, one finds:

- Users or clients or players
- The various providers
- Network administrator
- System administrator
- Database administrator
- The system (that recovers the various data or events)
- Mobile devices (since the geographical position is given by the device and not by the user)
- …

Messages between the actors and the system are essential in order to have something working. They transport information which will trigger off some actions by the receiver. A message can be used for different purposes:

- Deployment (platform)
- Publication (games for example)
- Maintenance
- Subscription
- Download
- Redirection (lobby server -> game server)
- Data (provider -> user [new information, acknowledgment after processing an event, …])
- Personalization (user interface management)
- Inter client Communications.

The context modeling materializes itself by the system-actors communication that can be represented by a synthesis diagram translating this dynamic context. One uses in this case a collaboration diagram.

The 2TUP technique is thus an iterative and incremental technique. Each stage of the process are also iterative and incremental. This aspect is considerable for a surer development. Thus, the project is built step by step, validated progressively and refined through its development until the final result.

3.2 Left track

Functional needs are formalized and detailed from the first stage. At the same time, they are complemented by technical needs which are in the right-hand side branch. Indeed, it is when one defines the functional needs that also appear the technical needs.

The following list constitutes a sample of the functional requirements raised within the project. They will be labeled by "req" followed by a number. This example shows that the enumeration of needs makes it possible to specify the scope, to refine the analysis and to reveal elements related to the design.

Req1 : save the game from a menu

Req2 : associate an user profile to a game save file…

Req5 : menus, buttons, sliders, images supported by the software …

Req25 : Characters or creatures attributes : name, dominant color, % of strength, state, geometrical shape.

Req26 : each character or creature belongs to an social class identifiable by a single attribute …

Req52 : A main menu items : new game, resume game, pause, change profile, quit, options, load a game …

Req70 : *x* levels with basic quests …

Req75 : software and hardware architecture …

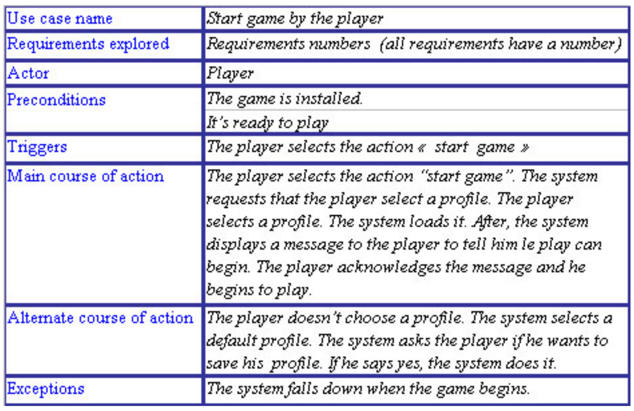
Req(n-1) : a given programming language in the software is developed using.

Req(n) : events' synchronization with a global clock.

It is then advised to classify all these requirements by group of functionalities. Each of them forms a component. From our previous list, it is possible to group:

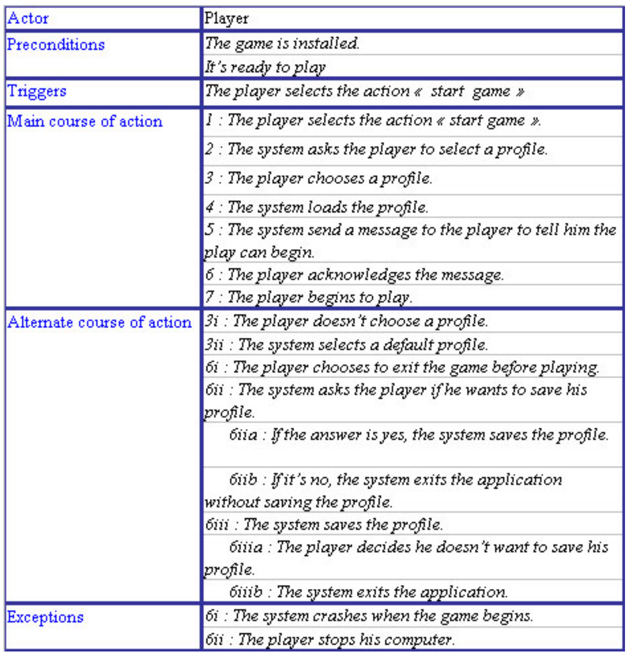
- req25 and req26 into a component called "statistics"
- req75 and req(n-1) into a component "non functional requirements"
- req70 into "different levels"

The requirements will also make it possible to fill up the use cases description forms.



Use case name	*Start game by the player*
Requirements explored	*Requirements numbers (all requirements have a number)*
Actor	*Player*
Preconditions	*The game is installed.* *It's ready to play*
Triggers	*The player selects the action « start game »*
Main course of action	*The player selects the action "start game". The system requests that the player select a profile. The player selects a profile. The system loads it. After, the system displays a message to the player to tell him le play can begin. The player acknowledges the message and he begins to play.*
Alternate course of action	*The player doesn't choose a profile. The system selects a default profile. The system asks the player if he wants to save his profile. If he says yes, the system does it.*
Exceptions	*The system falls down when the game begins.*

The transformation of this diagram, rather of narrative type, is done by explaining actions more precisely. The following structure is then obtained:



Actor	Player
Preconditions	*The game is installed.* *It's ready to play*
Triggers	*The player selects the action « start game »*
Main course of action	*1 : The player selects the action « start game ».* *2 : The system asks the player to select a profile.* *3 : The player chooses a profile.* *4 : The system loads the profile.* *5 : The system send a message to the player to tell him the play can begin.* *6 : The player acknowledges the message.* *7 : The player begins to play.*
Alternate course of action	*3i : The player doesn't choose a profile.* *3ii : The system selects a default profile.* *6i : The player chooses to exit the game before playing.* *6ii : The system asks the player if he wants to save his profile.* *6iia : If the answer is yes, the system saves the profile.* *6iib : If it's no, the system exits the application without saving the profile.* *6iii : The system saves the profile.* *6iiia : The player decides he doesn't want to save his profile.* *6iiib : The system exits the application.*
Exceptions	*6i : The system crashes when the game begins.* *6ii : The player stops his computer.*

Functional needs are modeled this way through use cases. They could also be represented in a more concrete way by using some forms filled by testers in order to take into account their remarks, criticisms and reactions.

3.3 Right track

The capture of the technical needs can be carried out as soon as one knows which materials will be used for the project's development. Materials include more precisely machines, networks and tools. At this level, the number and the type of supported materials are defined by technical constraints.

3.3.1 Some technical characteristics

Unlike PCs or game consoles, a mobile terminal does not have multiple accessories and peripherals. To be compatible with most mobile devices, the interaction scheme has to deal with very reduced resources. The set of elements considered as minimal is, in our case, the digital keyboard and the small color screen.

In addition, on most powerful and expensive devices, more functionality is accessible which make it possible to consider richer interactions. For example, the use of a touch screen, of voice recognition, of minis joystick can influence the game design and thus contribute to enrich the playing experience.

3.3.1.1 Screen

Conditions of use of a graphic interface for the mobile devices are different from those of office computers. For instance, the display's size is smaller and there is often no pointing device. For these simple reasons, the directives and recommendations of UI programming are not the same.

The screen of a mobile phone is truly the significant and delicate point to manage. The resolutions in pixels, the number of colors and the screen's size converge very slowly towards a minimal comfortable standard. So it is necessary to manage several possible resolutions. The difference of resolution from one device to another prohibits the massive use of large bitmaps in a game. Sprites with a size much smaller than the smaller device's resolution can only be used. Larger images for full screen applications stick to particular resolutions and are not transposable in another resolution without loss of information and thus of quality. In addition, the reduced memory capacity offered by the Java Virtual Machine (32Kb usually) on mobiles does not make it possible to easily store many bitmaps.

Vectorial graphic are preferable because they can adapt to the resolution of the platform. Moreover, vectorial data do not take a large amount of storage capacity and their rendering is accelerated. The drawback with this solution comes from the code's size and the processing time needed. It is necessary to draw various geometrical primitives composing an object. In the bitmap solution, a simple memory copy is sufficient. Nevertheless, we prefer the vectorial solution.

3.3.1.2 Keyboard

The other important limitation is the absence of pointing device. On the majority of the mobile phones, it will be necessary to be use exclusively the digital keyboard. Thus, it will be necessary to make a reduced use of forms because text input is difficult and requires a lot of time even when the user has a long experience with Short Messaging System.

However, it is not rare to see games being played with the keyboard. For example, earlier First Person Shooters games were played exclusively with the keyboard. The mouse is used only since the point of view is managed like a synthetic camera having 6 degrees of freedom. One can thus imagine that an isometric 3D FPS game could have a certain success. However, for a player on PC or console, this means a retrogression of 10 years.

3.3.1.3 Other interfaces

A touch screen would make it possible to enrich the interaction by providing a pointing system. Accessing to a mechanism of selection, of drag and drop would enrich interactions. The phone's vibrator can be used as a force feedback device. Joy pads offer exactly the same mechanism of vibration to transmit important information feedbacks to the player. The camera integrated into the recent telephones makes it possible to imagine an augmented reality game. The localization is certainly the most interesting element to integrate into the game design of multi-players games in which the same part of the real world is shared.

3.3.1.4 Operating capabilities

Programming a game for mobile phones can be very painful and frustrating. First of all, the screen is tiny and does not allow displaying rich information distinctly. In addition, the memory is extremely limited since the majority of the phones only offer 32Ko of dynamic storage in which must reside a program. Knowing that a program is often written in Java, in which classes are greedy in memory, one sees that only little place is given for development. To gain some memory, the usual solution simply consists in limiting the number of these classes in a mobile application. Lastly, the embedded processor and graphical units are hundreds of times slower than the ones on an average computer.

We do not present all the technical needs here. The point is to show that originality of a mobile game will not come from graphics or interaction. These two aspects are absolutely deplorable. At best, one can consider that in a few years, the mobile platforms will have high-performance 3D processors and screens with acceptable resolutions. This time is not so far away since nVidia has released a 3D chipset for mobile phones. Until its integration, complex game designs requiring long periods of plays must be proscribed.

It is necessary to focus the game design on specific characteristics of the mobiles. The first is the continual proximity of the user with his phone. Mobile devices are ideal for games requiring a regular follow-up. These popular games are mushrooming on the Internet (management of a farm, virtual creatures, goldfish ...). This tamagoshi type of game matches well with mobile phones use and capacities. The user will be able to take care of his virtual creatures when he/she wishes to do it. In addition, those tasks do not take much time which corresponds well to the average use time of mobile phones.

The mobile phone is essentially a communication device. This fact must be taken into account in the game design. The integration of SMS and vocal communications together with data transfer is something that must be explored. It can be useful for the notification of important events in a persistent game or to propose realistic relations between players.

Another thing interesting to explore is the localization. New games appear taking into account the real position of the player. According to the place where he/she is, a hunting game is able to adapt and propose enigmas.

In the same way than for the capture of the functional needs, it is necessary to identify the technical use cases and to express them textually with description forms. It is at this level that technical constraints, like those stated above, lead to defining the number and the type of material supports.

In our study, the technical exploitation of a network is of primary importance. Specifications around it are being realized through connection via one or more models of hardware configuration. Materials or machines impose a certain number of constraints. In the same way, the type of connections generates constraints related to the communication, to the band-width.

To be the most exhaustive, we have to list all possibilities and make them as generic as possible. It will allow offering many choices as for the project’s realization. Thus, it will be possible to adapt the specifications to the various hardware and/or software supports.

3.4 Middle track

The more the functional needs are exact, the more the technical needs are identified. The use cases and the level of abstraction will make it possible to carry out the technical analysis. This is succeeded when all the technical difficulties are stated both on the software specification side as the material side.

4 CONCLUSION AND FUTURE WORKS

Our approach allowed to make a state of the art of methodologies and to study rapid application development techniques. The 2TUP unified process held our attention for elaborating the game design or mobile phones games.

Underscoring the functional and technical requirements which are on the left and right tracks of the Y is the beginning of this study. This stage precedes the conception based on UML diagrams.

Our goal is to continue this work, to run the whole Y and of course to apply to a concrete case.

Our first conclusions are rather positive as for the use of the method and the language. It is necessary to await the next results to determine if this approach could be adopted, developed even adapted to help with the development process of mobile games.

ACKNOWLEDGMENTS

Eric Gressier-Soudan is gratefully acknowledged for sharing with us his ideas and remarks.

REFERENCES

[BEL05] A. Belangour, J. Bézivin, M. Fredj – The design decision assistant tool : a contribution to the MDA approach – Premières journées sur l’ingénierie dirigée par les modèles _ IDM’05 30 juin/1er juillet 2005.

[CHU99] Church - FADT (Formal Abstract Design Tools), 1999.

[DES04] Les design patterns http://prografix.games-creators.org/document/204 - 2004.

[KRE03] Kreimeier - Game Design Methods: A 2003 Survey – Gamasutra March 3, 2003.

[OMG05] Software Process Engineering Metamodel Specification version 1.1, OMG document formal / 05-01-05, 2005.

[PRI05] J. Printz, « La dynamique des processus projets. Comment évaluer les méthodes agiles » Séminaire *Méthodes Agiles* – Journées du CMSL/CNAM, mars 2005.

[ROL00] A. Rollins, D. Morris, "Game Architecture and Design", Coriolis Ed, 2000.

[ROL03] A. Rollins, E. Adams, « on Game Design », New Riders 2003.

[ROQ04] P. Roques, F. Vallée, « UML2 en action » - Eyrolles – 2004.

[VEG04] L. Vega, « Modélisation et analyse spatiale et temporelle des jeux vidéo basées sur les réseaux de Pétri », thèse soutenue en décembre 2004.