Challenges in Market Driven Requirements Engineering

Final Term Paper

by: Mairon Toçi

5/4/2012

Submitted to:

Dr. Lawrence Chung
Challenges in Market Driven Requirements Engineering

Mairon Toçi
University of Texas at Dallas
mtoci@programmer.net

Abstract

Requirements engineering for market-driven software packages entails special challenges. Market-driven software development consists of developing packaged software to a market rather than to a specific client. It has been argued that the requirements engineering process for market-driven software differs from customer specific software. This paper presents the results gathered from the mobile games industry in understanding of the complexity and challenges of market driven requirements engineering processes. Of course there are a lot of things to consider, but the most characteristic problem in market-driven RE process is: “How does a company handle requirements if they don’t know their customers?” I will try to answer this question by focusing my research on the results of two cases described below and backing up my answer with additional support. At the same time, implications of these challenges and the ways to handle these problems will be described even further in this paper.

1. Introduction

The development of market-driven software products has gained increased interest from researchers and practitioners. These products have to satisfy different and distributed customers who may have very different needs. Usually the customers and final users are identified after the software is produced and is launched in the market. This fact reflects how difficult is the work that the developers of the product face while building it due to the insufficiency of information usually gathered from customers and final users. In this paper, I’m going to investigate the challenges faced during the requirement engineering phase of market-driven product, by focusing on the mobile video game. As it is discussed throughout the paper, mobile games development presents several distinctive challenges to requirements engineering practice. Mobile games development is an attractive market with projected revenue of $17.6bn by 2011 [1], in which games are mainly developed by small companies. The development process is considered especially creative but still rarely follows a well defined software engineering process. Contrary to most software applications, no one is obliged to use a video game. This simple fact has a lot of consequences. Creating attractive games based on emotional attributes like fun and enjoyment is difficult [2]. So a question that stays is: How does one handle requirements when he doesn’t know its customer, their expectations, and their preference changes? After considering the findings presented in the sector of mobile games, I will try to explain in general these challenges in the market-driven software products.

1.1 Motivation

The motivation of this paper is to create a better understanding of the complexity and challenges of Market driven Requirements
Engineering processes. Considering the results gathered from two examples in the mobile game industry, I will try to give a frame on how are the requirements handled in this sector of software development where the customer is not known.

1.2 Problem
Market-driven software products consist of a huge industry. In this paper, we are more concerned in finding the challenges that are faced during the requirement engineering phase of the production of this type of software. Of course there are a lot of things to consider, but the most characteristic problem in market-driven RE process is: “How does a company handle requirements if they don’t know their customers?” This statement consists of at least the following questions:

- How does a company select a target group?
- How does a company create adequate requirements, when these requirements are not delivered by customers?
- How can you fulfill different stakeholder’s points of view?
- Who makes the decision when is a conflict among stakeholders?
- How does a company check if a requirement is satisfied?

1.3 Solution
Everything looks abstract in this industry. As you can’t find always the right requirement suitable for a project (because you don’t know the customer), in the same way you can’t say that you can handle a solution for the challenges faced in this sector. Anyway, a lot of techniques and models are available to cope with the problems faced in RE of market-driven software. Some of these are presented in this paper.

1.4 Related work
Most of the authors who have written about this field have the same opinion when it comes to identify the challenges in the market-driven requirement engineering. Anyway a lack of concrete solution on how to handle these challenges is missing in the literature. Sometimes ideas that are applied in general in requirement engineering are suggested, not taking in consideration how difficult and complex are those to be applied in the market-driven sector. Some software challenges in new media applications are described by [3], and a report on the current practices of market-driven engineering processes for software products in [4]. After being mentioned in [3] the challenges in requirements engineering for mobile games development are considered in [2], by a study made in a Brazilian company: Meantime. A good list of the top-10 market-driven software development hypotheses related to challenges was written in [5], and some solutions are mentioned in [1], [6], [7].

1.5 Outline
Section 2 of this paper talks in general about market-driven requirements engineering. The focus of the challenges discussed here for market-driven RE is presented in section 3. Section 4 describes two case studies on development of two different mobile games: Frogman by Meantime and Memichi at Banff National Park Case. After introducing these two games, all the findings present there are described in section 5. Also in section 5 these challenges are generalized to help the market-driven RE case, by including also appropriated solutions. This research paper ends with the conclusions in section 6.

2. Market Driven-Requirement Engineering

Today, in the software engineering industry, Requirements Engineering is a topic that is widely discussed. RE is one of the first steps done during the software process of a new project. All tasks that are going to support the RE are to elicit requirements, prioritize the requirements, write a specification and validate the requirements.

Requirements engineering for market-driven software is different from RE for a particular software project. In the customer specific RE, the customer is well-defined and the requirements specification often acts as a contract between the developer and the customer.
When developing software products with the intent to sell them in market, the RE process should be able to invent requirements based on foreseen end-user needs and select a set of requirements resulting in a software product which can compete on the market. Market-driven requirements engineering is an interesting topic, way different in comparison to the traditional requirements engineering. In MDRE customers aren’t known. Due to this fact, no feedback is received from the customers.

In literature market driven RE are known to have this specifics [4]:

**The primary goal is time-to-market.** Since we are talking about market and competitiveness, time-to-market is considered as a “survival attribute”. Frequently a set of requirements of a lower priority may be excluded from certain realization if the deadline is near. Meeting the time to market is essential.

**Requirements are invented.** In this sector of software production there are only potential customers and users, so even if a requirement is good enough, you can’t say that it is satisfying. The requirements are, therefore, initially more or less invented by the development organization, and only later elicited from the customers and users.

**Requirements are rarely written.** Actually this is a serious concern. In this sector of software development, requirement documents are rarely written. Instead, requirements are most often communicated verbally within the development project, and maintaining requirements documentation is usually perceived as overhead.

However, by reading this paper, you will understand that the challenges of market driven requirements engineering aren’t only from the technical side. Social and organizational challenges have their impact here as well. First of all, the social challenges are faced in the coordination of the marketing and development departments, their ways of understanding and viewing the project, lack of communication etc. Also the organizational challenges are coordination and enforcing communication. Finally, there are also technical issues as release planning, techniques for requirements prioritization etc.

### 3. Challenges in Market-Driven RE

Talking about requirements engineering in market-driven software projects it’s quite interesting. The main reason is because in this kind of projects you don’t know your costumer. And if it’s not sufficient, you will be dependent of their feelings and preferences during the realization of your product. It is difficult to give solution to the problems in this segment of software development, but in this paper, I will try to segregate the existence of some of the problems and perhaps point out some guidelines that could help solve at least some issues. Here I’m going to cover some of the challenges that are evident in market-driven requirement engineering. After considering two cases of development of mobile games in section 5, I will try to give a frame of all the findings from these cases related to these challenges discussed.

Through considering three different researches [5] [2] [7], I came to the conclusion that the followings are the main challenges that the development of market-driven requirements engineering encounters:

#### 3.1 Target group

“How does a company select a target group?”

If you are going to say that you can see in every project the other four challenges that are described in this paragraph, you can’t say so for this first one. Selecting a target group is really difficult for mass-market-driven products. Why? Because you can’t select it! When a company develops a product, they don’t have a clear view of neither the customer nor the end user. They need to get advice from system’s stakeholders and do a lot of research to gather requirements from different viewpoints. If these actors think right from the beginning, less costly changes are going to be made to the project later. This is essential! Anyway, planning for future conflicts is normal. Conflicts will always exist when you develop a system which is going to serve many people with different expectations.

#### 3.2 Find adequate requirements

“How does a company create adequate
requirements, when these requirements are not delivered by customers?"

The elicitation of requirements for market-driven software products is known to be difficult [7]. A market-driven product usually starts from innovative ideas that fulfill need. After the initial market analysis shows profit, if the product is developed, it is time to make the requirement specifications. Clear goals can make the requirement phase much easier. You can judge easier if a requirement fits the goals or not if you know the goals of a project. In any case, new product development is particularly difficult because the product must present innovative features to charm customers.

Time-to-market is another pressure evident when developing these kinds of products [9]. We don’t face it often when we develop for a single customer or a specified target group, but this is important here. Market demand changes fast and it is determinant on the release of the product in time. If you miss this point, someone else will cover for you. If you don’t want to convince the customers that your product is better than your competitors, you better release it first.

3.3 Prioritization of stakeholders

“How can you fulfill different stakeholder’s points of view?”

When eliciting requirements, there will most definitely be split opinions about what specific requirements actually are supposed to do, and how they will do it. There will probably be issues if specific requirements should be used or thrown away. Somehow there has to be a prioritization of stakeholders. They may have various interests that pull the system in different directions which may require negotiations to resolve. This is mostly present among non-functional requirements, where difficult agreements often need to be made among requirements related to cost, flexibility, performance etc. Knowing how to compromise in order to please the stakeholders is important.

3.4 Managing conflicts

“Who makes the decision when is a conflict among stakeholders?”

I have already started considering the importance of managing conflicts in the previous paragraph. Team communication, good decision-making and flexibility are part of an adequate managing of a conflict. When there are disagreements about requirements among stakeholders, perhaps because the requirements are ambiguous or in conflict with each other, it is a good idea to contact the customer and book a meeting. In the meeting, the issues will be discussed and if the different parties cannot come to a conclusion, the customer has the final say. This is not the case here though. We don’t know our customer! It looks like the conflict can’t be solved easily. So really, who has the final word? Who will make the decision?

3.5 Is the requirement satisfied?

“How does a company check if a requirement is satisfied?”

When the system is finally ready to be tested, a new, but old problem arises. Who can confirm that the system works as it should? Of course, at this point in the project we have more or less all the information necessary to build our system, so what we can do is to make sure that the system is working the way we want it or the information says it should.

4. Investigating in mobile game requirement engineering

Considering the mobile games industry for understanding how the market-driven requirement engineering really works, is like adding challenge to the challenge. It is one of the most profitable sectors of the software development industry, so it’s huge, but in the same time it is a big mess. The main issue is with the development process which is considered especially creative, but still rarely follows a well defined software engineering process. There are a lot of requirement challenges that are related with this sector. A critical one for example is domain portability. You have to consider different features of the devices regarding user interfaces such as screen size, number of colors, sounds or keyboard layout; the application size; device-specific bugs; carrier specific requirements;
internationalization etc [1]. Anyway, I brought this list just to understand the complexity in writing requirements in this sector. In this paper I’m not going to cover these things. My purpose it to use the challenges found in this sector and generalize those in order to better understand the RE in market-driven software engineering.

4.1 Frogman by Meantime Case

Below I’m going to consider a study presented from two researchers. The first author is a specialist in requirement engineering from UFPE and the other is a project manager from Meantime Mobile Creation, which is an expert in mobile games development. Meantime is one of the biggest mobile developer publisher-distributors in South America. For better understanding the development challenges of this sector and their relation to the requirements engineering process, I’m going to describe the Frogman game developed by Meantime. Using the Frogman case, I will go through two issues:

- How to identify the critical requirements for mobile games?
- How to satisfy the requirements of very diverse and globally dispersed stakeholders?

With all the new games that we are facing today in the Smartphone era, Frogman looks like an outdated game. It still brings a really good example. Frogman is a casual game designed for everyone. The following is a short description of this game [2]:

“Join the strangest and funniest circus ever. At the Freak Circus an act draws the spectators’ attention: the jump of the Frogman. For this act, the Frogman climbs several platforms to perform dangerous jumps towards a barrel full of water. Help the Frogman reach his target so that he won’t crash onto the floor.”

There are two main activities that lead the designing of a game: idea generation and game design specification.

Idea generation is the entire brainstorm period which leads the team in understanding what they really think is a good new idea to work on. The workflow of generating a good idea for this company starts with discussing market trends and successful cases between game and graphic designers. After having a focus on the game story, the designers start analyzing other similar games that are in the market and the success that these games have among players. This phase is really difficult, because it is related with the understanding what casual players perceive as fun, and this is very subjective. Based on information gathered, the theme and game mechanic was decided. As the game was developed to users in different geographic locations, the game had to be simple enough to be played by everyone. Another requirement was that the theme should be familiar to users from different countries. Using all the ideas, a demo was presented to some casual gamers and marketing staff (because they are going to distribute the game). Gathering stakeholder’s feedback some changes and improvements were made to the final concept. The key requirement for a good game concept for Meantime is defining the right target audience for the game. Even that this requirement is really important, a right answer will never exist. Having an attractive game for a group of players, doesn’t mean that you can inspire another group. So understanding the correct interests and desires of a particular market segment is key to designers. At this point, Meantime faced difficulties because the carriers that had all the information to make a good market research never revealed the customer’s preferences. Anyway, Meantime’s team used parallel information sources from mobile game websites to resolve this lack.

In other words, the Game Design Document (GDD) is more like a requirement document. But in this industry it represents more than that. The GDD includes details about the game concept and technical details of the game. It is the main artifact of the pre-production phase and is used to guide the development process. Similarly to a requirements document, it describes the requirements for the game, but it also includes creative elements such as: storyline, description of sound and graphical effects, etc. All the goals that should be accomplished by playing this game are described here. Since it’s not in the interest of this paper, I’m not going to go into further details on this. After the first realization of the GDD, the team reviewed the document to check if all features were coherent and how some things could be
improved. Usually in this phase some conflicts are encountered. Same as for the Frogman team, another example in the next section, Memichi at Banff National Park, illustrates the same conclusion. The team is composed by game and graphical designer leaders and technical team leader that usually don’t agree on everything. In Frogman case, the technical team for example asked to clarify more the GDD, as it was not detailed properly to serve for the implementation phase. On the other side, the game and graphical designer team think that leaving flexibility to this document is a good thing. Many challenges were faced during the preproduction phase of the game, such as:
- Create a game mechanic with simple rules and user interface that was accepted in many countries
- Create a theme that was funny, innovative, and easy to understand in many countries
- Include features that could make the game more fun
- Find the best way to implement these features considering all the device restrictions
- Prove that the game would be a success to Meantime stakeholders (e.g. players, carriers, external publishers and company directors);

### 4.2 Memichi at Banff National Park Case [3]

Here is the development of another mobile game. After this second case, I will take into consideration the challenges related to requirements engineering and analyze the results from the experience of these two companies.

“Memichi was a context and location aware game application that ran on Smartphone’s. The game was supposed to: provide unique user experience, entertain, and engage visitors at Banff national park, Canada. A multi-disciplinary team of technologists, designers and multi-media production crew developed it. Broad game content ideas were generated in "participatory design" brain storming and creativity sessions by the team. Later the game designers evolved this content through multiple iterative design cycles. In each cycle, they required reasonably complete software version for each alternate content scenario before they could choose one and freeze some specifications for the final software. To make this possible, the software engineers built a tool which automatically generated software code within few hours, once the content is formally specified. The designers specified alternate contents and the tool automatically generated software versions to choose from. The tool architecture and its development were carefully planned so that the core modules were available in the early phases of the project. During the project the software engineers went on developing and maturing the tool whose design was decided a priori. This insulated them from the frequent changes in the content specification. The content production crew went on producing the multi-media content artifacts, whereas, creative designers went on experimenting with the alternative version of the design and evolved it. This tool and appropriate project management allowed the three application development flows to independently progress and yet interface at well defined points. It allowed incorporation of multiple system features at any time; hence the game application manifested rich context awareness. The application recognized a pre-determined set of hand gestures in addition to location. It also allowed generation of device optimized high performance codes, hence the application enjoyed high quality audio-video rendering even on mobile phones. It could build different versions of the application for different handheld devices other than smart phones.”

### 5. Findings

These two cases give a lot of feedback for the challenges mentioned in section 3. Below I’m going to cover all the findings from these two cases and I will also try to emphasize the disadvantages of some solutions that are widely used.

#### 5.1 Target group

Suppose for a moment that we conducted a good research and we found our target group. This could happen if we have in our hands a
really good idea that will satisfy mostly anyone out there. They are all exited to finally have our product in their hands, and we develop it. But wait a minute! What happens if they don’t like it after we lunch it? What happens if they had different expectations? And even if they were expecting exactly this product, what happens if they changed their mind during the development time? Is our product supposed to die after its first version? The mobile game industry suggests that real players are important sources of information to design attractive games. However, it’s still really difficult to identify the right target group. At Meantime, for instance, the idea generation is mainly conducted by game designers and there is little direct contact with real players (only informal chats with friends and staff’s kids who act as stakeholders). Consequently, designers may be biased to include their personal perception of what constitutes an interesting game for a specific target audience. In this industry, some requirements, like fun, are highly dependent on customers. What one customer considers as fun doesn’t mean that it should be generalized for the entire market. Meantime experience suggests that early user involvement could improve the game design and provide important feedback on gameplay requirements.

In general, interviews and studies are widely used to understand what are the customer’s needs and preferences. This is not an easy thing to do. Interviews and studies how different people perceive a specific thing take a lot of time. A solution to this can be to take a very small group and let them represent a large group of stakeholders. An advantage of this method is that the requirement elicitation will go much faster. Also, if you want to clear something during the development phase, you are going to call only this small group. A disadvantage can be that you can’t generalize the majority of all users by the thoughts of a small group. With this said, we can agree that a perfect product will never exist.

5.2 Find adequate requirements
Experience suggests that the needs of potential game users cannot be fully anticipated. This is related with the high competitiveness that is faced to deliver new games to the market. Therefore, the time to understand user requirement during the game design phase is really limited. As mentioned before, time to market is really essential in this industry. Another key difficulty involved in eliciting requirements for games is that games are designed to entertain users. Described in two words, the game must amuse and entice users. It is terribly difficult to deal with some requirements that can say: the must seduce the user in the first minutes of the game; otherwise the casual player will look for another distraction. However, it is still an open issue in requirements engineering practice in understanding of enjoyment and fun attributes. Another problem in finding adequate requirement can be due to the lack of understanding different vocabularies through the team. At the company who developed the Memichi game, the “prototyping requirement elicitation” cycles can be expensive due to difficulty in communication and idea sharing. Capturing artist’s visions, even in the informal qualitative narrative form, is quite difficult because artist and technologists do not share a common vocabulary. Both practitioners have to go through a learning process to understand the subtle meanings and connotations of each other’s domain specific terminologies, models and their semantics.

A solution to handle the requirement elicitation process is to have a special marketing department. This is not because they randomly know better the market, but because they work in parallel with the real needs of the market. The employees from this department visit different trade fairs, conferences and they talk directly to customers and end users. Their input can be used later to create high level requirements. In regard to these high level requirements, some problems exist. First of all, the engineering staff must be able to handle requirements from many different sources, as mentioned customers, stakeholders, marketing department etc. Another problem can be that there are a lot of requirements and to find conflicting requirements or requirements that depend on each other becomes harder. Because most of the requirements are written in a natural language, an automated lexical analysis can be helpful in these kinds of situations. Although it does not replace
the human judgment, these tools are effective. For this purpose, I’m going to mention two widely used and known models for the management plan of the product lifecycle. They have been developed at Ericsson Radio System and Telelogic [7].

They call the models for RDEM [Appendix A] (Requirement Driven Evolution Model) and REPEAT [Appendix B] (Requirements Engineering Process At Telelogic) respectively. The two models have been developed in parallel but completely independent. They are quite similar in many respects. Both RDEM and REPEAT have lifecycle models with different states for each individual requirement in its progress towards release. They have a string focus on roles and responsibilities. The general differences behind the two models are that Telelogic REPEAT is more focused on time-to-market, whereas Ericsson’s RDEM is more focused on quality. The lifecycle approach makes the requirement management more flexible so that new requirements can be introduced when needed and requirements can change based on changes in the market situation. A major problem with these type of models can be the release plan, but also to decide what to include in the current release. Many requirements have dependencies to each other. For Example “X must be done before Y” or “the cost for X decries if Y is done”.

5.3 Prioritization of stakeholders

Considering the case of Meantime and Memichi, there are several groups of stakeholders, and each of them have a different idea of which group of requirements are more important. Some stakeholders, in the two cases discussed are e.g. players, publishers, carriers, manufacturers etc. Nobody mentioned in this list cares much for the needs of the others. If their specific needs are achieved, everything is fine. But this doesn’t make any sense. Who can build a system (in our case a game) that satisfies all the requirements that these stakeholders say are necessary to include. Some things have to be dropped. Maybe a market analysis can be a solution, but how can we be sure that the result reflects the real world? An example that happed in Sweden illustrates this perfectly:

Not long ago, it was a parliamentary election in Sweden. Of course there were researches of what political party would win the election. The result said that the Social Democrats would lose, which, as it turned out, was not the case. In fact it was the total opposite. So, can we trust the result of a market analysis enough to use it for prioritizing?

By using a system like “CORE” [Appendix C], which is a system for organizing information from multiple stakeholders, the situation may become more pleasant. This model lets you create views based on stakeholders and their requirements. This will give a clearer picture of what stakeholders should have higher prioritization. Another approach to solve the problem is to use a model called “house of quality”, [Appendix D]. This model takes into consideration what requirements came from what stakeholder, and further more lets you see relationships between requirements. This way, it is possible to see if requirements are in conflict or not, and what stakeholders are involved in a certain requirement.

5.4 Managing conflicts

Game design is an intrinsically creative task, while the game production involves solid technical knowledge in fields such as mobile computing and artificial intelligence. Meantime’s and Memichi’s experience suggests that game design document must satisfactorily combine the design and technical views. The transition between pre-production and production needs to be carefully managed. The mobile game development asks specifically for flexible requirements. This does not mean that they need to be ambiguous. The cases indicate that clearer requirement engineering processes are fundamental to support the game design to inform the next phases of development. But since we are talking about games and art is involved anywhere, the specification of RE is suggested to allow the flexibility of this creativity process. Of course, this can lead in conflicts between designers and developers. This happens because game designers, who are especially creative individuals, generally try to include features that are very hard to implement. They argue that such features may improve the gameplay and the overall look and feel of the game. However, there is no formal process to
assess that; just common-sense is used. Normally, the final game is the result of tradeoffs among creative design, technical constraints and platform constraints.

A solution to manage conflict in a team, caused as a consequence of RE, is to assemble a committee of experts. Even that they are called “experts”, this does not mean that what they say will match the real world. In each case, the developing company must approach this problem if they want to win something at all. They will probably have to rely on market researches and expert committees, as well as their own hunch. One way would be to wait and see what happens, but that is not recommended, at least not if one wants to profit from the idea. There will be competitors who might not wait, and if there is a market, they will get there first.

5.5 Is the requirement satisfied?
This is a big question that is not only a challenge for the market-driven RE. How can we deal with this? Sometimes the use of formal proof techniques to prove that the system does what it is supposed to is an alternative. Also we can check for performance through different tests. However, all these methods will not be worth at all if we were wrong from the beginning. No existence of people to satisfy the requirements makes it hard to get useful critique about the system. Sometimes, even when we manage to find accurate requirements, we reach a point when it becomes hard to say if it is actually wanted, and more so if we have no customers. Perhaps the best solution is to use ones instincts and go with it. In market-driven segment, there is nobody to tell if the requirement is correct.

6. Conclusions
This paper reports on interesting issues discovered in the mobile game industry study on market driven requirements engineering. These findings are discussed before and after the experience brought from two game development companies. After collecting the theoretical issues from the literature, and considering the cases of Frogman and Memichi games, I have given some reports and possible solutions.

It is obvious that the requirements engineering phase for market-driven systems changes a lot from developing a system for a single customer or a specific target. The main reason of this paper is to be aware that you need to know this important aspect before you start to work on a project that has these characteristics. Usually, mobile games are developed by small or medium companies. It will be a good idea for these companies, but not only them, to make models of the processes used to develop if their intent is to produce more than one product. Using some kind of process patterns can make their subsequent work much easier. This is important since the key of success in market-driven product development doesn’t have fixed guidelines. This kind of software products encounter a lot of uncertainties, especially during the phase where the requirements are written. So each company needs to develop by following the flow that makes more sense to them.

Another important point to highlight is maintenance. Today systems will not be released only once, so having in place models to reuse will make the production of the next versions less expensive. We talked in this paper about the importance of using an automated lexical analysis to manage conflicting requirements or requirements that depend on each other. Beside the fact that this will help the maintenance phase, this is especially helpful when the requirements are collected by different people that maybe don’t meet each other too often. We have found that managing stakeholder’s requirements can be a problem in this type of market. There are many reasons related to this as for example the simple fact that customers and end-users don’t come from the same organization, sometime they are not even from the same country. In managing possible conflict in this situation, the use of RDEM or REPEAT model is a good option.

Prioritization of stakeholders is a must do thing. Some methods used in customer-developer situations, could probably be used in the market-developer situation as well. We have already mentioned “the house of quality” or CORE. There are however weaknesses. These two would for instance be good tools for handling conflicting stakeholders and their requirements. When it is time to perform tests,
however, there is no single person to ask about the requirements, because there will probably not be one or two persons to ask, but a whole market.

In this paper the communication gap between marketing and developers, resulted in insufficient time estimates and requirements quality, and as a result it was a big issue. Also the balance between marketing and developers’ requirements decisions is also recognized as a dilemma. Having a committee of experts in the organization could be a solution.

Perhaps there is no way of handling the fact that you cannot discuss with the whole market, and that there is no customer to tell you that this is right or that is wrong. Perhaps we just have to live with this fact, or maybe someday someone will find a general solution. This paper was written to present the challenges in market-driven requirements engineering and try to create awareness in readers mind. If one day they are going to deal with projects in this field, hopefully they will find a situation with consciousness and without a clouded mind.

References

[1] Experiences with Mobile Games Product Line Development at Meantime - Vander Alves, Tarcísio Camara

[2] Challenges in Requirements Engineering for Mobile Games Development: The Meantime Case Study - Carina Alves, Geber Ramalho, Alexandre Damasceno - Universidade Federal de Pernambuco – UFPE- Brazil

[3] Software engineering challenges in new media applications – Amitava Biswas (Texas A&M University), Jagmit Singh (Ontario College of Arts & Design)


[9] Challenges in Market-Driven Requirements Engineering; an Industrial Interview Study - Lena Karlsson, Åsa G. Dahlstedt, Johan Natt och Dag, Björn Regnell, Anne Perssson
Appendix A
The RDEM requirements lifecycle model
Progression of requirements in RDEM is defined in terms of what is known about the requirements. This knowledge is contained in the requirements attributes, and the state model can be viewed as a definition of certain levels of knowledge. The state model is designed to reflect the natural stages in a development process, and the decision points between states can be viewed as milestones.

Source: Requirements Lifecycle Management and Release Planning in Market-Driven Requirements Engineering Processes - Carlshamre and Björn Regnell

Appendix B
The REPEAT requirements lifecycle model
REPEAT is an RE process that manages requirements throughout a whole release cycle. It covers typical RE activities, such as elicitation, documentation, and validation, and has a strong focus on requirements selection and release planning. Management of requirements changes due to, e.g., new market demands, is an important function.

Source: Requirements Lifecycle Management and Release Planning in Market-Driven Requirements Engineering Processes - Carlshamre and Björn Regnell
Appendix C
CORE by Vitech
CORE is a requirements tool with the benefit of integrated system engineering. This means that unlike a typical requirements tool, where you simply trace your requirements to architecture, CORE allows you to do this parallel to a detailed analysis that not only traces but verifies and justifies your design.

Source: Vitech  http://www.vitechcorp.com/

Appendix D
House of quality
There are many different forms of the House of Quality, but its ability to be adapted to the requirements of a particular problem make it a very strong and reliable system to use. Its general format is made up of six major components. These include customer requirements, technical requirements, a planning matrix, an interrelationship matrix, a technical correlation matrix, and a technical priorities/benchmarks and targets section.
Source: Steps in Understanding the House of Quality Jennifer Tapke, Allyson Muller, Greg Johnson, Josh Sieck