Proceedings of the 6th WSEAS International Conference on Applied Informatics and Communications, Elounda, Greece, August 18-20, 2006 (pp400-405)

Communication Gaps and Requirements Uncertainties in the Information Systems Design

DENISS KUMLANDER
Department of Informatics
Tallinn University of Technology
Raja St.15, 12617 Tallinn
ESTONIA

Abstract: - The information systems engineering technologies and requirements gathering techniques are evolving on the permanent base. This evolution highlights some issues that were hidden so far or appeared with new techniques. In this paper we review requirements formulating and information system design problems produced by communication gaps, and uncertainty of requirements. Sources of those problems are described and some guidelines are proposed on avoiding or dealing with those problems. A "supporting" design and collaborative teams are proposed to stabilise requirements and construct information systems with less mis-modelling and mistakes.

Key-Words: - Software design, communication gaps, uncertain requirements

1 Introduction

The ultimate goal of developing information systems is to provide customers with tools that will help them run their business in a better way. Nowadays increasing competition and globalisation of business demands much higher quality of the released software, much shorter development cycle and increased flexibility of defining requirements. The proper software implementation over the low quality one provides benefits for both projects sides. A customer benefits due running their business properly and hopefully having certain advantages over their competitors. A project team has in the result a better image. Both sides benefit from saving a lot of resources because of decreasing the number of rebuilds and changes.

There are different modern methodologies supporting the process of achieving the described goals, capturing real requirements and providing a correct design including automated design check verification from a "programming" point of view [2, 6]. At the same time real software projects still demonstrate results that are far from our expectations. Main reasons of that are uncertainty of requirements and communication gaps. Requirements uncertainty is caused by quickly changing business world, wrong initial propositions of a person formulating requirements on what she or he would like to see in the result and many other sources. Communication gaps - by inability to describe and provide information on what a person is sure about.

In this paper all our previous researches in fields of analysing gaps and addressing uncertain requirements in the software engineering [4, 5] are concentrated and applied to the information systems engineering field. First of all high real approaches to information systems' design are reviewed in the chapter 2. The next two chapters describe communication gaps and requirements uncertainty. Thereafter ways to deal with those problems are presented in the chapter 5. The proposed techniques were applied in several companies and one of those is presented in the chapter 6 as a case review. The last section concludes the paper.

2 High Level Design Approaches

There are two main approaches in the design and requirements defining field used so far. The first approach bases on an assumption that software companies and their designers know much better what a customer needs and therefore could contact the customer and come up with a solution. This approach is also known as a consulting. The second one is opposite – information systems are built using only functional specifications provided by customers, although a software company can assist in formalising customers' requirements. This approach means that information system design will not be started until all documents and requirements are provided and all questions are answered. Questions occurring during the design phase of a project are also answered by customers. In both cases a set of documents and models are built and verified together with customers. Unfortunately many projects show that neither of those approaches can be applied without having certain problems. The real truth lies somewhere in between: it is always good to demand customers to provide as exact information as possible, but it is also hard to require customers to be designers. Main problems here can be the following: customers usually do not have necessary skills and hardly understand any case diagrams that designers can produce. Such difference in skills, knowledge and stereotypes can be a source of misunderstandings or communication gaps. Besides, practically each software designer used to work with uncertain requirements. Under uncertain requirements we mean requirements, which are changed during the design stage or after this stage is finished and a project is moved to the implementation phase. Sometimes designers try to fix requirements and do not accept any later changes, but this approach is rather wrong in many cases from our point of view. An information systems' design is made not because of the design, but to anticipate customers' needs. From another point of view it is impossible to redesign a product on a permanent base and especially if a product is on the final stage. That's why we are going to propose a new approach to information systems design and requirements collecting for cases where requirements are uncertain.

3 Communication Gaps

A communication gap is a term indicating the transformation of information during a communication. Basically it means that information, which is sent, is not equal to the received one after transmission. Mainly we concentrate here on the communication between persons and do not mean corruption of information in the communication channels like emails, mails etc. There are different reasons of distortion and main are listed below.

One reason is a physical distance between a customer workplace and a designer workplace. The designer in this situation cannot just walk to the customer office, talk face-to-face and ask to review the design / gathered requirements or do other things the designer needs to be done. Besides such a distance force them also to communicate in a "none-visual" manner that usually makes a communication between two different people much more problematic. It is also hard to organize "enough" meetings with customers as they are usually occupied with their business.

Another sources of gaps come from a generally problem of communication between any two persons that are explained by a difference in experience, skills, available information, life's and work's environments and culture backgrounds. This problem can be seen especially during interviews etc, when interviewer can miss important information, can miss an area to ask about the customer have not explained enough etc.

Another common problem is presenting a model in a form, which is unfamiliar to customers who have to verify that. This can be seen as a sub-reason of the previous group, but we rather exclude it from the previous part as it is very important and is connected to the modelling rather than to experience and so forth.

So far we have identified the next "communication" problems that can arise while designing an information system and gathering requirements:

- Impossibility to do/force to do something if it is needed;
- Insufficient quantity/quality of models/design reviews;
- Loss of information during a communication because of different experience, available information and so forth
- Inability to explaining fully the received requirement basing on "computer" models, i.e. models the customer is not familiar to work with.

All those reasons cause a certain probability that the information system will not correspond to requirements in the end of the project.

4 Uncertain Requirements

In this part of the paper we are going to review sources of uncertain requirements, classify those and identify why those exist.

Today movement and changing of information in business becomes faster and faster. That's why it is practically impossible to collect all information a customer may need before designing the information system that should assist in their everyday operations. It becomes normal that requirements can be a subject of change already on the project starting stage and this knowledge should be addressed in the project and requirements processing. There are two major types of changes. The first type is expected changes, i.e. when we know that something new can appear soon and it has to be addressed already now in the future software. Another type is unexpected changes, which can be sub-divided into internal or external changes. Internal unexpected changes are those that are caused by incorrect information in the requirements of a project. This could include wrong definitions, descriptions or even ideas of how it should work. Some of them cannot be identified by reviewing models or case diagrams. External unexpected changes are changes that come from an environment where software is designed or has to work. The environment includes both business environment of customers and a customers' internal environment. First of all we mean here changes that cannot be foreseen basing on reviewing requirements, reviewing software made by those requirements or experience of project team members. So, those changes are external from the project point of view. Another possible cause of requirements' changes during a project could be a need to quickly and adequately react on changes in the business world. Companies need to be flexible to remain competitive. Global markets are much more demanding and much quicker changing – all this will be also a challenge for a software design and should be classified as uncertain requirements that need corresponding methods to address in software design.

A specific problem for designing information systems comes from necessary to anticipate needs of a range of users having totally different requirements and views on the system. Usually users describe during an interview only his part and it becomes a challenge to obtain a full picture from those pieces of information. Sometimes those descriptions are in a conflict with each other. The only person that can resolve such conflict is a top-manager of both sides, but his availability is very limited usually therefore the question which side's requirement to fulfil can produce an uncertainty again.

Potentially dangerous can be a pure requirements' documentation that leaves enough space for misinterpretations. Both sides can try to benefit from it, but usually it will lead to a conflict that nobody would like to have. This will also increase a risk of un-satisfiability of the result.

Notice that communication gaps described earlier are also leading to uncertain requirements.

Reconciliation of uncertainty sources

- Missed information:
  - Requirements includes only individual opinions of users and do not provide a full picture;
  - Requirements are initially subject to change: since the customer doesn't have enough information at the moment, but will after some time; there is a high risk that requirements will have to be adjusted; the customer can foresee with some probability what part of requirements will be adjusted;
  - Unexpected changes – external changes: the customer would like to receive more than s\he originally planned because of changes in the environment;
  - Pure documentation;

- Errors
  - Unexpected changes – internal changes: Customer can find that his original ideas do not correspond to what he would like to have and will change his mind. It is a common case if sale persons who are weak in the formal thinking formulate requirements;
  - Error because of communication gaps like information loss, pure communication, inability to review design or model, inability of customers to provide a full information etc.

Needs to change requirements because of errors can arise on any project stage, but tend to not be discovered until the project end in many cases. The last fact generated a lot of jokes and funny pictures showing a huge difference between what was asked and what was delivered.

5 Ways to Address Communication Gaps and Requirements Uncertainty

It is impossible nowadays to define requirements purely inside a software company or to have complete and detailed requirements specifications from a customer. The first case is unacceptable by customers since often they want software companies or software departments to meet their needs, since software are made for them. The last software design method cannot be used nowadays because of requirements uncertainty, which we had described in the previous chapter. Besides software people have a lot of experience in technical details and similar projects and could help a lot to formulate requirements and functional specifications "right". The first part of this chapter addresses communication gaps problem that need to be resolved before the uncertainty can be processed. Of course each case of gathering requirements, designing and working with customers differs from others, therefore not all of the listed below advices can be used directly, but applying most of those will increase the quality of the result information system a lot.

The following principles can be used to avoid communication gaps:

- A person that is responsible for the project on the customer site has to be defined;

Information systems are usually built for external customers and sometimes it is hard to contact them. It is even harder to make somebody to review anything if your workplace locates somewhere else. Therefore a special person that is responsible for the project is needed at the customer site. All questions can be forwarded to this person and processed by him. This is the person that moves the project forward and is a customer representative. Notice that this person should be both responsible and interesting in the project. There should be some amount of reserved time that he or she can spend on the project, so other business responsibilities will not interfere.

- "Useful" meetings with the customer have to be established. Those should be well prepared and have a good timing (consider different time zones);
- Define rules, good practices, processes in the requirements gathering and models reviewing with customers as clear and simple as possible;

- Force to underwrite requirements documents and models/design documents – especially your variant.

The designer has to ensure that his document has been read, hopefully understood and accepted. It will secure his future work since customers tend to skip reviewing phase and this responsibility can make them review gathered requirements and built models. It can be useful to “play” through information systems’ scenarios and cases.

- Iterational development, shorter development circles [1, 7] should be used;

It is highly advisable to divide an information system construction project into a set of steps/iterations. An output of each development iteration is a part of information system (iteration’s features) ready to be used. The main reason of dividing into iteration: a model that looks correctly at the beginning could not be so obvious at the end due uncertain requirements and hidden issue that can be detected only using a fully functional system. Customers and the product team fill much more comfortable in the iterational development situation since they have a better view on the work progress. It is important to synchronize customers’ ability to review and the production cycle. Ideally each reviewed part of the information system should go live. This requires a decision on how the information system will be run, will it replace an old one (if any exist) or will be run in parallel etc.

It is possible to address uncertainty of the requirements after the communication gaps are avoided and stable requirements are defined. Today the information system’s engineers and their customers have to be flexible. The uncertain requirements problem arises more and more with this flexibility and we need to transform it into an opportunity. In this paper a collaborative team and the supported software engineering [5] are proposed to be used if uncertain requirements exist. The supporting engineering principle defines that the design should help (support) a process of formulating requirements, i.e. requirements that will be in the result both correct and correspond to real customers’ needs, instead of being just after requirements are gathered. It as an excellent way:

- for customers and designers to collaborate in a better way;
- to decrease number of mistakes and make requirements less uncertain on early stages;
- to address uncertainty directly in a project; keep them in mind and leave enough space for later changes.

The supporting design should help to find and provide to the design stage all information and requirements from customers including information on the requirements uncertainty; use design as an additional tool in formulating requirements etc.

As it was described above, the information system should be designed so that it will be possible to release the product in series of steps and those steps should be synchronised with customers’ ability to review and accept the system. A plan should contain features that are under a question because of the information lack as well by postponing those features to last steps. The plan that still considers features that are postponed helps to design first steps / features so that adding postponed features will not be a big challenge and will not require rebuilding of the entire information system. A possibility to start using a product starting from first releases will show disadvantages of the design and mistakes of requirements that could be still addressed until the project end and potentially can gain additional benefits to customers in case their old system is too bad or does not exist at all. Notice that reviewing prototypes or steps with customers helps to create the collaborative team.

Another important principle of the supporting design is to use documents that will simplifier discussion of different issues with customers, who do not have any special knowledge about UML, designing databases, case models and so forth. The design document should contain as many parts that can be reviewed together with users as possible. For example, it could be modelling a user interface basing on visual information like pictures or other UI prototypes [3]. Today too much documents are generated, which are irrelevant or is hard to understand. Such documents can neither be verified with customers nor developers will use as documents are too complicated. Besides we will not suggest having two different documents – one for customers and another for internal use with a lot of design details. There could be a problem to synchronise those as design models can raise questions about requirements that could be reviewed with customers. There should be an easy way to state question and incorporate answers into models again. Notice again that the design documentation is a base for collaborating between customers and designers.

Principles described so far should be supported by certain infrastructure. None formal reviewing process of documents and implemented steps needs to be established. Notice that perfectly formulated requirements documentation and iterational development of the information system provide just a possibility to avoid mistakes on early stages. The reviewing process is a process converting this possibility into reality. The biggest risk here is a “formal” reviewing, i.e. a review without having it really done.

The described approach to the information systems requirements gathering and design can be applied to a variety of cases where communication gaps and requirements uncertainty exist. The approach can be combined with other approaches and is an additional approach to major information system modelling techniques existing nowadays.

6 Case Review

The proposed approach has been adopted in several companies we were working for. The overall idea and requirements uncertainty problem came from those companies and motivated us to find a solution. One typical company will be described below to give an overview on how the approach is applied.

The company is a big global one developing hardware, software, providing telecommunication services and many others. We have been working for a department producing software components. This department is divided into several branches located in different countries. The overall number of workers is around 50 including developers, software designers and some consultants working with customers. A project we started to work with was very good illustration of what was happening in the department before. The marketing team decided right before a major release that a new functionality should be added into that as soon as possible. It resulted in developing the functionality that was not stable, i.e. that was developing in parallel. There were a functional specification developed by the business analyst and a design document, basing on which developers were working. Those documents were done rather despite of each other, since the design document sometimes contained parts that the business analyst wasn't thinking about, but developers have found and vice versa – the requirements document contained some thought that were not reflected in the design document. Sometimes the specification update was forced by the design document.

The following symptoms were discovered in that company in a lot of projects:

- Software has to be developed by requirements developed in parallel;
- Company should be flexible enough to allow that;
- Some requirements have been driven by the development team that do the "what-if" analysis during the development process;
- Two documents on the project exist, but neither accurately reflects actual requirements, design and current project status and therefore cannot be used directly for the specification underwriting without a time consuming synchronisation efforts.

The supporting design principle described in this article was the best way from our point of view to avoid failing for the project, anticipate the way they were already trying to work and even increase standard efficiency of the development process. A specification formulated quickly usually contains a lot of uncovered issues. The quick development arise those question making sometimes the design document to be better that the specification, especially when those questions are quickly answered online by the marketing/consultants team. In other words some answers appear quicker in the design document than in the specification one. The development team drives formulation of the specifications in this case because they need answers quickly, because they find problems and therefore this team supports the requirements stabilisation process a lot. The synergy between the business analysts' team and the design and development team was really sufficient and it moved projects forward dramatically.

The iterational development were adopted as the only way to see practically in the real-time how the functionality is done, test it and basing on it identify potential problems and unanswered questions. Notice that in some projects each iteration step was done during several weeks or even days, which is much less than in the agility development process. So small iterations worked perfectly for a functionality that is about to be added into the project in a hurry right before a version is released. It was the only efficient way to fulfil the requirements and release the version in the predefined time frame.

Another important element we used was meetings with customers to verify the developed functionality. This address the problem of uncertainty, i.e. how the functionality should be done as even consultants are not quite sure sometimes about how it will be used. Notice that this was possible only because of applying the iterational development described before. Such meetings saved a lot of time in formulating requirements and made possible to release a lot of projects in time. Unlike a common vision that customers are not fond of spending their time on reviewing projects in development we have found that they are. Mostly it is explained by the fact that they see it as a possibility to affect the project on early stages and ensure that the future versions will perfectly match their needs.

Besides the designer of the development team was forced to write all answers and rewrite specifications in his own language and submit it back to the business analyst working with this project. It was done to minimise a number of errors in the design and worked very well. It was even surprisingly how many parts of the requirements that look to be clear can be understood in another way. Notice that it was not said "wrongly" since verification of those mistakes demonstrated that requirements often contained improper formulations that were understood differently by different backgrounds' persons.

The last but not least element that was applied was defining a person that is responsible for a project. Consider that each project usually involves 3-4 departments: business analysis, development, marketing, testing; every department has a manager and sometimes also a person specifically managing that project. The problem was mainly in communication between different departments, i.e. between all those persons. There was a clear need for a person that could drive the project, ask each department to do their work, synchronise efforts and so forth. An overall efficiency of the project development increased a lot when one "central" person was defined.

In conclusion we should notice that all those steps allowed stabilizing requirements, releasing the project in time with the new required functionality, shortened the development cycle and made the general management team much happy than before. Instead of a long list of failed projects the company management see now successful projects resulting in functionality that is needed right now. This improved both flexibility of the company and the company customers satisfaction rate.

7 Conclusion

The main goal of information systems engineering is satisfying today and future customers needs as much as possible. The closer results of the information system development to the requirements the more advantages have both sides of the project. Unfortunately it is hard to ensure that real customers' requirements are met in the project even having advance modelling techniques because a variety of reasons. In this paper two main reasons were reviewed: communication gaps and requirements uncertainty. A communication gap is a term indicating the transformation of information during a communication. The distortion of information can happen because of a physical distance during the communication, differences in skills and experience of communicating people and some others. It is important to address those issues to stabilize requirements and avoid errors produced by the requirements gathering process. Another requirements problem is their uncertainty that we have to deal with during the information system development process. The uncertainty is produced by the quickly changing business environment, missed information and certain errors in customers' inputs on the project. Sometimes customers are not sure about their needs and wishes, or there could be conflicts of different users groups. A set of techniques is proposed either to decrease uncertainty or to address those in the information system development process. The last technique is very important as makes possible to work with uncertainties on early project stages and decrease a risk of re-building the information system dramatically.

References:

[1] B.W. Boehm, A spiral model of software development and enhancement, *Computer*, Vol. 21, No. 5, 1988, pp. 61-72

[2] T.T. Dinh-Trong, A Systematic Approach to Testing UML Design Models, *Doctoral Symposium, 7th International Conference on the Unified Modeling Language (UML)*, Lisbon, Portugal, 2004

[3] K. Hadelich, H. Branigan, M. Pickering, M. Crocker, Alignment in dialogue: Effects of visual versus verbal-feedback, *Proceedings of the 8th Workshop on the Semantics and Pragmatics of Dialogue, Catalog'04*, 2004, pp. 35-40

[4] D. Kumlander, Providing a correct software design in an environment with some set of restrictions in a communication between product managers and designers, *Proceedings of the Fourteenth International Conference on Information Systems Development: Pre-Conference*, 2005, pp. 1-11

[5] D. Kumlander, On software design and development supporting requirements formulation, *Proceedings on the 10th WSEAS International Conference on Computers*, 2006, *to be published*

[6] M. Rauterberg, O. Strohm, Work organisation and software development, *Annual Review of Automatic Programming*, Vol. 16, 1992, pp 121-128

[7] P.R. Reed, *Developing applications with Visual Basic and UML*, Addison-Wesley, 1999