Requirement Engineering Processes and their effects on Software Architecture

Sanam Narula[*]
University of Waterloo
s5narula@uwaterloo.ca

ABSTRACT
Requirement Engineering is the process of documenting the software requirements that are needed by the client for the software that is desired. It is one of the early and important stage in Software Development Life Cycle(SDLC). These requirements often changes as the project progresses. These changing requirements effects the architecture of the software being developed. These changes in requirements can be because of several reasons such as inappropriate requirement analysis, changing requirements on customer side, miscommunication on developer side and many more. In this paper we will discuss why these problems occur, what kind of effects they have on software architectures and the following phases in SDLC. These problems can be averted or their effect can be minimized by following proper requirement engineering processes. In this paper we will discuss various tool and techniques that can be used from requirement engineering.

Keywords
Software Architecture, requirement analysis

1. INTRODUCTION
Requirement Engineering is one of the crucial phase of software development because of the reason that all later phases are dependent on this phase [16]. Incorrect requirements leads to severe losses in software development. There are various cases such as well known project failure of Denver International airport baggage handling system and Ariane 5 which failed just because of the improper requirement analysis. Improper requirements can not only delay the project but can also result in project failures.

Software architecture is the blue print of the project. It is a description for the structure of system. It is known as the first artifact that can be analyzed to identify how well quality attributes are achieved [15]. It is also important because of its organizational and business significance. It supports the decisions which affect the software development and it's maintenance. Getting these decisions right is important to avoid project delays and failures [1].

Software architecture is normally followed by the requirement phase in software development life cycle. Once the requirement documents are finalized the architecture of the software is made in design phase and both these stages are interrelated. if the requirement changes take place, the software architecture may get affected and some time these requirement changes can result in change of the type of architectures. These problems of requirement phase can be eliminated with the use of various tools and techniques. This can result in significant time as well as money saving in various projects.

[*]Graduate student in Department of Computer Science

2. REQUIREMENT ENGINEERING
Requirement Engineering is the process of collecting requirements from the client and stakeholders using various tool and techniques and documenting them in a way that is helpful in analysis. One of the early product of requirement phase is SRS (Software Requirements Specification) document [3]. It describes the behaviors and the characteristics that are expected by the software. A quality SRS is one that contributes to the successful completion of the software project in terms of money as well as the time. It is a detailed document which include the needs of the user and contains the viewpoint of all the stakeholders [18]. Figure 1 shows the various inputs that are considered while doing Requirement Engineering. It also shows the output that is generated after applying requirement engineering processes.

2.1 Importance of Requirement Engineering
As we discussed in previous sections that Requirement phase is the first phase in SDLC which means that following phases depend on this phase. Any miscommunication in requirement phase can lead to problems in the subsequent stages. The main problems that come in absence of good requirement engineering processes are:

2.1.1 Cost of fixing missing requirements
Cost of fixing a requirement error increases as the time increases in the project. As the error keeps on propagating and fixing it becomes more and more difficult. Sometimes in later stages these bugs or missing requirements can result in big side effects such as the change in whole architecture of the project. These kind of changes results in delayed and unstable projects. There are various investigations done to calculate the loss in projects that happens due to unstable requirements. One of those is the series of studies conducted in the 1970's by Barry Boehm. He concluded that an error that is not discovered until acceptance testing can cost ten times as much to fix than if it was found during programming phase. If the requirement error was not found till the delivery time of the software in that case the cost to fix that error can be 100 times more [2]. These problems occur

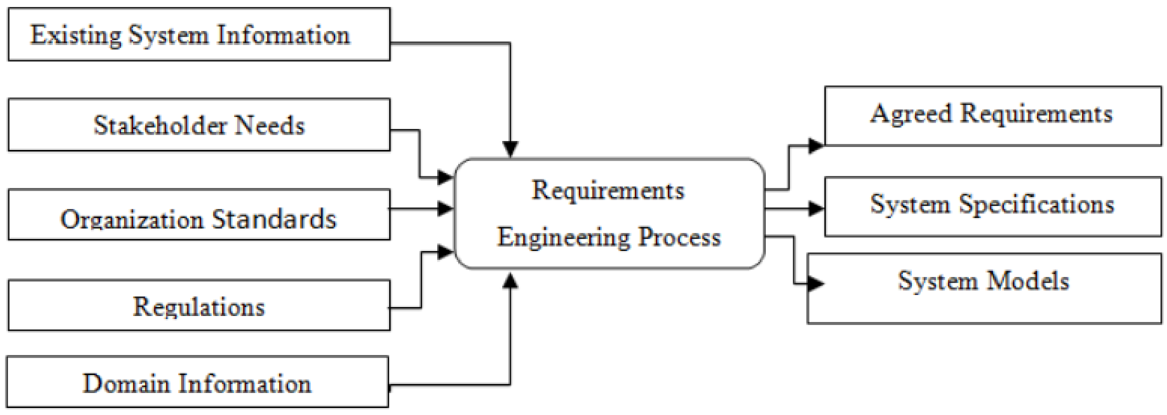

Figure 1: Input/Output of Requirement Engineering Processes [16].

because the design decisions are made on the wrong assumptions by the stakeholders or the customer is not clear about the product that he/she wants. There may be cases of miscommunication between various stakeholders or the reason may be that requirements were not well documented.

2.1.2 Project Failures

As we have discussed few examples of projects in Introduction section which are one of the costliest failures in the software industry such as Ariane 5 got failed because of improper requirement engineering. According to a survey conducted by ESPI in 1995 concluded that about 40-60% of all the bugs found in a software project can be traced back to errors made during the requirements stage [16]. Another survey that was conducted by Standish Group Study, 1994 concluded that 13.1% projects fail due to the incomplete requirements and 8.8% projects fail due to the abrupt changes that are made in requirements [10]. These two studies clearly shows the importance of following best practises and processes while performing requirement analysis and elicitation.

3. SOFTWARE ARCHITECTURE

Software Architecture is a description of system structure. It gives a very high level design of software. It is the backbone of the project. It is made in such as way that it includes the view of every stakeholder [15] such as :

- Developers must understand the working requirements of the system.
- Management should be able to make schedule for proper milestones that needs to be achieved on different time line.
- Testers should be able to make test cases properly by understanding the working structure.
- It should be informative and clear so that a new developer or any other person can easily understand it.

There are various types of architectures which are followed in software life cycles depending on what kind of system we are implementing. Figure 2 shows some of the architecture that are followed widely in industry. Normally in a project more than one architecture may be used if there are various modules. Whenever the requirement changes there may be changes in the architectural designs. Sometimes the requirements due to which the particular architectural style was chosen are dropped leading to the removal of the architectural style altogether which leads to change in significant amount of code leading to the time and economic problems.

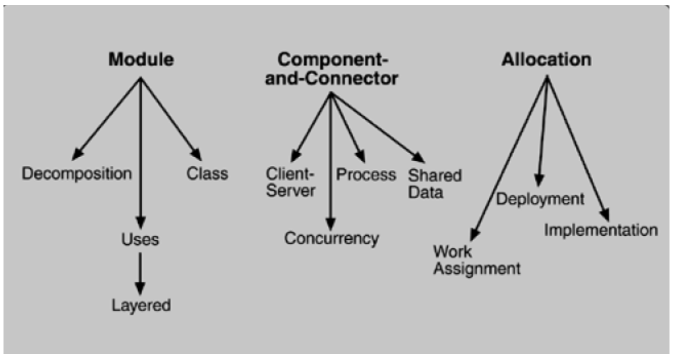

Figure 2: Common software architecture structures.

There are various domains of software development such as Game development in which the life time of the project is only 2-6 months. In such scenarios a wrong decision in architecture style due to the missing requirement can lead to the dropping of project.

4. PROBLEMS IN REQUIREMENT ENGINEERING

Various problems that come in Requirement engineering are:

4.1 Documentation

One of the biggest problems that come in Requirement engineering is the documentation. It is very difficult to document everything specially when the documentation gets outdated after few days. In most of the organizations the requirements are communicated through mails and other informal ways of communications which leads to requirement leaks. In case of projects which have a duration of 2-6 months documentation is considered a big overhead by the stakeholders. This happens due to the time pressure that less importance is given to requirement documentation which leads to delay in projects. A proper documentation of requirement can result in a well maintained SRS document which is referred by design stages to make architecture decisions and even the later stages such as functional testing.

4.2 Fragmented Environment

In few Organizations where stakeholders are of different backgrounds such as game development there exist a fragmented environment. There are stakeholders from different backgrounds so they work in different domains with different knowledge sets. As designers work for finalizing the design as to what the UI of the product will look like. There are often scenarios where a pipeline model of development of model is followed in which there is very less communication between the teams. This communication gap increases the misconception about the final product. Different stakeholders have different view of the system which leads to requirement understanding problems.

4.3 Informal Discussions

It is not possible to have a formal meeting every time something needs to be discussed between stakeholders. This leads to the use of different mediums of communication such as email, phone calls etc. Such communications are not formal so no documentation is kept for such kind of interactions. This should be avoided and each such communication should be documented in order to have a track of changing requirements.

4.4 Poor Quality of Requirements

Requirement Engineering is not just about documenting everything. Over-documentation can also lead to a problem where stakeholders start neglecting the requirement document. In many cases the quality of requirement is really poor which means that stress is laid on requirements that are not as important and are a side product [?]. This makes the real requirements look obscure and thus the SRS document look ambiguous. The problem occurs because normally such a documentation task is given to an inexperience person as documentation is considered as non-technical task in various organizations. This assumption gives the most technical work in the hand of an inexperienced engineer.

The other reason that is responsible for poor requirements is that in all the meetings only one person is given the task of a scribe. This person is responsible to document the requirements discussed in the meeting. The person assigned for documentation although experienced in his domain has a very little knowledge of the other domains. This may lead to poor quality of requirements.

4.5 Inadequate Requirement Validation

One of the most important part of requirement Engineering is that all stakeholders should validate their requirements in order to confirm that the requirements completely and correctly specify their needs [9]. But these requirements are rarely validated by the stakeholders. In some projects requirements validation is dropped due to a lack of funding or lack of schedule. This results in requirements that are incomplete as they fail to specify the stakeholder needs. The resulting system may then be unacceptable to many stakeholders. If the problems are fixed in later stage it result in delayed projects and increased cost. So, it must be ensured that requirements validation is a fundamental component of any requirements method [9].

5. ARCHITECTURE DESIGN AND REQUIREMENT DEPENDENCY

As we already discussed in Introduction section that Architecture design is the part of design phase of Software Development Life Cycle. The design phase follows the requirement phase. The architect build the software architecture design based on the requirements provided. If a change occurs in requirements in a later stage it directly impacts the architecture design. The impact of the change in requirements can be small and in such case a component of the architecture design need to changed or a new component needs to be added. In cases where requirement change is very large it can result in a change in architecture of the system. This change has an adverse impact on the following phases also such as on coding and implementation phases.

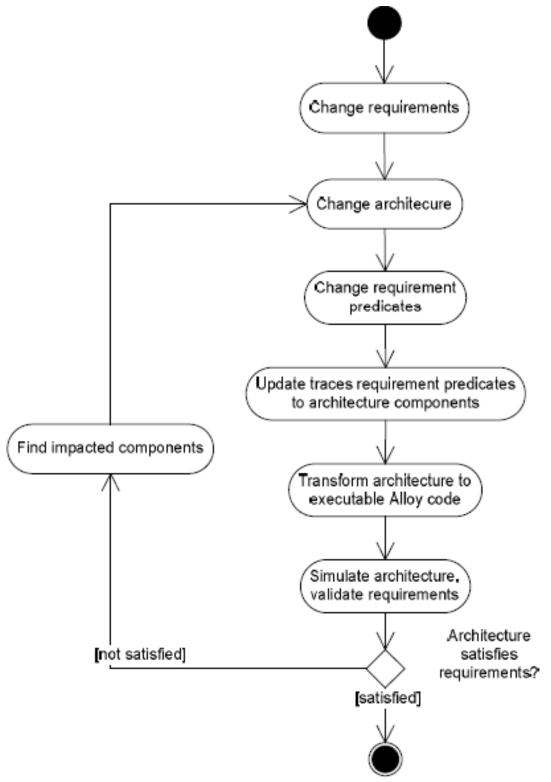
Change requirements
Change architecure
Change requirement predicates
Update traces requirement predicates to architecture components
Transform architecture to executable Alloy code
Simulate architecture, validate requirements
Find impacted components
[not satisfied]
Architecture satisfies requirements?
[satisfied]

Figure 3: Change impact analysis [17]

Figure 3 shows how changes in requirements affect the system. It shows the impact of the requirement change on the software architecture of the project [17]. The first stage in the figure 2 say the change in requirements which can be a change in functional requirement of the system. The new requirements update the architecture of the existing system. Change in architecture signifies that the architect changes the existing architecture by adding a new component or changing the architecture as a whole. When new requirements are added there may be the cases that the old requirements become obsolete. It is very important to update the requirement predicates so that updates in requirements are always documented. Next stage shows the changes in architecture which happens due to the change in requirement predicates. After changing the requirements the modification of architecture is done and requirements are validated. After the requirements are validated the changes are approved and made to the existing system.

6. TOOLS AND TECHNIQUES

There are a lot of tools and techniques that can help in eliciting and documenting requirements such that the effects of changing requirements on the project architecture can be reduced. We cannot eliminate the chances of requirement change completely as it is an inherent part of the SDLC. Some requirements generate too late in the development cycle. We can reduce the losses in time and money due to the missing requirements that occurs due to miscommunication and improper understanding of the stakeholder. In this section we will discuss various tools an techniques used for requirement gathering and documentation. Selection of a proper tool is also important as the tool selected should be appropriate for the type and complexity of the project.

6.1 Interviews

Interview is one of the oldest tool that is used in requirement engineering [11]. It is a one-to-one meeting with the customer. Normally a Business Analyst is a person who interacts with the customer in such a interview to elicit requirements from the customer. It is the responsibility of Business Analyst to get more and more information about the needs of the customer. After understanding the needs of the customer the business analyst explains the requirements in details to other stakeholders.

6.1.1 Types of Interviews

Interviews are often categorized based on their structure [5]:

1. Structured Interviews
   In these kind of interviews the questions are close ended and the person being interviewed is not allowed to say anything outside the context of the project. The data collected in such an interview is easier to analyze and is oriented because of the limited scope of the questions that are asked. This approach is mainly followed when there are a lot of interviewee and the questions covers all the scenarios.

2. Semi-structured Interviews
   In this kind of interviews, interviewer ask some predefined questions same as it is done in case of Structured interviews but other random questions are also asked in order to get more insight in the interests and understanding of the interviewee.

3. Unstructured Interviews
   These kind of interviews are very informal and there is no restriction on the scope of the questions that needs to be asked. These kind of interviews are very useful when the interviewer is an experienced professional and can get good insight in interviewee understanding very quickly. Since, there is no scope limit so these interviews can go beyond the scope of the project and can result in wastage of time. So, it is very important for the interviewer to have a control of the interview process.

6.2 Prototyping

Prototyping is one of the most effective tool for requirement analysis and elicitation. It is widely used in cases where customer is not sure about the requirements in the beginning of the project. In this approach a version of the product is made and is shown to the customer. The customer analyzes the prototype and mentions the new requirements that are needed and then again a new version is created including those requirements

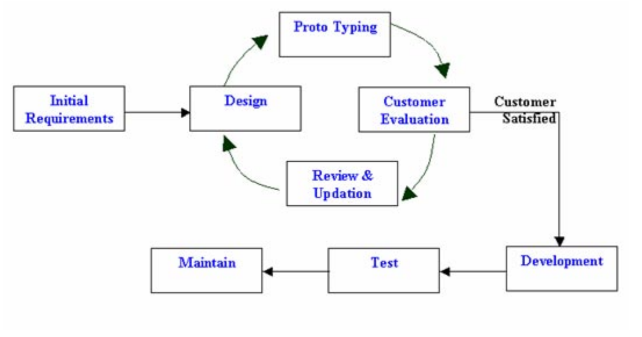

Figure 4: Requirements gathering using prototyping technique. [19]

Figure 4 shows the working that is followed in case of prototyping. Each cycle introduces new requirements and these requirements are included in the next cycle. This process is repeated until the customer agrees with the product. This process provides the detail analysis of each prototype. This tool is very useful in softwares development of GUI interfaces because it is very difficult to finalize GUI without looking them as a working product. This technique is also useful in application which are new to the market and some user study is needed to know the requirements as if the user will use such an application or not. If the process is not followed properly there may be redundant cycles of prototypes which can result in monetary losses. It is very important to analyse every prototype carefully so that no detail is missed.

6.3 Brainstorming

Brainstorming is a tool of requirement collection that is widely used in softwares that are based on creative ideas. In these software products the first step is to finalize the idea on which the software will be developed. Game development is one such domain in which the projects are dependent on ideas and brainstorming is one of the main tool that is used. In case of game development where there are a lot of stakeholders such as developers, designers and artists brainstorming plays a very important role. In a brainstorming session people from different teams sit together and give a lot of ideas. A light discussion is done on each idea. This discussion is not very deep as there are a lot of ideas and normally 2 to 3 sessions are kept in order to come up with a

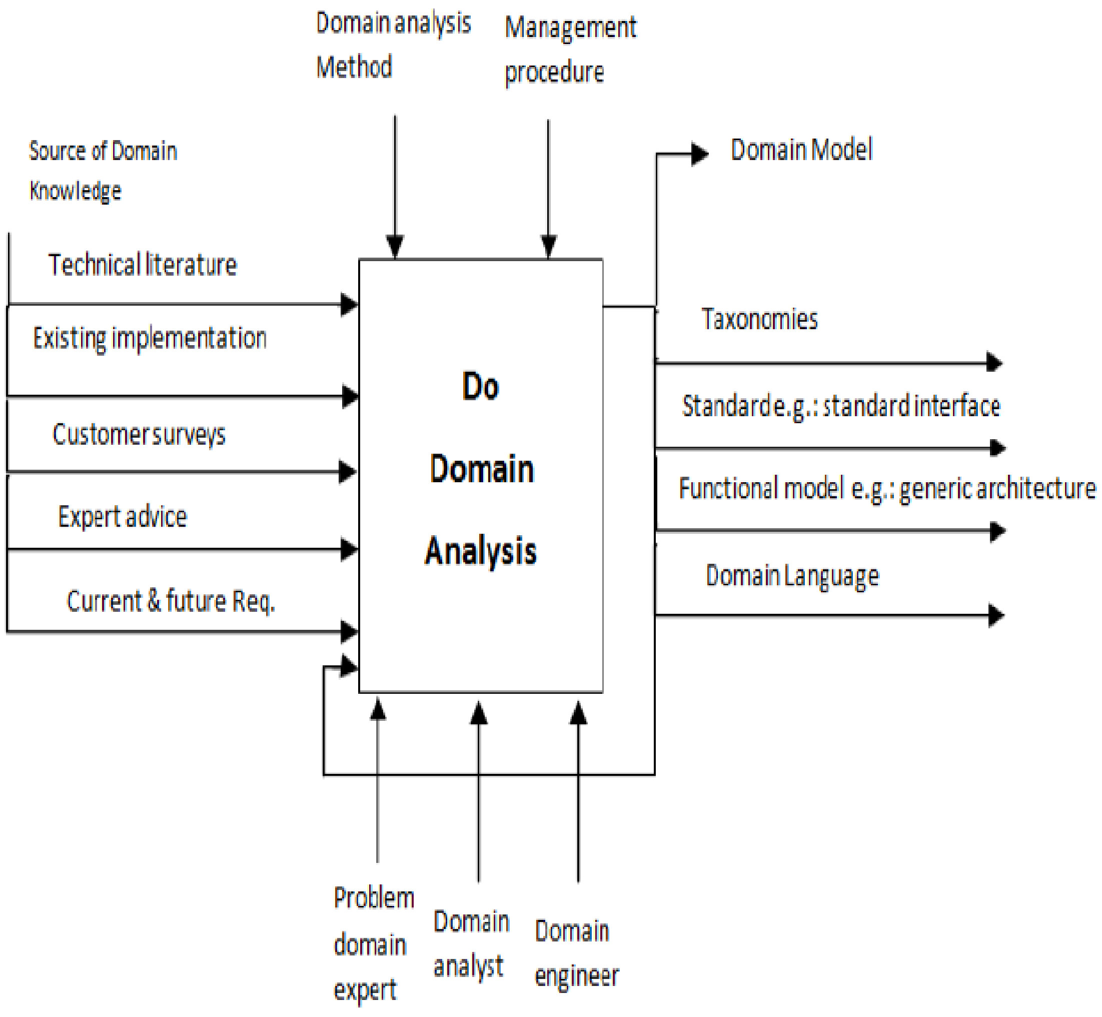

Figure 5: SADT diagram of domain analysis [14].

refined idea. This is a perfect tool in case where creative requirements need to be finalized because all the stakeholders can discuss the things at one place. This tool is not capable of resolving major issues as the people from different backgrounds can be biased towards their opinion and sometime it may be difficult to arrive at an unanimous decision.

An example that can explain the situation that may arise in such a case is that a game designers may not understand the limitation of AI required when designing non-player characters while software engineers may not understand the creative vision [8]. It is very important in these cases to resolve these problems in early stages of the requirement engineering. Every decision should be well documented so that all the stakeholder are in agreement over the decisions made. This can decrease the abrupt changes that occurs in requirement at later stages of development cycle.

6.4 Questionnaires

Questionnaires is another tool that is used to collect requirements. Questions in a questionnaire is designed in such a way that they can elicit more information. Questions can be closed as well as open ended. This approach is much faster as compared to the Interviews. The drawback of this tool is that the information collected may not provide holistic view of the system. The questionnaire limits a user to a set of questions and the user is not able to say more about the system which leads to the loss of information.

6.5 Domain Analysis

Domain Analysis is a process to identify requirements in a way that it can be reused in creating new systems from the existing one [14]. Version control systems are maintained so as to know if a similar requirements come then the code can be reused in those projects. The source code is at very low level in this approach.

It is a requirement elicitation technique [13]. In this technique a domain expert investigates the domain area of the system to get the requirements. This tool should be used in projects which are build upon the existing systems as an enhancement or an upgrade. Figure 5 shows the SDET context view of domain analysis. It includes everything from code documentation to the history of design decisions. The user manuals for existing systems are also managed and well documented. The main aim of the tool is to make the information readily available. This technique is used in collaboration with various other techniques. In case of interviews the domain expert can use this information that the domain expert already have to ask questions that will be more beneficial in eliciting requirements from the customer. This technique requires a lot of skills and expertise.

6.6 Surveys

Surveys are preferred when the scope of the project is very large and requirement engineering needs to be done at a very large scale. This technique covers a large geographical area. These are done mostly in case of general purpose software development [16]. The analysis of data can be done easily. There are various third party software such as Monkey Survey which are getting famous in this domain. The availability of such softwares has made this technique relatively less costlier as compared to other techniques.

6.7 Group Work

Group work is a technique that is used in projects that require collaboration between people of various background or in cases where there are more stakeholders. There are cases when more than two organization work for a customer on the same project. In these cases it is very important for the stakeholders to work together during requirement phases. This technique is really effective in resolving the conflicts. This approach streamlines the working and requirement analysis is done in detail. The drawback of this approach is that it is very difficult to arrange quick meetings as the stakeholders have different priorities and it is very difficult to arrange meetings for group work.

6.8 Observation

The observation technique is an effective means of deciphering how a customer needs the system to behave. It is done by an observation of the work environment of the customer. It increases the requirement analyst's familiarity with the culture and working style of a group of people. This technique is mainly used to verify the requirements and elicit some new requirements on the spot.

It is an authentic tool because the analyst goes to the proper environment in which the system needs to be deployed. He observes the use cases that come on the customer's environment. This gives deep insight in the product that is needed by the customer. It can be used in projects which include [4]:

- Improving an existing system.
- Customer is not able to explain their working and needs.
- Validity of requirements collected from other tools is questionable.
- Current implemented system need to be validated.

Observation should be organised in a way such that all the required data element are pre-determined. This reduces the uncertainty in observation process. This ensures that the analyst will be able to focus on the specific process which needs to be observed. Otherwise the analyst may lost the focus and get wrong observations. Observation is the collection of data that the analyst gathers and after gathering data it is refined to extract relevant information from the gathered data. The information that is extracted should clearly explain the requirements that are needed by the customer. Observation technique can be divided in two types:

1. Active Observation:
   In case of Active Observation the analyst can ask questions to any worker while the observation session is going on. The questions that are asked are normally not pre-determined and are made on the spot based on the experience of the analyst and the situation of the workplace.

2. Passive Observation:
   In case of passive observation there is no interaction between the analyst and the worker while the observation is being performed. There can be a questioning session after the observations session and the analyst will ask some pre-determined questions.

The data gathered during observation session is authentic and reliable. Observation technique is not suited for every business environment. Observations are normally expensive because it can only be performed by visiting the customer's work place which can be expensive and incur heavy cost.

6.9 Joint Application Development

Joint Application Development is a tool that can be used in cases when there are a lot of stakeholders in the project. This approach is very popular among companies following the agile practises in their development model. It provides a very fast way of requirement elicitation. It handles the rapid changeability in the requirements very effectively by providing a direct communication between the stakeholders. Since all the the stakeholders are present at the same place it is very easy to solve big issues between various stakeholders. It educates the customer about the development processes and a developer about the customer's requirements. It ensure that business objectives are met with in time. Figure

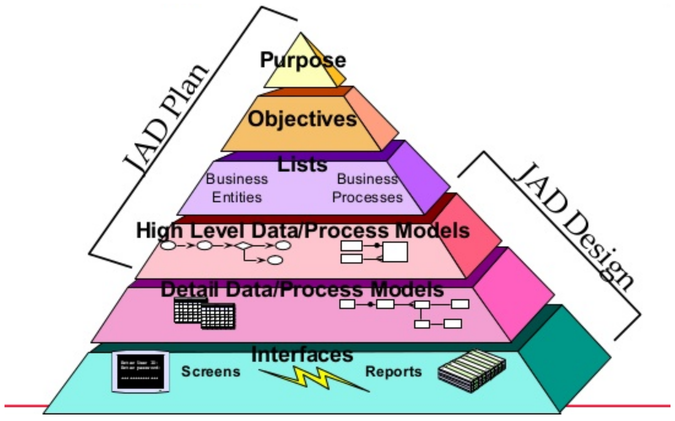

Figure 6: Phases of JAD [7].

6 shows the various phases in case of JAD. it consist of two main divisions:

1. JAD Plan:
   The JAD Planning phase is conducted when an idea has been finalized by the individuals within an organisation for a system. The aim of this phase is to get the system's high-level requirements and scope. High level design of the product is finalized.

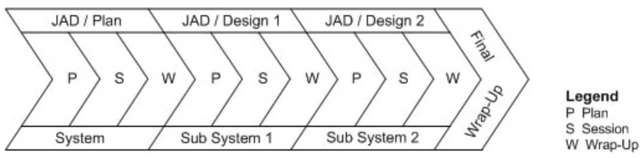

Figure 7: Stages in JAD. [6]

   There are three activities that are performed in any JAD plan as shown in figure 7.

   - Planning: In this activity high level requirements are identified, Number of sessions required are estimated. Visuals are prepared to facilitate the later phases.
   - Conducting the session: In this activity proper task briefing is done to all the stakeholders in the session. Stakeholders try to arrive at a unanimous approach that need to be followed in order to solve the problems. Requirement Validation is performed in this session in order to confirm that all the requirements are being met by the system.
   - Wrapping up: This activity is most important as it gives the concluding document which is normally a detailed specification document. The document lists all the requirements that needs to be met by the system. A presentation is prepared in order to pitch the idea to a potential sponsor.

2. JAD Design
   As you can see in Figure 7, the JAD design is performed on more detailed sub modules. In this phase the persons that are needed in the sessions are decided according to the need of the process. Training needs of the participants are identified. it results in high level process flow diagrams and reports.

Sometimes following JAD can also lead to requirement problems. As in JAD processes are done at a rapid pace so sometimes requirement validation is not given importance because of the time pressure. This should be taken care that requirement validation should be considered an important phase of the requirement engineering. JAD also requires experienced persons of the domain so that they can lead the group effectively.

It is not necessary that using all the tool will result in very good requirement elicitation but the correct selection of the tool is important. Different projects have different requirements and it depends on the complexity of the project that which could be the best tool for requirement engineering. Selection of the tool is usually taken by a domain expert. The domain expert has a knowledge of the system being developed and usually is an experienced professional. If proper requirement tool is selected it can give significant advantages and can lead the project on a smooth track. If the requirements are elicited properly and early then the project can easily be completed in time.

7. CONCLUSION

In this paper we discussed the importance of requirement engineering and how they can affect the later stages of Software Development Life Cycle. We also discussed the various problems that occurs in Requirement Engineering and what are their effects on the software architecture. We elaborated on the problems as to what are the reasons because of which the requirement leaks occur. We also discussed various tools and techniques that can be used to avoid these problems. We also stressed on the importance of selecting the right tool for the correct situation.

8. REFERENCES

[1] Bass, Len; Paul Clements; Rick Kazman (2012). Software Architecture In Practice, Third Edition. Boston: Addison-Wesley. pp. 21âĂŞ24. ISBN 978-0321815736.

[2] B. W. Boehm and P. N. Papaccio. 1988. Understanding and Controlling Software Costs. IEEE Trans. Softw. Eng. 14, 10 (October 1988), 1462-1477. DOI=10.1109/32.6191.

[3] Pierre Bouque and Robert Dupuis , ed (2004). Guide to the Software Engineering Body of Knowledge -2004 Version. IEEE Computer Society .pp. 2-1. ISBN 0-7695-2330-7.

[4] Business Analyst Learning, using the observation technique for requirements elicitation, May 2013.

[5] Courage Catherine, Baxter Kathy, Understanding Your Users: A Practical Guide to User Requirements Methods, Tools, and Techniques, (2005)

[6] Chambers, Joint Application Development: tinyurl.com/k77hsq7

[7] Crosby J, Joint Application design, April 2013: tinyurl.com/ouf2ejp

[8] D. Callele, E. Neufeld, and K. Schneider. âĂIJRequirements engineering and the creative process in the video game industryâĂIJ, Procs. of the 13th IEEE International Requirements Engineering Conference, 2005, pp. 240-250.

[9] D. Firesmith. âĂIJCommon Requirements Problems, Their Negative Consequences, and the Industry Best Practices to Help Solve ThemâĂİ. Journal of Object Technology, Vol. 6, No. 1, pp 17-33, Jan.-Feb. 2007.

[10] E David, Drehmer, M.Dekleva Sasa, A note on the evolution of Software Engineering Practices, (2001).

[11] Foddy W, Constructing questions for interviews and questionnaires. Cambridge University Press, Cambridge, (1994)

[12] H. Gumuskaya, "Core Issues Affecting Software Architecture in Enterprise Projects", In Proc. of World Academy of Science, Engineering and Technology, 9(1):32-37, 2005.

[13] Richardson J, Ormerod TC, Shepherd A, The role of

task analysis in capturing requirements for interface design.Interacting with Computers, (1998)

[14] R. Prieto-Diaz and G. Arango, editors, Domain Analysis: Acquisition of Reusable Information for Software Construction. IEEE Computer Society Press, (1989).

[15] Salehin, Md Rounok, "Missing Requirements Information and its Impact on Software Architectures: A Case Study" (2013). Electronic Thesis and Dissertation Repository. Paper 1811.

[16] Shams-Ul-Arif, Qadeem Khan, and S.A.K. Gahyyur. Requirements engineering processes,tools/technologies, & methodologies.International Journal of Reviews in Computing,2009.

[17] S. Looman. Impact analysis of changes in functionalrequirements in the behavioral view of softwarearchitectures, August 2009

[18] Singh, Rajinder."Comprehensive Study of Impact of Requirement Engineering Processes on Rework ". International Journal of Computer Trends and Technology (IJCTT) V9(5):227-241, March 2014. ISSN:2231-2803.

[19] Qastation, Prototype Model, 2008: tinyurl.com/o26gqz5.