

CHALLENGES FOR REQUIREMENT ANALYSIS

1. Harsimranjeet Kaur Bhattal

Co-ordinator CSE Deptt., GGSCMT Kharar
simmbhattal@gmail.com

2. Navreet Kaur

Lecturer, MCA Deptt., GGSCMT, Kharar

3. Rupinder Kaur

Lecturer, CSE Deptt., I.E.T. Bhaddal

Abstract - In systems engineering and software engineering, requirements analysis encompasses those tasks that go into determining the needs or conditions to meet for a new or altered product, taking account of the possibly conflicting requirements of the various stakeholders, such as beneficiaries or users. Systematic requirements analysis is also known as requirements engineering. It is sometimes referred to loosely by names such as requirements gathering, requirements capture, or requirements specification. The term requirements analysis can also be applied specifically to the analysis proper (as opposed to elicitation or documentation of the requirements, for instance). Requirements analysis is critical to the success of a development project. Requirements must be actionable, measurable, testable, related to identified business needs or opportunities, and defined to a level of detail sufficient for system design.

Keywords: Prototype, use case, domain expert, basecamp, stakeholder, JRD, SRS,

1. INTRODUCTION

Requirements analysis, also called requirements engineering, is the process of determining user expectations for a new or modified product. These features,

called requirements, must be quantifiable, relevant and detailed. In software engineering, such requirements are often called functional specifications. Requirements analysis is an important aspect of project management.

Requirements analysis involves frequent communication with system users to determine specific feature expectations, resolution of conflict or ambiguity in requirements as demanded by the various users or groups of users, avoidance of feature creep and documentation of all aspects of the project development process from start to finish. Energy should be directed towards ensuring that the final system or product conforms to client needs rather than attempting to mold user expectations to fit the requirements.

Requirements analysis is a team effort that demands a combination of hardware, software and human factors engineering expertise as well as skills in dealing with people. Why user experience disasters happen at the start of web projects explains that before designing an

interactive solution, you have to understand the problem: who are the future users, what are their current practices and what are their needs? This article lists some common objections/challenges against user requirements analysis and why you should not believe them.

2. MAIN TECHNIQUES USED FOR REQUIREMENT ANALYSIS

(a) Stakeholder interviews

Stakeholder interviews are a common method used in requirement analysis. Some selection is usually necessary, cost being one factor in deciding whom to interview. These interviews may reveal requirements not previously envisaged as being within the scope of the project, and requirements may be contradictory. However, each stakeholder will have an idea of their expectation or will have visualized their requirements.

(b) Joint Requirements Development Sessions (a.k.a., Requirement Workshops)

Requirements often have cross-functional implications that are unknown to individual stakeholders and often missed or incompletely defined during stakeholder interviews. These cross-functional implications can be elicited by conducting JRD sessions in a controlled environment, facilitated by a Business Analyst, wherein stakeholders participate in discussions to elicit requirements, analyze their details and uncover cross-functional implications. A dedicated scribe to document the discussion is often useful, freeing the Business Analyst to focus on the requirements definition process and guide the discussion.

(c) Contract-style requirement lists

One traditional way of documenting requirements has been contract style requirement lists. In a complex system such requirements lists can run to hundreds of pages.

(d) Measurable goals

Best practices take the composed list of requirements merely as clues and repeatedly ask "why?" until the actual business purposes are discovered. Stakeholders and developers can then devise tests to measure what level of each goal has been achieved thus far. Such goals change more slowly than the long list of specific but unmeasured requirements. Once a small set of critical, measured goals has been established, rapid prototyping and short iterative development phases may proceed to deliver actual stakeholder value long before the project is half over.

(e) Prototypes

In the mid-1980s, *prototyping* was seen as the solution to the requirements analysis problem. Prototypes are mock-ups of an application. Mock-ups allow users to visualize an application that hasn't yet been constructed. Prototypes help users get an idea of what the system will look like, and make it easier for users to make design decisions without waiting for the system to be built. Major improvements in communication between users and developers were often seen with the introduction of prototypes. Early views of applications led to fewer changes later and hence reduced overall costs considerably.

However, over the next decade, while proving a useful technique, prototyping did not solve the requirements problem:

Managers, once they see a prototype, may have a hard time understanding that the finished design will not be produced for some time.

Designers often feel compelled to use patched together prototype code in the real system, because they are afraid to 'waste time' starting again.

Prototypes principally help with design decisions and user interface design. However, they can't tell you what the requirements originally were.

Designers and end users can focus too much on user interface design and too little on producing a system that serves the business process.

Use cases

A **use case** is a technique for documenting the potential requirements of a new system or software change. Each use case provides one or more *scenarios* that convey how the system should interact with the end user or another system to achieve a specific business goal. Use cases typically avoid technical jargon, preferring instead the language of the end user or *domain expert*. Use cases are often co-authored by requirements engineers and stakeholders.

Use cases are deceptively simple tools for describing the behavior of software or systems. A use case contains a textual description of all of the ways which the intended users could work with the software or system. Use cases do not describe any internal workings of the system, nor do they explain how that system will be implemented. They simply show the steps that a user follows to perform a task. All the ways that users interact with a system can be described in this manner.

A use case should:

Use cases can be effective for establishing functional requirements for some but not all types of projects. Use cases focus on users' interactions with the system, and as such work well for end-user applications. However use-

cases are much less valuable in projects where deep complexity does not lie in user interactions, such as: batch processing, data warehousing, or systems with complex computations or detailed calculations.

Use cases are not suited to capturing Non-Functional Requirements. However Performance Engineering specifies that each critical use case should have an associated performance oriented non-functional requirement.

Software requirements specification

A software requirements specification (SRS) is a complete description of the behavior of the system to be developed. It includes a set of use cases that describe all of the interactions that the users will have with the software. Use cases are also known as functional requirements. In addition to use cases, the SRS also contains nonfunctional (or supplementary) requirements. Non-functional requirements are requirements which impose constraints on the design or implementation (such as performance requirements, quality standards, or design constraints).

Recommended approaches for the specification of software requirements are described by IEEE 830-1998. This standard describes possible structures, desirable contents, and qualities of a software requirements specification.

Stakeholder identification

A major new emphasis in the 1990s was a focus on the identification of stakeholders. It is increasingly recognized that stakeholders are not limited to the organization employing the analyst. Other stakeholders will include:

- those organizations that integrate (or should integrate) horizontally with the organization the analyst is designing the system for
- any back office systems or organizations
- Senior management

3. CHALLENGES

Stakeholder issues

Steve McConnell, in his book *Rapid Development*, details a number of ways users can inhibit requirements gathering:

- Users don't understand what they want or users don't have a clear idea of their requirements
- Users won't commit to a set of written requirements
- Users insist on new requirements after the cost and schedule have been fixed.
- Communication with users is slow
- Users often do not participate in reviews or are incapable of doing so.
- Users are technically unsophisticated
- Users don't understand the development process.
- Users don't know about present technology.

This may lead to the situation where user requirements keep changing even when system or product development has been started.

Engineer/developer issues

Possible problems caused by engineers and developers during requirements analysis are:

- Technical personnel and end users may have different vocabularies. Consequently, they may wrongly believe they are in perfect agreement until the finished product is supplied.
- Engineers and developers may try to make the requirements fit an existing system or model, rather than develop a system specific to the needs of the client.
- Analysis may often be carried out by engineers or programmers, rather than personnel with the people skills and the domain knowledge to understand a client's needs properly.

The management issues:

What managers in the developing company think the website should offer, is certainly interesting information, but these managers are not the future users. What they want on the site is not (necessarily) what users want on the site.

What managers know about future users is what they need to know for their job. Since their job is not designing websites, they probably don't have the kind of knowledge about users that is needed to design interactive systems.

The creator's issues:

Creativity and technology provide solutions. We still need to answer a real problem with this solution or it will not be used.

Do you want to design a product that the creators think is great or a product that users think is great?

The expert issues:

A user experience designer can't be expert in all domains. His expertise consists of methods to learn about user needs and skills to translate these into an interactive solution. A detailed understanding of users' needs is a competitive advantage. Companies developing an interactive system should not rely on public knowledge about users and their market. To obtain and maintain a competitive advantage with interactive systems, you have to know more about users than your competitor.

The budget issues:

The cost of "learning something about users you didn't want to know" at the start of a web project is small compared to the cost of learning it after launching the website.

The planning issues:

User requirements analysis can be scaled to the size of the project. It's better to do a 5-day analysis than no analysis at all. User requirements analysis should be planned from the beginning of a project.

Time-to-market is less important than being in the market with a product that is used by users. First mover advantage is often misunderstood. *You* don't create a first mover advantage by launching a website. It's *the user* who creates the first mover advantage by using your site.

The novelty issues:

If your system is so new that nothing is related to it, there's a big chance that nobody will understand nor use the system. If users can't relate to your product, you probably don't have a market. If you don't know the current way of working, how will you know whether the new way is more effective? Are you sure you will not create new problems with the system? On what are you going to base design decisions for the new system if you don't know the current way of working?

The ignorant user issues: User requirements analysis is not about asking users what system they want. User requirements analysis is about understanding users' current practices and the problems they encounter. Users are not interaction designers. They often don't know the possibilities and constraints of interactive systems. It's the job of an interaction architect to vision a solution that answers real user needs.

People are not all that different. There are patterns in their practices and needs. User requirements analysis is about recognizing these patterns.

This is a simple thing that shouldn't happen, but it does. A software update impacted a group that was left out of requirements. As a result, it cost more to fix it later in the project. [37signals' Basecamp](#) or a similar tool is a great way to record all communications between customer and designer. Basecamp is a web-based application that sends emails to affected parties along with a link to the message for replying. At the least, have one assigned point of contact for the customer and for the Web design firm. Ensure that contact is a reliable person and can do the job. No company wants to assign a contact whose answers will mostly be, "I don't know." It's OK not to have all the answers as you have a team to work with. The person should understand what goes on and respond appropriately.

The ubiquitous user issues: Involving a small sample of everybody is better than involving nobody at all. It's better to base your design on a thorough understanding of 10 users than on statistics about 10000 users. Statistical market research shows how often something happens in the target audience. It does not help us understand how or why it happens.

Now that more and more applications are used globally, international cultural differences can become apparent. However, patterns of users' behavior and needs will still appear and its important to take into account possible cultural differences when designing user experiences. There aren't that many products nor interactive systems that are really used by "everybody". Targeting "everybody" with an interactive system is not easy, nor cheap. User requirements analysis can contribute to defining precise target audiences that are more likely to use your product than others.

The academic issues: User requirements analysis uses some techniques derived from social science research. However, it is not like scientific research in its goals nor the way it is conducted. User requirements analysis is entirely focused on designing effective interactive systems. It does not pretend to answer research questions beyond that goal. In addition, timing and costs of the analysis are reduced to meet development demands.

The marketing issues: Statistical market research alone is not sufficient to design effective interactive systems. It shows how often something happens in the target audience. It does not help us understand how or why it happens.

The secrecy issues: Even a top secret product has to be optimized. Perhaps it's possible to find trustworthy users or user representatives.

The busy user issues: They spend time now or later struggling with product that are not adapted to their needs.

This is often true because much of development has to do with technology that's beyond the customer's knowledge. In terms of Web design, understanding the customer's mission, vision, goals, and other important parts of the project. In software development — especially large and complex software with many interfaces — requirements don't always affect customers. Requirements often focus on the back-end, processing and system interfaces. This is over marketing's head.

What works is sending an email (BEFORE requirements discussions begin) to one contact from each major group that has ever been affected by the software. The contacts reply whether or not they need to be involved in the project. If they do, then they should attend the requirements meetings. What about the problem of having too many people in the meetings? That's where having one contact helps. That contact should be communicating with the group.

The testing issues: Usability testing can reveal poor usefulness of a product. Only user requirements analysis shows how to fix it.

4. CONCLUSIONS

In this paper the techniques available for requirement analysis and challenges have been investigated to understand the problems of future users, their current practices and their needs. Also some common objections/challenges against user requirements analysis have been discussed in detail. We have concluded that requirements analysis is a team effort that demands a combination of hardware, software and human factors engineering expertise as well as skills in dealing with people either existing or future users..