Design for Requirements Engineering

Prakash Hegade

School of Computer Science and Engineering, KLE Technological University, Hubballi-580031
prakash.hegade@kletech.ac.in

Abstract: Requirements engineering is a fundamental and critical part of the software development process as every further step is influenced by it. Requirements engineering refers to the process of defining, documenting, and maintaining the project requirements. Interviews, brainstorming, task analysis, Delphi technique, prototyping, etc. are some of the methods for requirements collections where the stakeholders can be customers, business manuals, standards, existing similar projects, experts, etc. The modern digitized society and rapidly growing start-up culture present several gaps in the current process that needs immediate addressing. This paper breaks down the requirement process, its challenges, into various facets and discusses the methods to cover the existing gaps. Inducing a design aspect with wireframes into requirements that play a vital role, requirements are further drawn from infrastructure, competitor landscape, and culture. We call this 'Design the Requirements' approach. The paper systematically compares and classifies the traditional and our approach for a part of the restaurant application case study. The results show that contemporary projects are complex than what we consider to be and need a broader horizon of rational thought processes. The approach works towards the evolving and multifaceted modern society.

Keywords: design, requirements, wireframes

1. Introduction

The application development life cycle for any system usually consists of planning, creating, testing, and deploying. A system can be composed of software, hardware, or a mixture of both software and hardware. A software development life cycle to be in specific essentially consists of requirements collection, design, implementation, testing, and post-implementation reviews. The phases of the software development life cycle come with respective milestones, deliverables, and documentation to trail and administer each of the phases.

Prakash Hegade
School of Computer Science and Engineering, KLE Technological University, Hubballi-580031
prakash.hegade@kletech.ac.in

The objective of the software development life cycle is to produce quality software meeting customer expectations, which is developed in the estimated time frame and cost. The process of development is carried through various available models depending on the software need and requirements — the deliverables of each phase act as a feed into the next phase.

Requirements gathering or elicitation is the first stage in the software development process. Requirements engineering talks about the function points to be achieved and the constraints that need to be adhered to in building a real-world system. It is also concerned with the relationship of these factors to precise specifications of software behavior, and to their evolution over time and across software families (Zave, 1995).

Requirements engineering is usually conducted by the experienced team members with inputs from all the stakeholders and domain experts in the industry. Planning for the quality assurance, requirements, and recognition of the risks involved is also done at this stage (Dick J et al., 2017). This stage gives a clearer picture of the scope of the entire project and the anticipated issues, opportunities, and directives, which triggered the project. Requirements gathering stage needs teams to get detailed and precise requirements. It helps companies to finalize the necessary timeline to finish the work of that system (Pohl, 2010).

Requirements engineering is open-ended. It is interdisciplinary. It concerns translation from informal observations of the real world to mathematical specification languages. For these reasons, it can seem chaotic in comparison to other areas in which computer scientists generally do research (Zave, 1995).

Poor requirements have a significant impact on the results of systems or projects. The way civil engineers use Blueprint in building construction; likewise, Requirements are the blueprints for the software under construction. When there are poor requirements, this can lead to poor designs and tests, which in turn will cause delays in

development and testing (Mamoun, 2015) — with the current trends in the change of software development complexity, growing start-up culture, interconnected devices, digitization and modernization present new challenges to be tackled in the software development process and to be precise, in requirements engineering (Cheng and Atlee, 2007). The research question that is considered and addressed in this paper is - Do we have sufficient tools and techniques at disposal for requirements engineering towards a digital society and emerging start-ups culture?

Ian Sommerville, quotes on his blog, "We also need to cover practical methods, such as scenarios and stories, used by product managers to understand what might appeal to users. People buying software products don't have 'requirements,' and conventional requirements engineering is not very relevant" (Sommerville, 2018). This is important for the reason that most universities teach Software Engineering using his textbook. He also further quotes in his article that he is not coming up with a revised edition of the book but onto something new completely different. This paper works at one probable approach towards the identified goal.

The paper is further divided into the following sections: Section 2 presents the literature survey. Section 3 presents our methodology addressing the research question. Section 4 presents a case study. Section 5 presents results and discussion, and final section 6 presents the conclusion and future scope.

2. Literature Survey

Gathering requirements accurately and selecting the appropriate technique can assist in ensuring that all the systemic requirements are captured well. The major issues concerning the requirements engineering that are addressed through literature are shown in Fig 1.

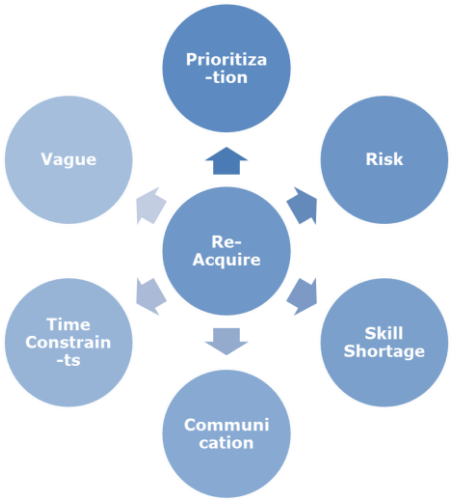

Fig. 1 Requirements Engineering Issues

The shades indicate the priority of the address where the darker shade indicates the high priority, which in fact is 'prioritizing' the requirements. Fig 1 shows the major parameters that are used to evaluate and factors that affect the collected requirements.

The research efforts in requirements have been classified Zave, 1995). Research directions in the domain have been studied (Cheng and Atlee, 2007). Requirements have been social modeled on social concepts and strategic analysis of relationships among social orders (Eric et al., 2011). Agile and traditional requirements have been compared (Paetsch et al., 2003). Goal-driven requirements engineering have been devised and analyzed (Yu and Mylopoulos, 1998), (Van et al., 2000). Scenario-based requirements engineering have been supported (Sutcliffe et al., 1998).

Viewpoint oriented requirements definition has been proposed to handle the requirements knowledge structure (Kotonya and Sommerville, 1996). Requirements have been treated as the success factor for software projects (Hofmann and Lehner, 2001). Various dimensions of requirements engineering have been studied (Pohl, 1994). The role of natural language in the process has been studied as well (Ryan, 1993). Metrics have been devised (Costello and Liu, 1995), and communication problems have been noted (Al-Rawas and Easterbrook, 1996).

Along with various research efforts from numerous directions, there are also commonly used and adapted methods. Below describes and gives a summary of the major ones followed in the process. The list is not exhaustive and is referenced from (Ian, 2011), (Jalote, 2012), (Nancy, 2019), (Cheng and Atlee, 2007) and (Loucopoulos and Karakostas, 1995).

A. Brainstorming

A group discussion always does a better job than individual contribution. Usually, brainstorming is used in identifying all possible solutions to problems providing a means for requirements gathering.

B. Document Analysis

Document Analysis is a vital gathering technique. Evaluating the documentation of a present system can promote to make the process documents. It also helps in performing the gap analysis.

C. Focus Group

A focus group is formed with customers and user representatives to gain product feedback. The feedback collected usually addresses the needs, opportunities, problems, etc. It can also be used to refine and validate the already elicited requirements.

D. Interview

Interviews help in understanding requirements better and to understand the perspective of each interviewee. It is helpful

in understanding user expectations. It positively benefits in knowing the objectives and goals of the system.

E. Observation
By watching users, a process flow, normal flow, exceptional flow, and etc. scope of improvements and opportunities can be determined. Observation can either be passive or active. They help in knowing the inherent process features.

F. Survey
A questionnaire survey can be used to gather requirements. The survey usually insists the users to choose from the given options to agree/disagree or rate something. A well-designed survey must provide qualitative guidance for characterizing the market.

G. Prototyping
Prototyping can be beneficial at gathering feedback. Prototypes are effectively done with fast sketches of storyboards and interfaces. Prototypes in some situations are also used as official requirements. They are also supported by user stories and scenarios.

H. Reverse Engineering
When a project does not have enough support or notes of the current system, reverse engineering can determine what a system does, where we deconstruct and work on the design.

3. Design the Requirements

The existing methods do not bridge the gaps present in the digitized society. The software's being developed presently is not the same as the ones that were developed decades or years ago. Design the Requirements (DTR) method brings in the design aspects into the requirements phase. In the traditional software development life cycle, the design comes after requirements collection, which comprises of high and low-level designs. DTR induces the design approach into requirements to improve the requirements collection process.

A. Design Goals
DTR has three major design goals: Current Trends, Ease of Design, and Re-Track. The design goals of the DTR are detailed below:

1) Current Trends: The requirements should inherently capture the current market trends and state-of-art. The customer might or might not be aware of, but requirements should capture them.

2) *Ease of Design:* The requirements process should ease the design phase. There should be a thin line of demarcation between requirements and design. The process should naturally flow from one phase to another as a continued step.

3) *Re-Track:* The requirements need to be tracked and monitored until the end of the system and as well during post-implementation testing.

B. DTR Process
The DTR process can be seen below, as shown in Fig. 2. Along with the traditional requirements collection process, DTR has three additional steps through which the scenario needs to be collected and analyzed. DTR also introduces wireframes that bring the design aspect into the picture.

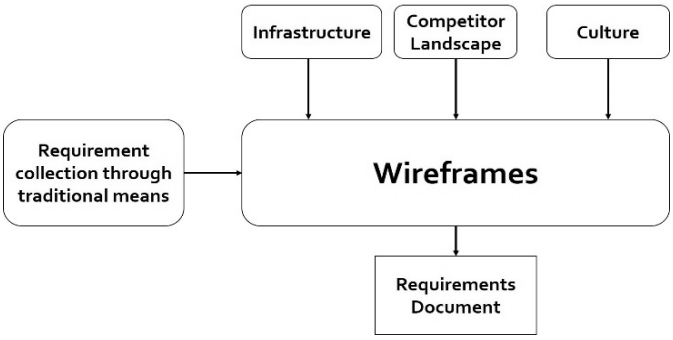

Fig. 2 DTR Process

The three major props of the DTR are shown in Fig 3 below.

Fig. 3 Three Props of DTR

1) Infrastructure: The infrastructure needs to be carefully studied from the stakeholder. The objective of this step is to understand where will be the product being developed get deployed. The objective can be achieved by visiting the places and having a walkthrough or by a recorded video from the customer or by setting up a meeting. Table 1. gives the checklist for infrastructure (which is a minimal sample set only). It has a set of questions for which the answer needs to be gathered and not limited to.

Table 1. Infrastructure Checklist Sample

Si. No.	Question
1.	What is the vision of the customer for the product?
2.	What is the approximate budget for the product?
3.	What is the infrastructure owned by the customer? (Do they have a chain of stores? Where will be the product put to use? etc.)

2) Competitor Landscape: Not only the start-ups but any product that is being developed must also consider the competitor landscape. It is essential to carry out the Serviceable Market Area (SAM) and Total Market Area(TAM) analysis. Any product that does not stand with the current state of the art and beyond the competitor would stand obsolete. The objective of this step is to gather all the competitor features, compare, and co-relate by generating the difference map and within the available limits on what could be incorporated within the product being developed.

3) Culture: Culture plays a major role in the software being developed. One needs to understand the roles and responsibilities of the product. Not knowing the culture could lead to misled objectives. Culture helps to understand the history and know the domain well. It is necessary to understand what will thrive and what will perish in the given domain. Along with the requirements, the past has to be studied, which is appropriate for the developing product.

C. Wireframes

Once we complete the three components, we then build the wireframes. The wireframes are extended requirements for the design. Wireframes are developed to help and understand the requirements better. They are carried out in the time span of 5-6 days. Wireframes help us to visualize the product and understand the flow better. Indeed, it is yet another way to collect the requirements. There are also ample tools available to build simple wireframes. Wireframes are weaker interfaces of the project that is being developed.

After carrying out the above-mentioned process, as explained in sub-sections B and C, we then prepare the final requirements to document using a trackable spreadsheet.

Case Study

This section presents a case study and compares the traditional and DTR approach. The case study is picked form a real-time project being developed at Transil Technologies. Only one requirement is considered for the demonstration.

The project is to automate the hotel and restaurant customer and engagement services. For the concern of privacy, the vision and mission of the product and the details are not mentioned. A particular requirement is selected to demonstrate the DTR process. One of the requirements for the user, where the user being manager of the hotel, recorded was:

Requirement (U-015): The user shall be able to bill the customer ordered food.

Following the DTR approach,

Following observations were recorded with respect to three components:

- With respect to infrastructure, the additional requirement captured were: there were two kitchens in the hotel, they had a take-home service, and the hotel had three dining rooms, and these aspects had to be considered for billing.
- With respect to the competitor landscape, the requirement noted was that there has to be an option to work with food delivery services like Zomato, Swiggy, etc. and have a billing integrating coming from those third party channels.
- With respect to culture, the requirement noted was adding gift coupons to the bills during the festival and seasonal periods.

With the considered requirements, from traditional and from DTR, a wireframe was designed capturing the necessary details and can be seen in Fig 4.

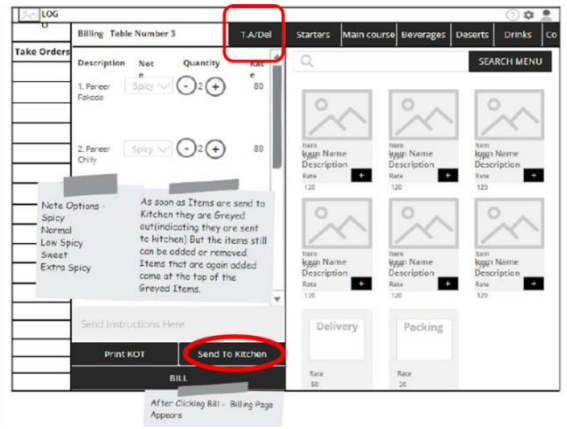

Fig. 4 Wireframe Design

As seen in the figure, the wireframe helped in understanding the requirement better and capture the new apparent requirements. After the wireframe, the 'Send to Kitchen' module was introduced, which otherwise was not a consideration. Also, the take-home delivery billing option came into consideration, which otherwise was missing.

4. Results and Discussion

In continuation of the requirements mentioned in section 4 for the Transil project on customer engagement and services, we here summarize the DTR approach. DTR fares relatively better in capturing the requirements and also helping out in the future design process. As seen in the case study, DTR helped in identifying the new requirements that otherwise seemed obvious and not captured.

Following is the summary of traditional vs. DTR approach. In traditional we have a single requirement, as shown in Table 2.

Table 2. Traditional Requirement Collection

Si.No.	Requirement
1.	The user shall be able to bill the customer ordered food.

In DTR, we have the requirements captured as following, as mentioned in Table 3, which are aggregated results from the three props and the designed wireframe.

Table 3. DTR Requirement Collection

Si. No.	Requirement : The user shall
1.	be able to bill the customer ordered food.
2	be able to send the order to the kitchen
3.	provide a take-away or a home delivery service

4.	be able to provide services through third-party services
5.	be able to add a gift coupon to the bill
6.	be able to bill for a customer from any of the three dining rooms

ID	Product Backlog Item	User Story	File Name	Function Description	Owner	Estimated Time	Actual Time	Comments
1	Task	Bill Customer						
1.1	Task_1	Order Type	The hotel manager can decide the type of orders	hotel/bill	The order given by the customer can be of any given type: eat-in, take-home or home-delivery. Additional charges need to be applied accordingly.	Prakash	3 hours	5 hours	Synchronization error debugging costed additional time
1.2	Task_2	Apply Coupon	The hotel manager can verify and apply coupon code	hotel/coupons	There can be different types of coupons: starter, festival, discounts etc. The code and respective amount needs to be added to the final bill.	Sujay	2 hours	1 hour	The task was modification on existing code
1	Task_3	Bill	Generate the bill for the customer	hotel/bill	Generate the final bill for the customer. The bill also includes the GST and 1% service tax.	Jay	3 hours	3 hours	Task completed on time

Fig. 5 Requirements Monitoring

Working towards the third design goal, the challenge is to track the requirement over the product timeline. Though there are many tools, it makes the process cumbersome. A simple spreadsheet application was used for the purpose. The sheet can be seen in Fig 5. The figure only captures part of the sheet maintained. The spreadsheet tracks the following information:

- Function point id and its subtasks
- Name of the activity
- User story
- Name of the function/file
- Description of the task it does
- Owner of the task
- Time taken to complete the task including start and end date
- Phase wise user story follow-up
- Any other comments.

We usually cover various ways to track the progress of requirements and the entire project, but all that industry needs is a simple tool like a spreadsheet that makes the tracking easy and captures the required user story.

Feedback was collected from developers and a customer involved with the project, and below presented is the analysis. The company had six developers who were involved in the project, and the questions and results are presented in Table 4. The rating scale used was one to five, where one is the lowest, and five is the highest.

Table 4. DTR Developer Feedback

Si. No.	Question	Average Score
1.	Rate on the effectiveness of understanding the system using DTR	4.33
2	Did DTR help in understanding function points better?	3.83
3.	Effectiveness of spreadsheet in tracking the project timelines	4.0

A customer who runs hotel chains in Hubballi was visited on two days. Day-1 was used to understand the requirements which the customer had. Day-2 involved explaining market trends and trigger towards DTR support. It was challenging to get the quantifed feedback from the customer for the process. So instead one single question was asked to be rated on the scale of one to five where one is lowest, and five is highest. The feedback is presented in Table 5.

Table 5. DTR Customer Feedback

Si. No.	Question	Day-1 Score	Day-2 Score
1.	How connected did you feel on each day of discussion while we were collecting the required data for the project?	3	5

5. Conclusion

DTR certainly certifies to be a promising and better approach for modern society. The results evidently show that contemporary projects are complex than what we think and need a broader horizon of rational thought processes. The process is also detailed out in the ebook BluePrint for Software Engineering (Prakash, 2019). Requirements engineering demands this flavor in teaching pedagogy.

DTR extends the canvas of requirements engineering to design, but that directly impacts the production and management time further in the life cycle positively. DTR needs standardization yet with respect to building the detailed formalized design template and the process workflow to carry out the methodology. The process also needs to be further validated and quantified by applying on other project developments and analyzing the collected feedback at various steps. DTR appears to be a promising

methodology that can be adapted at the student projects and also while teaching Software Engineering course.

Acknowledgment
I want to express my gratitude to Transil Technologies Private Limited, Hubballi, for all the support provided for the work. The experiments were carried out with the industry support of collaborative projects assigned to student teams of KLE Technological University.

References

Al-Rawas, A., & Easterbrook, S. (1996). Communication problems in requirements engineering: a field study.

Cheng BH, Atlee JM. (2007) Research directions in requirements engineering. In FoSE'07 2007 Future of Software Engineering 2007 May 23-25, Minneapolis (pp. 285-303). IEEE Computer Society.

Costello, R. J., & Liu, D. B. (1995). Metrics for requirements engineering. Journal of Systems and Software, 29(1), 39-63.

Dennis Alan, Haley Barbara Wixom, Roth Roberta M. 2014). Systems Analysis and Design. pages 89-93. John Wiley & Sons.

Dick J, Hull E, Jackson K (2017). Requirements engineering. Springer.

Eric, S. K., Giorgini, P., Maiden, N., & Mylopoulos, J. (Eds.).(2011). Social modeling for requirements engineering. MIT Press.

Hofmann, H. F., & Lehner, F. (2001). Requirements engineering as a success factor in software projects. IEEE, (4), 58-66.

Ian Sommerville. (2011) Software Engineering. 9th Edition, Addison-Wesley Publication.

Ian Sommerville (2019). What should we teach in software engineering courses. Link: https://iansommerville.com/systems-software-and-technology/2018/03/18/what-should-we-teach-in-software-engineering-courses/. 18 March 2018.

Jalote, P. (2012). An integrated approach to software engineering. Springer Science & Business Media.

Kotonya, G., & Sommerville, I. (1996). Requirements engineering with viewpoints. Software Engineering Journal, 11(1), 5-18.

Loucopoulos P, Karakostas V. (1995) System requirements engineering. McGraw-Hill, Inc.

Mamoun Eid. (2015) Requirement Gathering Methods. umsl.edu.

Nancy Dehra. (2019) Project Planning for PM's, www.brighthubpm.com.

Paetsch, F., Eberlein, A., & Maurer, F. (2003). Requirements engineering and agile software development. In WETICE 2003. Proceedings. Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003. (pp. 308-313). IEEE.

Pohl, K. (1994). The three dimensions of requirements engineering: a framework and its applications. Information systems, 19(3), 243-258.

Prakash Hegade (2019). BluePrint for Software Engineering. First Edition. Smashwords. Link: https://www.smashwords.com/books/view/953345.

Pohl K (2010). Requirements engineering: fundamentals, principles, and techniques. Springer Publishing Company, Incorporated.

Ryan, K. (1993). The role of natural language in requirements engineering. In [1993] Proceedings of the IEEE International Symposium on Requirements Engineering (pp. 240-242). IEEE.

Sutcliffe, A. G., Maiden, N. A., Minocha, S., & Manuel, D. (1998). Supporting scenario-based requirements engineering. IEEE Transactions on software engineering, 24(12), 1072-1088.

Van Lamsweerde, A., & Letier, E. (2000). Handling obstacles in goal-oriented requirements engineering. IEEE Transactions on software engineering, 26(10), 978-1005.

Yu, E., & Mylopoulos, J. (1998). Why goal-oriented requirements engineering. In Proceedings of the 4th International Workshop on Requirements Engineering: Foundations of Software Quality (Vol. 15, pp. 15-22).

Zave P. (1995) Classification of research efforts in requirements engineering. In Proceedings of IEEE International Symposium on Requirements Engineering (RE'95) 1995 Mar 27 (pp. 214-216). IEEE.