ENGR 6991 Report

Prepared For

Dr. Yong Zeng
Canada Research Chair in Design Science
Associate Professor
Concordia Institute for Information Systems Engineering,
Concordia University,
1515 Ste-Catherine Street West,
Montréal, Quebec, H3G 2W1, Canada.

Prepared By

Tanvir Khan
ID 6019285

Rajbir Kaur
ID 6004628

Research Directions in Requirement Elicitation and Modeling

ABSTRACT

Defining the system requirements is the most important activity in the development of a software system. This activity usually requires close collaboration of stakeholders with different and even contradictory needs and perspectives. It is a critical part of software development because errors at this stage propagate through the development process and very hard to repair later. Another point is that the software development team are often geographically distributed from their end users which create communication challenges that impact the effectiveness of requirement elicitation. This paper shows the current research trend in requirement elicitation. It also compares a number of elicitation techniques and proposes which method is applicable based on the software project classification. It describes the main areas of RE practice and highlights some key future research issues.

The second part discusses the "Requirement modeling" part of Requirement engineering. A Requirement is a software capability needed by the user to solve a problem to achieve an objective or a capability that must be met or possessed by the system or system component to satisfy a contract, standard, specification, or other formally imposed documentation. Requirement Modeling is an analysis activity during which an initial graphical description of the relationship between system entities is proposed. This part is divided into three sections: Section 1 explains about the Requirement modeling its use. Principles, techniques, tools etc. In section 2 various approaches to requirement modeling is discussed and comparison with the similar approaches is done. In section 3, different types of requirement modeling methodologies are discussed briefly. And at the end of this paper State of art of Requirement Modeling is done.

Keywords: elicitation, techniques, state-of-the-art, requirement artifacts, stakeholders, non-functional requirements, requirement engineers, use cases, communication.

REQUIREMENT ELICITATION

1. Introduction

Though Requirement Engineering is common sense, it is perceived to be difficult, not well understood and poorly done. It is also about management and hence issues in relation to requirements and management blend to show how requirements can be used to manage software development.

Requirements are the basis for every project. It starts with defining the stakeholders – users, customers, businesses, developers, suppliers involved in the system. To be well understood by everybody they are generally expressed in natural language and the main challenge lies here – to capture the need and/or problem completely and unambiguously. Once the communication is established among the stakeholders, requirements drive the project activity. The need of the stakeholder may be many and widely varied and can be in conflict with each other. These needs may not be clearly defined at the start of the project or even they can be constrained by factors outside their control or may be influenced by other goals which themselves change in course of time.

Our approach taken in this article is based on current research in Requirement Engineering; however it has not only taken the academic view but has also built on current experience of working in industry to enable software engineers to understand the requirements more successfully. The consequences of having no or little requirements are many and varied. There is ample evidence around the software industry that failed because requirements were not properly organized. However well the software may appear to work at first; if it is not the system that users want or need, it is useless.

The requirements development phase may have been preceded by a feasibility study, or a conceptual analysis phase of the project. The requirements phase may be broken down into:

- *Requirements elicitation* - gathering the requirements from stakeholders
- *Analysis* - checking for consistency and completeness
- *Definition* - writing down descriptive requirements for developers and
- *Specification* - creating an initial bridge between requirements and design.

2. Understanding the Classification

2.1 Main classification of requirements

Requirements are mainly placed into two categories:

- *Functional requirements* - describe the functions that the system is to execute. They are sometimes known as capabilities.
- *Non-functional requirements* - requirements are the ones that act to constrain the solution. Non-functional requirements are sometimes known as quality requirements.

They can be further classified according to whether they are performance requirements, maintainability requirements, safety requirements, reliability requirements, or one of many other types of requirements.

2.2 Other types of requirements

There are several other classes of requirements such as :

- *Users' requirements*- list the tasks and goals of the user or consumer. User requirements are intended to make the tool or product easier to use, faster, less error prone. User

requirements are separated from other requirements to provide clarity and a focus on user experience. Under obsolete requirements classifications the inclusion of user requirements in other requirements has resulted in poor user experiences and, in extreme cases, the rejection of the application or product.

- *Business requirements*- list the goals of the business. At the highest level, these goals are to increase revenue, decrease costs, improve data management, increase knowledge transfer, improve efficiency, and so on. For example, an internal software application is made only when someone in the business recognizes a need for a better-than-current system or process and implements a project to fill that need.
- *Process requirements*- specify the limitations on the development processes, methods, techniques that are allowed to be used in the construction of the target system or software. In a product manufacturing example, some process requirements would be manufacturing requirements, or the conditions, processes, materials, and tools required to get the product from the design board to the end users.

3. Why Requirements Engineering is Difficult?

70% of the systems errors are due to inadequate system specification and 30% of the system errors are due to design issues.

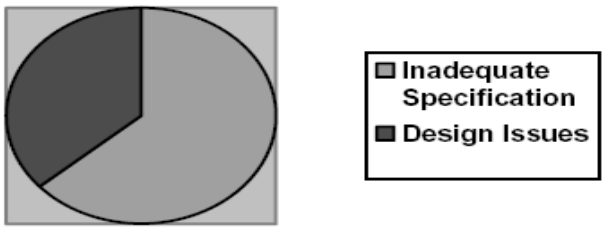

Figure 1: Reasons of software project failures

RE is the process of defining the problem that the proposed software will solve and other Software Engineering activities are the process of refining and defining a software solution. The following issues make RE a tough job:

3.1 The requirement artifacts need to be understood by all the stakeholders; such as developers, customers, software architects, etc. To be easily understood by all the parties, detailed notations should be avoided and Requirement Engineers find it hard to make the documents easily readable.

3.2 Usually it becomes tough to make the assumptions and dependencies of the environment on the system.

3.3 As the system needs to operate on a combined environment (human-behaviour, hardware devices, physical world phenomenon, etc) , Requirement Engineers find it hard to cope up with the complexity of the behaviour that pose on the system.

3.4 Different stakeholders of the system usually have conflicting requirements and the RE must have a balanced set of requirements to satisfy almost all of them.

3.5 As the RE is a group of technology-oriented people, there remains a communication gap between them and the customers which leads to the incapability of negotiation.

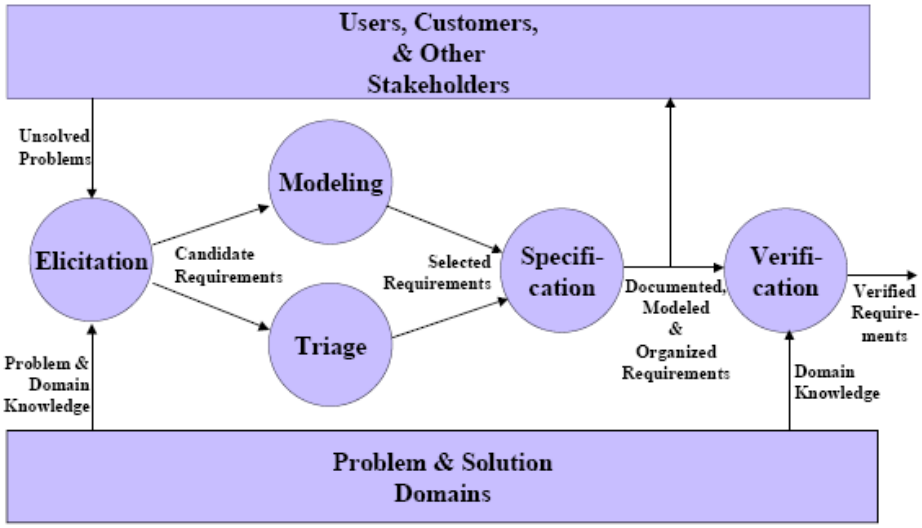

Figure 2: Basic features of Requirement Engineering

4. What is Requirement Elicitation?

This is the very first step of Requirement Engineering. It's all about understanding the goals, objectives and motives for making a software product. Most of the researches on elicitation focus on technologies for improving the accuracy, precision and variety of the requirement details. Information gathered during requirements elicitation often has to be interpreted, analysed, modelled and validated before the requirements engineer can feel confident that a complete enough set of requirements of a system have been collected. Therefore, requirements elicitation is closely related to other RE activities.

5. Classification of Elicitation problems

McDermid [56] gives a classification of elicitation problems:

- *Problems of scope* The boundary of the system is ill-defined, so that unnecessary design information may be given, or necessary design information left out.
- *Problems of understanding* Users have incomplete understanding of their needs, analysts have poor knowledge of the problem domain, user and analyst speak different languages, "obvious" information may be omitted, different stakeholders may have conflicting needs or perceptions of their needs, requirements are often vaguely expressed.
- *Problems of volatility* Requirements evolve over time, either because of changing needs or because of changing perceptions by the stakeholders.

6. Starting as a new field of research

The Requirement elicitation studies began during the 90's. By the early 90's RE became as a field at its own right. RE professionals had two series of international meetings – the IEEE conference and symposium and the establishment of an international journal published by the 'Springers'. The field has become large enough to support small meetings and workshops in various countries by the late 90's. The basic questions that have been addressed over the years are:

1) *What aspects to model in the 'why-what-how' range*- determines the ontology of conceptual units in terms of which models will be built - e.g., data, operations, events, goals, agents, and so forth.
2) *How to model such aspects*- determines the structuring relationships in terms of which such units will be composed and linked together - e.g., input/output, trigger, generalization, refinement, responsibility assignment, and so forth.
3) *How to define the model precisely*- determines the informal, semiformal, or formal specification technique used to define the required properties of model components precisely.
4) *How to reason about the model*- determines the kind of reasoning technique available for the purpose of elicitation, specification, and analysis.

The seminal paper by Ross and Schoman opened the field. The companion paper introduced SADT as a specific modeling technique. SADT was a semiformal technique in that it could only support the formalization of the declaration part of the system under consideration - that is, what data and operations are to be found and how they relate to each other; the requirements on the data/operations themselves had to be asserted in natural language. The semi-formal language, however, was graphical. Shortly after, Bubenko [62] introduced a modeling technique for capturing entities and events. RML brought the SADT line of research significantly further by introducing rich structuring mechanisms such as generalization, aggregation and classification [44]. It was the first requirements modeling language to have a formal semantics, defined in terms of mappings to first-order predicate logic.

7. **Elicitation techniques**

The Requirement Engineers can choose one or a combination of the requirement techniques to gather all the requirements for a software system. The driving factors behind choosing one are – time, resources available, what kind of information needs to be elicitated, etc.

7.1 Traditional techniques include a broad class of generic data gathering techniques such as questionnaires and surveys, interviews, and analysis of existing documentation (Organisational charts, process models or standards, and user or other manuals of existing systems).

7.2 Group elicitation techniques include brainstorming and focus groups, as well as RAD/JAD workshops (using consensus-building workshops with an unbiased facilitator) [1]

7.3 Cognitive techniques include a series of techniques originally developed for knowledge acquisition for knowledge-based systems [2]. This includes:

7.3.1 *Protocol analysis* - an expert thinks aloud while performing a task, to provide the observer with insights into the cognitive processes used to perform the task)

7.3.2 *Laddering* - using probes to elicit structure and content of stakeholder knowledge

7.3.3 *Card sorting* - asking stakeholders to sort cards in groups, each of which has name of some domain entity

7.3.4 *Repertory grids* - constructing an attribute matrix for entities; by asking stakeholders for attributes applicable to entities and values for cells in each entity).

7.4 Model-driven techniques provide a specific model of the type of information to be gathered, and use this model to drive the elicitation process. These include goal-based methods, such as KAOS [3] and I* [4], and scenario-based methods such as CREWS [5].

7.5 Prototyping can be used where there is a great deal of uncertainty about the requirements, or where early feedback from stakeholders is needed [6]. Prototyping can also be readily combined

with other techniques, for instance by using a prototype to provoke discussion in a group elicitation technique, or as the basis for a questionnaire or think-aloud technique.

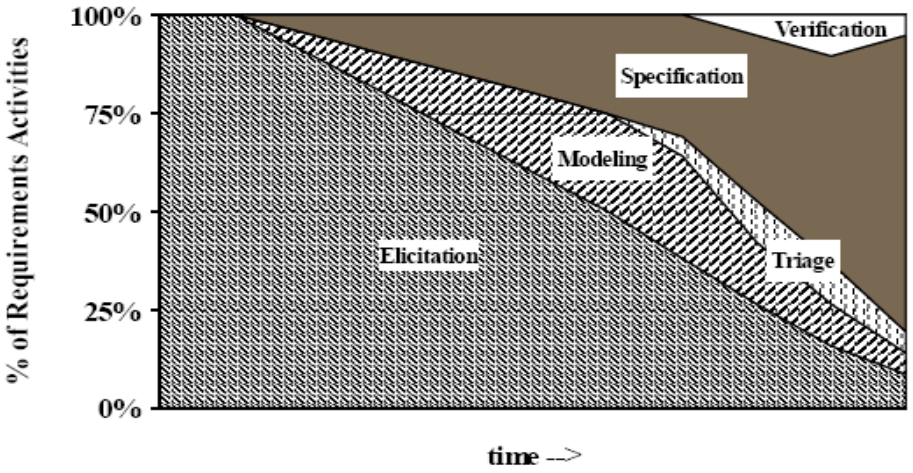

Figure 3: Percentage of requirement activities in respect of time

8. **Requirements elicitation technique selection**

Although some analysts think that just one methodology or just one technique is applicable to all situations, one methodology or technique cannot possibly be sufficient for all conditions [7, 8, 9, 10, 11, 12].

Analysts select a particular elicitation technique for any combination of four reasons:
(1) It is the only technique that the analyst knows
(2) It is the analyst's favourite technique for all situations
(3) The analyst is following some explicit methodology, and that methodology prescribes a particular technique at the current time
(4) The analyst understands intuitively that the technique is effective in the current circumstance.

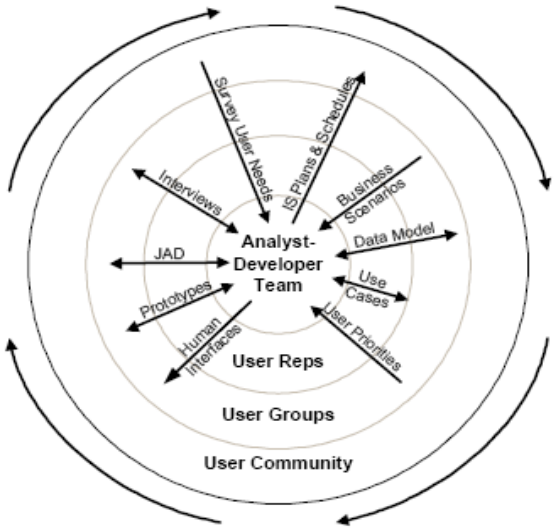

Figure 4: Different types of elicitation techniques and user categories

Unfortunately, most practicing analysts do not have the insight necessary to make such an intelligent decision; they usually rely on one of the first three reasons.

9. Representative Approaches

9.1 Use Cases

A use case is a technique used in software and systems engineering to capture the functional requirements of a system. Use cases describe the interaction between a primary actor —the initiator of the interaction—and the system itself, represented as a sequence of simple steps. Actors are something or someone which exist outside the system under study, and who (or which) take part in a sequence of activities in a dialogue with the system, to achieve some goal: they may be end users, other systems, or hardware devices. Each use case is a complete series of events, from the point of view of the actor [61].

According to Peter Hantos [66], the term “Use Case” in the software arena is applied in two different ways. According to the strict interpretation, use case is the name of an object oriented analysis and design methodology, facilitating the creation of a full object model from the use cases. The second interpretation of the term is more general, and it is not tied to any particular analysis or design methodology. Use cases or scenarios are used simply to describe the system operation in external terms only, without getting into any details regarding the internal processes of the system.

According to Bittner and Spence, "Use cases, stated simply, allow description of sequences of events that, taken together, lead to a system doing something useful"[53]

9.2 Interviews

Interviewing is a method for discovering facts and opinions held by potential users and other stakeholders of the system under development. Mistakes and misunderstandings can be identified and cleared up. There are two different types of interviews:

• *Closed interview* where the requirements engineer has a pre-defined set of questions and is looking for answers
• *Open interview* without any pre-defined questions, the requirements engineer and stakeholders discuss in an open-ended way what they expect from a system.

There are three types of one-on-one interview:

- *Unstructured* – the interviewer will begin with talking points but will allow the participants to go into each point with as much or little detail as he/she desires. The questions or topics are open ended, so the interviewee has the independence to choose his/her own way to give the response.
- *Structured* – It is the most controlled type and similar to a verbal survey. The interview consists of primarily of closed-ended questions and the interviewee must choose from the option provided. Open-ended questions may be asked but the interviewer will not ask the questions which are not on the script.
- *Semi-structured* – It is the combination of structured and unstructured types. The interview may begin with a set of questions to answer but goes beyond that list at times.

9.3 Surveys

It can be an extremely effective way to gather information about the users. The problem is that a reliable survey can be very difficult to design and more importantly, respondent rate can be low. According to experienced survey researchers, response rate estimation is somewhere between 20% to 60%. There are some ways to boost up the response rate:

- Reduce the number of open ended questions – they can make the survey longer to complete and decrease the return rate, responses can be difficult to comprehend as the respondents use their own terms and phrases.
- Make the questionnaire easy to comprehend.
- The survey should come with return postage paid and self-addressed envelop.
- Follow up with gentle reminder.
- Include a personalized cover letter and instructions to fill the form.
- Make sure the standard information is included (contact information, approximate time to complete, purpose, etc)
- Question wording should be clear and unbiased.

Using the surveys has the benefit of being anonymous and it is easier to ask questions that could be odd to ask face to face (questions like age or salary-range, etc). It's better to avoid 'required' fields in the survey question set. The respondents may then provide inaccurate information just to move forward with the survey.

9.4 Focus Groups

In a focus group, five to ten individuals are brought together to discuss their experiences or opinions around topics introduced by a moderator. The session typically lasts one to two hours and is good for having quick understanding of user perception about a topic or concept. They often bring out spontaneous reactions and ideas. Since there is often major difference between what people say and what they do, observations should complement focus groups. They can support the articulation of visions, design proposals and a product concept. Additionally, they help users in analyzing things that should be changed, and support the development of a 'shared meaning' of the system [64].

Focus groups have the advantage of allowing more natural interactions between people than questionnaire interviews, or even open ended interviews.

9. 5 Joint Application Development (JAD)

This is done by a series of sessions where users and developers work closely to capture the requirements. It has recently become popular in Requirement Engineering, especially for Information Systems applications, because of their claim to greatly accelerate the development of requirements [55]. In particular, participants will certainly be unable to pose tacit knowledge. Also, even though group facilitators try to avoid imposing their own categories on participants, there is no guarantee that the participants will in fact share categories with each other. Moreover, because participants may have widely different status within the organisation, there is a danger that some will not feel free to say what they really think, especially if it is unpopular. Finally, it will often be difficult for non-technical participants to assess the significance of technical decisions.

This method is closely related to focus groups, and can be expected to suffer from some of the same problems. In particular, participants will certainly be unable to articulate tacit knowledge. Although this method appears promising, we believe its potential limitations should be studied empirically.

9.6 Field Studies

A field study means several visits to the user's location and can be conducted in any environment in which a user lives or works. This type of studies can last for a couple of hours to a full day or even weeks depending on the goals and resources of the study.

The main advantage of this method is – the requirement engineer has a chance to observe the users completing the tasks in their own environment. He can observe the task flows, inefficiencies and challenges directly. The information can then be used to understand the user requirements for the software. It is more advantageous to observe user behaviour than to ask the user to describe how s/he works because of issues with memory and social desirability. Moreover, users often know more than they can describe at a single or multiple interviews.

9.7 Rapid Prototyping

Rapid Prototyping is an iterative process, because the developers go through all the conventional software product engineering steps while creating a prototype. It has also incremental process characteristics, because the new prototypes are not only representing enhanced quality, but enhanced functionality as well. While prototyping is a critical part of the hardware development process, the described Rapid Prototyping activity to create actual products is only feasible in the software environment [66]

9.8 Introspection

"Introspection is the first and most obvious method for trying to understand what properties a system should have in order to succeed. It amounts to imagining what kind of a system I would want if I were doing this job, using this equipment, etc" [63]

The problem with this method is – it may be done by a person who doesn't have experience of actual user and thus can state the properties in a wrong way.

9.9 Participant Observation

The observer attempts to develop a legitimate role within the community of interest and becomes a part of the community to capture the requirements.

9.10 User-Led Requirements Construction (ULRC)

ULRC is essentially a social process that addresses a major problem in the requirements elicitation process: user–developer culture gap [65]. This can be overcome by training users to build the requirements models themselves. The approach comprises three rounds.

Round one consists of user training, where the developer transfers knowledge and skills to the user for the building of the requirements model.

Round two consists of constructing a model of the current domain, used as the basis for constructing a model of the future domain.

Round three is an iterative process to improve the quality of requirements.

9.11 Soft Systems Methodology (SSM)

As a methodology, SSM offers a set of guidelines, which can be applied to 'messy, changing, ill-defined situations' [51]. The core idea is that people work through seven phases of the methodology in order to analyse complex systems to plan and determine appropriate changes.

10 Methodology-based techniques

10.1 Requirement Elicitation using QFD

Quality Function Deployment (QFD) is defined by Soto as "a systematic process for motivating a business to focus on its customers" [14]. It is based on market research: understanding customers' needs and desires, and the effectiveness of relevant products in meeting those needs and desires. In QFD, cross-functional teams identify and resolve the issues involved in developing products, etc., to satisfy their customers.

Why QFD?

Once a team has identified the customers' wishes, QFD is used for two basic purposes [15]:

• To improve the communication of customer needs throughout the organization.

• To improve the completeness of specifications and to make them traceable directly to customer wants and needs.

QFD requires that representatives of the different organizations involved in producing the product be involved in its definition. Consequently, these representatives discuss the meaning of the customers' wishes and work together to ensure that they come to a common understanding. Communications throughout the organization are greatly improved. This process will also uncover many issues whose resolution will lead to a more complete specification.

The "Whats" Room: This room contains the requirements, as identified by the QFD team. "Typically there are many customer requirements, but using a technique called affinity diagramming, the team distils these many requirements into the 20 or 30 most important needs." In affinity diagramming, the team discusses the initial requirements provided by the users, and clusters them into a smaller number of more general requirements.

The Importance Ratings and Customer Competitive Assessment Rooms: When QFD is to be used, market research has to be designed around the expectation that the QFD team will use it. Market research provides information about the varying priority of expressed customer needs (the Importance Ratings room) and about the strengths and weaknesses of both the client's and competitors' existing products. Note that the Importance Ratings room is associated with the "Whats" room, while the Customer Competitive Assessment room feeds separately into the HOQ's relationship matrix.

The "Hows" Room: The "Hows" room requires completion of the "Whats" room. In the "Hows" phase, the team develops metrics for success in the "Whats" previously identified. Each "What" requires at least one "How", and some "Whats" may require more than one.

The Relationships Matrix Room: After completion of the "Whats", "Hows", and Customer Competitive Assessment rooms, it is possible to start building the Relationships Matrix. The team attempts to define the relationship of every what to every How. The relationship may be strong, medium or weak, or there may be no relationship at all. In any event, the matrix must be completed.

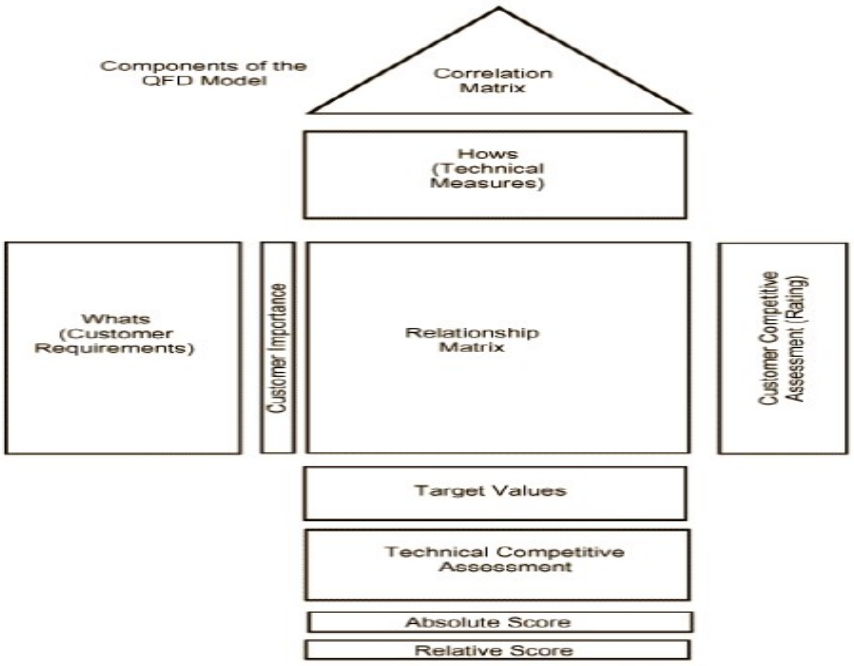

Figure 5: Quality Function Deployment

10.2 Key characteristics

It was introduced in the 80's by some large US companies. Lee and Thornton [67] considered limitations of the method. Product key characteristic is the most important type of this technique. In functional terms, the product key characteristics for a particular product are those that are highly constrained. The effects of product key characteristics can be categorized. A

method is proposed for identifying high-risk characteristics which add a significant risk to successful product delivery, because of sensitivity in terms of cost, reliability and the production schedule.

11. Risk Analysis Using CMM

The Capability Maturity Model (CMM) is divided into five maturity levels [69]:
(1) *Initial* - decision support process for managing risk is characterized as ad hoc, and occasionally even chaotic. Few processes are defined and success depends on the individual's abilities and experience. Remove an individual and the processes may change dramatically commiserate with the next individual's level of ability & experience.
(2) *Repeatable*- basic management processes are established to document the management of the organization. The necessary process discipline is in place to repeat earlier successes on similar tasks, based on previous experience of the organization.
(3) *Defined*- process for standardizing, documenting, integrating risk management into the normal decision-making processes of the organization. All decisions use the approved detailed version of the organization's standard risk management process for decision-making.
(4) *Managed*- detailed measures of management decisions made, the formal process of managing risk and the quality of the risk management (planning, including setting context, risk identification, assessment of risk, evaluation of risk, mitigation of risk to an acceptable level, the monitoring of risk and review of the whole process). All the business processes and the output products or services are quantitatively understood and controlled.
(5) Optimizing- continuous process improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies to measurement of the public affected by the decisions.

12. Choosing among interview, survey and focus group

One to one interviews can take significant time to conduct and more resources than a survey. However, there may be times when there is a need to collect large samples of data to feel more confident that the results obtained can be applied to the user population as a whole. If we are looking for statistically significant results that can be generalized to the population as a whole, surveys are a better choice.

Individual interview is a better choice than focus group if we are looking for detailed information that can't be obtained from group discussions or surveys. There can be a follow up of responses and clarification which is not possible in a survey. One more important point is – participants can't influence each other's responses which is a major problem with focus groups. In a focus group, there is room to ask a limited number of questions to the participants and there is a limitation of the depth of discussion with each participant because the moderator needs to hear from each participant equally.

Focus group is a better choice if we are looking for multiple points of view in a short period. They can be used at the middle or at the end of the development cycle. They can also be a great activity to help gather initial requirements. To be more specific, a focus group can help to understand surprising or contradictory results obtained via surveys or identify the reasons why user satisfaction level is poor.

13. State of the art of Requirement Elicitation research

This section provides a foundation of the different research areas of requirement engineering from which to explore future research directions. We have constructed a matrix

which shows the classifications of the RE techniques and which research papers try to address that particular issue.

Requirement issues	Techniques and tools	Strategies/Methodologies	Notations
Elicitation	Invariant generation [16] Simulation [20] Animation [22,26,30]	Inventing requirements [17] Identifying stakeholders [33] Metaphors [21,24] Contextual requirements [25,34]	Non-functional requirements [18,23] Agents [19,36] Policies [40] Goals [27,29,32] Scenarios [31,38,43]
Requirements Management	Stability analysis [45] Traceability [28,35,39,49]	Scenario management [48]	Variability modeling [37,46]
Requirement Analysis	Requirement selection [42,47] Setting priority [58] Conflict analysis [52,60] Checklists [54] Linguistic ambiguity [50,59]	Conflict management [41] Negotiation [57]	

14. Evolving Requirements

Successful software systems always evolve as the environment in which these systems operate changes and stakeholder requirements change. Therefore *managing change* is a fundamental activity in RE [70]. The RE team has to manage the changes of the requirement documentation. It requires the techniques of configuration management, traceability and version control [68]. Most common requirement changes are adding or deleting the requirements based on the stakeholders needs. It can also be triggered just because they missed the requirements initially.

Managing requirements involves making new set of requirement documents, recognizing change through ongoing requirement elicitation, evaluation of the risk and systems in their operational environment. It has become important to develop system product families so that the softwares grouped together in the same family share the similar requirements and architectures but only differ in key requirements.

REQUIREMENT MODELING

Section 1

1.1 What is Requirement Modeling?

- Requirement modeling is the part of Requirement Engineering. It is one of the most important parts of software development life cycle. It is the process of describing functional requirements of the system and developing useful and proven models.
- It is formal description of all the business activities, information required to support those activities, for the purpose of development of those activities.
- The process of identifying the requirements and modeling business processes; mapping requirements to abstract objects.
- It can be the assurance to requirements specifications, requirement validation and requirement management.
- It is used to describe the business activities and the information required to support those activities for developing systems.
- Modeling is an analysis activity during which an initial graphical description of the relationship between system entities is proposed [20].
- It is the process of specifying system behaviour , responsibilities , states , modes ,timings and performance .[36]

1.2 Benefits from the Requirement Modeling:

- It improves the quality of the software requirement specifications that will in turn increase the quality of developed software.
- It helps to reduce the time required to develop the large and complex software as it will help the developers to easily develop the software.
- It promotes and re-engineer of systems by providing the engineers clear specifications of each and every requirement the system offers.
- Enhance the developer's productivity.
- Helps to reduction of software faults early in the software development life cycle.
- Helps in reducing time wastage in implementing unused and unimplemented features.

1.3 Principles of Requirement Modeling [28]:

- Business requirements should be expressed into the form of fully integrated data and function models.
- Requirements should only be model that has a specific purpose.
- Stakeholder's participation is very important.
- Modeling should be simple and easy to understand by all.
- As requirements evolve over time so modeling should be able take into account requirements change with time.
- Textual documentation should also be integrated with formal models appropriately.
- Requirements model should be made at enterprise planning level.
- Business requirements are subject to change. All the data models and function models should be made at abstract and generalization level.
- The validity of the information is changing always so the data models should allow for those changes in future at any time.
- Information regarding process measurement and decision making should be expressed in the data model.
- The most important one is the quality of Requirement modeling.

1.4 Tools used in the requirement modeling are:

- Requirement management tools, formal model checkers, software auto coding tools, programmable logic devices tools, cost modeling tools etc.
- Oracle designer
- Diagramming tools (Visio, power point)
- Text documents
- Central repository

1.5 Techniques used for Requirement Modeling:

- Process Modeling
- Data flow diagramming
- Entity-relationship modeling
- State Transition modeling
- Function-hierarchy modeling
- Business rule modeling

Dealing with requirements is always challenging task as complex and big projects involve large number of development teams and sometimes it all become confusing. Moreover requirements involve hierarchies and various relationships among each other. Requirements errors are very expensive [38] and if they can not be detected earlier in life cycle, the project quality degrades very much, it will be very late and unable to meet the customer's expectations [37]. So in order to overcome these problems requirements should be elicitated early and modeled by prioritization of the requirements so that designers can bound the projects[38].

1.6 Use of Requirement Models:

Requirement modeling technology brings requirement models that help to check the consistency, completeness, and redundancy of the requirements.

- Models are used for representing and organization of business requirements.
- They possess reliable contents and structures, which are helpful in guarantying clarity, carefulness and strength in the work products.
- They are used for representation of diagrams and support the synoptic view of the contents of model, for communication among people at different levels.
- They can be used by different types of software tools to perform functions such as- model validation, or code generation.
- These models can be used to formalize the textual requirements in business requirements documents. These can be:
- Data models that consist of entities, attributes, domains and relationships.
- Function models: includes functions and rules.

Section 2

2.1 Object – oriented and Process-oriented methodologies in Requirement modeling:

- Process – oriented methodologies include the data flow diagrams representing data flows and processes, a data dictionary and detail process specifications.

- Object-oriented method uses the concept of an object, An object is the collection of data (attributes) and processes (services) that manipulate the data. An object implements the key idea in system development that of encapsulation and information hiding. Services are invoked via message passing between them.
- Differences between them :

Let me explain this concept by using an example: for example we have to build the air traffic management system. for building that system we need to know all the data and processes that has to be taken into account, for example data might be (pilot attributes) and processes might be departure , arrival etc. So to make any system data and processes are the most important things to consider that will make a complete target model of the application. In process-oriented methodologies, system is represented as series of processes that transforms data. Processes are linked via data flows. Whereas object-oriented methodologies use the concept of objects and process that manipulate the data.

2.2 Scenario-Based Requirement Modeling

I am going to discuss here scenario – based Requirement Modeling:
Scenarios represent ways to use a system and to achieve different tasks. They are single and complete sequence of steps describing an interaction between actors and a system. These are mostly used in requirement engineering, as they are easily understood by all the stakeholders. These helps to describe the system functionality, discover design alternatives and explain system weaknesses. The scope of scenarios can vary from system-internal to organization and to ordinary cases. Scenario can be represented formally or informally. Scenario manipulation represents the scenario development and existence over time.
As we all know about Aspect-oriented software development (AOSD) used for representing cross cutting concerns in software development [2] interesting part is that scenarios help in representing aspects .It describes how to compose aspectual scenarios as well as non-aspectual. Non -aspectual scenarios are modeled as UML sequence diagrams and aspectual are modeled as Interaction Pattern Specifications (IPS's). Finite state machines are modeled as UML state machines and aspectual finite state machines will be modeled as State machine pattern specification (SMPS's) [2].
As I discussed above the four views in scenarios [3]: Purpose, contents, form, lifecycle. I tried to explain them here:

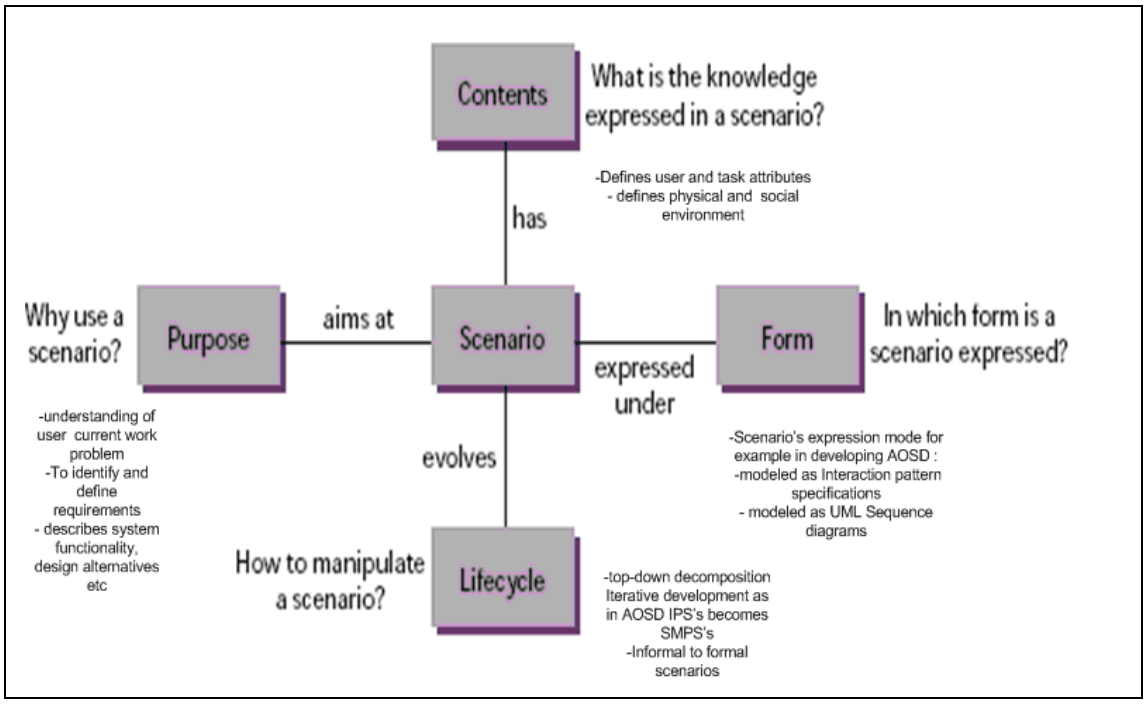

Figure 1 (Four views on scenarios)

2.3 Use of Scenarios in Requirement Engineering

- Scenario based requirement Engineering is used for interdisciplinary development with constant communication between developers and customers [3]. In order to develop scenarios the knowledge of full domain is required. Mostly customers and users involved in the projects want to talk more about concrete scenarios rather then abstract scenarios.
- Projects have different stakeholders and they have different goals so reaching the overall agreements between the stakeholders is difficult task. Scenarios reduce this complexity and help to reach on the agreements between the projects, they also helps in discussing design alternatives in the projects.
- Helps in Aspect-oriented software development in representing cross-cutting concerns [2].
- Scenarios role in usability engineering: helps in completely understand user's task. Reduce the risk of failure of software projects by focusing early on validity of requirements to build usable systems. Helps to bridge the gap between usability engineering and software engineering by helping in validation of requirements, by context analysis and providing reasonable study of design proposals [1].
- Helps to detect the state changes that has been effected on the execution of use cases and also helps to identify the pre- states and post-states obtained by use case executions [30].

2.4 Integrating expressiveness of Requirement modeling

Integrating the various approaches to Requirement Engineering nowdays is very essential to combine the knowledge and various techniques that are already developed before. I am going to discuss here three ways in which several views of the product can be modeled.

1. Modeling multiple views with the help of conceptual graphs, OMT object diagrams, data flow diagrams.
2. Integrative Modeling used to combine several views of the product.
3. Integrating expressiveness of Requirement Modeling approaches.

The idea behind these three of the approaches are almost same they all combine several views of the product with the help of different techniques.

In integrating expressiveness of requirement modeling approaches, five approaches are combined: standard IEEE 830-1998, Use-case approach, Goal oriented approach, Aspect-oriented approach, variability management approach. So all these approaches are combined to form the Meta model of core concepts and then these concepts can be extended to form the goal model that represents goals, requirements, and operationalization by inheriting from artifacts. It also shows dependencies between variants [5].

In integrating requirements modeling approach: requirement model integrates between informal textual descriptions and object – oriented visual executable models. In Requirement Modeling system that consists of User Interface (UI) that allows the user to create, edit, update, and view versions of requirements. The requirements viewer (RV) used by UI to view versions, requirement editor is used by UI to edit an update, Scenario Viewer / Editor (SVE) is used by RV or RE to view or edit scenarios. Requirement Database (RDB) is used by UI to create updates [4]. In Multiple viewed requirements modeling helps to form conceptual graphs with the help of object diagrams and data flow diagrams. [6]

2.5 Comparison between these approaches:

Modeling multiple views with the help of conceptual graphs is very easy to understand and it offers multiple viewed development environment with integrating and validating each view [6]. Where as integrating various requirements modeling approaches is sometimes hard because it's hard to compromise between their semantics [5].

An integrative modeling approach helps the user to connect and integrate elements of different views [4]. Where as multiple viewed requirement modeling conceptual feedback is a valuable feature [6]. Both these approaches easily support natural language interactions and both do not impose much structures and methodologies. Where as combining integrating expressiveness of requirement modeling approaches creates problems when defining highly expressive models [5].

The most important feature of (3) is that it helps to include more number of dependencies and refinements in artifacts as compared to other requirement management tools.
The strength of (1) is that they conceptual graphs can be analysed for inconsistencies and incompleteness. The main feature of (2) is the scenarios links provided.

All of these approaches serve various distinct features and they motivate these motivate industrial development teams to adopt these new methodologies.

2.6 Object Oriented Modeling Method for Requirement Modeling:

As we all know that today object-oriented methods have gain lots of popularity in developing software systems, these techniques are used for requirement modeling and use for modeling interactive, behavioural and declarative aspects of the software system. The basic approach in this modeling method is to capture the real world entities called objects and defining the relations between them, forming the classes, methods, messages, and inheritance etc.
I am going to discuss here four research papers on Object-Oriented Modeling approach:

Object-Oriented Requirement analysis, object-oriented Requirement Modeling Based on UML, NDHORM: An OO approach to requirement modeling, The use of Object-oriented models in Requirement Engineering: A field study.
1) An OO based requirement modeling based on UML is the method of forming the OOA requirement model using the UML diagrams notations. The OOA model considers requirements using three aspects: object model, dynamic model and function model. It consist of these steps: From the analysis of problem domain form the initial OO model for the system that forms use case diagrams and sequence diagrams, next the initial model is refined by making class diagram and defining attribute and services of the class. And a super class is formed by finding the common characteristics of the classes, and then final diagrams are made and prototyping and implementation goes on. [7].
2) NDHORM: an OO approach to requirement modeling is also a method of developing the requirement model that aims at finding the objects of problem domain. It consist of three steps: forming ORM (Object-Relationship Model) , Class-Relationship Model(CRM) and State – Transition Model(STM). [9]
3) Object-Oriented analysis Requirement Analysis technique discusses about the Goal-oriented and object-oriented analysis. [10]
4) The field –study discusses about use of Object-Oriented models in requirement requirements engineering and discusses the conceptual model in requirement modeling [8].

2.7 Comparison between these approaches:

All of these approaches are discussing OO techniques for requirement modeling with the use of different ways. In the conceptual model discusses an Object-oriented modeling procedure that leads to create formal and informal models, informal means making picture diagrams etc and formal means making state transition diagrams [8], the same concept is used in modeling based on UML that also forms the use case diagrams as well as class diagrams but it doesn't take into account the state transition diagrams that are also part of UML [9]. And theses diagrams are necessary for the analysts to do good analysis of the system and for also easy to identify the state and transitions.
One of the interesting concepts I found in OO requirement analysis paper is the involvement of goal-oriented approach to cover the non-functional requirements of the system but neither of the papers on OO method covered this issue [10].

NDHORM model is multiview model that includes the views with respect to user, and analyst point of view, which is the major benefit of using it [9]. Conceptual model discusses about the mapping between informal and formal models which is good way of modeling the requirements from user as well as analysis's point of views. Also the concept of Mental modeling that is "modeling in the mind" during the elicitation process [8]. OO requirement analysis gives a good discussion on goal – oriented approach to deal with Non-functional requirements and functional requirements.

2.8 Case tools used for Requirement Modeling:

I am going to discuss here case tools used for Requirement Modeling:
STROM: Software tool for the Organization of Requirement Modeling. A strong software tool able to streamline the requirements specification phase of software engineering process [14].
ST Tool: Secure Troops methodology: A case tool for modeling and analyzing requirements [13].
SCR: Software Cost reduction requirement method. A case tool for developing formal requirement specifications in SCR tabular notations [11].

AUTO-RAID: A tool that supports integrated requirement model, and operations for converting textual requirements to design model [12].

All theses tools are used for requirement modeling purposes and used for developing requirement specifications with the use of different techniques.
Comparison between four of these tools:
ST-Tool is a case tool for designing and verification of functional requirements, as well as support for verifying officially the inconsistencies and correctness with the help of different model-checkers [13]. The same property of checking inconsistencies and correctness is given in ST-Tool and SCR that has tools called consistency checker and verifier. Whereas the STROM does not support these functionalities which doesn't help the requirement analyst to check for these properties, but compared to other tools the STROM has a very big advantage of providing support for traceability matrix for generating use case diagrams from scenarios and also it tells how important is the this use case for the system [14] .One of the main advantages of using Auto-RAID is that it supports hierarchical structuring of requirements specifications and it all the requirements are stated category wise like architectural requirements, data requirements can be categorized differently and helps the requirements analyst to do tasks effectively [12]. whereas in ST-Tool , it provides the user interface for designing specifications where user has to manage the all the components and functionalities of the tool and there is no way of managing the graphical objects in some structured way , because everything is messed up after the all the objects are managed[13]. One of main points I found in the STROM is that it deals with text aspects of use case modeling which is very important for understanding [14]. As we can some of other tools no other tool is using text aspects. Properly designed use cases can help to assist the design decisions after the specification phase. SCR tool is use to represent formal specifications that are expressed in tabular notations. As today formal specifications are becoming more important as there use can lead to reliability and robustness of the design of the system. This tool also supports for consistency checker, verifier, and a simulator for symbolically executing the specifications. But these formal models are used only for high assurance system for example in military and airline systems etc. but for simple system there is no need to apply these methods [11].

2.9 Use Case Requirement Modeling:

I am going to discuss Use case modeling that is the requirement modeling with the help of UML (use case diagrams). All the papers discuss about the use case modeling and every paper is discussing different other methods also.
One of the paper discusses about the non-functional requirements as we all know that use case diagrams usually discuss about functional requirements and they do not include non-functional requirements in them [16], so in this use case diagrams are extended to include the non-functional requirements as well for example: time and concurrency, fault –tolerance, security etc. So this paper deals with good concept of including non-functional requirements of the system. Two papers are discussing about business modeling. One of the papers says that object-oriented techniques are not good for business modeling as; it does not facilitate top-down view and doesn't provide enough information for modeling business processes [16]. Also business people do not understand properly the there processes so it becomes difficult in applying the use cases in this way. Also it doesn't provide way to capture the business needs –mission, goals etc of the business. So they take another approach called process diagrams to do business modeling and for business people also this method is very easy as they can easily caught on to process mapping techniques. As were as business modeling in other paper concerned it deals with Role – diagram, sequence diagrams and process diagram (same as the previous one I discussed).[15]As shown in the figure below :

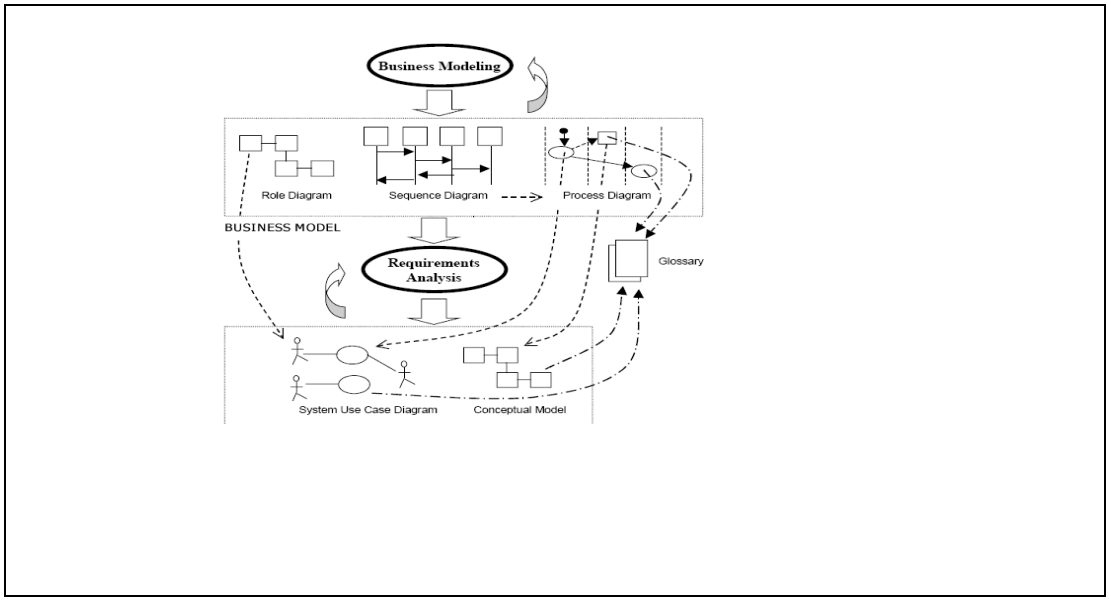

Figure 2: Relationship between Business and Requirement models

The Role diagrams are used to identify the roles and links between them as well shows the characteristics between the roles. Also it shows the sequence diagram that shows the sequence of actions about the system tasks. Finally the process diagrams, it is similar to activity diagram, and all the business processes are hierarchically organised in these diagrams. It also shows some business rules and constraints also on the business rules [15]. One of the paper discuss about business rule diagrams that identifies complex structural constrains that can not be expressed by class diagrams. Two types of business rule diagrams can be formed one called Always and other called Never. Always will contain all the properties that always hold And Never can properties that should not be occur[30].
The formation of conceptual Model from the use case diagram is a very good point of Requirement Modeling, as in this case the after the creation of use case diagram, the initial conceptual model is formed which is called the Class diagram. Each object will become a concept [15]. The good explanation of conceptual modeling is given in one of the papers, it clearly explains about the requirement analysis phase, conceptual model and execution model in any software development system. It describes the translation of requirement model into conceptual model and then to execution model. The conceptual modeling consists of creating the objects models, dynamic models and functional models [25]. Although all the papers above I discussed explains about advantages of use case modeling but sometimes the use case models can lead to failure of software projects for example when :- use case do not provide sufficient details , inadequate input from business representatives , improper selection of system functionality , neglecting some of the business rules and constraints , keeping use case model inconsistent with the domain model and not developing abstract use cases. So requirement engineers have to keep in mind the above points before doing use case modeling[34]

2.10 CRUD (Create, retrieve, update and delete) functionality in Use Cases [15]:

I found this functionality important and interesting as you can create separate use cases and create only one that includes all the CRUD functionality into one use case.

As with the help of CRUD representations the number of use cases can reduce and it improves the use case diagram. In this paper it explains about all the general concepts about use case diagrams that is formation of general use case diagrams and also concepts about aggregation and generalization mostly used notations in use case diagrams.
Example of use of CRUD is explained in CRUD table that shows flow down of use cases into classes. Its showing the various classes for example cashier, Sale line etc. and various use cases for example sale item, Return purchased item etc, and showing the various instances of classes by assigning a set of classes to each use case[25].

Section 3

Here I am going to discuss about various methodologies related to requirement Modeling:

3.1 Object – oriented and Process-oriented methodologies in Requirement modeling[31]:

- Process – oriented methodologies include the data flow diagrams representing data flows and processes, a data dictionary and detail process specifications.
- Object-oriented method uses the concept of an object; An object is the collection of data (attributes) and processes (services) that manipulate the data. An object implements the key idea in system development that of encapsulation and information hiding. Services are invoked via message passing between them.
- Differences between them :

Let me explain this concept by using an example: for example we have to build the air traffic management system. For building that system we need to know all the data and processes that has to be taken into account, for example data might be (pilot attributes) and processes might be departure , arrival etc. So to make any system data and processes are the most important things to consider that will make a complete target model of the application. In process-oriented methodologies, system is represented as series of processes that transforms data. Processes are linked via data flows. Whereas object-oriented methodologies use the concept of objects and process that manipulate the data.

3.2 Requirement Modeling with Rosetta SLDL [23]

Rosetta SLDL (System level design language) is a information technology that supports the multi-technology, complex, multi-discipline systems**.**
Today, system engineering is facing a lot of problems example of quality, security, cost issues etc. and due to this many of the projects do not finish at all .one of the major reasons for these problems are due to improper requirement engineering. Requirement problems include incomplete, not consistent, misunderstood, ignored requirements.
As the projects grow with size, more number of people will be working on the projects and more problems will arise.
So in order to overcome these problems the need for good requirement specifications and analysis is must. SLDL (System level design language) represents the complete description of requirements that help the developers to easily develop the software's.
Requirement modeling with SLDL helps to represents the system with various views and from different perspectives. Various modeling technologies describes the systems with only few

aspects and do not allow to represents each and every components including hardware and software's.

3.3 Connecting requirement specifications with user interaction model:

The main goal of software engineering is to provide the good requirement specifications, as requirements are one of the strongest parts of software engineering, without good requirements good software cannot be developed. And one of other major parts of the software engineering is HCI (human computer interaction) that is user interaction at proper level. So it becomes very important to link the software requirements specifications and user interaction in very accurate manner. So in order to do this Use of CTT in MDA-based development framework Obtaining Functional requirements:
The very first step is obtaining functional requirements with the help of various techniques: for example defining function refinement trees, use case diagrams (as discussed above in section 2) and all the requirements are captured with the help of requirement models used (discussed above).

3.4 Modeling Interaction with CTT task model [35]

CTT (Concurrent task tree model) is projected by paterno and it is used to express the human computer interaction as graphical task decomposition. Tasks in this case are: abstraction, interaction, application and user task) and relationships between them. In this task model: a forest of trees is drawn to show the all the interactions between the user and the system, with root of tree is leaf of Functional Requirement Tree. For more complex tasks the use cases are further broken down into subtasks and this decomposition is continued until the individual data elements that make up the message is communicated to the software system to be build. The final CTT will be contains nested tasks that consist of data entry and process.

3.5 Creating user interface:

We already discussed the requirements specifications part above and modeling interactions with CTT task model. The next step is to describe the design of final user interface.
Use of CTT in MDA-based development framework:
OlivaNova model execution (ONME) is a MDA-based technology that automatically derived the user interface. With the overall design, the final source code of the design is obtained by OMNE that will result into three-layer application, which includes application logic, the database and the user interface. As a result the final windows are obtained by using CTT model and pattern language UI.

Correspondence between CTT and final widgets:
Each data introduction has corresponding widgets, every task has ok button to apply, and every abstract task calls another window to fulfill the entries.

3.6 Fuzzy to Formal Requirement Modeling:

As we all know the importance of Requirements specifications, and the high risk of requirement volatility in developing any software system. So it's important for requirement engineers to formally represent the requirements. In this paper, set of Meta models is defined to represent the requirements formally. And this approach not only describes functional requirements but non-functional requirements also [19].

Functional and Non – Functional Requirements [20]:
Functional Requirements:

- Describes the functionality or services that the system is expected to provide.
- They address the input-output behavior of a system.

Non-Functional Requirements:

- Global constraints on software system functionality.
- Non-functional requirements = system quality attribute.

Enterprise modeling and objective Modeling:
Consisting of Objective models (describes the reason or motivation of activities), an activity and usage model (describes the functions and processes of enterprise), an actor model (types of actors involved in enterprises), conceptual models (describes set of concepts), and information systems requirements models.
Process modeling:
It is as important as system modeling. It is handled in three different ways:
Process meta-model (model consisting of generic concepts which allows to model any RE process as process consisting of various situations, decisions etc).

3.7 Feature – Oriented Requirement Modeling

Feature –Oriented Requirement Modeling is used to capture users high-level expression of desired system behavior in terms of application features, analyze the relationship between features, and organizing the features into feature-oriented requirement specifications. The main sources of getting the features are the users knowledge about the problem domain. Features can be divided into: functional features and non-functional features[22].
Features: A service the system provides to fulfill one or more stakeholders needs. Needs and features together form the software requirements [20].
Example:

Application Domain	Example of Feature
Elevator Control	Manual Control of doors during fire emergency

Features can be organized at various levels [22]:
High-level features (example of business level)
User-level features (example of supporting specific user class)
Functional feature (example of supporting specific functions)

3.8 Requirement modeling for Real Time software development [24]

For Real Time software development, the paper discusses the system perspective from three different views:
Processing Perspective: The basic approach is to identify all the system event-response pairs and assigning processes to each pair. The system is described using data flow diagrams. Requirements are defined by using all the events. The context diagrams are drawn to specify the system interfaces and boundaries. Event –Response diagram are drawn to specify the all the events and responses in the system.

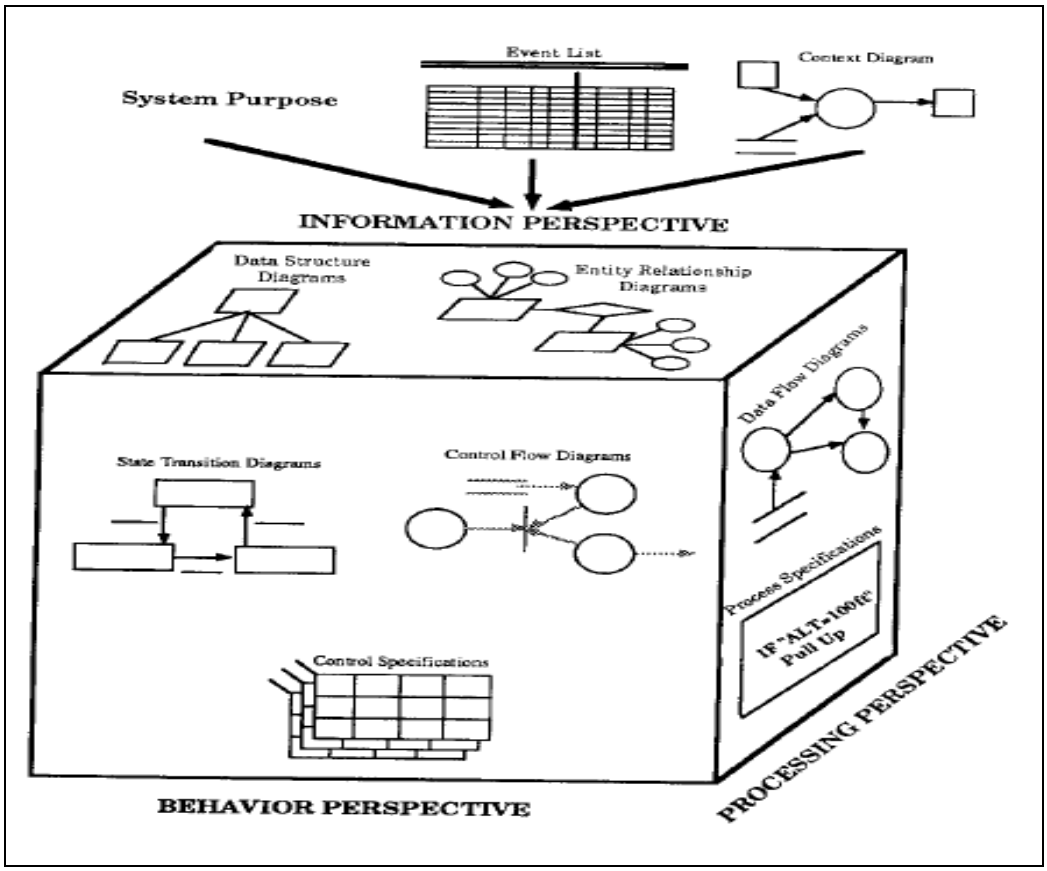

Figure 3 System Perspectives [24]

Information Perspective:
In this perspective the entity relationship diagrams (ERD) and data structure diagrams (DRD) are drawn to identify all the specific system objects.
Behavior Perspective:
The behavior perspective describes the behavior of system using control flow diagrams (CFD) and control specifications (CSPEC) .
All of these perspectives are integrated with each, the processing perspective adds to the development of information and behavior perspective with developing interface definitions and detection processes. The behavior perspective is linked to processing perspective through its process activation facilities.

3.9 Requirement Models:

I am discussing here three models that are used in formal requirement specifications:
On of the model called reference model is based on five main artifacts called WRSPM (domain knowledge (W), Requirements I, specifications (S) , A program that implements the specifications` , a programming platform , (p). This model is independent of the choice of the language for expressing the various artifacts, and all the artifacts are described using formulas of church's higher order logic. [21]

3.10 Relationship between Environment and system:

Ve,s W^M^P=> R

The requirements allow all the events the environment performs and all the events the events system performs. W restricts the actions that environment can perform by restricting e. The requirements R, describe more restrictions, in which all possible actions are desired. P, when evaluated on M, describes the class of possible system events.
Specifications:[21]
Requirements are implemented into two parts: first when requirements are developed, and second when the programming is carried out.
If S(system) property takes W(domain knowledge) into account in saying what is needed to obtain R (Requirement), and P(program) is an implementation of S for M(machine that implements P), then P implements R as desired .

The four variable models [26] also used to describe system behavior, and describe four set of variables – monitored and controlled variables and input and output data items.
Relations NAT, REQ, IN, OUT are used in this model.
REQ, NAT– describes the black box specification of system from monitored to controlled quantities.
IN – defines mapping from monitored quantities to the input data items.
OUT- defines the mapping from the output data items.
-The difference between these two models is that the reference model describes above, provides what is missing in four variable models. By matching corresponding parts of the two models, the correspondence does not make it clear whether IN and OUT should consider the system or environment parts , but the above reference model provides the missing obligations , even if they interpret them as being in the environment itself.
-Both models insist on rigid division between system and environment control of designated terminology.

Formal Requirement model[11]: This model is used to represent requirements for high assurance systems, it shows:
A formal notation, such as SCR notation, is used to specify the requirements.
An automated consistency checker tests the specifications for syntax and type correctness , coverage , determinism and other application – independent properties.
The specification is executed symbolically using a simulator.
At last the requirements are analyzed for application properties.
The difference between Formal requirement model and four variable model is that Four variable model represents naturally continuous quantities, for example pressure and temperature, as continuous variables whereas formal requirement model represents these quantities as discrete variables[11].

- In order to make the requirement model, the analyst and client need to know the domain knowledge of the problem in context and take deep understanding of the domain to form the requirement models, as shown in fig 4 below. They take into account the goal theory also to do analysis of the overlooked or forgotten goals in the problem domain. And if some of the goals are not taken into account they again analyze new goal theory [27].

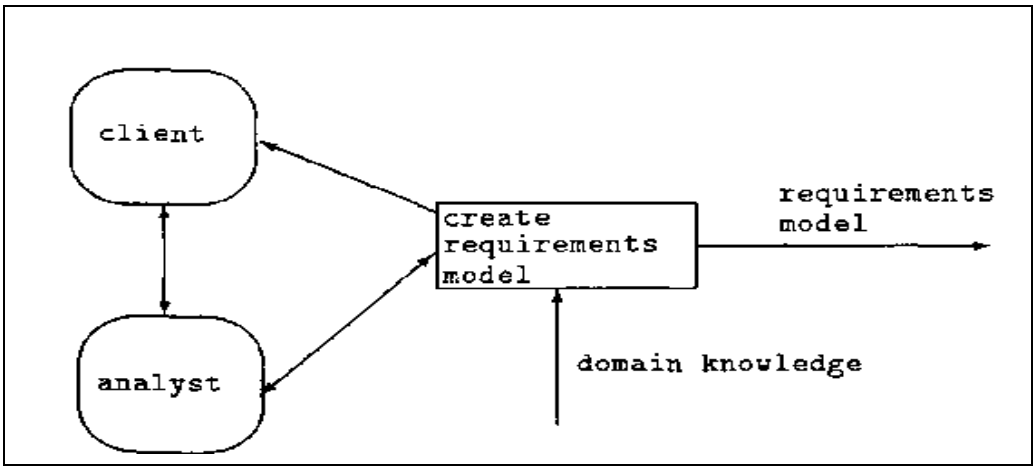

Figure 4 Scenario of use [27]

3.11 Analysis of software system requirement models [29]:

I am going to discuss here survey of seven requirement models:

Goal hierarch icy	State chart	hypertext	Domain networks	Use case hierarchy	logic	Conceptual state machine
-This approach helps to describe the system and environment at the high level of abstraction. -This model is represented by several notations and graphs represent ting goals and their relationships.	-State charts have been developed for use on real time systems, control plants etc. -State charts provides three views on the system, structural, functional, behavioural. -Views are described using: module charts, activity charts, state-charts.	-Hypertext systems are used for managing requirements in the format where there are links between them. -It covers Both functional and non-functional requirements. Three phases in this model:- Requirement documentation, Requirement discussion, Requirement evolution.	-Use of structured domain knowledge. -domain knowledge is modeled as network of nodes and links. - The nodes contain formal specificatio n fragments and links contains relationship between them.	Hierarchical Use case modeling : Environment level : Containing actors , use cases, Structure level : Describes each use case as a episode. Event level : Details the flow of events in each use case episode.	Suitable for knowledge intensive systems for example: air traffic control systems. -Syntax expressed in prolog	- Primarily used for functional requirements of system. - deterministic machine with single start and terminating state.

3.12 Understanding of Quality in Requirement Models [32]:

In order to understand the quality of requirement models, we need to take into account the following:
1) Distinguish between your tasks that is , what you're trying to achieve in modeling and how you can achieve that task.
2) The core of quality here is syntax, semantics, and pragmatic because modeling is usually expressed in language.
3) Communication between participants involved in modeling is also the core quality concept.

3.12.1 The main quality types in the requirements models are:

- Physical Quality: The basic physical quality features here are externalization (information of any social actor has been externalized by use of modeling) and internalize ability (model is constant and available for audience to make sense of it) .
- Syntactic Quality: association between the model and the language in which it is developed.
- Semantic Quality: association between domain and the model. Two goals here are:
- Validity: statements made in the model are correct and relevant. Second: statements are complete and contain relevant information with regard to the domain.
- Pragmatic Quality: association between the model and audience interpretation of it .

3.12.2 Language Quality regarding the Modeling languages used:

- Domain appropriateness: modeling language should be able to capture the domain.
- Comprehensibility appropriateness: easy for participants to understand it.
- Executability appropriateness: extent of formalization to enable executions.
- Knowledge externalizability appropriateness: knowledge of domain expressed in the language .

3.13 State of the art of Requirement Modeling

Requirement modeling is the vast research area that contains a number of areas to be discussed – Requirement Elicitation, Requirement management, Requirement analysis and Requirement Modeling. We have constructed a matrix that shows the classification of important Requirement Modeling Techniques and also the corresponding methods and papers that are concerned with them.

Requirement Modeling Methodologies	Strategies	Tools used
Scenario-Based Requirement Modeling	-Explanation of all the four views.[3] -Scenarios in usability engineering[1] - Scenarios in aspect-oriented software development .[2]	UML[2]

Object –oriented Requirement modeling	-OO requirement model[8] -NDHORM OO method - conceptual model[9] Object –oriented and goal oriented analysis[10]	UML[8],[9]
Case Tools used in requirement modeling	-Method to streamline the requirements specification phase of software engineering process[14]. - Method for modeling and analyzing requirements[13]. - Method for developing formal requirement specifications in SCR tabular notations [11]. - Method that supports integrated requirement model, and operations for converting textual requirements to design model [12].	STROM[14] ST-Tool[13] Auto-Raid[12] SCR[11]
Use Case Requirement modeling	-Use case diagrams[15], [16],[17] -Business modeling[16],[18] - Conceptual modeling [18]	UML[15], [16] , [17], [18]
Integrating expressiveness of requirement modeling methodologies	-Modeling multiple views with the help of conceptual graphs, OMT object diagrams, data flow diagrams[6] -Integrative Modeling used to combine several views of the product[4] -Integrating expressiveness of Requirement Modeling approaches. [5]	UI[4], UML[6]
Requirement Models	-WRSPM[21] -Four variable model[26] -Formal requirement model11] -Goal hierarchy [29] -Hypertext[29] -State chart[29] -Domain networks[29] -Use case hierarchy [29] -logic[29] -conceptual state machine[29]	-

A. CONCLUSIONS

A.1 Elicitation

Much of the research and practice in requirement elicitation for software systems during the past decades may be brought within a unified framework as the development of theories, methodologies, tools and application experience. Different research papers focus on different elicitation techniques and the research on this field has to go a long way to make a collaborative study among all the people involved in the RE activities. Few of the papers and articles deal with the generic terms of software requirement elicitation while most of them focus in a specific software system. As many of the researchers talk about different methodologies and techniques, it becomes hard for the Requirement Engineers to choose from the options and when to apply which method. Although choosing the right method depends largely on the specific software system, its budget and time constraints; a clear guideline should exist to follow for requirement elicitation. If the researchers can't provide such a guideline, software teams will continue to avoid applying rigorous RE techniques which will obviously result in increasing percentage of software project failure in the coming years.

A.2 Modeling

Through this paper I have tried to cover almost all the techniques and methodologies used for "Requirement Modeling." All the research papers discussed about different methodologies of requirement modeling for example object-oriented techniques, scenarios based, use case modeling, requirement models etc. I compared all these approaches to find the advantages and disadvantages of using all of them. Some of the research papers discussed techniques to requirement modeling that I have only explained briefly in the paper to get the basic ideas of how to use them, when to use them and the benefits of using them. At the end I have added the State of art of Requirement Modeling and the future work required in the filed of Requirement Modeling.

B. FUTURE RESEARCH DIRECTIONS

B.1 Elicitation

- More research should be done on how to find an adequate mix of business users and IT professionals to have a quiet interaction for requirement elicitation. Both the groups must have equal input at the design process so that the resulting product satisfies all the stakeholders.
- This paper talks about a number of elicitation techniques. For a specific software project, multiple methodologies can be combined. There should be more research on how to combine two or more methodologies in the same project.
- Most of the research papers described at this article provide a theoretical look at communication process, more real world research is necessary to have a clear idea about user-analyst relationship, how the user and designer interaction take place, etc.
- Requirement elicitation can't work in isolation from the organizational and social context in which the new software system will need to operate. This view encouraged the use of participant observation.
- Another way if RE analysis is to describe the current properties of the environment and what effect should the system will have on the environment. This section of thought requires more attention.

- There is a strong relationship between the evolving requirements and software architecture. As the analysts choose the architecture based on the current requirements, there should be a consideration about the way the requirements can change later. More theoretical research can be done in this respect.
- In many domains of applications, the requirement models can be reused which saves the effort and resources of the Requirement Engineer to elicit the requirements from the very beginning.
- None of these papers emphasizes on the term 'Requirement engineer'. Broadly, it refers to the professionals who do the requirement analysis and modeling. As they are coming from many different backgrounds, there should a unified definition of these professionals and their required set of soft and hard skills to perform their job. Among the soft skills, communication is of critical importance because requirement elicitation solely depends on communicating among the parties.
- There should be more research on developing richer models to capture the non-functional requirements.
- Engineering style research is needed to integrate requirements technologies into a coherent requirement process. Most of the research projects deal with a single RE problem (elicitation, modeling, traceability, etc). Therefore, the state of the art RE research is a collection of the individual research technology with a little focus on how to combine techniques effectively. Further research is due on how to integrate RE technologies so that RE professionals know how to apply individual technologies effectively and synergistically.
- Effective use of the pattern or reusable artifacts are time-saving option for RE. A reusable requirement should be accompanied by standard pattern fields. However this is not enough to facilitate reusable artifacts. More attention from researchers yet to come on how to apply and adapt individual patterns.
- Most of the papers have theoretical aspect of RE techniques. The researchers have to work in partnership with the practitioners to understand the industry-related RE issues.
- A repository of artifacts can be made by both the groups – RE researchers and practitioners. Such a repository will help all the professionals related to RE to share the best practices. It will also help for potential reuse of requirement artifacts.
- Large industrial project data should be available to the researchers so that the can verify the effectiveness of each RE method. As the researchers usually work in isolation with the practitioners, getting this data is of crucial importance.

B.2 Modeling

- A set of case tools are required for requirements specifications easily and do tasks more effectively then people and also free people to do some tasks productively.
- Extension of further tool support for scenarios is required to overcome the problems that occur in scenario management and usage.
- Further studies and experiments are required in Object –oriented methods to fully mature this work.
- To limit task duplications and function overlaps in the projects, conceptual qualified use case diagrams should be employed.

- If users want to port there diagrams from one UML tool to another, this requires capability to import and export to XML, a standard XML based file format for diagrams.
- A design and implementation of requirements specifications automation systems that can automatically transform a graphical object-oriented requirements model to formal specification in an object-oriented language.
- Further work should be done on the more use of formal requirement models that uses formal notations to specify requirements and detect early errors easily. They should also lead to reduce software developments cost. More emphasis should be given because these days formal specifications are getting popularity.
- Use of strong modeling languages and tools are required to that can support modeling and validating requirement models.
- Further development of various product metrics in connection to the use of modeling languages for different modeling tasks.
- Research is required in future whether the applicability of a particular methodology depends on the nature of the problem to be solved. Performance for all the methodologies should be checked on various application domains.
- Further research in the use of conceptual models is required to gain more knowledge of large organizations has been developed and which tools and techniques are useful for specifications for design and implementation of usable systems.
- Extending the use of graphical descriptions techniques like UML for representing the varying and multiple version requirement models.
- Research on integrating various tools into various platforms for automatically verifying the requirement specifications.
- Research is required on various software development policies that would eliminate the ill-formed requirements models.

REFERENCES

References for requirement elicitation

[1] Maiden, N. & Rugg, G. (1996). ACRE: Selecting Methods For Requirements Acquisition. Software Engineering Journal, 11(3): 183-192.

[2] Shaw, M. & Gaines, B. (1996). Requirements Acquisition. Software Engineering Journal, 11(3): 149-165.

[3] Van Lamsweerde, A., Darimont, R. & Letier, E. (1998). Managing conflicts in goal-driven requirements engineering. *IEEE Transactions on Software Engineering*, 24(11): 908-926.

[4] Chung, L., Nixon, B., Yu, E. & Mylopoulos, J. (2000). *Non-Functional Requirements in Software Engineering*. Boston: Kluwer Academic Publishers.

[5] Maiden, N. (1998). CREWS-SAVRE: Scenarios for Acquiring and Validating Requirements. *Automated Software Engineering*, 5(4): 419-446.

[6] Davis, A. (1992). Operational Prototyping: A New Development Approach. Software, 9(5): 70-78.

[7] Kotonya, G., and I. Sommerville, Requirements Engineering, Wiley, 1998.

[8] Macaulay, L., Requirements Engineering, Springer, 1996.

[9] Maiden, N., and G. Rugg, "ACRE: Selecting Methods for Requirements Acquisition," Software Engineering Journal, 11, 5 (May, 1996), pp. 183-192.

[10] Davis, A., and A. Hickey, "Requirements Researchers: Do We Practice What We Preach," Requirements Engineering Journal, 2002.

[11] Glass, R., “Searching for the Holy Grail of Software Engineering,” Communications of the ACM, 45, 5 (May 2002), pp. 15-16.

[12] Yadav, S., et al., "Comparison of Analysis Techniques for Information Requirements Determination," Communications of the ACM, 31, 9 (September 1988).

[13] McDermid, J.A. “Requirements Analysis: Problems and the STARTS Approach”, in IEEE Colloquium on ‘Requirements Capture and Specification for Critical Systems (Digest no. 138), 4/1-4/4. IEEE, November 1989.

[14] Soto, Alejandro and Preeti S. Malik, “QFD”. iSixSigma Dictionary, http://www.isixsigma.com/dictionary/QFD-103.htm, 2003.

[15] International TechneGroup Inc., “The Basics of QFD”, http://www.iti-oh.com/cppd/qfd/qfd_basics.htm, 2004.

[16] R. Jeffords and C. Heitmeyer. Automatic generation of state invariants from requirements specifications. In Proc. of ACM SIGSOFT Found. on Soft. Eng. (FSE), pages 56–69, 1998.

[17] N. Maiden and S. Robertson. Integrating creativity into requirements processes: Experiences with an air traffic management system. In Proc. of the IEEE Int. Req. Eng. Conf. (RE), pages 105–116, 2005.

[18] L. Chung, B. Nixon, E. Yu, and J. Mylopoulos. Nonfunctional Requirements in Software Engineering. Kluwer,1999.

[19] E. Letier and A. van Lamsweerde. Agent-based tactics for goal-oriented requirements elaboration. In Proc. of the IEEE Int. Conf. on Soft. Eng. (ICSE), pages 83–93, 2002.

[20] J. M. Thompson, M. P. E. Heimdahl, and S. P. Miller. Specification-based prototyping for embedded systems. In Proc. of ACM SIGSOFT Found. on Soft. Eng. (FSE), pages 163–179, 1999.

[21] Y. Pisan. Extending requirement specifications using analogy.In Proc. of the IEEE Int. Conf. on Soft. Eng. (ICSE),pages 70–76, 2000.

[22] C. L. Heitmeyer, J. Kirby, B. G. Labaw, and R. Bharadwaj.SCR*: A toolset for specifying and analyzing software requirements.In Prof. of the Int. Conf. on Comp. Aid. Verf. (CAV), pages 526–531, 1998.

[23] T. Gilb. Competitive Engineering: A Handbook for System Engineering, Requirements Engineering, and Software Engineering using Planguage. Butterworth-Heinemann,2005.

[24] C. Potts. Metaphors of intent. In Proc. of the IEEE Int.Req. Eng. Conf. (RE), pages 31–39, 2001.

[25] T. Cohene and S. Easterbrook. Contextual risk analysis for interview design. In Proc. of the IEEE Int. Req. Eng. Conf. (RE), pages 95–104, 2005.

[26] J. Magee, N. Pryce, D. Giannakopoulou, and J. Kramer. Graphical animation of behavior models. In Proc. of the IEEE Int. Conf. on Soft. Eng. (ICSE), pages 499–508, 2000.

[27] P. Bresciani, P. Giorgini, F. Giunchiglia, J. Mylopoulos, and A. Perini. TROPOS: an agent-oriented software development methodology. J. of Auto. Agents and Multi-Agent Sys., 8(3):203–236, 2004.

[28] J. Cleland-Huang, G. Zemont, and W. Lukasik. A heterogeneous solution for improving the return on investment of requirements traceability. In Proc. of the IEEE Int. Req. Eng. Conf. (RE), pages 230–239, 2004.

[29] N. G. Leveson. Intent specifications: An approach to building human-centered specifications. IEEE Trans. on Soft.Eng., 26(1):15–35, 2000.

[30] H. T. Van, A. van Lamsweerde, P. Massonet, and C. Ponsard. Goal-oriented requirements animation. In Proc. Of the IEEE Int. Req. Eng. Conf. (RE), pages 218–228, 2004.

[31] A. Alfonso, V. Braberman, N. Kicillof, and A. Olivero. Visual timed event scenarios. In Proc. of the IEEE Int. Conf. on Soft. Eng. (ICSE), pages 168–177, 2004.

[32] A. van Lamsweerde. Goal-oriented requirements engineering: a guided tour. In Proc. of the IEEE Int. Req. Eng. Conf.(RE), pages 249–263, 2001.

[33] H. Sharp, A. Finkelstein, and G. Galal. Stakeholder identification in the requirements engineering process. In Proc. of the 10th Int. Work. on Datab. & Exp. Sys. Appl., pages 387–391, 1999.

[34] A. Sutcliffe, S. Fickas, and M. M. Sohlberg. PC-RE a method for personal and context requirements engineering with some experience. Req. Eng. J., 11(3):1–17, 2006.

[35] J. H. Hayes, A. Dekhtyar, and S. K. Sundaram. Advancing candidate link generation for requirements tracing: The study of methods. IEEE Trans. on Soft. Eng., 32(1):4–19,2006.

[36] E. Yu. Agent orientation as a modelling paradigm. Wirtschaftsinformatik, 43(3):123–132, April 2001.

[37] S. Buhne, K. Lauenroth, and K. Pohl. Modelling requirements variability across product lines. In Proc. of the IEEE Int. Req. Eng. Conf. (RE), pages 41–52, 2005.

[38] A. Cockburn. Writing Effective Use Cases. Addison- Wesley, 2001.

[39] M. Sabetzadeh and S. Easterbrook. Traceability in viewpoint merging: a model management perspective. In Proc. of the Int. Work. on Trace. in Emerg. Forms of Soft. Eng.,pages 44–49, 2005.

[40] J. D. Hay and J. M. Atlee. Composing features and resolving interactions. In Proc. of ACM SIGSOFT Found. On Soft. Eng. (FSE), pages 110–119, 2000.

[41] W. N. Robinson, S. D. Pawlowski, and V. Volkov. Requirements interaction management. ACM Comp. Sur.,35(2):132–190, 2003.

[42] C. Potts, K. Takahashi, and A. Ant´on. Inquiry-based requirements analysis. IEEE Soft., 11(2):21–32, 1994.

[43] W. Damm and D. Harel. LSCs: Breathing life into message sequence charts. Form. Meth. in Sys. Des., 19(1):45–80,2001.

[44] S.J. Greenspan, J. Mylopoulos, and A. Borgida, “Capturing More World Knowledge in the Requirements Specification”, Proc. ICSE-6: 6th Intrnational Conference on Sofiware Enginering, Tokyo, 1982.

[45] D. Bush and A. Finkelstein. Requirements stability assessment using scenarios. In Proc. of the IEEE Int. Req. Eng.Conf. (RE), pages 23–32, 2003.

[46] K. Schmid. The product line mapping approach to defining and structuring product portfolios. In Proc. of the IEEE Int.Req. Eng. Conf. (RE), pages 219–226, 2002.

[47] A. Sutcliffe, W.-C. Chang, and R. Neville. Evolutionary requirements analysis. In Proc. of the IEEE Int. Req. Eng. Conf. (RE), pages 264–273, 2003.

[48] T. A. Alspaugh and A. I. Ant´on. Scenario networks for software specification and scenario management. Technical Report TR-2001-12, North Carolina State University at Raleigh, 2001.

[49] R. Settimi, E. Berezhanskaya, O. BenKhadra, S. Christina, and J. Cleland-Huang. Goal-centric traceability for managing non-functional requirements. In Proc. of the IEEE Int. Conf. on Soft. Eng. (ICSE), pages 362–371, 2005.

[50] D. Berry and E. Kamsties. Ambiguity in Requirements Specification. Perspectives on Software Requirements, chapter 2. Kluwer Academic Publishers, 2004.

[51] Checkland P. Soft systems methodology: rational analysis for a problematic world. Wiley, New York, 1989

[52] P. Carlshamre, K. Sandahl, M. Lindvall, B. Regnell, and J. N. och Dag. An industrial survey of requirements interdependencies in software product release planning. In Proc. of the IEEE Int. Req. Eng. Conf. (RE), pages 84–93, 2001.

[53] Kurt Bittner, Ian Spence (2002). Use Case Modeling. Addison Wesley Professional, 2-3. ISBN 0-201-70913-9.

[54] K. S. Wasson, K. N. Schmid, R. R. Lutz, and J. C. Knight. Using occurrence properties of defect report data to improve requirements. In Proc. of the IEEE Int. Req. Eng. Conf. (RE), pages 253–262, 2005.

[55] Dorine C. Andrews. JAD: A crucial demension for rapid applications development. Journal of Systems Management, pages 23-31, March 1991.

[56] McDermid, J.A. “Requirements Analysis: Problems and the STARTS Approach”, in IEEE Colloquium on ‘Requirements Capture and Specification for Critical Systems (Digest no. 138), 4/1-4/4. IEEE, November 1989.

[57] H. In, T. Rodgers, M. Deutsch, and B. Boehm. Applying WinWin to quality requirements: A case study. In Proc. Of the IEEE Int. Conf. on Soft. Eng. (ICSE), pages 555–564, 2001.

[58] A. Moreira, A. Rashid, and J. Araujo. Multi-dimensional separation of concerns in requirements engineering. In Proc. of the IEEE Int. Req. Eng. Conf. (RE), pages 285–296, 2005.

[59] F. Chantree, B. Nuseibeh, A. de Roeck, and A. Willis. Identifying nocuous ambiguities in natural language requirements. In Proc. of the IEEE Int. Req. Eng. Conf. (RE), pages 59–68, 2006.

[60] J. H. Hausmann, R. Heckel, and G. Taentzer. Detection of conflicting functional requirements in a use case-driven approach. In Proc. of the IEEE Int. Conf. on Soft. Eng. (ICSE), pages 105–115, 2002.

[61] Ivar Jacobson (1992). Object-Oriented Software Engineering. Addison Wesley Professional. ISBN 0-201-54435.

[62] K.L. Heninger, “Specifying Software Requirements for Complex Systems: New Techniques and their Application”, IEEE Transactions on Software Engineering Vol.6 No.1, January 1980,

2-13.

[63] G. Joseph, L. Charlotte. Techniques for Requirements Elicitation. In Proc. of the IEEE Computer Society (RE), pages 152–164, 1993.

[64] Linda A. Macaulay: Requirements Engineering, Springer Verlag, 1996.

[65] Flynn DJ, Jazi MD. Constructing user requirements: a social process for a social context. Information System 1998;8(1):53–83

[66] H. Peter. A Systems Engineering View of Requirements Management for Software-intensive Systems.

[67] Lee, D. J. and Thornton, A. C. The identification and use of key characteristics in the product development process. In Proceedings of the 1996 ASME Design Engineering Technical Conferences and Computers in Engineering Conference, Irvine, California, 1996 (American Society of Mechanical Engineers, New York).

[68] Estublier, J. (2000). Software Configuration Management: A Roadmap.

[69] Fox, Nevill, "Capability Maturity Model (RM-CMM) for Risk Management", Silicon Rose, http://www.siliconrose.com.au/Articles/RiskCMM.htm, 2005.

[70] Bohner, S. A. & Arnold, R. S. (Ed.). (1996). Software Change Impact Analysis. IEEE Computer Society Press.

References for requirement modeling

1) Dzida W., Freitag R., Makinh "Use of Scenarios for validating Analysis and design". In IEEE Transactions on Software Engineering, Vol 24, No.12, December 1998.
2) Araujo J.,Whittle J., Kyoo Kim D., "Modeling and Composing Scenario-Based Requirements with Aspects". In proceedings of the 12th International Requirement Engineering Conference.
3) Weidenhaupt K., Pohl K., Jarke M., Haumer P., "Scenarios in System Development Current Practice".
4) J. polajnar, D. Polajnar , K. Pruden , "An Integrative Approach to requirement Modeling". In proceedings of the 1999 IEEE Canadian Conference on Electrical and Computer Engineering.
5) Navarro E., Letelier P., Ramos I., "Integrating Expressiveness of Modern Requirement Modeling Approaches". In proceedings of 2005 ACIS conference on Software Engineering Research.
6) Delugach H.S. " An Approach to conceptual Feedback In multiple Viewed Software Requirement Modeling".
7) Lu M., Zhao X., Li M., "Object- Oriented Requirements Modeling Based on UML", 1999.
8) Dawson L., Swatman P., "The use of object-oriented models in requirement engineering -A Field study".
9) Jiazhong Z., Zhijian W. "NDHORM: An OO approach to Requirement Modeling ". Software Engineering Notes vol 21 no 5, September 1996.
10) Mylopoulos J., Chung L., Yu E., "From object- oriented to goal – oriented Requirement Analysis". Communications of the ACM, January 1999.
11) Heitmeyer C., Bull A., Gasarch C., Labaw B., "SCR: A Toolset for Specifying and Analyzing Requirements". 1995.
12) Schatz B., Fleischmann A., Geisberger E., Pister M., "Model Based Requirement engineering with AutoRAID".
13) P. Giorgini , F. Massacci , J . Mylopoulos, A. Siena, and N. Zannone, "ST-Tool: A CASE Tool for Modeling and Analyzing Trust Requirements", 2005
14) Dascalu S., Debnath N. , Debnath N. , Akinwale O. , " STORM Software Tool for the Organization of Requirement Modeling" , 2006 .
15) M. Peter, B. pat, " The Rationale for OO Associations in Use Case Modeling ", In Journal of Object Technology, Vol 4, No 9, 2005.
16) P. Andy, "Requirement Engineering: Use Cases and more ".
17) D. Adenekan, "Qualifying Use Case Diagram Associations".
18) J. Gracia Molina, M. Jose Ortin , M. Begona , N. Joaquin , T. Ambrosio "Towards Use case and conceptual Models through business modeling ."
19) J. Bubenko , C. Rolland , P. Loucopoulos , V. DeAntonellis, " Facilitating "Fuzzy to Formal Requirement Modeling ." , IEEE , 1994.
20) Comp 6481 : Lecture notes (Dr. Olga).
21) Gunter C. A , Gunter E.L. , Zava P. , Jackson M., " A Reference Model for Requirements and Specifications ." IEEE 2000.
22) Liu D. , Mei H. , "Mapping requirements to software architecture by feature orientation ."
23) B. Darrell , " Requirement Modeling Technology A vision for better , faster , and cheaper systems ."
24) Buescher T.," Requirements Modeling For Real-Time Software Development .|",IEEE 1990.
25) E.Insafran ,O. paster , R. Wieringa "Requirement engineering based conceptual modeling ." Requirement engineering(2002) Springer –Verlag London Limited.
26) Constsnce L . Heitmeyer , Ralph D. , Jeffords and Bruse G. labaw , "Automated Consistency checking of Requirement specifications" .ACM Transactions on software engineering and methodlogy , Vol 5, No 3, July 1996.

27) F.David,G.Stewart "Formal Requirement models from Domain knowledge ."Department of computing king's college London .
28) IM/IT standards and guidelines, "Requirement Modeling and Specifications." March 31, 2003.
29) H. Elizabeth , D. Philip , "Analysis of software systems Requirement models."IEEE, 1996.
30) G. Murali Krishna , "Requirement Modeling Experience from Insurance project". Proceedings of fourth IEEE international conference on Software Engineering and Formal Methods , 2006.
31) A. Ritu , P.S Atish , T. Mohan " Cognitive Fit in Requirement Modeling : A study of object and process methodologies ." Journal of Management Information systems , 1996.
32) K. John , "Integrating the Understanding of Quality in requirements Specifications and Conceptual modeling" .Software Engineering notes Vol 23 , No 1, January 1996.
33) F.H . Hubert , L . Frenz " Requirement engineering as a success factor in Software projects ." IEEE Software , July/August 2001.
34) C . Periannan "How Use Case Modeling policies have affected the success of various projects ." Conference on Object Oriented Programming Systems Languages, 1997
35) A. MF , MA Peraze-Quinones " Using Task models to generate multi platform user interfaces while ensuring usability ." Conference on Human factors in computing systems ,2002.
36) L.Mitch , P . colin , R. Charles " A Review of state of the practice in Requirement Modeling ."IEEE, 1992.
37) S. Pete , S. lan , V . Stephen "capturing the benefits of Requirement Engineering", IEEE , march/April 1999 .
38) F. A Stephen ," Fast , cheap Requirements : Prototype or else ." IEEE Software.
39) J.Yanbing , X. Chunxiao , H. Wie , Y. Jijiang " On procedure strategy of Constructing the SOA's modeling language ." Proceedings of the 2005 International workshops on Service-Oriented systems, IEEE , 2005 .
40) D. Bernhard , " A process model for Requirement Engineering of CCOTS ."Proceedings of tenth international workshop , IEEE , 1999.