ILMENAU UNIVERSITY OF
TECHNOLOGY

Integrating Requirements Prioritization and Selection into Goal Models

Doctoral thesis
for attaining the academic degree of

Doctor of Engineering (Dr. -Ing.)

presented to the Faculty of Information and Automation
Software Architectures and Product Lines Group

by M.Sc. Arfan Mansoor
(12. December 1983)

1. Reviewer: Dr.-Ing. Detlef Streitferdt
2. Reviewer: Prof. Dr.-Ing. habil. Wolfgang Fengler
3. Reviewer: Associate Professor. Ghulam Rasool

Submitted on: 26.10.2016
Defended on: 30.05.2017

urn:nbn:de:gbv:ilm1-2017000222

Dedication

To the never ending memories of my father and grandmother
To my mother for her ongoing love and support
To my wife, brothers, sisters and lovely nephews and nieces

Acknowledgements

Thanks to Almighty Allah(SWT).

I would like to thank my guide Dr.-Ing. Detlef Streitferdt for giving me an opportunity to work with him as a PhD student. I am indebted to him for his persistence during my work. He spent long hours with me discussing all aspects of my thesis. His valuable suggestions helped me a lot, not only in my work but also in difficult times. Without his guide and support this work would not have been possible.

I would also like to thank professor Ilka Philippow who initially supported me at the Ilmenau University of Technology. Furthermore, I would like to thank Dr Oswald Kowalski, Dr. Patrick Mäder, Franz-Felix Füßl, Stefan Wendler and late Heiner Kotula for their friendly attitude and support during my PhD. My special thanks to Nils Würfel for his technical support at TU Ilmenau.

Special thanks to my late father Mansoor Ahmed Gill who always has been a source of inspiration for me. Moreover, I sincerely thanks my brothers for their support, patience and encouragement throughout my educational timespan and my friends here in Ilmenau, Germany and back in Pakistan.

Finally, thanks to my wife for understanding ups and downs in last one year.

Abstract

Requirements engineering is the first main activity in software development process. It must address the individual goals of the organization. The inadequate, inconsistent, incomplete and ambiguous requirements are main obstacles on the quality of software systems. Goal Oriented Requirements Engineering (GORE) starts with abstracts high level goals. These goals are refined to lower levels until they are assignable to agents. During GORE analysis, decisions need to be made among alternatives at various positions. Decisions involve different stakeholders which may contradict with each other based on certain criteria.

In the context of GORE, the support for identifying and managing the criteria for requirements selection process is required. The criteria are based on stakeholders needs and preferences and therefore stakeholders opinions need to be involved in selection process. It helps to identify the importance of requirement according to stakeholders understandings and needs. It also helps in the understanding of interaction between system and stakeholders (stakeholders involvement in making important decisions) and by documenting the stakeholder preferences early in GORE, helps to identify inconsistencies early in the requirements engineering.

Software quality requirements are essential part for the success of software development. Defined and guaranteed quality in software development requires identifying, refining, and predicting quality properties by appropriate means. Goal models and quality models are useful for modelling of functional goals as well as for quality goals. This thesis presents the integration of goal models with quality models, which helps to involve stakeholders opinions and the representation of dependencies among goals and quality models. The integration of goal models and quality models helps in the derivation of customized quality models. The integrated goal-quality model representing the functional requirements and quality requirements is used to rank each functional requirement arising from functional goals and quality requirement arising from quality goals. Triangular Fuzzy Numbers (TFN) are used to represent stakeholder opinions for prioritizing requirements. By defuzzification process on TFN, stakeholders opinions are quantified. TFN and defuzzification process is also used to prioritize the identified relationships among functional and non-functional requirements. In the last step development constraints are used to re-prioritize the requirements. After final prioritization,

a selection algorithm helps to select the requirements based on benefit over cost ratio. The algorithm makes sure that maximum number of requirements are selected while fulfilling the upper cost limit. Thus the whole process helps in the selection of requirements based on stakeholders opinions, goal-quality models interaction and development constraints.

The thesis also presents an integrative model of influence factors to tailor product line development processes according to different project needs, organizational goals, individual goals of the developers or constraints of the environment. Tailoring is realized with prioritized attributes, with which the resulting elements of the product, process and project analysed are ranked. An integrative model for the description of stakeholder needs and goals in relation to the development process artefacts and the development environment specifics is needed, to be able to analyse potential influences of changing goals early in the project development. The proposed tailoring meta-model includes goal models, SPEM models and requirements to development processes. With this model stakeholder specific goals can be used to support binding a variable part of the development process. This support addresses soft factors as well as concrete requirements.

Zusammenfassung

Requirements Engineering ist der erste Schritt im Softwareentwicklungsprozess. Er dient zur Aufnahme organisationsabhängiger Ziele und Anforderungen. Unangemessene, inkonsistente, unvollständige oder mehrdeutige Anforderungen können die Qualität von Softwaresystem stark negativ beeinflussen. Goal Oriented Requirements Engineering (GORE) beginnt mit der Entwicklung von übergeordneter Zielen, welche in weiteren Entwicklungsstufen verfeinert werden, bis sie einer verantwortlichen Person zugewiesen werden können. Während einer GORE Analyse werden an verschiedenen Stellen Entscheidungen über Alternativen getroffen.

Diese Entscheidungen betreffen unterschiedliche Akteure, die sich in ihren Ansichten widersprechen können. Im Rahmen von GORE wird die Unterstützung zur Identifizierung und Verwaltung von Kriterien zur Auswahl von Anforderungen benötigt. Diese Kriterien basieren auf den Vorstellungen und Vorlieben von Stakeholdern, daher ist eine Integration aller Stakeholder in den Auswahlprozess erforderlich. Dies soll dabei helfen, die Bedeutung bestimmter Anforderungen auf Basis der betroffenen Personen zu identifizieren und aufzuarbeiten. Darüber hinaus hilft GORE bei der Kommunikation zwischen System und Akteuren durch ihren Einbezug in wichtige Entscheidungen. Durch frühzeitige Dokumentation des tatsächlichen Stakholderbedarfs können Inkonsistenzen im Requirements Engineering frühzeitig ermittelt werden.

Die Bestimmung von Software Qualitätsmerkmalen ist wesentlicher Erfolgsfaktor in der Software Entwicklung. Zur Gewährleistung einer qualitativen Softwareentwicklung und eines entsprechenden Produktes sind die Identifizierung, die Verfeinerung und die Vorhersage von Qualitätseigenschaften jederzeit durch geeignete Maßnahmen erforderlich. Goal Models und Quality Models sind wertvolle Werkzeuge zur Ermittlung und Modellierung funktionaler und nicht-funktionaler Anforderungen und Ziele. Diese Arbeit enthält einen Lösungsansatz zur Integration von Goal Models und Quality Models, der dazu beitragen soll, Stakeholder und Abhängigkeiten zwischen Goal und Quality Models einzubeziehen und sichtbar zu machen. Die Integration von Goal Models und Quality Models soll zur Ableitung spezifischer Quality Models beitragen. Somit kann das integrierte Goal-Quality Model, welches die funktionalen Anforderungen und die Qualitätsanforderungen vereint, zur Priorisierung aller funktionalen Anforderung, die sich aus den funktionalen Zielen ergeben, und aller Qualitätsanforderun-

gen, die aus Qualitätszielen resultieren, dienen. Zur Priorisierung der Anforderung auf Basis der Stakeholderbedarfe werden Triangular Fuzzy Numbers (TFN) verwendet. Nach der endgültigen Priorisierung dient ein spezieller Algorithmus zur Einschätzung und Auswahl der Anforderungen auf Basis einer Kosten-Nutzen-Analyse. Dieser Algorithmus stellt sicher, dass unter Einhaltung einer von der Organisation gewählten Kostenobergrenze die maximale Anzahl der Anforderungen umgesetzt werden kann. Der gesamte Prozess dient demnach zur Anforderungsanalyse unter Berücksichtigung verschiedener Interessengruppen, Abhängigkeiten, sowie durch den Einbezug von Grenzen, die sich beim Zusammenspiel von Goal-Quality Models und der Softwareentwicklung ergeben können.

Darüber hinaus enthält die Arbeit ein integratives Modell, um Entwicklungsprozesse während der Erstellung von Produktlinien an Einflussfaktoren, wie Projektbedürfnisse, Organisationsziele, individuelle Ziele von Entwicklern oder an Umweltbedingungen anzupassen. Dieses sogenannte Tailoring wird durch Priorisierung von Attributen erreicht, welche verschiedene Elemente des zu erzeugende Produktes, des Prozesses oder des Projektes analysieren und nach Bedeutung sortieren. Ein integratives Modell zur Beschreibung von Stakeholderbedürfnissen und -zielen in Bezug auf die Artefakte des Entwicklungsprozesses und die Besonderheiten einer Entwicklungsumgebung wird benötigt, um potenzielle Einflüsse sich verändernder Ziele frühzeitig während der Projektentwicklung zu analysieren. Das hier vorgestellte Tailoring-Meta-Model beinhaltet Goal-Models, SPEM Models und Requirements hinsichtlich Entwicklungsprozesse. Mithilfe dieses Modells können stakeholderspezifische Ziele dazu verwendet werden, um einen variablen Teil eines Entwicklungsprozesses projektbezogen zu gestalten. Auf diese Weise können weiche Faktoren genauso integriert werden, wie konkrete Anforderungen.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Software development process is mainly divided into four stages; vision, definition, maintenance and development. Each stage has different focus of activities. In vision phase focus is on 'why' i.e., why this system is required. In definition phase focus is on 'what' i.e., what needs to built the outlined vision. Development phase is focused on design and implementation of the system while in the maintenance phase, system changes and enhancement in the system are carried out. Requirements engineering is the starting point of development process at which the system services and constraints are established or in other words elicited.

Most of the problems in development process are tracked down to shortcomings in the requirements gathering and requirements specification phase. Some studies show that 40% - 60% of defects in software projects are because of poor requirements [Lam00a]. It is necessary to identify and catch errors early in the software development life cycle because correcting an error later is more difficult, more time consuming and it will also costs much more. Correcting an error after development costs 68 times more than correcting it before development and it may go up to 200 times [Lam00a]. Poor requirements are also cause of delays and over budgets in software development life cycle. Therefore, requirements are one critical success factor for any software development project.

In 10th RE conference (RE02) requirements engineering is defined as goal-driven: "Requirements Engineering (RE) is the branch of system engineering concerned with the real-world goals for, functions of, and constraints on software-intensive systems. It is also concerned with how these factors are taken into account during the implementation and maintenance of the system, from software specifications and architectures up to final test cases." Goals have been used as high-level abstraction medium for the structuring and abstracting the contents of requirements [Don04] .

Goals are used for identifying, organizing and justifying software requirements [Ant96] and goal oriented requirements engineering (GORE) deals with the use of goals for eliciting, elaborating, structuring, specifying, analysing, negotiating, documenting, and modifying requirements [VL01]. GORE is an incremental approach in which high level goals are identified then these high level goals are refined and classified into different categories. Different types of goal categorization have been purposed. Finally the requirements and assumptions are elaborated to meet these goals. A goal under the responsibility of single agent in the software-to-be is called a requirement while a goal under the responsibility of single agent in the environment of the software-to-be is called an assumption.

One of the major highlights of Goal Oriented Requirements Engineering (GORE) is the concept of early requirements analysis [Lap05]i.e., instead of answering 'what' needs to be implemented GORE first focus on 'why' and 'how' questions (not 'how' to implement but 'how' to identify new goals) . GORE also helps to answer 'who' and 'when' questions [DLF93] i.e., 'why' a certain goal is required, 'how' that goal can be achieved and 'who' is responsible for that goal. Goal elicitation, refinement and analysis of goals, assignment of goals to agents, and alternative system proposals are some of the main areas of research in GORE. This research will focus on GORE in general and on finding alternative system proposals in specific.

1.1 Motivation

Several studies show that requirements problems are major cause of cost overruns and project delays. A survey of 8000 projects under 350 organizations was conducted in US and mostly the causes of failures were identified to poor requirements - more specifically, the lack of user involvement (13%), requirements incompleteness (12%), changing requirements (11%), unrealistic expectations (6%), and unclear objectives (5%) [Lam00a]. The inadequate, inconsistent, incomplete, ambiguous requirements are numerous and they have critical impact on the quality of the resulting software [LL02].

[LL02]describes requirements gathering, establishing the detailed technical requirements including all the interfaces to people, to machines, and to other software systems as the hardest single part of building a software system. No other part of the work so cripples the system if done wrong. No other part is more difficult to rectify later. Therefore, the iterative extraction of product requirements are the most important task that requirements engineer performs for the client.

Requirements engineering approaches so far discussed are in the what-how range. The idea of goals have introduced 'why' concerns in the early stage of requirements engineering i.e., 'why' the system was needed and does the requirements specification captures the needs of stakeholders. The idea of goal also emphasized the understanding of organizational context for new system [Lap05]. Goal based requirements engineering is concerned with identification of high level goals to be achieved by the system envisioned, the refinement of such goals, the operationalization of goals into services and constraints and the assignment of responsibilities for the resulting requirements to agents such as human, devices and programs [DL96]. Requirements engineering must address the contextual goals, functionalities to achieve these goals and constraints restricting how these functions are to be designed and implemented [Lam00a]. These goals, functions, and constraints have to be mapped to precise specifications of software behaviours and their evaluation over time and across software families [Zav97].

Although Eric Yue was the first one who explicitly stated the representation of goals for requirements completeness – the requirements are complete if they are sufficient to establish the goal they are refining [Yue87] but the idea of goal was already recognized as essential component of requirements engineering by Ross and Schoman [VL01] as they stated "Requirements definition must say 'why' a system is needed, based on current or foreseen conditions, which may be internal operations or an external market. It must say 'what' system features will serve and satisfy this context. And it must say 'how' the system is to be constructed" [RS79]. Unified Modelling Language (UML) also mentions the importance of goals as higher-level abstractions "In my work, I focus on 'user goals' first, and then I come up with use cases to satisfy them; by the end of the elaboration period, I expect to have at least one set of system interaction use cases for each user goal I have identified" [FS97]. From 10th requirements engineering conference the notion of goal has been explicitly stated in requirements engineering "Requirements Engineering (RE) is the branch of systems engineering concerned with the 'real-world goals' for, functions of, and constraints on software-intensive systems. It is also concerned with how these factors are taken into account during the implementation and maintenance of the system, from software specifications and architectures up to final test cases."

The difficulties during the GORE lead to several challenges:

C1 One start with initial high level goals and keep refining them until they are reduced to functional requirements satisfying these goals. The goals identified at the start may be of contradictory nature, for example, there may be technical contradictions or physical contradictions. In technical contradiction, alternative solution improve

one aspect at the expense of another while in physical contradiction solution may be required to be in two states at once.

C2 Analysis of these contradictory nature goals is required to facilitates the discovery of trade-offs and search of the full space of alternatives rather than a subset [Myl06].

C3 Each alternative found represents a particular way to satisfy the goal [MCL+01]. This leads to the selection of alternative system proposal [LL00].

C4 Decision making is required at various positions [Myl06] in GORE e.g., during the decomposition of an objective into sub-objectives there are different alternatives and one need to select the best one according to certain criteria. During conflict analysis, resolving critical risks, assigning a goal to particular agent or operationalizing particular objectives different solutions can be available and the selection process is required.

C5 Alternative solutions will help to capture variability in GORE [Lap05]. Instead of going for one solution, all alternatives solutions are considered and then one solution is selected. This leads to customizable solutions depending on the stakeholder preferences.

C6 For stakeholder preferences, there is need to capture stakeholders opinions into GORE.

C7 A better understanding is required on the prioritization and evaluation of goals based on stakeholder preferences.

C8 For quality properties, the impacts measurement of quality requirements on goals is necessary.

1.2 Goals of the Thesis

This work is focused on the requirements phase of the software development which is the starting point of development process where system services and constraints are established or elicited. The work particularly focuses on goal oriented requirements engineering(GORE) and provides a systematic mean to *involve stakeholder preferences into GORE, prioritization of goals based on stakeholder preferences, tailoring of quality models based on goals and their integration into goal models. Next, it evaluates the impact of quality goals on high level identified goals and produces two prioritized lists*

for functional requirements and non-functional requirements. Then it quantifies the impact of development constraints. In the last it focused on selection of requirements based on cost constraints.

The research goals related to this work are:

Goal 1 Evaluating existing goal methods/frameworks for requirements analysis.

Goal 2 Establishing the means for stakeholders representation into GORE.

Goal 3 Providing a prioritization scheme for high level goals and quality goals which is based on stakeholder preferences.

Goal 4 Providing a goal based tailoring process for quality model, for the quality requirements and their integration into goal models.

Goal 5 Once the integrated goal-quality model is obtained, establishing the influences (positive or negative) of quality goals on each other.

Goal 6 Measuring the impact of quality goals on each other and of high level goals.

Goal 7 Involving development factors into prioritization of requirements and selection of requirements based on cost constraints.

Goal 8 Alternative selection of system solutions based on already established prioritization scheme.

Goal 9 Provide the tailoring of product line development process based on goals.

1.3 Contributions

This thesis contributes to the area of requirements engineering generally and to GORE more specifically. Following are the contributions made by this thesis in the field of GORE:

- It presents a through investigation of GORE concepts and frameworks and identification of limitations regarding stakeholders involvement in GORE. In the process thesis presents an approach to involve the stakeholders opinions into GORE by allowing them to rate the importance of each requirement .

- To achieve the overall goal which is the decision support and prioritization and selection of requirements in GORE, the thesis combines ideas from goal-oriented requirements engineering with quality models. An integrative model based on goal models and quality models is presented. The goal graphs of GORE are used for transition from high level goal to lower level requirements. Then the lower level requirements are presented to different stakeholder to accumulate their opinions regarding importance of these requirements. Based on their inputs, each requirement is quantitatively evaluated. The integration helps to model explicitly the dependencies between goals, and quality requirements. Furthermore,

- The integration of goal models and quality models helps in resolving conflicts among goals. It helps to quantitatively evaluate the dependencies regarding the influence of goals and quality requirements to support prioritization. The dependencies are quantified using fuzzy numbers and by using multi-criteria approach each alternative is ranked.

- By using the quantification of stakeholder opinions, dependencies and development constraints a requirements prioritization technique is implemented. Proposed approach provides two prioritization of both functional and non-functional requirements. The approach not just provide the rank/order of requirements but also a selection algorithm which helps to select maximum number of requirements by not exceeding the cost upper limit.

- The thesis also presents an integrative model of influence factors to tailor product line development processes. The tailoring process is based on project needs, organizational goals, and individual goals of the developers or constraints of the environment. Tailoring is realized with prioritized attributes, with which the resulting elements of the product, process and project analysed are ranked. An integrative model for the description of stakeholder needs and goals in relation to the development process artefacts and the development environment specifics is needed, to be able to analyse potential influences of changing goals early in the project development. With this model stakeholder specific goals can be used to support binding a variable part of the development process. This support addresses soft factors as well as concrete requirements.

1.4 Outline of the Thesis

Chapter 2 discusses the fundamentals software requirements, requirements engineering(RE) process, requirements statements characteristics, requirements specification characteristics and evaluates the traditional RE definitions. This chapter concludes with different types of requirements, their use for this work and why GORE is useful as requirements engineering.

Chapter 3 elaborates the state of the art for GORE. Main concepts, definitions and the strengths of GORE are discussed. Goal based requirements analysis is presented and most widely used frameworks based on GORE are discussed. The findings from these frameworks that are useful for this work are highlighted in this chapter.

Chapter 4 discusses the need of decision making at the early stage of requirements engineering. This chapter highlights the decision points in GORE which are prerequisite of effective prioritization. Factors that influence decisions in GORE are established specifically the importance of non-functional requirements as decision factors.

Chapter 5 presents classification of quality model and comparison of these quality models. The main highlights of this chapter is the integration of goal models and quality models and an integrated meta model is the research output of that integration.

Chapter 6 presents an approach for the prioritization of functional goals. The prioritization is based on stakeholder opinions. Triangular fuzzy numbers are used in the process because of their accurate representation in vague situations. Quality goals are also prioritized and the interactions between quality goals and functional goals are quantified.

Chapter 7 presents the extension of approach used in last chapter for alternatives selection. A variant of TOPSIS method is used for alternative selection. Score obtained by TFN and defuzzification process are used as weighing criteria by TOPSIS to rank best alternative.

Chapter 8 discusses the use of goal models for product line development. For tailoring product line development process a tailoring meta-model is presented. This meta-models integrates goal models, SPEM process models as well as requirements. With this model stakeholder specific goals can be used to support binding a variable part of the development process.

Chapter 9 presents the evaluation of the proposed approach. For evaluation, student experiment was conducted. The experiment was performed in two rounds; In first round proposed approach is compared to most widely used approach in the literature and in second round it is compared with five other approaches. In the end a survey in the form of questionnaire is given to participants to evaluate proposed approach.

Chapter 2

Fundamentals of Software Requirements

This chapter starts with brief introduction of requirements engineering process and why goal-oriented requirements engineering is needed. Next the main concepts used in goal oriented requirements engineering are defined. In last part of this chapter, some major goal oriented frameworks will be discussed and their usage for this thesis is highlighted.

2.1 Software Requirements

Before describing RE process, it is important to understand *what requirements are*. A number of definitions of requirements exits in literature but the most used in research and academia are presented in table 2.1:

Evaluation of Traditional Definitions All of these definitions take requirements engineering as a whole and there is no distinction made between early phase and late phase requirements engineering. Use of Goal Oriented Requirements Engineering (GORE) helps to distinguish between early phase and late phase requirements engineering. In early stage requirements engineering the focus is on high level goals, on stakeholder needs and interests. Early-stage requirements engineering is characterized by uncertainty, ambiguity etc. Late-stage requirements engineering concerns future objectives and how these may be operationalized in terms of systems components. Here focus has been on analysis for ambiguity, incompleteness and inconsistency. It focuses on achieving the completeness, consistency, and precision on moving towards the final specification document.

Table 2.1: Software Requirements

Author(s)	**Description**
Definition by Jones	Software requirements document is a statement of needs by a user that triggers the development of a program or system.
Definition by Alan Davis	Software requirements document is a user need or necessary feature, function, or attribute of a system that can be sensed from a position external to that system.
Definition by Ian Somerville	Requirements are a specification of what should be implemented. They are descriptions of how the system should behave, or of a system property or attribute. They may be a constraint on the development process of the system.
Definition by IEEE	IEEE defines software requirements as: • A condition or capability needed by user to solve a problem or achieve an objective. • A condition or capability that must be met or possessed by a system, or system component, to satisfy a contract, standard, specification, or other formally imposed document. • A documented representation of a condition or capability as in 1 or 2
Definition by Webster's Dictionary	Requirement is something required, wanted or needed (there is difference between wanted and needed and it should be kept in mind).

2.2 Requirements Engineering (RE) Process

Software requirements engineering is a process which enables to systematically determine the requirements for a software product. The process involved in developing system requirements is collectively known as Requirements Engineering process. RE process is shown in figure 2.1. The focus is on functionality of the system to be built i.e., what the system needs to do. In contrast, in goal driven methods the importance is on why a certain functionality is needed and how it can be implemented.

The major activities performed in RE process are: requirements elicitation, requirements analysis and negotiation, requirements specifications, requirements validation. These activities are represented in RE process as shown in figure 2.2.

Figure 2.1: RE Process [KS98]

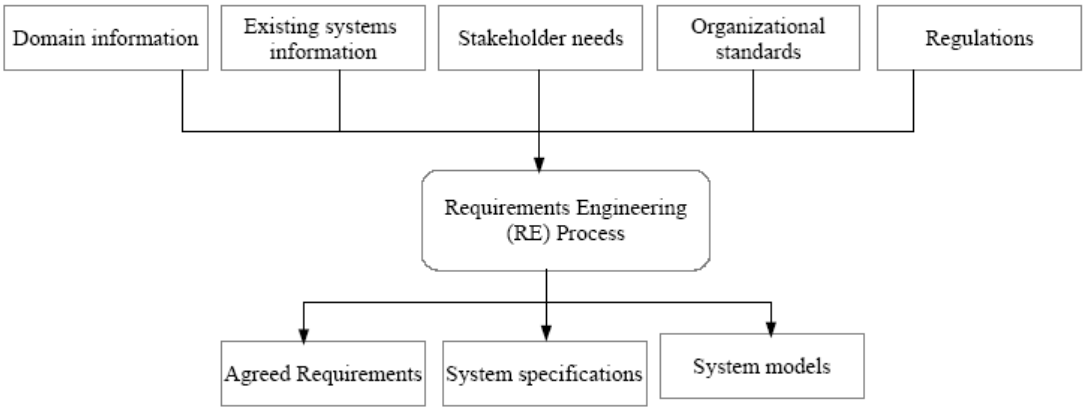
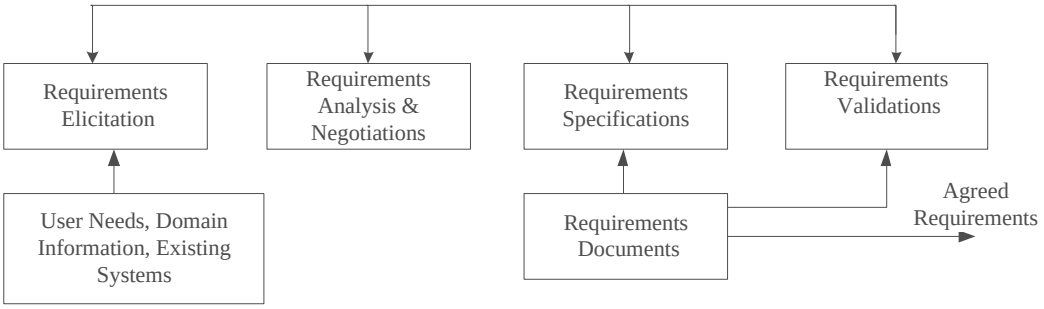


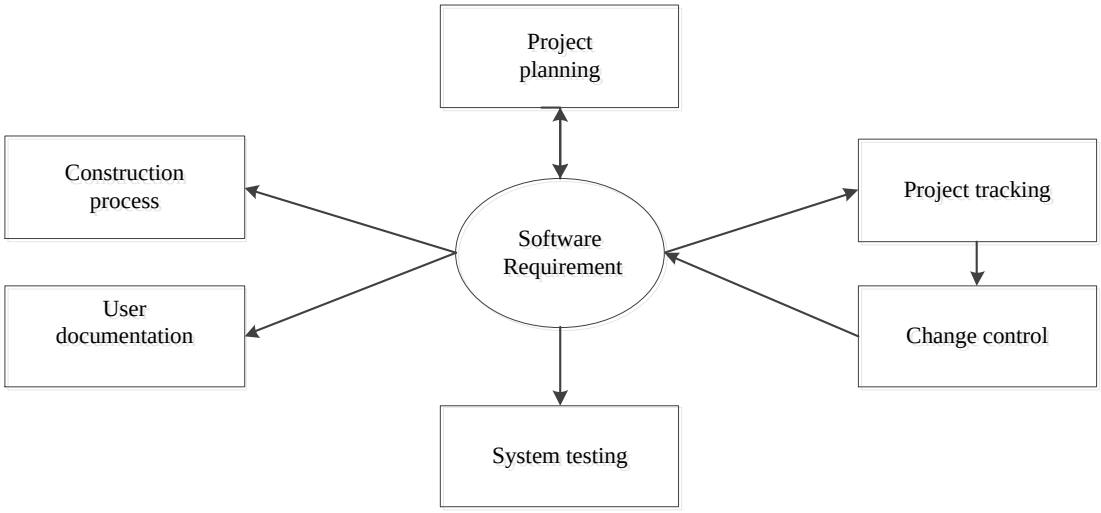
Figure 2.2: RE Process Activities [KS98]



Requirements documents produced as output of the RE process are used throughout software development cycle. They are used in project planning to determine time, effort and outlays in the project development. Requirements documents are used as base reference point in designing and coding phase of the software development. Project managers use these requirements documents to monitor and track software progress to meet deadlines. The central role of the software requirements documents in the entire development process is depicted in the figure 2.3.

Two kinds of documents are produced during RE phase i.e., Requirements Statement and Requirements Specification. They are also called Requirements Definition and Functional Specification. Requirements Definition are used to document user requirements and Functional Specification are used to document functional requirements. Requirements documents should include functional and non-functional requirements while the project requirements (e.g., staffing, schedules, costs, milestones, phases, reporting

Figure 2.3: Central Role of Requirements Documents



procedures etc.), designs, and product assurance plans (e.g., configuration management plans, verification and validation plans, test plans, quality assurance plans etc.)

2.2.1 Requirements Statement Characteristics

Requirements statement document must possess the following characteristics:

- Complete: Each requirement must fully describe the functionality to be delivered
- Correct: Each requirement must accurately describe the functionality to be built
- Feasible: It must be possible to implement each requirement within the known capabilities and limitations of the system and its environment
- Necessary: Each requirement should document something that the customer really needs or something that is required for conformance to an external system requirements or standard
- Prioritized: An implementation priority must be assigned to each requirement, feature or use case to indicate how essential it is to a particular product release
- Unambiguous: All readers of a requirement statement should arrive at a single, consistent interpretation of it
- Verifiable: User should be able to devise a small number of tests or use other verification approaches, such as inspection or demonstration, to determine whether the requirement was properly implemented.

2.2.2 Requirements Specification Characteristics

Requirements specification document must possess the following characteristics:

- Complete: No requirement or necessary information should be missing
- Consistent: No requirement should conflict with other software or higher-level system or business requirements. For example, the following requirements may be conflicting with each other.

  - All programs must be written in ADA

  - The program must fit in the memory of the embedded micro-controller

  These requirements conflict with one another because the code generated by the ADA compiler was of a large footprint that could not fit into the micro-controller memory.
- Modifiable: One must be able to revise the Software Requirements Specification when necessary and maintain a history of changes made to each requirement.
- Traceable: One should be able to link each requirement to its origin and to the design elements, source code, and test cases that implement and verify the correct implementation of the requirement.

2.3 Requirements Types

In traditional approaches requirements are categorized into five major classes. Each of them is briefly discussed except non-functional requirements which are discussed in more detail. Non-functional requirements are focused because they will be used in quality goal trees (part of solution concept) in later chapter.

1. Functional requirements
2. Non-functional requirements
3. Domain requirements
4. Inverse requirements
5. Design and implementation constraints

2.3.1 Functional Requirements

These are the statements describing what the system does; they capture the functionality of the system. Functional requirements are the statements of services the system should provide. These statements will represent the reaction to particular inputs, behaviour in particular situations, abnormal behaviour etc. sequencing and parallelism are also captured by functional requirements. Usually the customers and developers have their focus on functional requirements.

2.3.2 Non-functional Requirements

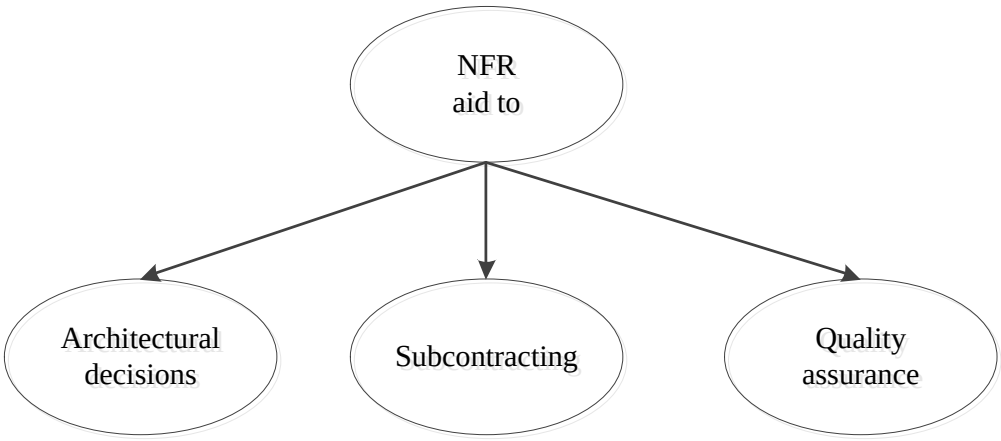
Requirements documents not only describe the services system performs but they must also describe the constraints under which it will operate. These constraints are restrictions for software developers about the design and construction of software. These kinds of requirements are called non-functional requirements. The attributes of functional requirements may be timing, performance, reliability, accuracy, security, ease of use, regulations, standards, contracts etc. Non-functional requirements arise through user needs, external factors, safety regulations, privacy legislation, budget, constraints etc. Sometimes failure to meet non-functional requirements may make the whole system unusable e.g., failure of performance requirements in real time control system will make the control function to not operate correctly thus making the system unreliable. Non-functional requirements are usually divided into three classes:

1. Product requirements: usability, reliability, portability and efficiency requirements
2. Organizational requirements: standards, Implementation and delivery requirements
3. External requirements: interoperability, ethical, privacy and safety requirements

Use for Proposed Approach Specifying non-functional requirements is key for the later stages of software development activities. Missing non-functional requirements can be a major threat of project and product success [CPL]. Non-functional requirements will be used in quality goal tree (represented in next chapter) which is used to represent the interdependencies between quality attributes. These interdependencies represent the positive and negative influences between the quality attributes and therefore help in making the important decisions and trade-offs. Non-functional requirements help to enable following key success factors in the development of the software-based systems [CPL01].

1. Identify the quality requirements that will impact the architectural decisions.
2. Identified quality requirements help in effective subcontracting. If only functional requirements are specified the contractor can get into a situation where subcontractor fulfils the functional requirements but it is still not useful because of insufficient quality.
3. Identifying measurable non-functional requirements in early phase requirements engineering at high level goals helps in early quality assurance.

Figure 2.4: Non-functional Requirements Aid



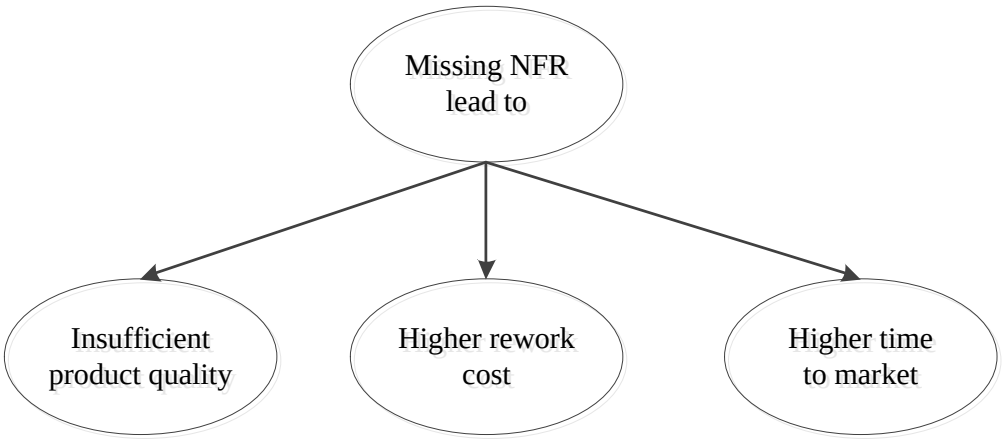
On the contrary missing the non-functional requirements can lead to [Jorsh]:

1. Product does not fulfilling the quality needs will be delivered with lower quality
2. If the product does not fulfil the quality needs and higher management decides for rework to match the quality expectations, the project will be consuming more effort than planned and time to deliver the product will be postponed. It will result in higher rework cost and
3. Higher time to market

2.3.3 Domain Requirements

Domain requirements can be both functional and non-functional requirements. They come from the application domain and represent the fundamental characteristics of the application domain. Domain requirements may not be explicitly mentioned but there absence may cause dissatisfaction.

Figure 2.5: Missing Non-functional Requirements Effect



2.3.4 Inverse Requirements

These requirements indicate the indecisive nature of customers about certain aspects of the product. They explain what the system should not do, for example, the system should not have red colour in interface etc.

2.3.5 Design and Implementation Constraints

These are guidelines within which the designer must work. They restrict choices available to customers, for example, the system should be developed using open source and shall run Linux operating system.

2.4 Why GORE

In GORE importance is given to high-level goals as opposed to their operationalizations into constraints to be ensured by agents through appropriate actions which are derived from these higher level goals at later phase requirements engineering. Instead of starting directly from lower level process or action oriented descriptions as usually done in traditional (current) RE approaches, GORE starts from system level and organizational objectives from which such lower level descriptions are progressively derived.

2.5 Summary

This chapter discussed the basic concepts of software requirements, including requirements definition, requirements process, requirements statements characteristics, requirements specification characteristics, different requirements types present in literature. The chapter focused to highlight why GORE is useful and to discuss how different requirements types are used in this thesis.

Chapter 3

State-of-the-Art GORE Concepts and Frameworks

3.1 Goal Oriented Requirements Engineering

Goal oriented requirements engineering refers to the use of high level goals for requirements elicitation, elaboration, organization, specification, analysis, negotiation, documentation and evolution [VL01]. From this definition the major activities of GORE are identified:

1. Elicitation
2. Analysis
3. Elaboration
4. Refinement
5. Specification

3.1.1 Goals, Terms and Definitions

Many authors have defined *Goals* according to their specific purpose. Most widely used GORE terms and their definitions are given in table 3.1 and table 3.2 gives the definitions of *Goals* in literature.

Table 3.1: GORE Terms and Definitions

Term	**Definition**
Object	An object is thing of interests which can be referenced in requirements. Entity, relationship, event, and agent are specialization of object concept.
Entity	An autonomous object which exits independently from other object(s).
Event	An instantaneous object defined In GORE by name and definition [Let01].
Action	Actions define state transitions. The attributes of action include PreCondition, TriggerCondition, postCondition. The pair (PreCondition, PostCondition) defines the state transition. An action can be applied only if its PreCondition holds whereas it must be applied if its TriggerCondition becomes true.
Agents	Active objects which are capable of performing operations, monitoring and controlling objects and can take the responsibility for goals. Agents may be software agents, hardware devices or humans.
Relationship	A subordinate object defined by a set of features. A feature is either an attribute of relationship or the ordered list of concepts linked by relationship together with their respective roles and cardinality e.g., 'Borrowing' may be a relationship linking 'Borrower' and 'Bookcopy' concepts. [DL96].
Attribute	In GORE an attribute is defined by characteristics like its name, informal definition, domain of values, and unit of values e.g., [Let01]. Number of attributes can be attached to goals [VL01].
Requirement	A goal under the responsibility of a single agent in the software-to-be is called requirement.
Assumption	A goal under the responsibility of a single agent in the environment of the software-to-be is called assumption.
Constraints	A constraint is an operational objective to be achieved by the composite system. It can also be defined as the limit on the achievement of the goal e.g., 'LimtedBorrowingPeriod' may be defined as constraint to make sure the availability of Book(s) in the library. Goals are made operational through constraints i.e., the goal can be achieved provided the constraints operationalizing it can be met. For example the goal 'RegularAvailability' is met by implementing 'LimitedBorrowingPeriod' constraint.

Table 3.2: What are Goals

Author(s)	Definition
Dardenne at al. 93	"A goal is a non-operational objective to be achieved by the composite system. Non-operational means that the objective is not formulated in terms of objects and actions available to some agent in the system; in other words, a goal as it is formulated cannot be established through appropriate state transitions under control of one of the agents"
Anton 97	Goals are high level objectives of the business, organization, or system. They capture the reasons why a system is needed and guide decisions at various levels within the enterprise.
Zave 97	Goals are formulated in terms of optative statements which may refer to functional and non-functional properties and range from high level concerns to lower-level ones.
Rolland et al. 1998	A goal is defined as something that some stakeholders hope to achieve in the future.
Phol and Haumer 1997	The goal represents the objectives an actor wants to achieve when requesting a certain service.

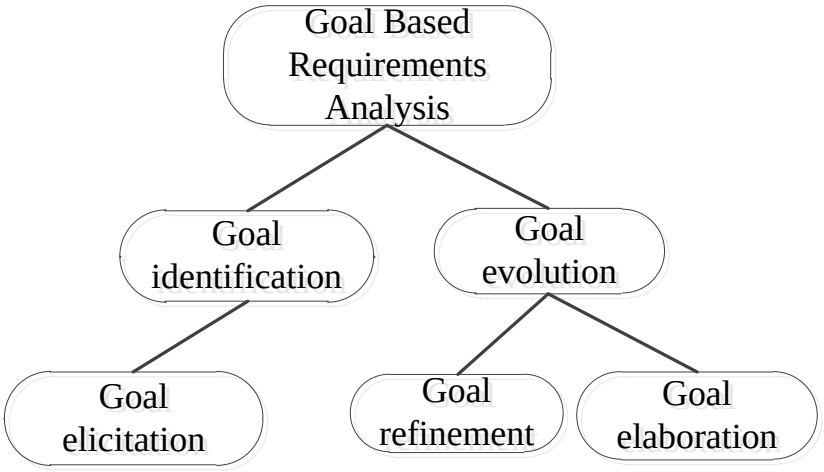
3.2 Goal Based Requirements Analysis

GORE concerns are classified into major categories i.e., goal analysis and goal evolution. Goal analysis is the process of exploring gathered documents, ranging from information about the organization, (i.e., enterprise goals) to system specific information (i.e., requirements) for the purpose of identifying, organizing and classifying goals [Ant96]. Goal evolution concerns the how goals are changed from when they were identified to when they are operationalized. Goal evolution process is further refined into goal refinement and goal elaboration. Because stakeholders change their minds and goals have to be operationalized into requirements the goals and their priorities are likely to change. In former case it is called goal elaboration and in later goal refinement [Ant96].

3.2.1 Goal Identification

Sometimes goals may be explicitly stated but most often they are implicit and elicitation process needs to b undertaken to identify goals. A preliminary analysis of current system is important source for goal identification. The main sources for identifying goals are to look for intentional keywords in documents provided like interviews, transcripts, mission statements, policy statements [Don04] etc. A common approach is to

Figure 3.1: Goal Based Requirements Analysis



find deficiencies that can be formulated, negate these deficiencies to produce first list of goals [VL01]. Once high level goals are identified then goal refinement, goal abstraction, and goal elaboration processes are used to identify further goals. Scenarios, use cases and initial goal model for these processes are used to identify goals. Goal analysis deals with identification, organization and classification of goals. Three main activities involved in goal analysis are [AP98]:

- Explore: Deals with examination of available information
- Identify: Extracting goals and identifying responsible agents from the information available
- Organize: classifying and organizing identified goals according to goal dependency relationship. Goal should be classified in particular to their target condition desired [AP98].

After the initial identification of goals the main approaches to identify further goal are:

- Goal elicitation by refinement
- Goal elicitation by abstraction
- Goal elicitation by obstacle analysis
- Goal elicitation by scenarios
- Goal elicitation from constraints

3.2.1.1 Goal Elicitation by Refinement

Goal elicitation by refinement is used to identify more concrete goals form high level goals so that these goals can be easily operationalized and implemented [Ant96]. 'How' questions are used to identify new off springs [Don04] and then AND/OR refinement links are used to link these goals. The formulations of subgoals entail the formulation of parent goal [LL00]. Subgoals may also need to be refined further until one can get assignable subgoals. For example one subgoal identified for high level goal 'Borrower request satisfied' by asking 'How' questions in library management system may be 'Book request satisfied' which may further be refined into subgoals (again by asking 'How' questions)like 'Maintain regular availability', 'Achieve availability notified' and 'Maintain enough copies' [Lap05].

3.2.1.2 Goal Elicitation by Abstraction

In some situations goals may be identified which are refinements of some parent goal and somehow they were missed in the initial goal identification process. In these cases 'Why' questions are used to elicit more abstract goals from already identified goals e.g.„ asking 'Why' question about the goal 'Maintain minimum distance between trains' yields more abstract goal 'Avoid train collision' [Don04].

3.2.1.3 Goal Elicitation by Scenarios

Scenarios also help in identification of new goals. Initial set of functions may be vague and confusing and scenarios are used to elaborate these by asking and listing different activities. Scenarios are comprised of actions or behaviours which may be mapped to goals [Let01]. In fact there is bidirectional relationship between goals and scenario. When a goal is identified a scenario can be authored for it and when a scenario is authored it can be analysed to yield goals [SMMM98]. Therefore one can say that goal elaboration and scenario elaboration are intertwined processes. Scenarios may help in elicitation of goals and goals may help in elicitation of scenarios.

3.2.1.4 Goal Elicitation by Obstacle Analysis

Obstacles provide a mechanism for anticipation of exceptional cases. This helps in finding more robust requirements. Once an obstacle is introduced it is refined much the same way goals are refined. There are number of approaches for obstacle resolution like obstacle elimination obstacle prevention, goal substitution, agent substitution, goal de-idealization, obstacle mitigation, goal restoration [LL00]. Some of these techniques like goal substitution, goal restoration, obstacle prevention and obstacle mitigation help

us to find or define some new goals [LLb98]. For example in goal substitution, an alternative goal refinement procedure is selected for higher level goal in which obstructed goal and obstructing obstacle is no longer present similarly goal restoration strategy consist of adding new goal to make obstacle disappear.

3.2.1.5 Goal Elicitation through Constraints

In some situations new goals may also be identified from constraints operationalizing some goals. For example a constraint which states that "Before an employee can advance their certification level, they must take a course which officially qualifies them for achievement". This constraint helps us to identify a goal "courses which employee qualifies" because system must know which courses an employee can take [Ant96]. This is the case in which one may find requirements first and then identify goal from these requirements. Constraints also help to identify the situations where goal priorities may change e.g., consider a constraint which specifies that a meeting must be scheduled on a specific day and if no room is available or no one can attend meeting on that day then one have to re-examine goal priorities [Ant96].

3.2.2 Goal Refinement

Requirements completeness process goes beyond the stakeholders words to discover the goals driving the development process. Requirements describe the detail implementation plan for general goals. Goal refinement is intended to reduce the risk of incomplete requirements [AP98]. Goal refinement process deals with decomposing goal into subgoals until these goals can be assigned as responsibility of single agents [DLF93]. In literature number of refinement techniques has been purposed. The idea is to provide to provide complete and correct refinement [DL96]. The main approaches for goal refinement are:

- Agent driven decomposition: A goal is decomposed into subgoals that involve less number of agents. First a group of agents are identified which are involved in the achievement of parent goal. This group of agents is then divided into subgroups that can achieve corresponding subgoals according to their abilities or to some known schedule [Let01]. A catalog of agent based refinement tactics for refining unrealisable goal until realizable subgoals are achieved is purposed in [LL02].
- Case driven decomposition: A goal is decomposed by case analysis i.e., normal case or exceptional case.

- Time driven decomposition: A goal is decomposed into subgoals that need to be achieved successively over time. In this technique a state 'milestone' is identified and that state has to be reached in order to achieve target state.

Formal refinement patterns are useful for number of reasons [DL96] e.g., they allow formal reasoning to be hidden from Requirements engineers, they help in detecting of incomplete refinements and they allow choices underlying the refinements to be made explicit. Numbers of formal refinement patterns are purposed in literature which are proved correct once for all. These patterns are generic and can be instantiated to different situation. The pattern library should have following properties [DL96].

- Relevance: library should provide patterns that are actually needed by requirements engineers
- Retrievability: relevant patterns should be retrieved easily

3.2.3 Elaboration Method

Once goals are found from goal identification and goal refinement process next step is to identify agents, identify objects concerned by goals, identify actions describing state transitions, operationalization of actions and assigning responsibilities to agents. These steps may be running in parallel with possible backtracking at every step [LL00]:

- Identifying Objects
- Identifying Agents and Agents Assignments to Goals
- Identifying Operations and Operationalizations of Goals

3.2.3.1 Identifying Objects

During the goal identification and goal refinement process the objects concerned by these goals are also identified. The identification and characterization of objects from goals ensures that only those objects are identified which are relevant to goal [DLF93]. The identified objects and attributes are defined by relating them to real-world quantities they belong [Zav97]. An object may be classified as entity, relationship, event or agent based on whether the object is autonomous, subordinate, instantaneous or active. The objects characteristics are declared as attributes and relationships links to other objects [Let01]. Each object has name and definition; name is used to identify object while definition is usually a natural language statement. Object instances may

change over time e.g., a person may be an instance of 'student' object at one time but he may no longer be instance of that object in future. Objects are also not necessarily disjoint. An object instance may be member of several instances at same time [Zav97] e.g., a person can be instance of two different objects 'teacher' and 'student' at the same time.

3.2.3.2 Identifying Agents and Agents Assignments to Goals

Agents are active system components which have choices of behaviours to ensure goals they are assigned to. They may be classified as human agents, physical devices and programs etc. Each agent is responsible of performing some action. These actions are present in the capability list of agent. Basic preconditions and postconditions for that action are also specified in that capability list. The identification of agent is made together with their categories and by the actions they are capable of performing on objects [DLF93]. This identification of agent and assignment of agent to the actions also helps in determining terminating condition for goal refinement [Let01]: goal refinement stops when a responsibility can be assigned to single agent. Terminal goals assigned to agents in the software-to-be are known as requirements while goal assigned to agents in the environment of the software-to-be are known as assumptions or expectations.

Agents are normally identified by interaction between client and analyst but it is not necessarily that all agents will be identified at early stages of analysis. Some agents might be identified at later stages when operationalizing goals [DLF93]. Agents can be classified into hierarchies when a specific agent inherits the characteristics of general agent e.g., 'Professor' may be specific agent of more general 'Employee' class agent. Generic agents belong to domain specific knowledge base. Responsibility assignment of goals to agents is declared by responsibility links. A goal is assigned to an agent only if the agent has the sufficient capabilities to ensure that goal [LL02]. The meaning of responsibility assignment of an agent to goal is that the agent responsible for goal is the only one required to restrict its behaviour to ensure the goal [Fea87]. It requires that goal be operationalized by strengthening operations performed by that agent assigned for goal. The requirements assigned to an agent are defined in terms of quantities monitored and controlled by agent. By term 'Monitor' means that agent directly reads the value of the attribute and by term 'Control' means that agent can write the value of the attribute. In addition to monitor and control variables there are internal variables i.e., variables internal to that agent. An internal variable of an agent is an attribute that is controlled by that agent and monitorable by no other agent. The monitoring and control properties of an agent help in determining the responsibility assignment of an agent to the goal; a goal assignable to an agent must be defined in terms of variables

monitored and controlled by that agent [Fea87]. Some heuristics for assignment of goal to agent are purposed in [DFL91], although some of these heuristics conflicts with each other. These heuristics are:

- Each agent defines its private goals and wished goals. If possible none of the goals for which agent is responsible should be in conflict with its private goals. For this it is necessary to check for conflicts between system goals and agent's private goals.
- Minimize multiple responsibilities to avoid overloaded agents.
- Minimize number of agents to avoid coordination problems.
- Minimize over communication between agents.
- Three attributes motivation, ability and reliability are defined and if there are more candidate agents for an action, assign action to an agent that has high values for ability and reliability attributes.

3.2.3.3 Identifying Operations and Operationalizations of Goals

In this step, the operations relevant to goals are identified and then requirements on operations are derived so that goals may be satisfied. So, this step consists of two sub steps [Let01]:

- Identify operations: goals refer to state transitions; these state transitions are identified in this step. Domain pre- and post conditions of these state transitions are also identified.
- Derive requirements on operations: domain pre- and postconditions does not ensure the satisfaction of goals and therefore one need to identify strengthened pre-, trigger and postconditions on these operations so that goals can be satisfied.

After the terminal goals assigned to agents next step is to derive operational software specifications. Goals refer to specific state transitions for which operations causing these transitions are identified and domain pre- and postcondition capturing these state transitions are also identified [VL01]. To operationalize leaf goals constraints are introduced which are formal assertions to objects and actions available to agents. Transforming goals into constraints is not an easy task because there may be several operationalizations available for same goal and one have to decide about best operationalization. Constraints provide information about the requirements that must be met for a goal completion and they also provide insight into issues when goal priorities

changes [Ant96] e.g., consider a constraint which specifies that meeting must be held on specific day but for certain reasons meeting cannot took place on that day then goal priorities must be re-examined and in the result constraints help to identify new goals and new actions. Note that strengthening pre-, triggered and postconditions are defined in addition to domain pre- and postcondition. There is distinction between domain pre- and postcondition that captures application of operations in the application domain and required (strengthening) pre-, triggered and postcondition which capture requirements on operations that are necessary to achieve operations [LL02]. These required pre-, triggered, and postconditions produce requirements on the operations for corresponding goals to be achieved [VL01]. Assignment of an agent to constraints is represented by responsibility relationship as [DLF93]:

- Responsibility (ag, C) iff agent 'ag' is among the candidates to enforce constraint 'C' through some appropriate behavior prescribed by 'Ensuring links'.
- Ensuring relationship is defined as follow: Ensuring (act, C) iff the application of action 'act' under strengthened conditions Pre $\bigwedge$ StrengthenedPre, Trig $\bigwedge$ StrengthenedTrig, Post $\bigwedge$ StrengthenedPost guarantees that constraint 'C' holds in the initial and final states of 'act'. Ensuring (obj, C) iff the restriction of 'ob' states to the strengthened condition Inv $\bigwedge$ StrengthenedInv guarantees that constraint 'C' holds in the initial and final states of any action on 'ob'.

Constraints may be divided into two classes i.e., Hard Constraints and Soft constraints [DLF93]. Hard constraints may never be violated while soft constraints may be temporarily violated. Hard constraints may be safety and time critical constraints e.g., 'no planes on same portion of air corridor' is hard constraint. Since soft constraints may be temporarily violated therefore every soft constraint must have one restoration action with them.

3.3 Goal Classifications

Goals are discussed in several contexts in literature and therefore they are classified into different categories by different authors. Goal classification yields more focused set of questions which analysts selectively employ depending upon the nature of the proposed system [AP98]. This section briefly covers different kind of classifications found in literature.

3.3.1 Classification by Patterns

According to temporal behaviour the following patterns are identified:

- Achieve P $\longrightarrow \Diamond$ Q
- Maintain P $\longrightarrow \Box$ Q

The operator $\Diamond$ ensures the property Q will hold some time in future while $\Box$ ensures that property Q will hold always in future. In this classification "Achieve" shall lead to a system behaviour (referred by the property Q) in the future while "Maintain" restricts the possible system behaviour in the future [VL01].

3.3.2 Classification by Type

At the root of the hierarchy there are system goals and private goals [DFL91]. System goals are then further classified into subcategories of satisfaction goals, information goals, robustness goals, consistency goals safety and privacy goals. These are the specialized categories of goals which fall under the general categories of functional and non-functional goals [VL01]. For example satisfaction and information goals are classified as functional goals [DLF93] while accuracy goals are classified as non functional goals [MCN92].

3.3.3 Classification by Target Condition

Goals are also classified according to desired target condition. Two classes of goals are proposed; achievement goals and maintenance goals. Achievement goals are fulfilled when their target condition is achieved while maintenance goal is satisfied as long as its target condition remains true. Maintenance goals are usually high level goals with which associated achievement goals should comply [Ant96]. Achievement goals are usually extracted from process descriptions and therefore they represent operational strategies while maintenance goals are derived from organizational policies and therefore they represent organizational goals [Ant96].

3.3.4 Classification by Nature of Goals

According to nature of goals a distinction is made between hard goals and soft goals. A goal is classified as hard goal when its achievement criteria is sharply defined (e.g., buy a computer) [Don04] or whose satisfaction is established through verification techniques [DLF93]. Hard goals are related to functional requirements. They are true/satisfied or false/denied. A softgoal is one whose satisfaction cannot be done in clear cut sense and it is up to the goal originator to define when a goal is considered to be achieved (e.g., buy a fast computer). Softgoals are highly subjective in nature and mostly related to non-functional requirements. In literature the new term 'satisficing' has been introduced

for softgoals. Softgoals are represented by clouds while hard goals are represented by rounded rectangles.

Figure 3.2: Hard/Softgoal

3.3.5 Classification based of RE Activity

In [Kav02] author has identified four types of goals in relation to RE activities namely: current goals, change goals, future goals and evaluation goals. At requirements elicitation level one needs to understand current goals and the motivation for changing the current situations i.e., identifying the change goals. At requirements specification level focus will be on future goals and how these goals can be incorporated into system components i.e., relating to functional and non-functional components of the system. Finally at the requirements validation level focus is on evaluation goals.

3.4 Links in GORE

During whole GORE process from goal identification to goal refinement and to goal operationalization different kind of links are derived known as goal links. Goal links are used to relate goals with each other and with other elements of requirements model. Based on this definition goal links are divided into two main categories:

- Inter-goal links: these are used to relate goals with other goals
- Cross-goal links: these are used to relate goals to other elements of requirements model

Goal links identified in GORE are discussed in this section.

Refinement Links Refinement links are used to relate goal to subgoal and usually refinement links are represented in AND/OR graphs used in GORE. They may be classified into following two categories:

- AND-refinement links: they are used to relate a goal to set of subgoals. Satisfying all subgoals is necessary condition for the satisfaction of parent goal.
- OR-refinement links: they are used to relate a goal to an alternative set of refinements. Satisfaction of one subgoal is sufficient condition for the satisfaction of parent goal.

Contribution Links Usually the softgoal contribution is not in an absolute sense and contribution links are used to represent the softgoal contribution towards other goals (hard goals or other softgoals). A softgoal may partially contribute positively or negatively in 'satisficing' an other goal. In AND-decomposition positive contribution means if all subgoals are 'satisficed' then parent goal will 'satisfice' and negative contribution mean if subgoal(s) is satisfied parent goal will be denied.

Responsibility Links Relate goal(s) and there responsible agent(s). Responsibility links can be classified as outer level responsibility links and as instance declaration responsibility links [Let01]. The former are used to declare agent class responsible for the goal while instance level declaration specifies more precisely which instance of agent class is responsible for goal.

Conflict Links They are used to capture the situations in which satisfaction of one goal may prevent other to satisfy. Conflict links capture potential conflicts and as a result they are helpful in goal refinement and in finding alternatives.

Input/output Links They are used to relate operations to objects. Input and output links may also be used to declare which object attributes make the domain and co-domain of the operation.

Performance Links They are used to relate agent to operations. The agent assigned to the operation by performance link must be able to perform this specific operation.

Operationalization Links They relate goals to operations which ensure them through required pre-, post-, trigger conditions.

Wish Links Each agent is capable of performing some actions. These actions are defined in the capability list of the agent. The actions assigned to an agent must be in the capability list of the agent. The actions are defined in the capability list of an agent through wish links. Therefore one can say wish links are used in agent assignment. Normally a goal is assigned to an agent if that agent has wish for goal.

Monitoring and Control Links Agent interfaces are declared through monitoring and control links. In monitoring links agent directly monitor ('reads') the values of the object attributes while in control links agent directly control ('writes') the value of objects attribute.

Coverage Links Goal elaboration and scenario elaboration are intertwined processes. When a goal is identified a scenario can be authored for it and when a scenario is authored it can be analysed to yield goals [SMMM98]. Coverage links are used to relate goal(s) with scenario(s).

3.5 Benefits of GORE

GORE offers number of benefits over traditional RE approaches.

Wider Perspective GORE takes wider system engineering perspective as compared to traditional approaches . Goals are prescriptive statements that should hold in the system made of software-to-be and its environment; domain properties and expectations about the environment are explicitly captured during the requirements elaboration process, in addition to usual requirements specification [Lap05]. The relationship between systems and its environment are expressed in terms of goal based relationships [YM98].

Requirements Acquisitions Traditional modeling techniques help in modeling of requirements while GORE also helps in eliciting and refinement of requirements. Identified goals are refined and elaborated by asking 'why', 'how' and 'how-else' questions [YM98].

- 'Why' questions extracts abstract goals from specialized ones
- 'How' questions are used to identify offspring goals.

Obstacle analysis and conflict analysis are also used to identify new requirements. Goal along with scenarios are assumed to be main requirements elaboration ingredients [VL01].

Requirements Completeness Goals provide sufficient completeness criteria for requirements specification. Specification is complete with respect to set of goals if all the goals are proven to be achieved from the specification and from the known properties about the application domain to be considered [Yue87].

Requirements Clarification Goal oriented approach is an incremental approach in which goals are identified and clarified in an incremental way. Requirements analysis in terms of goals can be seen to tease out many level of requirements statements, each level addressing the demands of next level [YM98]. NFR framework is goal and process oriented approach used to clarify non-functional requirements.

Requirements Pertinence Goal models can be used to avoid irrelevant requirements; from goal models one can judge whether a particular goal contributes to some high-level stakeholder goal or not [Lap05]. A requirements is pertinence with respect to set of goals in the domain considered if its specification is used in the proof of one goal at least [Yue87].

Rational behind Requirements Instead of asking what the system needs to do GORE asks why certain functionality is needed and how it can be in implemented. Thus GORE provides a rational for system functionality by asking 'why' certain functionality is needed. A requirements appears in the specification because there exists some goal(s) which provide a base for it [DFL91], [VL01]. Requirements which does not contribute to a goal will not be considered therefore every requirements will have a rational for it. There are also different kinds of traceability links in goal refinement tree which help to find rational behind requirements.

Conflict Resolution Goals provide starting point for conflict and obstacle analysis. During requirements engineering process one may have to face different kinds of inconsistencies originating from elicitation of goals, from different requirements of each stakeholder, from different viewpoints and from different source documents. Various models and heuristics are proposed for resolving conflicts in [LLb98]. Meeting of one goal may interfere with other goals. These contributions among goals (positive or negative) can be modelled and managed and thus providing conflict resolution. One more advantage relating to this is because goals introduces the concept of early requirements analysis and therefore conflict analysis and divergence analysis is started at much earlier stage and thus providing more freedom to solve these conflicts and divergences [LLb98].

Traceability Goal refinement trees provide traceability links from high-level objectives to low level technical requirements to precise specification and to architectural design choices [Lam04]. Different kinds of goal links are also defined in goal models which relate goals to other elements requirements model. In addition to capturing positive or negative interaction between different goals these links are also used for

tractability. The hierarchical arrangement of questions ('why', 'how', 'how-else') is also helpful for traceability of requirements.

Robust Requirements Goals are also helpful in producing robust requirements through the introduction of obstacle and conflict analysis [Lam04].

Ideal communication In [Lam04], it is claimed that decision maker are more interested in well documented goal models than UML models providing an ideal communication interface between business managers and software engineers. Goal refinement offer right level of abstraction to involve decision makers for validating choices made among alternatives [Lap05]. In addition to alternative goal refinement models they were operationalizations, responsibility assignments.

Explorations of Alternatives In GORE there is great emphasis on alternative system proposals in which less or more functionality is automated. In GORE obstacle analysis, conflict analysis, alternative goal refinements and responsibility assignments help in finding alternatives which are missing from traditional approaches. Moreover during requirements elaboration process many alternatives are considered which may help to find some overlooked problems.

Capturing Variability The single goal model focuses on alternative goal refinements and alternative assignment of responsibilities. The quantitative and qualitative analysis of these alternatives helps to choose the best one. Therefore goal models help to capture the variability in problem domain [Lap05].

Better Documentation and Improved Readability Instead of starting from lower-level process or action oriented descriptions GORE gives importance to high level goals. Lower-level descriptions are then derived and documented from these high level goals (system-level and organizational objectives) therefore goal refinement provides a natural mechanism for structuring requirements documents and thus improving readability [VL01]. AND/OR structures are used to capture how goals are refined and abstracted. Overall goal structure maintains the division of responsibility among agents, ties specification components to informal text describing goal and it can be used to resolve conflicts [DFL91].

Requirements Management The higher level the goal is the more stable it is will be. Separating stable information from volatile is an important concern for require-

ments management and goals are much more stable than lower-level requirements or operations therefore they also help us in requirements management.

3.6 GORE Frameworks

After presenting the main goal oriented requirements engineering concepts are terms GORE approaches will be discussed. Although there are number of GORE approaches but only those are selected which are widely being used. Most of the other approaches are either extending these approaches or they are based on one of these approaches. Most important of them are Non-Functional Requirements framework(NFR) by Chung et al. [CCL99], i* framework by [Yu96], Goal Oriented Requirements Language (GRL) [GRL08] and Knowledge Acquisitions in automated Specification or Keep All Objects Satisfied (KAOS) by Lamsweerde [DLF93].

3.6.1 NFR framework

The non-functional requirements framework (NFR) presented by [CCL99] deals with modelling of quality requirements. It uses the concept of softgoals for quality requirements. Softgoals are goals which can not be fulfilled in their true scene. These are the goals without a clear definition and definite criteria for their fulfilment. But these are important as they can influence the design decisions. Because of their interdependencies and positive and negative influences on each other they are used for handling conflicts and trade-offs. The interdependency among softgoals is categorized according to influence on each other. They used qualitative terms to define the influences: strongly positive (++), weakly positive (+), weakly negative (-), strongly negative (–) and unknown (?). These influences are also named contribution types and are named make, help, hurt , break and unknown respectively. The softgoals and their inter-dependencies are represented in Softgoal Interdependency Graph (SIG). In SIG first, softgoals are established and then they are refined into subgoals by AND and OR decomposition. In process to enhance NFR framework some argumentation are proposed to enhance the SIG for domain specific knowledge. After decomposition the goals are fulfilled by operationalizing these goals. operationalization represented the solution(s) for softgoals. Figure 3.3 below represents elements of NFR framework.

For detailed information about NFR framework address the work of [CCL99].

Figure 3.3: NFR Elements

3.6.2 i* (i-star)

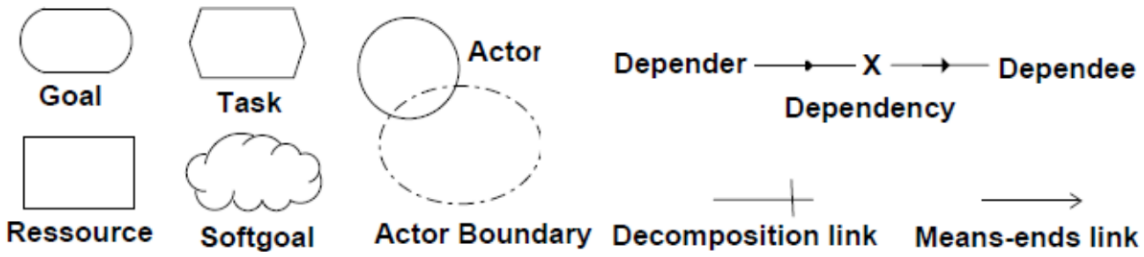
i* framework presented by [Yu96] is based on GORE concepts. It is similar to NFR framework used for early phase of system modelling. It was originally developed for modelling and reasoning about organizational context. i* is a combination of two modelling approaches: goal oriented modelling and agent oriented modelling. It support two types of modelling: Strategic Dependency model (SD) and Strategic Rational model (SR). It differs from other goal models like KAOS because it also represent the dependencies among the various actors in organizational context and it this actor dependencies are represented by SD model. SD model of nodes representing the actors and links connecting these actors. These links represent the dependencies between actors.

SR model is used to mode rational for each actor and their dependencies. There are four elements in the model to describe the dependencies: goal, softgoal, task, resource. These four elements are called intentional elements. SR model illustrate interdependency using different kinds of links: task decomposition links, mean-end links. When an actor is participating in a goal, softgoal, task or he requires a resource the task decomposition link is used. Mean-end link is used to describe why an actor would engage in a certain task. To represent the influences i* uses positive or negative contributions similar to NFR framework. Further the decomposition links and mean-end links are also similar to AND and OR decomposition of NFR framework. One advantage of i* over NFR framework is that it is not only useful to models system requirements but also helps to represent the organizational context. Of late i* framework has been extended (Tropos framework) to model the social context of the complex systems. This work is done by John Mylopoulos in the context of conceptual modelling and it uses the i* modelling framework. A number of other prototype tools are developed using the i* framework and Tropos. A few to mention are:

- TAOM4E (support tool for Tropos methodology)
- GR-Tool (Goal Reasoning tool)

- T-Tool (Formal Tropos tool)
- OpenOME (general purpose tool based on i*)
- SecTro (Automated modeling tool that provide supports for the Secure Tropos methodology)

Figure 3.4: i* Elements



The work of Eric S. Yu [Yu96]and John Mylopoulos [MCN92] gives a detail insight of i* framework and for further elaborated studies their work needs to be addressed.

3.6.3 Keep All Objects Satisfied (KAOS)

The KAOS - method another representative of goal-oriented RE goes back to the work of Lamsweerde et al. [LF91] and Dardenne et al. [DLF93]. KAOS stands for Knowledge Acquisition in autOmated Specification or Keep All Objects Satisfied. KAOS is described as a mulit-pradigm framework which provides various semi-formal and formal models at different levels of abstraction [Lap05]. Semi-formal models are used for modelling and structuring of goals, qualitative means are used for selection among the alternatives, and formal model are based on temporal operators are used for more accurate reasoning. It considers the system to be developed at two levels: an outer level also called graph semantic level with concepts, properties and relationships among the concepts and an inner formal level which is supported by temporal logic elements and is used for formally defining the concept (goal) [WPAOPL09].

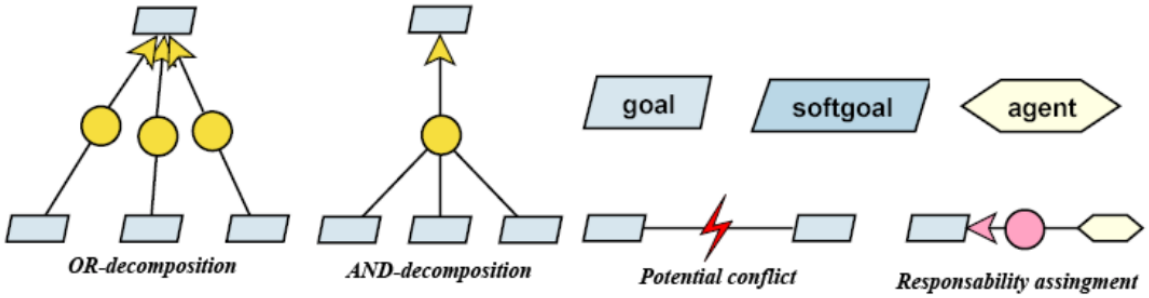
Goal(s) in the KAOS is defined as a ”prescriptive statement of intent about some system whose satisfaction in general requires the cooperation of some of the agents forming that system” [Lam04]. Goal are reached by certain conditions also called requisite. These conditions when operationalized into specification of software operations are known as requirements and called assumptions when they express the behaviours performed by external agents. Software agents performing operations necessary for

the fulfilment of certain requirements are the active components. Active (agents) and passive objects (entities, relations or events) are used to describe the structural model of the project. The dependencies of the agents with each other are represented by their interfaces made of objects, which are controlled by those agents. Obstacles and conflicting goals relations are used to integrate scenarios with KAOS.

In contrast to NFR framework's SIG model and i* framework's SD and SR model KAOS yields four kinds of models.

- Goal model is a set of goal graphs representing the goals in a top-down or bottom-up hierarchy. Goals are refined into subgoals by using the AND and OR relations. Other refinement pattern are also proposed by [Lam00a]. Subgoals describe how the overall goal can be achieved. Refining a subgoal ends when that subgoal may be associated with a single agent.
- Responsibility model represents the interfaces and responsibilities of agents that are placed on the respective agents through the assignment of requirements and expectations.
- Operational model represents the behaviours of the agents which are needed to cope with their responsibilities in the form of operations and tasks. With these operations and tasks associated objects are defined in the object model.
- Object model represents the formal specification of the objects and goals.

Figure 3.5: KAOS Elements



For details studies of KAOS framework book written by Lamsweerde [Lam09a] gives a details insight.

3.6.4 Goal Requirements Language (GRL)

The Goal-oriented Requirements Language (GRL) is another goal-based language used for goal and agent oriented modelling. Like NFR and i* framework the focus of GRL is on quality requirements. From 2008 GRL has been the part of User Requirements Notation (URN) approved as international standard. Since GRL is based on i* framework most of the elements (goal, softgoal, task, resources) are same with an additional element beliefs which is used to represent assumptions and relevant conditions (environmental). The relationship types: means-ends, decomposition, contribution and dependency are also with same meaning as in i* framework with just addition to correlation relationship which is used to describe the side effects of one elements to others. The tool support for GRL is provided by Organization Modelling Environment (OME). Figure 3.6 below represents GRL elements.

Table 3.3 gives the relevance of these frameworks for the work presented in this thesis.

Figure 3.6: GRL Elements

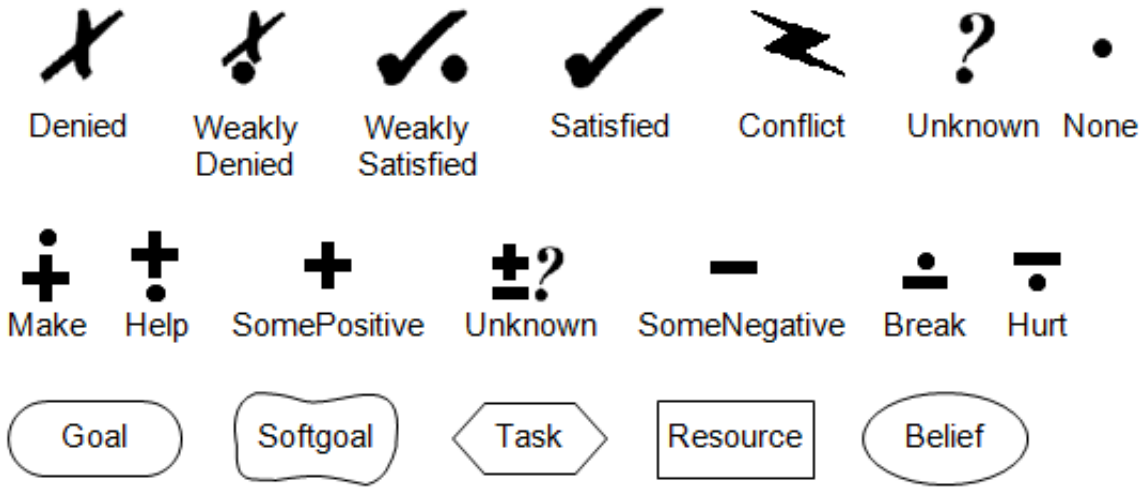


Table 3.3: GORE Frameworks and Their Relevance

Approach	**Relevance for Thesis**
NFR framework	Dependencies among the softgoals and their contribution links are useful for decision making. Alternative solutions are evaluated using the influences relations (also known as contribution links) and finally suitable solutions are selected. By using SIG, conflicts among the goals are detected and prioritization technique presented in this thesis is useful to resolve these conflicts. NFR framework provides a catalogue for certain quality goals. These catalogues can be used depending on particular situations where the same quality goals are being used.
i* (i-star)	The goal and softgoal concepts are used to represent the functional and non-functional (quality goals). The interdependencies among the goals are used in similar way to NFR framework by using the contribution links. The quality goal refinement is also done by same way as in NFR framework. But a prioritization technique is missing in i* framework which is necessary for conflict resolution among goals.
Keep All Objects Satisfied (KAOS)	KAOS model like other goal models refine the high level goals into subgoals. One advantage of KAOS is that it provides a catalogue, in addition to AND/OR refinement, for the refinement of goals. The catalogue consist of patterns and these patterns are used for refinement of goals. Among all the goal based frameworks discussed KAOS represents a very detail process for obstacle handling which is used for conflict management and risk analysis. These are used in comprehensive prioritization approach. KAOS provides a detail basis of task description as tasks are related to constraints and to an object which can be active (human or machine) or passive agent (event, entity or relation). KAOS does not provide explicit representation of non-functional requirements or quality goals which are used for design decisions at later stages to deal this issue, NFR framework or i* framework is used.
Goal Requirements Language (GRL)	GRL is based on i* framework therefore it uses the same concepts to model the quality requirements with just minor graphical notations differences.

3.7 Summary

This chapter discussed the main goal definitions and concepts of GORE. A complete goal based requirements analysis is presented and as the novelty of this analysis goal classification is discussed. Later the links present in goal models and benefits of GORE are figured out. In the last part of this chapter, GORE frameworks are evaluated and their use for this work is discussed.

Chapter 4

Decision Support in GORE

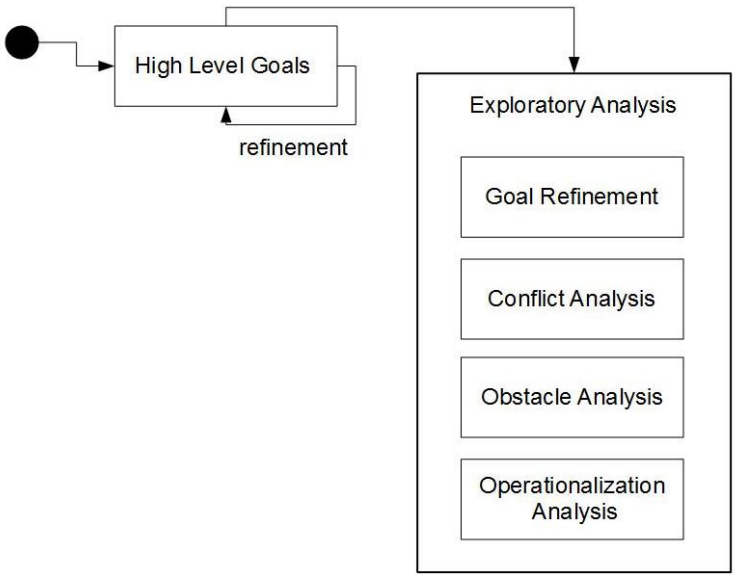
Requirements engineering start with high level customer problems or needs and move towards a detailed specification of these problems. One need to make various decisions in the requirements engineering process and the wrong or poor decisions here will lead to failures of software products or to products poorly fulfilling their functionality. Accordingly, there is a need of a decision making activity at the early stages of software development i.e., at the requirements engineering stage which can aid the discovery of trade-offs and to find alternatives. Decision making also needs an important consideration because it ranges from requirements elicitation to requirements negotiation and from requirements prioritization to requirements release planning. In this chapter the emphasis has been on the need of decision making in goal oriented requirements engineering and the mapping of decision making framework to GORE. It is discussed whether decision making needs to be introduced into one of the phases of GORE or to take this as a continuous activity which spans throughout all phases of goal oriented requirements engineering.

System development is a creative activity which requires iteratively twisting between problem space and solution space. It will be considered successful if the system meets its intended purpose and for this one need to have a thorough understanding of the system and user behaviour, the underlying technology and how the elements are going to interact with each other. The problem space is mainly focused on customer needs and problems and in the solution space the focus is on developing products, system architectures, standards and legacy systems [Leh05]. Based on this, requirements related to the problem space are considered to be external requirements (related to customer/user) and requirements related to the solution space are considered as internal requirements (related to solution and technical stakeholders). Goal oriented requirements engineering helps to capture and external requirements as well as internal requirements. In GORE there is a need to document the representation and justification of goal modelling choices i.e., why requirements engineers prefer one set of requirements over the others. The decision making activities that need to be incorporated into GORE might constitute strategic level decisions, management control decisions, operational control decisions [Reg01], [AW03]. The omission of decision making results in inconsistencies between requirements, weak traceability between expectations and their representations in goal diagrams, information loss on part of stakeholders modelling decisions [JFS08] etc.

4.1 Identifying Decision Points in GORE

In goal oriented requirements engineering (GORE) one start with initial high-level goals and keep refining them until the functional requirements satisfying (absolute fulfilment) or satisficing (partial fulfilment) these higher level goals are achieved. Because multiple stakeholders are involved in system development and these stakeholders might have different concerns, therefore, these initial goals contradict with each other and exploratory analysis needs to be undertaken to facilitate the discovery of trade-offs and to find alternative system proposals. During this analysis, there are situations where one can have various alternative options and there is need to select one option from many others e.g., in goal refinement many refinements are possible, in conflict analysis one have to choose among conflicting resolution options, during obstacle analysis different obstacle resolution techniques are available, during operationalization of a goal different operationalization options are available, similarly in responsibility assignment different assignments are possible etc.. In all these steps one have to decide about the best option according to one's needs. There are two options either to select the best option early in the analysis or support alternative options and let the stakeholders select the best option resulting in customizable solutions. Decisions have to be made

Figure 4.1: Goal Exploratory Analysis



among alternatives at various positions and these decisions need to be demonstrated and documented to make a requirements specification document more accurate and less deceptive [Lam09b]. The following question arises: *At which step should decision making be involved into the goal oriented requirements engineering process?* Furthermore, when a certain requirement is approved or disapproved there are a number of decisions that lead to approval or disapproval of this requirement. Usually only selected options are documented in requirement specification and discarded options are not documented. This information loss can cause problems when revising decisions and therefore there is need to have a good support for decision recording.

4.2 Importance of Decision support in GORE

There are many benefits of introducing the decision support in GORE:

- Decision support at goal level is useful to involve stakeholders early in requirements engineering phase. This is helpful in understanding the interaction between system and stakeholders (stakeholders involvement in making important decisions).
- It is easier for stakeholders to recall the reasons of their decisions if the rational for decisions is made explicit.
- If the rational is explicitly documented, it will result in better identification between requirements and other goal artifacts in the goal model.

- By explicitly documenting the rational of each decision, decision support in GORE is helpful for better change management capabilities.
- By documenting the stakeholder preferences early in requirements engineering the inconsistencies are pointed out by identifying the contradictions between stakeholders.

4.3 GORE and Decision Making Framework

Kontonya and Sommerville [KS98] consider decision making as an embedded activity in the requirements analysis and negotiation phase but in GORE, decision making is an activity which might appear in all phases of goal oriented requirements engineering. The approach adopted in [Reg01] is considered where decision making is considered as an evolutionary process that involves decisions which are continuous with iterations possible at each level. These decisions might include planning, objectives, resources, effective use of these resources and effectively performance of operations [Reg01]. Anthony [AW03] in his classic decision making model has distinguished decision making activity at three levels i.e. strategic decisions, management control and operational control. Accordingly, strategic level decisions are related to organizational goals while management control deals with decisions, which are related to identification and use of resources or in other words these describe management level goals or objectives. In these two stages it is determined whether a requirement is consistent with organizational product strategy and what resources are needed i.e. who is responsible for particular task and whether there is need of more effort for this task or not. Operational control concerns about the effective and efficient performance of the tasks. The qualitative assessment and dependency determination of requirements is carried out at operational control level.

At strategic level decisions might include the inclusion or removal of goals and therefore experienced and higher level staff is engaged in these decisions. The goal elaboration (goal analysis and goal refinement) technique is applied to refine goals into Subgoals until one get concrete level goals i.e. requirements and assumptions. Constraints, pre-, and post-conditions are also identified for the goals. Goals can also be classified depending upon the nature of proposed systems. At management control level possible ways of implementations of decisions made at strategic levels are explored. It is the stage where alternatives are analysed and assessed. Cost and benefit analyses of each alternative are also carried out here and then the best alternative is chosen for implementation. Decisions made at management control level include which development strategies to adopted, or what resources are needed etc., [AW03]. Requirement

priorities are also assigned here. These requirement priorities will help in the selection of alternative system proposals. To make sure that prioritisation produces accurate results, one need to involve all relevant stakeholders in the prioritisation process. Selection of stakeholders in prioritisation process depends upon the prioritisation criteria. There are different prioritisation criteria and selecting an appropriate criterion is again a decision making activity which requires the presence of relative stakeholders. For example, if a prioritisation criterion is usability then there is need to involve users or a group of users and if a prioritisation criterion is related to strategic importance of a requirement for the market segment then product managers should be involved in the prioritisation process. 'Importance', 'cost', 'damage', 'durability', 'risk', 'volatility' are some common criteria for requirement prioritisation [Poh10a]. The requirement prioritisation can be based on one criterion e.g., requirements can be prioritised regarding the importance for acceptance of the system or it is based on multiple criteria e.g., a specific requirement is prioritised according to the criteria of importance and development cost. In management control level the probable solutions and their implementation methodologies are obtained i.e., one move from problem space to solution space. This level complements goal operationalization, goal assignment to agents and alternative system proposals activities of GORE. Requirements implemented at the operational control level involve the decisions which are more structured as compared to the other two stages because the most preliminary analysis has been done in the earlier two stages. Mostly, decisions at operational control level relate to the quality and accuracy of the implementation of requirements. The proposed solutions are weighted and ranked to evaluate the individual alternatives. Decisions regarding product delivery and budget are also handled here. The overall mapping of decision making framework to basic GORE activities is shown in figure 4.2.

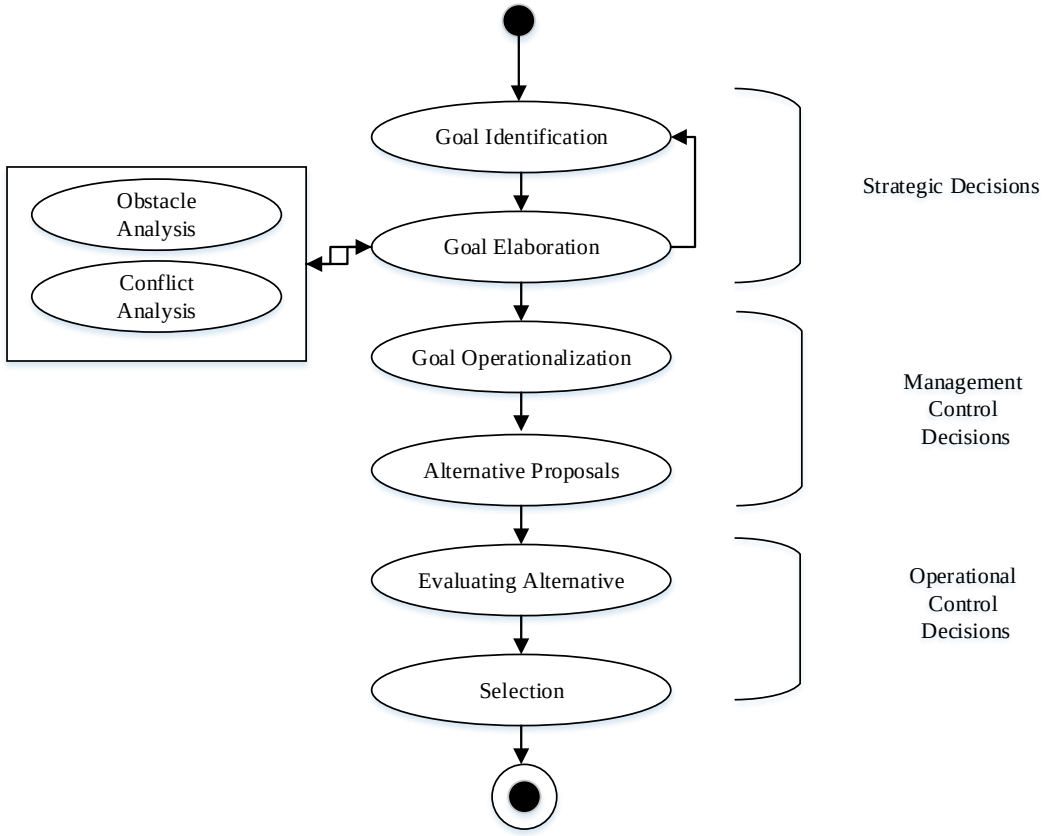
Based on the decision making framework three levels of requirements are as follow:

Organizational Level Requirements these include requirements related to business strategy, technology, marketing, benefits and profits.

Product Level Requirements are related to requirements for specific release, product architecture, resource management, implementation, change management.

Project Level Requirements are related to project planning, feasibility study, recruiting people, project management, quality control and validation.

Figure 4.2: Decision Making Framework and GORE



One also need to differentiate between decision making at individual level and decision making at organizational level i.e., group level decision making. For examples, individual level decisions might include developers taking decisions on their own about the implementation of a certain requirement. Similarly, individual level decisions might include management decisions about the imposition of certain requirement. Usually the individual level decision making at the developers site is not visible at the upper level i.e. requirement management groups etc. This also emphasizes on the need of incorporating individual choices in decision making models. In the requirements engineering process decision making should not be confused with problem analysis activities. In the problem analysis data is gathered and assessed while in decision making activity the gathered and processed data is used for decision making. Problems are identified and described in the problem analysis phase. These problems can be deviations from certain performance standards or they may be caused by some changes. At decision making stage the goals are stabilized and classified. Acceptance criterion is also defined for these goals. After that, possible solutions are explored for these goals. During this exploration, distinctive alternatives are uncovering . These alternatives are then

evaluated against the already established objectives. The alternative that is able to accomplish all objectives or most of the objectives fulfilling the acceptance criteria is chosen for implementation. Again, the selection of an alternative from various alternatives is the decision making activity which involves technical as well as non-technical factors. These factors might be objective or subjective depending on the decisions to be made, for example, in economic decisions financial factors are more important while in real time decisions security and timing factors are more important. Some of the important factors involved in decision making are: customers, organization process, organization objectives, financials, constraints, decision variables. The decision making activity along with its main steps that are important from a goal oriented requirements engineering perspective are represented in figure 4.3.

Figure 4.3: Decision Making Activities

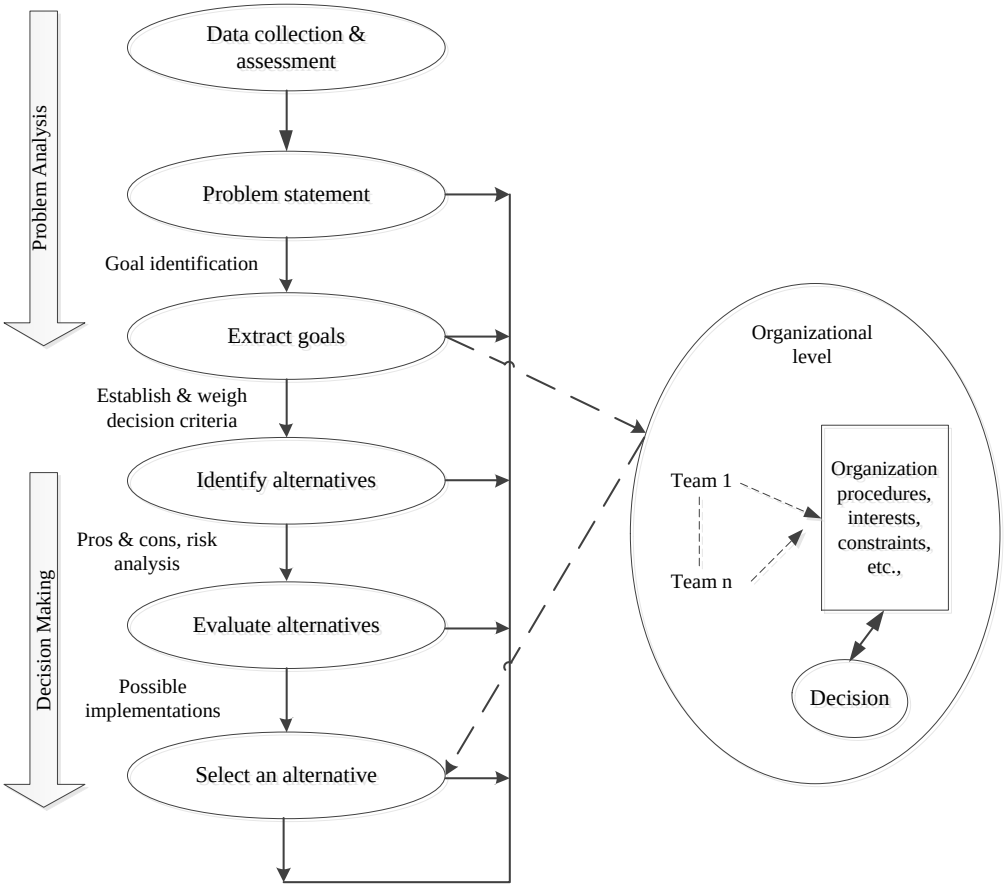


Table 4.1: Decision Influencing Factors

Sr	Organizational Level Factors	Technical Level Factors	Product Level Factors
1	Management	Hardware factors	Functional features
2	Domain knowledge	Domain specific factors	Non-functional features (quality factors)
3	Employee skills (analysis skilled employees, implementation skilled employees)	Software factors	Product constraints
4	Process and development environment	Architecture (product lines)	Maintenance
5	Schedule	Any particular standards	
6	Cost (hardware budget, software budget)		

4.4 Decision Influencing Factors

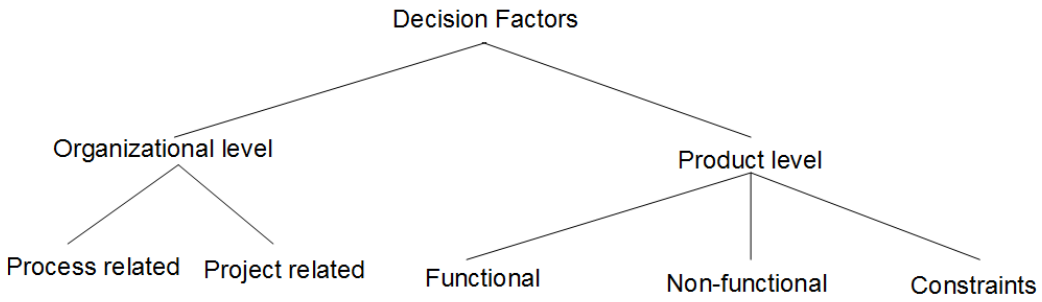
Several factors influence decision making. Understanding the factors that influence decision making process is important to understanding what decisions are made. That is, the factors that influence the process may impact the outcomes. This section categorizes the decision factors into three main categories:

- Organizational factors
- Technical factors
- Product factors

Table 4.1 gives influencing factors related to each of these categories.

Organizational factors are related to management decisions and they are of more concern at process and project level. They address project and process issues like cost, time or the maturity level the process has to fulfil. Requirements on specific development methods and techniques are also addressed here. For decision support at requirements level and in GORE the technical and product factors are of primary concern because of their importance at product level. Decision points in GORE are identified in goal graphs where functional and non-functional requirements influence the decision support as shown in figure 4.4. For the involvement of functional and non-functional requirements, a requirements taxonomy defined by [Jorsh] is used. This taxonomy was based on the results of a joint workshop of German-speaking NFR experts from industry and academia.

Figure 4.4: Decision Factors



As mentioned above at organizational level the decisions are more managerial ad product level decisions involve functional, non-functional requirements and product constraints. The requirement taxonomy classify functional requirements according to different domains. The business processes and interaction descriptions also known as operational scenarios are typical for information system domain. Whereas the behaviour descriptions and terms like stimuli and responses are representatives of embedded area. Neutral terms like functional descriptions and data items are also classified as functional requirements.

Table 4.2: Functional Requirements according to Domain

Domain	Requirements
Information systems	Business process requirements
	Interaction descriptions or operational scenarios
Embedded systems	Behavioral descriptions, response requirements, stimuli
Neutral	Functional descriptions, data items

Non-functional requirements are ones that restrict the solution space by constraining the qualities. Third category, product constraints, are usually known before the actual system development starts. They include constraints like:

- Architectural constraints
- Cultural constraints
- Operating environmental constraints
- Legal constraints
- Constraints imposed by physical laws

4.5 Non-functional Requirements for Decision Support

Non-functional requirements are considered from two perspectives:

- As requirements that describe the properties, characteristics or constraints of the system
- As requirements that describe the quality attributes that system must have

First type consist of business rules, external interfaces, development constraints and any other requirements that do not describe the functionality of the system. Quality attributes are properties of functional requirements that describe any other characteristic other than its functionality. An important part of quality attributes is that they should be measurable i.e., one or more metrics can be attached to the quality attribute e.g., response time, throughput time etc. There are three important characteristics of non-functional requirements that differentiate them from functional requirements:

- Functional requirements are related to specific functions while non-functional requirements are related to architecture and they might have affect on several functions. Due to this reason changes in non-functional requirements are more complicated.
- Functional requirements have hard criteria for their fulfilment i.e., when implemented they either work or do not work. While non-functional requirements might not be fully satisfied and they have a sliding scale of good or bad. For that reason they are difficult to test and usually are evaluated subjectively.
- Non-functional requirements might conflict with each other and therefore trade-offs are needed in these situations.

4.5.1 Identifying Terms of Non-functional Requirements

Non-functional requirements can be identified by specific terms like:

- -ilities: understandability, usability, modifiability, inter- operability, reliability, portability, maintainability, scalability, (re-)configurability, customizability, adaptability, variability, volatility, traceability, ...
- -ities: security, simplicity, clarity, ubiquity, integrity, modularity, ...
- -ness: userfriendliness, robustness, timeliness, responsiveness, correctness, completeness, conciseness, cohesiveness, ...

But this list is not exhaustive, there are also other keywords used to define non-functional requirements like: performance, efficiency, accuracy etc. [MZN10] have identified a total of 252 types of non-functional requirements. These 252 types consist of quality attributes (e.g., reliability, maintainable, performance), development constraints (e.g., time, cost), external interface requirements (e.g., user interface, human factors, look and feel, system interfacing), business rule (production life span) and others (cultural, political and environmental). Among these 252 types 114 types are specifically related to quality requirements. The top five most frequent types of quality requirements are: performance, reliability, usability, security, and maintainability.

4.5.2 Elicitation of Requirements

The requirements reside in scattered sources (stakeholder, text documents, requirements models etc.,) in different forms, e.g., as an idea, intentions or needs in the minds of stakeholders. For a successful requirements engineering process, all the relevant sources should be considered during requirements elicitation activity. The first goal of requirements engineering process is defined as "all relevant requirements shall be explicitly known and understood at the required level of detail." The general requirements engineering process was shown in figure 2.2. This process is decomposed into four major activities: requirements elicitation, requirements analysis, requirements specification and requirements validation. Each activity is defined as:

- Requirements elicitation: requirements elicitation consists of earliest activities in the requirements engineering process. The requirements are elicited from different sources (from customer, user, related documents) using different requirements elicitation techniques. The elicited requirements are known as customer requirements or user requirements.

- Requirements analysis and negotiation: customer requirements are analysed to discover problems especially problems related to inconsistent requirements (no requirements are contradictory), to identify the missing requirements (no needed services or constraints have been missed out) and to develop new and innovative requirements. The feasibility of requirements in the context of budget and schedule is also carried out at analysis phase. An important objective is to realize the relations among requirements and to find the requirements conflicts and overlaps. In case of conflicts the requirements are negotiated to find a compromise among the stakeholders.

- Requirements specification: requirements specification is about the representation of requirements that can be accessed for correctness, completeness and con-

sistency using natural language, graphical notations, mathematical notations or models.

- Requirements validation: the requirements documents are validated, before they are used as a basis for the system development, by customer and other concerned stakeholder to detect the deviations between the requirements documents and the stakeholder needs and wishes. During the validation activity, new requirements are developed and the process iterates until all requirements are validated and no more new requirements are elicited.

Eliciting clear, complete, and consistent requirements is a challenge and intricate task in requirements engineering due to number of reasons e.g., communication barriers that makes common understanding difficult. Most of the literature work has focused on the representation of requirements i.e., on requirements specification. The IEEE Guide to Software Requirements Specifications [IEE98] defines a good software requirements specification as being: unambiguous, complete, verifiable, consistent, modifiable, traceable, usable during operations and maintenance. In [Poh10b] the main goals of requirements engineering are characterized by three dimensions of requirements engineering which are:

- Content dimension: deals with understanding of requirements, all requirements should be known and understood in detail.
- Agreement dimension: deals with agreements among relevant stakeholders about known requirements.
- Documentation dimension: deals with documentation/specification of requirements in compliance with defined formats and rules.

Requirements elicitation process Interleaves the first two dimensions and therefore a good requirements elicitation process structures the foundation to build specification documents with desired attributes.

4.5.3 Requirements Elicitation Challenges

One of the main origin of project failures is the lack of due diligence at the requirements engineering phase. The study [PF] indicates 23.8 per cent projects were cancelled because of communication barriers between team members and end users, ambiguous requirements definition, and poor requirements management. Another study [Cha] shows that 90 per cent of system failures are tracked back to poor requirements elicitation. Problems of requirements engineering are grouped into three categories: problems of scope, problems of understanding, problems of volatility [Chr92].

- Problems of scope: problems of scope relate to determining the system boundary and the objectives of the target system. To little or too much information results in incomplete, ambiguous, not verifiable, and unnecessary requirements. These requirements do not reflect true user needs and they are not implementable under system constraints.

- Problems of understanding: problems of understanding occur because of user's poor or incomplete understanding of his needs, computer capabilities and limitations. Analysts do not have complete knowledge of domain. Communication barrier exist between user and analyst; both of them speak different language. There are conflicting and unspoken or assumed requirements from different stakeholders. These problems results in requirements which are often vague and un-testable.

- Problems of volatility: the requirements evolve over time and hence there are some requirements which are bound to change. The main reason of requirements change is that user needs evolve over time. Therefore, requirements engineering process should be iterative in nature to accommodate changes in the light of increased knowledge.

4.5.4 Requirements Elicitation Context

It is important to consider the context in which requirements are being elicited. Requirements elicitation process is followed in the following contexts: Organization, Environment, Project, Constraints imposed by people provide a good starting point for requirements elicitation.

- Organization context: Although requirements elicitation emphasis on the system's mission statements, but the overall organization context is often neglected. The requirements elicitation needs to understand the context of the organization where the system will be placed. The important factors of organizational context include: submitters of input, users of output, ways in which the new system will change the business process.

- Environmental context: Environmental context is necessary because the developing system must interface with the large system. Environmental constrains have a strong impact on requirements elicitation as for one type of applications; one may require methods and tools that are not necessary for other types of applications, for example, eliciting requirements for a real-time system will require different approaches than eliciting requirements for information systems. Important environmental factors include: hardware and software constraints, maturity

of the target system domain, certainty of the target system's interfaces to the larger system, the target system's role in the larger system.

- Project context: The project context also affects the requirements elicitation process. The factors of project context include: the attributes of the different stakeholder communities, such as the end users, sponsors, developers, and requirements analysts. Examples of such attributes are: management style, management hierarchy, domain experience, computer experience.

- Constrains imposed by the people: They are involved in the elicitation process, e.g., managerial constraints concerning cost, time, and desired quality in the target system.

4.5.5 Requirements Elicitation using Goals

According to Pohl [Poh10b] requirements elicitation in terms of tasks should facilitate:

- Identification of relevant requirements sources
- Eliciting existing requirements from identifies sources
- Developing new and innovative requirements

Identification of relevant requirements sources starts by exploring the gathered documents, ranging from information about the organization, (i.e., enterprise goals) to system specific information (i.e., requirements). For the identification of relevant requirements sources other approaches that complement goal based analysis are used. In [Poh10a] a two step procedure is proposed: in first step the relevant requirements sources are identified and in second step requirements sources are selected from those identified sources for eliciting and analysing requirements. The numbers of identified resources are restricted due to number of factors e.g., time, cost, availability of experts. In the first step techniques like interviews, workshops, or brainstorming sessions are used to identify relevant sources. The collected sources are added to the already identified sources. The process iterates until newly identified sources become sufficiently low or reaches to zero. For assessing the relevant sources a test named '100-dollar test' is proposed [LW00]. In this test each stakeholder is given 100 dollars to spend on the items and in the end the average amount of money spent on the items determines the relative weighting of that item. Now the requirements sources are prioritized depending on the amount spent on each item.

Table 4.3: Run Time and Development Time Qualities

Sr	Run Time Qualities	Development Time Qualities
1	Usability (ease-of-use, learnability, memorability, efficiency)	Localizability
2	Configurability and supportability	Extensibility
3	Correctness, reliability, availability	Evolvability
4	Performance (throughput, response time, transit delay, latency,etc.)	Composability
5	Safety (security, fault tolerance)	Reusability
6	Scalability	

After the identification of relevant sources, the next step is to elicit and analyse requirements. A preliminary analysis of current systems is important source for goal identification. Sometimes goals (high level) may be explicitly stated but most often they are implicit and an elicitation process needs to be undertaken to identify goals. The main sources for identifying goals are to look for intentional keywords in provided requirements sources like interviews, transcripts, mission statements, policy statements etc. A common approach is to find deficiencies that can be formulated, negate these deficiencies to produce first list of goals. Once high level goals are identified they are refined to elicit further goals until the system requirements are achieved. Scenarios, use cases and initial goal model are used to elicit system requirements.

Non-functional Requirements Elicitation from Functional Requirements Non-functional requirements emerge from functional requirements e.g., cash withdraw consists of quality and constraints: quality aspects represents the properties of the system that concerns stakeholders and these properties affect the degree of satisfaction of the system while constraints unlike qualities are not subject to negotiation, they are off-limits during design trade-offs. The qualities that are relative to users goals and judged by users are categorized as run-time qualities while qualities related to development organization's goals are categorized as development-time qualities [MB01]. Table 4.3 highlights the main run time and development time qualities.

Steps Involved Non-functional requirements elicitation from stakeholder goals consist of following steps:

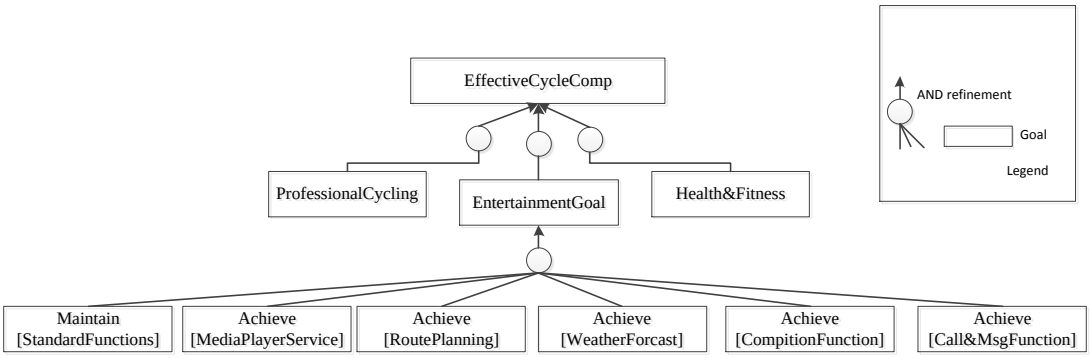
1. Identify stakeholders
2. Elicit goals for stakeholder
3. Establish softgoals (non-functional requirements) for each goal
4. Refine goals to subgoals
5. Identify softgoals for each subgoal

6. Refine softgoals to sub-softgoals

7. Establish links among goals and softgoals, subgoals to sub-softgoals

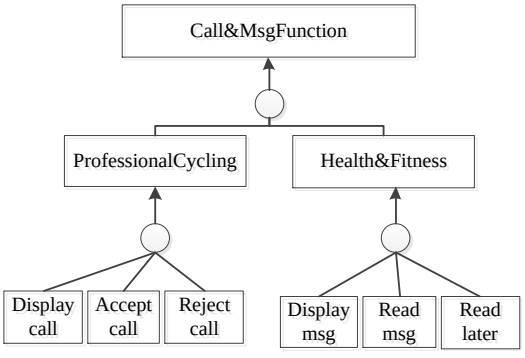
Cyclecomputer Example In cyclecomputer example user can have number of goals e.g., ProfessionalCycling, EntertainmentGoal, HealthAndFitnessGoal. EntrainmentGoal is selected for further analysis which is refined into number of subgoals: Maintain[StandardFunctions], Achieve[MediaPlayerService], Achieve[RoutePlanning], Achieve[WeatherForcast], Achieve[CompitionFunction], Achieve[Call&MsgFunction] Figure 4.5.

Figure 4.5: Cycle Computer Goals



Now lets take one subgoal from EntertainmentGoal i.e., Call&MsgFunction and refine that into further subgoals until requirements assignable to agents are achieved. The bold lines parallelograms in Figure 4.6 represents requirements and these requirements need a non-functional requirement i.e., usability.

Figure 4.6: Call&MsgFunction Subgoals



Functional Requirements from Non-functional Requirements Non-functional requirements at high level (whole system) lead to functional requirements at lower levels (subsystems). In that case non-functional requirements result in new functional requirements. A security requirement which is conventionally a non-functional requirement because it does not specify any specific functionality may lead to security subsystems to protect the system as a whole. For example a security requirement to prevent unauthorized access when refined results in functional requirement (user login): "The system shall include a user authorisation procedure where users must identify themselves using a login name and password. Only users who are authorised in this way may access the system data."

Cyclecomputer Example In cyclecomputer example there is one goal Achieve[SupportforTraining] which is a subgoal of a goal ProfessionalCycling 4.5. Achieve[SupportforTraining] is further refined into number of subgoals one of which is Maintain[CaloryConsumption]. Maintain[CaloryConsumption] requires security requirement (non-functional) for different users and therefore Maintain[UserLogin] is introduced in goal model.

4.6 Summary

This chapter discussed the importance of decision making at early stage of requirements engineering. After that the decision points in GORE are identified and the mapping of GORE as decision making framework is presented. The main factors that influence various decisions at requirements level are identified. In later part of this chapter the importance of non-functional requirements as decision factors is highlighted. The case study 'cyclecomputer' was presented as an example of requirements elicitation.

Chapter 5

Quality Models and Goal Models Integration

5.1 Quality Models Classifications

There are wide variety of concepts used for the classification of NFRs.

5.1.1 Boehm's Software Quality Tree [Boe76]

Boehm's quality tree was perhaps the earliest attempt to establish a conceptual framework of software quality. Boehm tree started by defining important software characteristic then metrics are defined to access the degree to which software has these characteristics. These metrics are then evaluated according to a number of criteria. In this way Boehm model presents three level approach. First level defining the highest level characteristic address three main questions:

- As-is utility: How well (easily, reliably, efficiently) it is used as-is?
- Maintainability: How easy is it to understand, modify and retest?
- Portability: Can I still use it if I change my environment?

Second level represents seven quality factors that represents the quality of the software system: These seven quality factors are:

- Portability (General utility characteristics): The characteristic portability defines the extent to which the system can be operated easily on configurations other than its current one

- Reliability (As-is utility characteristics): The characteristic reliability defines the extent to which the system can be expected to perform its intended functions satisfactorily
- Efficiency (As-is utility characteristics): The characteristic efficiency is about the efficient use of resources
- Usability (As-is utility characteristics, Human Engineering): the characteristic usability defines the extent to which the system is reliable, efficient and human-engineered
- Testability (Maintainability characteristics): Testability defines the extent that system facilitates the establishment of verification criteria and supports evaluation of its performance
- Understandability (Maintainability characteristics): Understandability defines that the system's purpose is clear to the inspector
- Flexibility (Maintainability characteristics, Modifiability): Defines the extent to which system facilitates the incorporation of changes, once the nature of the desired change has been determined.

Third level which is lowest level in Boehm's quality tree represents the metric hierarchy which given the foundation for defining the quality metrics. Figure 5.1 represent complete hierarchy between software characteristics from high level to low level. High level represents the uses of the software and low level is closely related to metrics that are used to perform the evaluations.

5.1.2 McCalls Quality Model (1977)

The McCall quality model has three major perspectives for defining and identifying the quality of a software product:

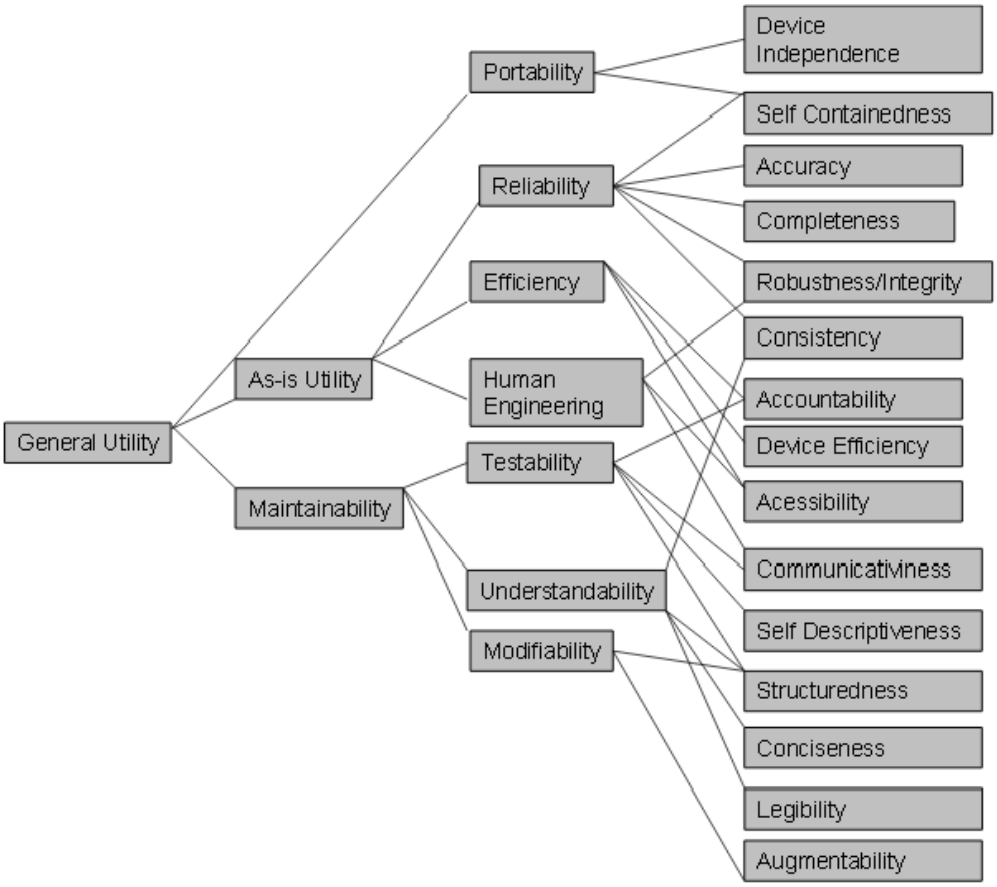
- Product revision: It defines the ability to undergo changes. It includes

  - maintainability: the effort required to locate and fix a fault in the program within its operating environment

  - flexibility: the ease of making changes required in the operating environment

  - testability: the ease of testing the program. The purpose is to ensure that it is error-free and meets its specification

Figure 5.1: Boehm's Software Quality Tree



- Product transition: Defines the product adaptability to new environments. Product transition is all about

  - portability: the effort required to transfer a program from one environment to another

  - reuseability: the ease of reusing software in a different context

  - interoperability: the effort required to couple the system to another system

- Product operations: Defines the operation characteristics. Quality of operations depends on

  - correctness: the extent to which a program fulfils its specification

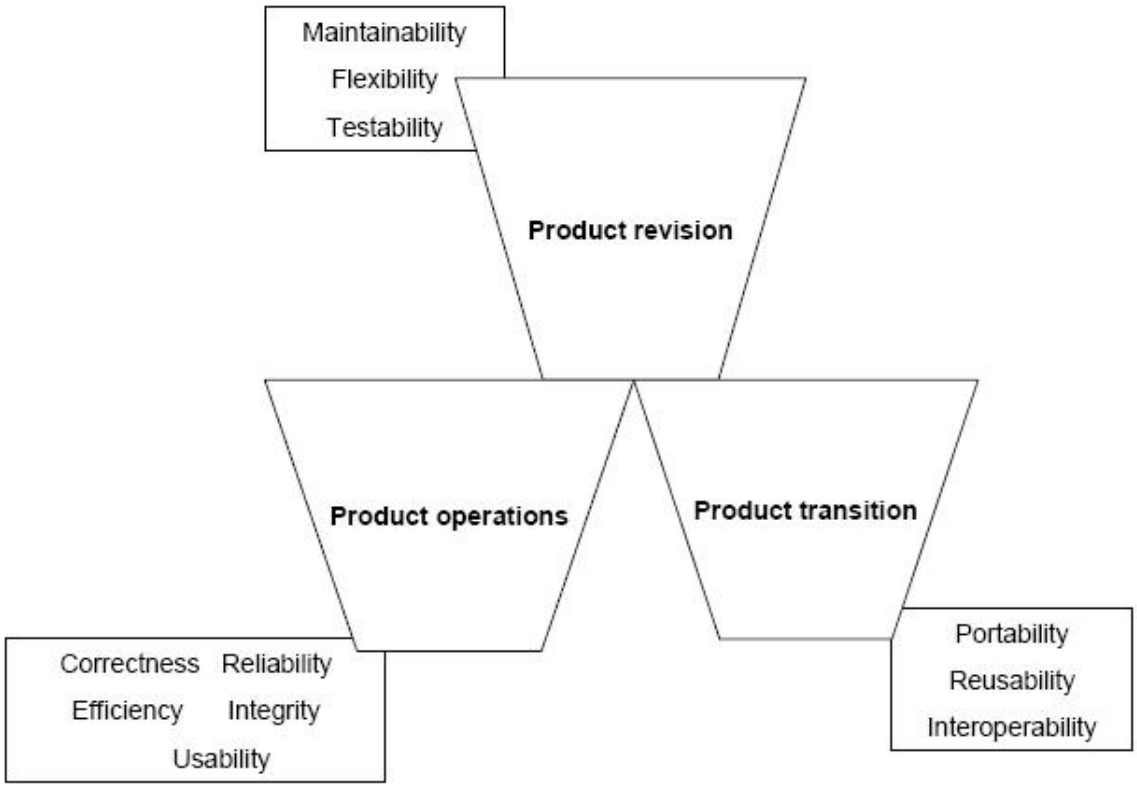
  - reliability: the systems ability not to fail

  - efficiency: deals with use of resources e.g., processor time, storage etc., It is further categorized into execution efficiency and storage efficiency

  integrity: the protection of the program from unauthorized access

  usability: the ease of the software

Figure 5.2: McCall's Quality Model



The model further describes 11 quality factors and 23 quality criteria. Quality factors are used to specify the external view of the product as viewed by the user. The quality criteria is used to describe the internal view of the product as as seen by the developer.

5.1.3 Romann Model [Rom85]

Roman defined NFRs as constraints and he divided NFRs into six types of constraints.

- Performance Constraints: They cover wide variety of concerns:

  - time/space such as response time, throughput, workloads, storage space etc.

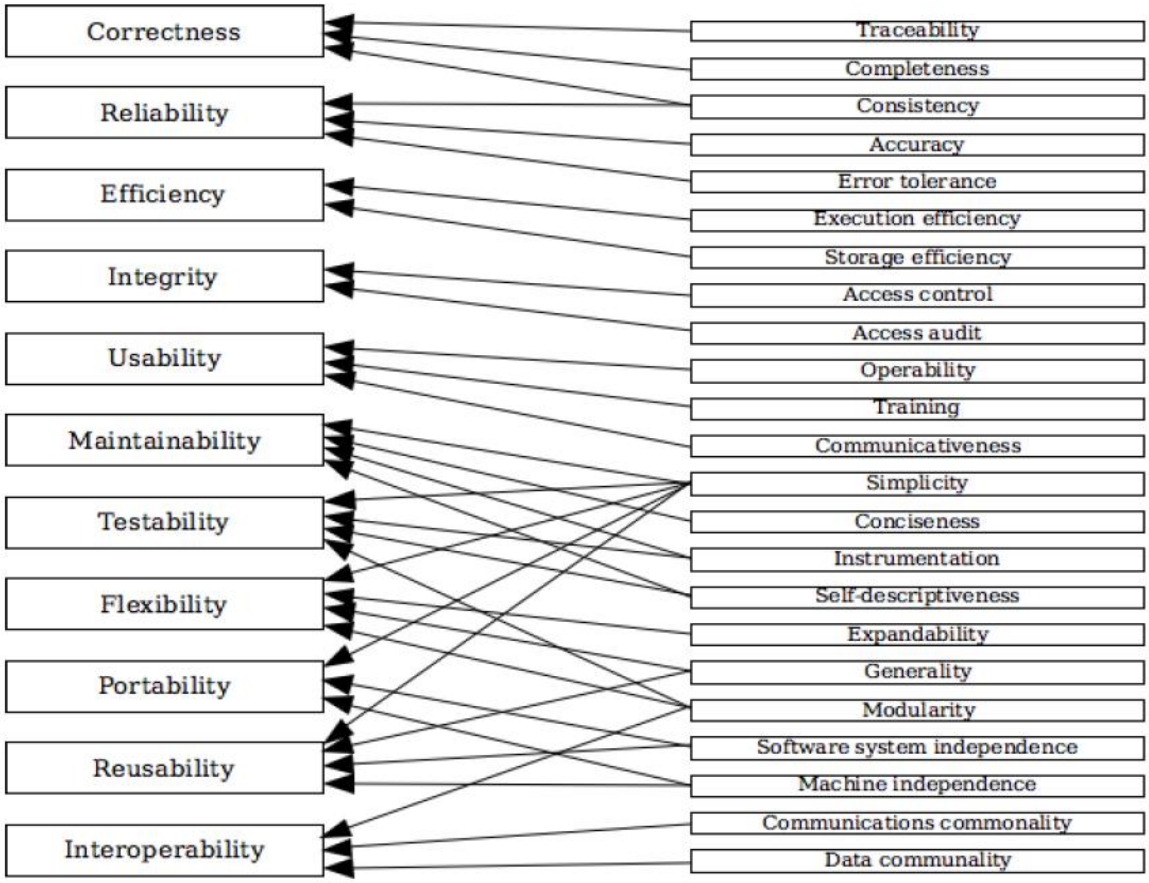
  - reliability deals with physical components and integrity of information maintained and supplied to the system

  - security constraints such as permissible information flow

  - survivability constraints such as related to defence system and system database prevent loss

- Interface Constraints: They define the ways the system interact with its environment, with users and other systems e.g., user friendliness

Figure 5.3: McCall's Quality Factors and Quality Criteria



- Operating constraints: They include physical constraints, personnel availability, skill level considerations, system accessibility for maintenance, etc.
- Life cycle constraints: They fall into two categories:

  - quality of the design which is measured in terms such as maintainability, enhanceability, portability

  – limits on development such as development time limitations, resource availability, methodological standards, etc.
- Economic Constraints: They define immediate and long terms costs. Thy may be limited to particular component and/or might consider the global marketing and production objectives
- Political constraints: They deal with policy and legal issues

5.1.4 Sommerville Model [Som95]

Ian Sommerville classified NFRs into product requirements, organizational requirements and external requirements.

- Product requirements: Requirements which specify that the delivered product must behave in a particular way e.g. execution speed, reliability, etc.
- Organizational requirements: Requirements which are a consequence of organisational policies and procedures e.g., process standards used, implementation requirements, etc.
- External Requirements: Requirements which arise from factors which are external to the system and its development process e.g., interoperability requirements, legislative requirements, etc.

Figure 5.4: Sommerville Classification of NFRs

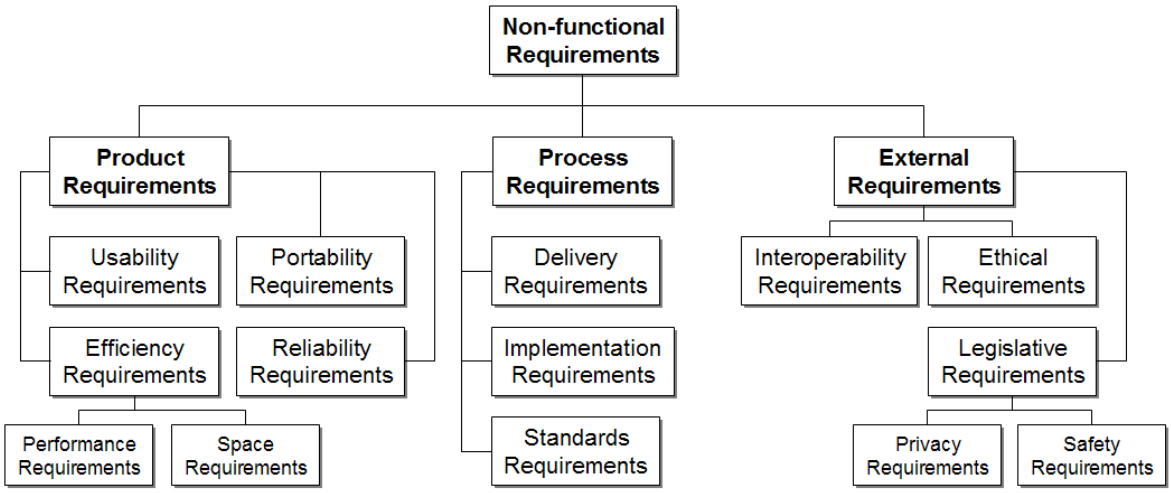


Figure 5.4 represents Sommerville [Som95] classification of NFRs.

5.1.5 Dromey's Quality Model [Dro95]

Dromey proposed a product based quality model. Dromey model argues that quality evaluation differ for each product. The focus in this model is on the relationships between the quality attributes and sub-attributes and it attempts to connect product properties with quality attributes. In this quality model, there are three main elements:

- Product properties: They influence the quality
- Quality attributes

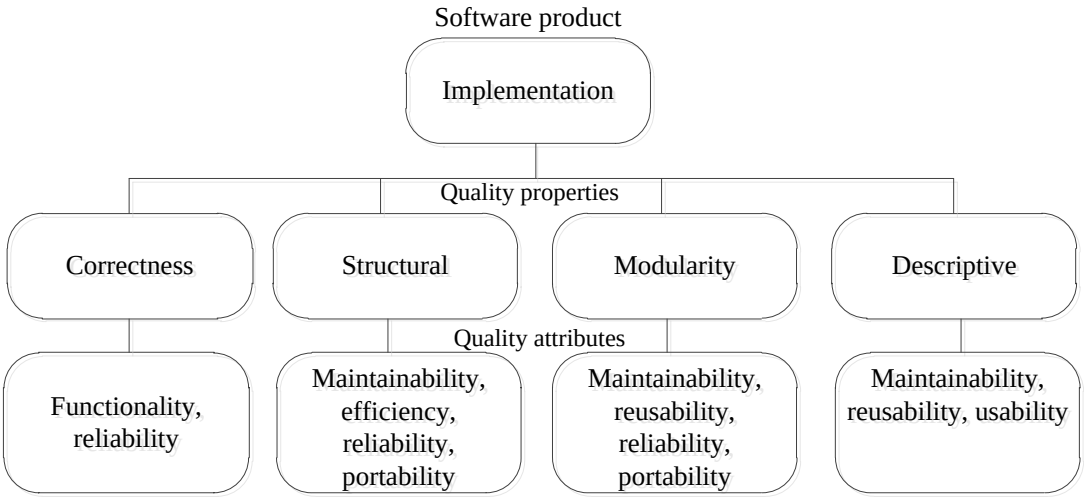
- Links: linking between product properties and quality attributes

Quality carrying properties are structured into four basic categories and then quality attributes are identified against each quality property:

- Correctness properties: represents minimal generic requirements for correctness
  - Computable: obeys law of arithmetic etc.
  - Complete: all elements of structural form are satisfied
  - Assigned: variable given value before use
  - Precise: accuracy preserved in computations
  - Initialized: Assignment to loop variables establish invariant
  - Progressive: each iteration decreases variant function
  - Variant: loop guard derivable from variant function
  - Consistent: no improper use or side effects
- Structural properties: deals with low level intermodule design issue
  - Structured: single entry/single exit
  - Resolved: data structure/control structure matching
  - Homogeneous: only conjunctive invariants for loop
  - Effective: no computational redundancy
  - Non-redundant: no logical redundancy
  - Direct: problem specific representation
  - Adjustable: parametrized
  - Range independent: applies to variables (arrays), types, loops
  - Utilized: to handle representational redundancy
- Modularity properties: deals with high level intermodule design issues
  - Parametrized: all inputs accessed via a parameter list
  - Loosely coupled: data coupled
  - Encapsulated: uses no global variables
  - Cohesive: the relationships between the elements of an entity are maximized
  - Generic: is independent of the type of its inputs and outputs
  - Abstract: sufficiently abstract

- Descriptive properties: deals with various form of specification/documentation properties
    - Specified: preconditions and postconditions are provided
    - Documented: comments are associated with all blocks
    - Self descriptive: Identifiers have meaningful names.

Figure 5.5: Dromey's Product Quality Model



5.1.6 FURPS/FURPS+ [Gra92]

This model was introduced by Robert Grady. The name comes from characteristics **F**unctionality, **U**sability, **R**eliability, **P**erformance, **S**upportability (FURPS). This classification addresses both functional and non-functional requirements as represented by letter F which stands for functionality. FURPS+ is used to represent constraints such as "the system will use 'xyz' database."

- Functionality: functional requirements are defined by inputs and expected outputs and may include feature sets, capabilities
- Usability: usability includes eliciting and stating requirements regarding user interface issues. They include human factor, accessibility, interface aesthetics, user documentation, training and consistency within the user interface
- Reliability: reliability includes availability, accuracy, and recoverability, predictability

- Performance: constraints on functional requirements such as speed, efficiency, throughput, response time, recovery time and resource usage
- Supportability: supportability includes number of other requirements like testability, adaptability, maintainability, compatibility, configurability, installability, scalability, localizability etc.

FURPS+ is used to specify design, implementation, interface and physical constraints.

- Design constraints: define constraints on design i.e., the approach one take in developing the system.
- Implementation constraints: constraints on coding e.g., platform, implementation language etc.,
- Interface constraints: constraints on external systems interactions.
- Physical constraints: constraints regarding shape, size, weight etc.

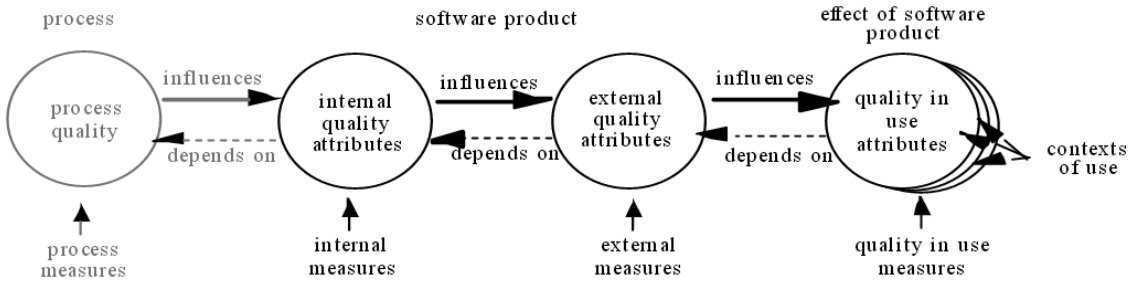
5.1.7 ISO 9126 Model [Sta04]

The ISO 9126 is an international standard for software quality evaluation and it is based on the McCall and Boehm models [Boe76]. This model represents the quality from three aspects [Sta04]:

- Internal quality: is about the design of the software i.e., it is the implementation. Falling to fulfil internal quality means, the system will be less responsive to changes in future. The internal quality characteristics are: Maintainability, Flexibility, Portability, Re-usability, Readability, Testability and Understandability.
- External quality: determines the fulfilment of stakeholders requirements i.e., is the system providing the required functionality? Is the interface clear and consistent? Its obvious measures are functional test and measures of the bugs of the product. The external quality characteristics are: Correctness, Usability, Efficiency, Reliability, Integrity, Adaptability, Accuracy, and Robustness.
- Quality in use: is the user view of the quality and depend upon achieving the external quality. internal quality influences the external quality which influences the quality in use. Quality in use is categorized into four characteristics: effectiveness, productivity, safety, satisfaction .

This model presents six top level quality characteristics (internal and external) which are further refined into twenty one sub-characteristics at the lower level.

Figure 5.6: ISO Quality Model in the Product life cycle [WS03]



- Functionality: provides the required functionality which meet stated and implied needs when the software is used under specified conditions.

  - Suitability: provide specific set of functions for specific tasks and user objectives

  - Accuracy: provide the agreed results with needed degree of precision i.e., correctness of functions.

  - Interoperability: The capability to interact with one or more specified components or systems i.e., the system does not work in isolation.

  - Security: protecting the information and data from unauthorized access and granting access to authorized persons.

  - Compliance: Adhering to standards, conventions or regulations in law whether industry or government.

- Reliability: maintaining a specified level of performance under specified conditions.

  - Maturity: concerns frequency of failures of the software.

  - Fault tolerance: Maintaining a specified level of performance in case of software faults.

  - Recoverability: re-establishing a specified level of performance and recovering the system to operational state (data and network connections) after failure.

  - Availability: the capability to be able to perform a required function at a given point in time under stated condition of use. It is a combination of maturity, fault tolerance and recoverability.

- Usability: the capability of the product to be understood, learned and used. Usability also addresses the environment of the product.

  - Understandability: ease with which the product functions are understood.

- Learnability: enabling the user to its applications.

- Operability: enabling the user to operate and control the product i.e., easily operated by a given user in a given environment.

- Attractiveness: to be attractive to the user

- Efficiency: capability to provide appropriate performance relative to the amount of the resources used under stated conditions.

  - Time behaviour: characterizes the response time, processing time for a given through put.

  - Resource behaviour: Characterizes the appropriate amount and types of resources used, i.e., memory, CPU, disk, network usage etc.

- Maintainability: capability of the software product to be corrected, improved and/or adapted to the changes in environment, requirements and specifications.

  - Analyzability: ability to identify the root causes of failures.

  - Changeability: amount of effort to implement the change.

  - Stability: avoiding unexpected effects of the changes.

  - Testability: efforts needed to test a system change.

- Portability: the capability of the product to be transferred from one environment to another.

  - Adaptability: ability to change to new specifications or operating environments.

  - Installability: effort required to install the product.

  - Co-existence: ability to co-exist with other independent software in same environment sharing common resources.

  - Replaceability: how easy is it to exchange a given component within a specified environment.

  - Conformance: adhering to standards or conventions relating to portability.

5.1.8 Comparison of Quality Models

Most of the quality models consist of layers. The number of layers are two (characteristics, sub-characteristics) or three; third layers usually consisting of metrics. Figure 5.8 gives a comparative analysis of quality model's characteristics.

Figure 5.7: ISO Quality Model Internal and External Quality Characteristics

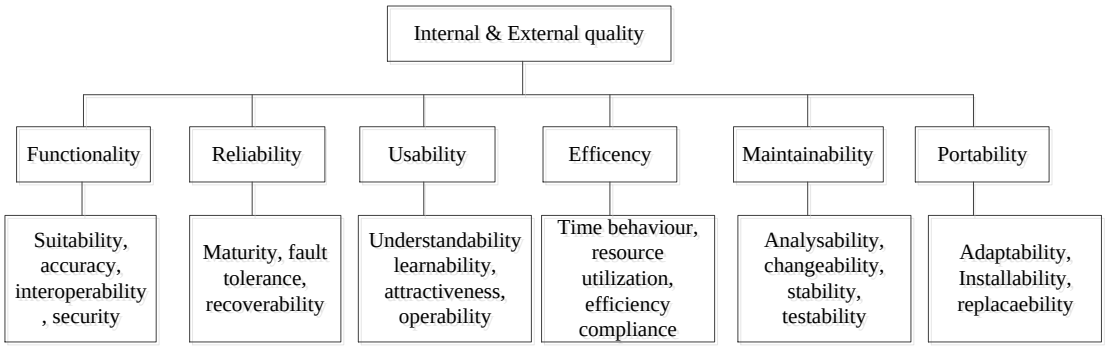


Figure 5.8: Comparison of Quality Models

Factors/Attributes/ Characteristics	Boehm's Model	McCall's Model	Romann Model	Sommerville Model	Dromey's Model	FURPS/ FURPS+	ISO9126 Model
Maintainability	*	*	*		*		*
Flexibility	*	*	*				
Testability	*	*		*			maintainability
Correctness	*	*	*				maintainability
Efficiency	*	*	*	*	*		*
Reliability	*	*	*	*	*	*	*
Integrity	*	*	*		*	*	*
Usability	*	*	*	*		*	
Portability	*	*	*	*	*		*
Reusability	*	*	*		*		
Interoperability		*	*	*		*	
Human Engineering	*		*	*		*	
Understandability	*						
Modifiability	*						maintainability
Functionality			*	*	*	*	*
Performance			*	*		*	
Supportability			*			*	
Clarity	*						
Documentation	*		*	*		*	
Resilience	*						
Validity	*						maintainability
Generality	*		*	*		*	
Economy	*		*				

5.2 Goal Model and Quality Model Integration

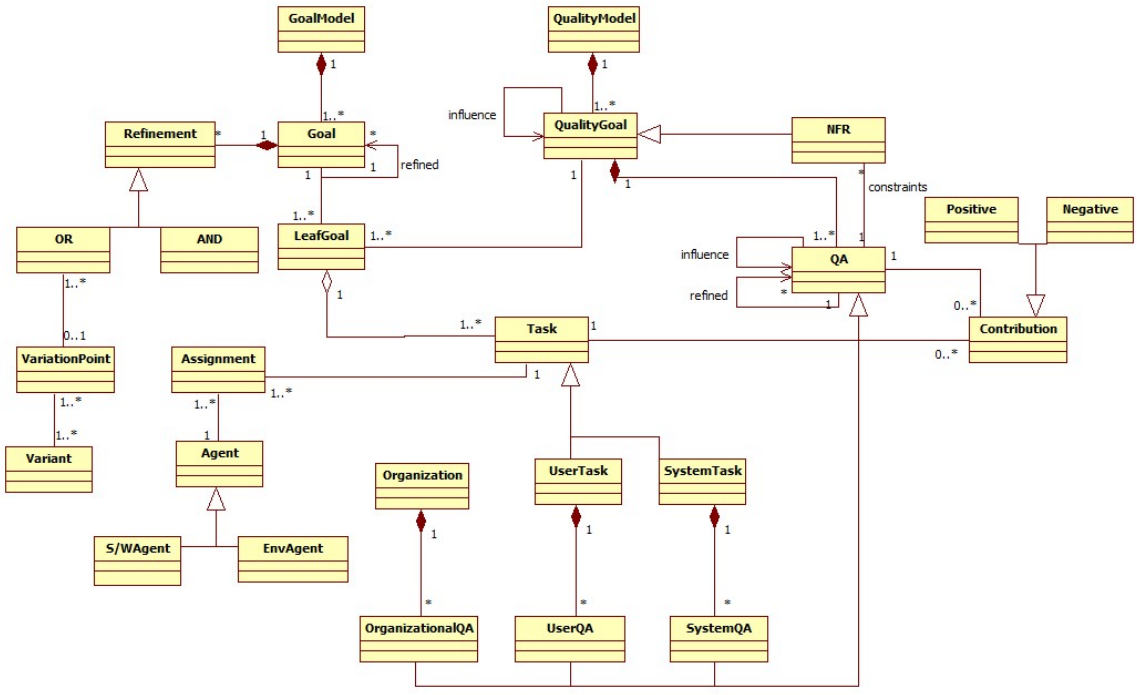
Mostly the quality models will not fit perfectly for the developing system [KLM11] and therefore the adaptation of these quality models for specific project is required. This section focused on the integration of goal models and quality models that helps in the derivation of customized quality models. In chapter IV, the decision influencing factors are identified. The adaptation of quality models is based on those organizational specific factors. The general tailoring process consist of three steps:

1. Specifying the goal: The process begins with specifying the higher level goal which defines the needs of the project or organization.

2. Specifying quality aspect: The quality related aspects belonging to identified goal are specified. For that, quality models are used. The quality model used to identify the quality aspects is called the reference model. In figure 5.8 all widely used aspects are identified. So, instead of using one particular quality model which may lack quality aspects present in other models this is used as reference model.

3. Tailoring the model: Once all quality aspects are chosen, the ones that are not needed in the final analysis are discarded.

Figure 5.9: Integrated Meta Model



The defined meta-model 5.9 is used to describe quality models in use, integrate the relevant attributes that are specific to stakeholder goals. The meta concepts GoalModel and QualityModel are central to overall meta model. A single goal have OR or AND refinements until the LeafGoal is achieved. LeafGoal can be the Task assignable to Agent or it may be a Quality Attribute(QA) derived from QualityGoal. QA influence other QA and it can also contribute to Task in positive or negative manner. Agents are of two types SoftwareAgent, EnvironmentAgent. Task is generalized form of UserTask and SystemTask having related User QA (UserQA) and System QA (SystemQA). For organizational specific QA, OrganizationalQA meta concept is defined. Each OR refinement may have variation points. Meta concept VariationPoint explicitly define the variability in goal model. VariationPoint represents the variation subject while Variant define concrete type of variation.

5.3 summary

This chapter discussed different quality models. A classification of quality models from various authors is presented. In comparison of quality models, the quality factors from all these models are presented. In the last an integration of quality models and goal models is discussed and an integrated meta model is presented as an output of that integration.

Chapter 6

Prioritization and Selection of Requirements: Three Tier Approach

Software quality requirements are essential part for the success of software development. Defined and guaranteed quality in software development requires identifying, refining, and predicting quality properties by appropriate means. Goal models of goal oriented requirements engineering (GORE) and quality models are useful for modelling of functional goals as well as for quality goals. In previous chapter, a goal based tailoring process for quality models is defined. Once the goal model representing the functional requirements and integrated quality goals are obtained, there is need to evaluate each functional requirement arising from functional goals and quality requirement arising from quality goals. The process consist of two main parts. In first part, the goal models are used to evaluate functional goals. The leaf level goals are used to establish the evaluation criteria. Stakeholders are also involved to contribute their opinions about the importance of each goal (functional and/or quality goal). Stakeholder opinions are then converted into quantifiable numbers using triangle fuzzy numbers (TFN). After applying the defuzzification process on TFN, the scores (weights) are obtained for each goal. In second part specific quality goals are identified, refined/tailored based on existing quality models and their evaluation is performed similarly using TFN and by applying defuzzification process. The two step process helps to evaluate each goal based on stakeholder opinions and to evaluate the impact of quality requirements. It also helps to evaluate the relationships among functional goals and quality goals.

The distinct purpose of software development is to satisfy various stakeholders needs [KR97]. There are multiple stakeholders involved in the system development and these stakeholders might have different concerns/opinions about the goals to be achieved by the system. Requirements engineering must provide a way to understand stakeholders needs so that high quality software systems are developed. Although stakeholders needs are placed at the most important place, their classification is regarded as the most difficult task. Each stakeholder might have different requirements and sometimes these requirements are of contradicting nature. Therefore, satisfying these requirements is a challenging task [Ito07]. The goal models of goal oriented requirements engineering(GORE) are used to identify and refine the high level goals. Finding the criteria based on GORE require high level goals to be analysed till leaf goals are achieved, that is, until operational requirements are achieved. These leaf level goals are used as criteria for the established high level goals.

There are multiple criteria in one goal model and each criterion may have different importance for various perspective stakeholders, that is, some criteria are more important than others [EK08]. Stakeholders opinions and preferences should be involved in the process to find the relative importance of each criterion. Normally, there is uncertainty and vagueness about selected criteria because of contradicting stakeholder interests and to find relative importance of criteria according to different stakeholders, multi-criteria analysis (MCA) is performed. These kind of problems are known as Multi-criteria problems and in general fuzzy set theory is adequate to deal with these problems [Che00].

One essential output of GORE is goal models. Goal model is a set of goal graphs representing the goals in a top-down or bottom-up hierarchy. Goals are refined into subgoals by using the AND/OR relations. In [Lam00b] a catalogue of refinement patterns is proposed. Subgoals describe how the overall goal is achieved. Refinement of a subgoal ends when that subgoal may be associated with a single agent. Most important GORE work includes Non-Functional Requirements framework(NFR) [CCL99], i* framework [Yu96], Goal Oriented Requirements Language (GRL) [GRL08] and Knowledge Acquisitions in automated Specification or Keep All Objects Satisfied (KAOS) [DLF93]. Functional goals are achieved by operationalization of them either by the system or by external actor while quality goals capture system qualities. The non-functional requirements framework (NFR) [CCL99] deals with the modelling of quality aspects.

GORE frameworks used the concept of softgoals for quality requirements. Softgoals are goals which can not be fulfilled in their true scene. These are the goals

without a clear definition and definite criteria for their fulfilment. Because of their interdependencies and positive/negative influences on each other they are used for handling conflicts and for making trade-offs. Dependencies among the softgoals and their contribution links are useful for the determination of quality goals impact on functional goals [CCL99]. Non-functional requirements are considered from two perspectives [Ant96]:

1. As requirements that describe the properties, characteristics or constraints of the system

2. As requirements that describe the quality attributes the system must have

First type consist of business rules, external interfaces, development constraints and any other requirements that do not describe the functionality of the system. Quality attributes are properties of functional requirements that describe characteristic other than its functionality. An important part of quality attributes is that they should be measurable i.e., one or more metrics can be attached to the quality attribute e.g., response time, throughput time etc. Quality aspects represent the properties of the system that concern stakeholders and these properties affect the degree of satisfaction of the system while constraints unlike qualities are not subject to negotiation, they are off-limits during design trade-offs. [Fra98] argue that quality requirements serve as basis for non-functional requirements in quality models. Quality models used for specifying non-functional requirements provide a hierarchical list of quality attributes also called quality aspects or quality factors. Although these quality model give a systematic structure to quality requirements, they are not consistent with each other [LHM+14], for example, understandability is a sub-quality of usability in IS0 9126 [Sta04], but is a sub-class of maintainability in Bohem's model [Boe76]. A comparison of quality models [Boe76], [RS79], [Rom85], [Som95], [Dro95], [Gra92], [Sta04] is presented in figure 5.8.

6.1 Fuzzy Numbers

The functional goals and quality goals help to identify the criteria for the acceptance of target system. There are requirements derived from goal models and quality models which are imprecise in nature. In literature, fuzzy numbers are very popular in engineering disciplines for their ability to represent imprecise and vague information. By using fuzzy sets, requirements are described using linguistic terms. These linguistic terms are then converted into formal representation by using membership functions described for fuzzy numbers [YT97]. Membership function is the set of real numbers

(R) whose range is the span of positive numbers in the closed interval [0,1], where '0' represents the smallest possible value of the membership function, while '1' is the largest possible value [LW92].

Fuzzy numbers depict the physical world more realistically than single-valued numbers. Among the fuzzy number Triangular Fuzzy Number (TFN) is capable of aggregating the subjective opinions [MD14]. A triangular fuzzy number (TFN) is described by a triplet (L, M, H), where M is the modal value, L and H are the left (minimum value) and right (maximum value) boundary respectively. TFN is used to represent stakeholder opinions for functional goals and quality goals which are established through goal models and quality models. Fuzziness of TFN is (L, M, H) is defined by the equation 6.1:

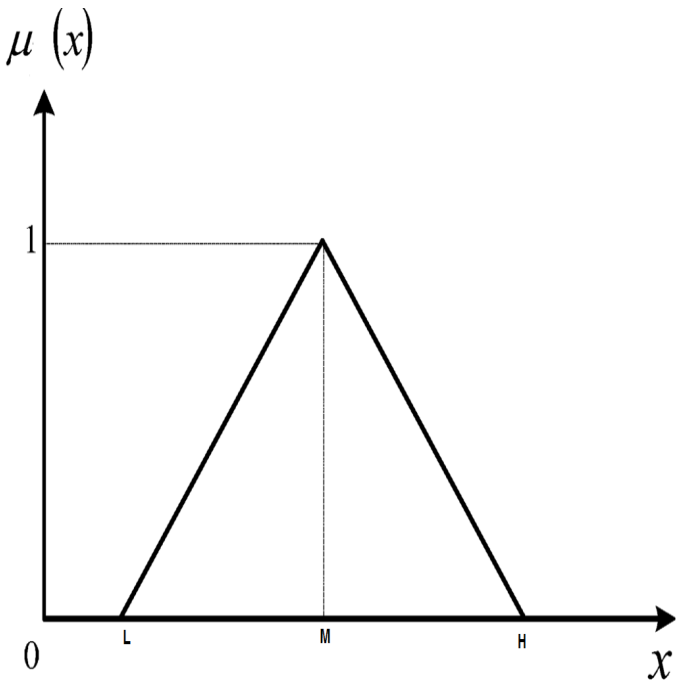
$$TFN(L, M, H) = \frac{H - L}{2M} \tag{6.1}$$

The membership function $\mu(x)$ for TFN is defined by the equation 6.2 and is shown in the figure 6.1 [Che00] .

$$\mu(x) = \begin{cases} 0, & x < L \\ \frac{x-L}{M-L}, & L \leq x \leq M \\ \frac{H-x}{H-M}, & M \leq x \leq H \\ 0, & x > M \end{cases} \tag{6.2}$$

Algebraic operations (addition, subtraction, multiplication and division) for TFN are

Figure 6.1: TFN Membership Function



performed respectively as [Zad99]:

$$(a_1, b_1, c_1) \oplus (a_2, b_2, c_2) = (a_1 + a_2, b_1 + b_2, c_1 + c_2) \tag{6.3}$$

$$(a_1, b_1, c_1) - (a_2, b_2, c_2) = (a_1 - a_2, b_1 - b_2, c_1 - c_2) \tag{6.4}$$

$$(a_1, b_1, c_1) \otimes (a_2, b_2, c_2) = (a_1 a_2, b_1 b_2, c_1 c_2) \tag{6.5}$$

$$(a_1, b_1, c_1) \div (a_2, b_2, c_2) = (a_1 \div a_2, b_1 \div b_2, c_1 \div c_2) \tag{6.6}$$

6.2 General Procedure

GORE is used for identifying and managing the criteria for higher level goals. The leaf level goals help in establishing the criteria which are used to accumulate stakeholder opinions. These criteria based on stakeholders needs and preferences help to identify the importance of requirements by using qualitative and quantitative reasoning techniques. After the relative importance of each leaf level functional goal, the quality models are used to identify quality goals of already accepted functional goals. Then the impact of quality goals among each other and among functional goals is determined.

The general procedure consists of the following steps:

1. Establishing leaf level functional goals for higher level goals
2. Involving stakeholders opinions
3. Finding scores of each leaf level functional goal
4. Identify quality goals related to functional goals
5. Establish links (contributions) among functional goals and quality goals
6. Measure the impact of quality goals and functional goals
7. Ranking quality goals

GORE is used to explore and establish the leaf level functional goals. These leaf level functional goals are then prioritized based on the stakeholders interests, for determining which of them are more important than others. It serves two purposes:

1. Involving the stakeholders opinions
2. Finding the relative importance

The output of this step is a prioritized list of functional goals. This list is then used to find the impact of quality goals which helps in the evaluation of quality goals among each other and on functional goal.

6.3 Methodology

Success of the software system depends on its capability to satisfy both functional and non-functional requirements. Traditionally, the functional requirements are given high priority while dealing with requirements at abstract level. Goal oriented requirements engineering has been used in representing the requirements at higher level. Goal models combined with quality models can represent both functional and non-functional requirements adequately. However, the impact measurement of contributions among quality goals and also between functional and quality goals is rarely addressed. Because of imprecise nature of the requirements, fuzzy number combined with goal models and quality models can sufficiently represent the requirements impact among each other by quantitative means.

In this section, the approach on how to use fuzzy number for functional goals and to find out among different functional goals, the ones which lead to better stakeholder satisfaction is presented. The proposed methodology consist of three levels. At **first level** all the identified functional requirements are prioritized according to different stakeholders using the fuzzy numbers. Stakeholder opinions are accumulated using linguistic terms and these opinions are converted to quantifiable terms using TFN and defuzzification process. These values are then normalized using the equation 6.10. Prioritized functional requirements based on stakeholder opinions are used as input for second level of prioritization. In **second level**, the interactions or dependencies between functional and non-functional requirements are determined and requirements are ranked based on these interactions. The interactions between requirements can be positive or negative. This stage ranks functional requirements and non-functional requirements as well. At **last level** of prioritization development factors like cost, time, effort and risk are used for prioritizing. The proposed methodology consist of following steps and is represented in the figure 6.2:

1. Identify all functional requirements
2. Identify relevant stakeholder
3. Assign stakeholders weights according to their importance. At least three perspective stakeholders should always be presented when prioritizing requirements [Dav03]. They are customers, developers and financial representatives. Stakeholders opinions are taken in linguistic terms.
4. Calculate score of each requirement based on stakeholders opinions and weights assigned to them. Stakeholder opinions represented in qualitative terms are converted to quantitative values by Triangular Fuzzy Number (TFN).

5. Defuzzification process is used to get crisp number
6. scores are normalized to get order/ranks of requirements

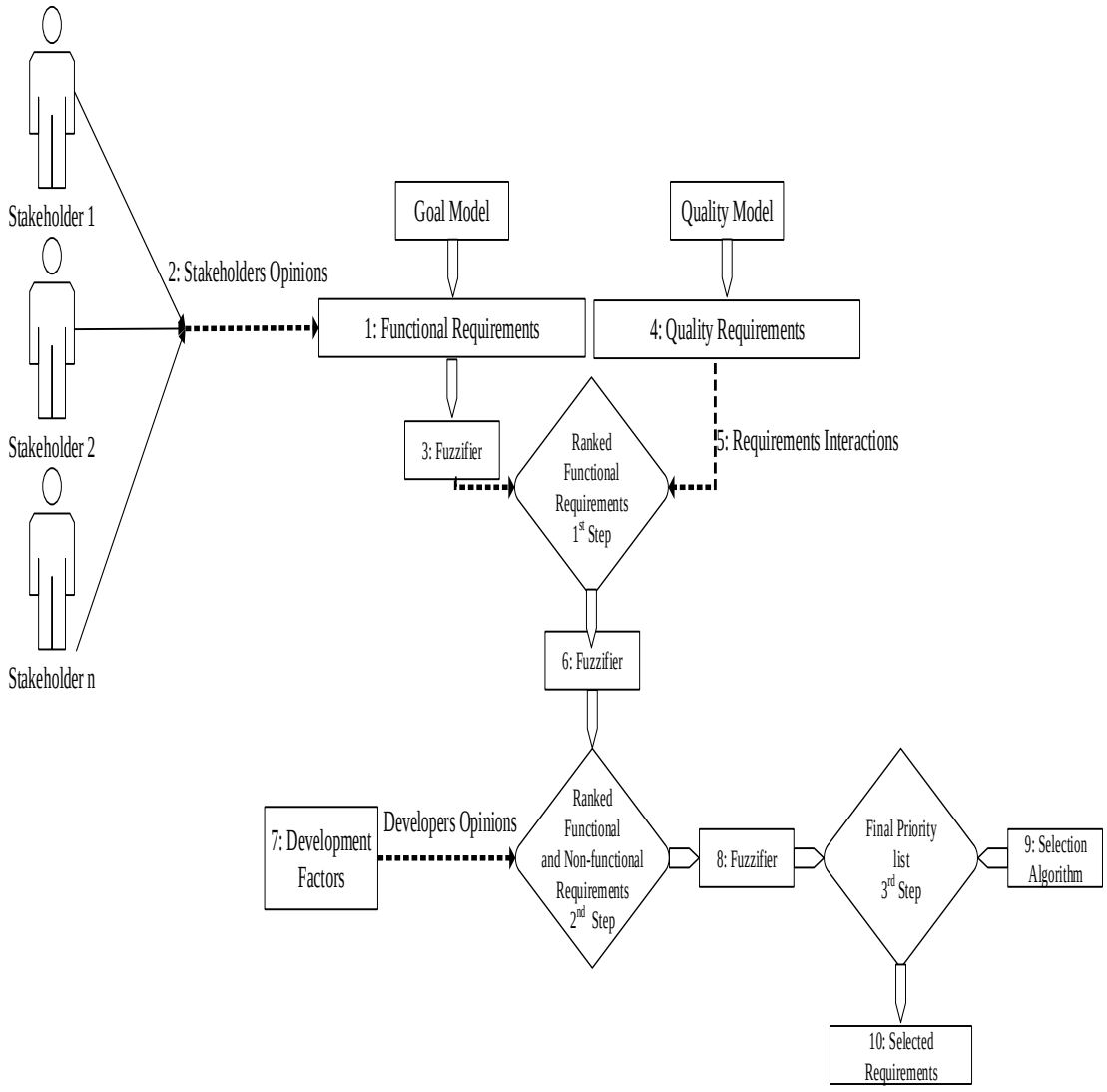
After these steps, a prioritized list is obtained based on stakeholders opinions. In next step, prioritization is refined based on non-functional requirements.

1. Identify non-functional requirements related to functional requirements
2. Identify the interactions (positive or negative) among functional and non-functional requirements
3. A relationship matrix is constructed based on functional and non-functional requirements interaction
4. Priority is calculated using fuzzy numbers and defuzzification process

After the two step process requirements are prioritized based on stakeholder opinions and non-functional requirements dependencies. In the last step, requirements are prioritized based on effort, time, and risk aspects.

The impact of non-functional requirements to the functional requirements is determined by using GORE. Higher level goals are modelled goal graphs are used to get the goal models. AND/OR diagrams which are essential output artefact of these goal models are used in the exploration phase of alternatives. The leaf nodes of goal models are used as criteria for functional requirements. These criteria are compared based on the weighted scores.

Figure 6.2: Proposed Methodology



6.4 Cyclecomputer Example

The 'cyclecomputer' system is used as case study which is developed in our research group. The aim of 'cyclecomputer' project is to develop a flexible and modular bicycle computer which is adaptable to the needs of the driver. A driver will be supported while riding the bike, for maintenance issues, for tour preparations, or to enhance the safety using the bike e.g., besides the normal cycling activities one could use the 'cyclecomputer' as a medical device which will support people having of health problems. It can be used for professional cyclist or just for entertainment purposes. A variety of sensors in 'cyclecomputer' provide a comprehensive view of bike, driver/rider and route. In addition to speed, temperature, altitude, geographic location, heart rate; measure-

ments like oil quality and pressure in the damper elements, brake wear or brake fluid quality are relevant to this project. Measurement of the quality framework on strain gauges is also an important requirement. This system will be attached to a bicycle, will process data from various sensors. All data is processed in the 'cyclecomputer' itself or it will communicate with a standard PC in the aftermath of a tour. One of the results of the requirements engineering phase is a goal model [MS11].

6.4.1 Establishing High level Goals

Though there are many goals related to 'cyclecomputer' but for space and simplicity considerations the following identified high level goal Achieve[TourPlaningServiceSatisfied] is selected.

6.4.2 Refine Goals to Leaf Levels (establish functional goals)

The above mentioned goal is refined using GORE until they are assignable to agents i.e., human agents or software agents. These leaf levels goals are used as criteria for functional goals. Quality goals which include non-functional requirements and often serve selection criteria are also refined based on quality models. The goals along with their subgoals and short description are presented in table 6.1, while figure 6.3 shows partial goal model for high level goal Achieve[TourPlaningServiceSatisfied].

6.4.3 Stakeholders and Their Opinions

6.4.3.1 Identifying Stakeholders

Though there are number of stakeholders in 'cyclecomputer' but following are the relevant stakeholders for goal Achieve[TourPlaningServiceSatisfied]:

1. Medical Cyclist: People who need a defined training / exercise due to any disease e.g., a heart disease. Medical cyclist can use pulse measurement, blood pressure, calory consumption by 'cyclecomputer' device.

2. Doctor (medical): The doctor will cooperate with a patient to set-up the correct tour plans.

3. Touring Cyclist: People who like to ride the bicycle for long trips (>100km) and they need specific services for their tours. The trips might take more than one day.

4. Analyst: analyse the touring details, analyse the cyclist.

Table 6.1: Partial Goal subgoal description

High level goal	**Sub-goals till functional goals**	**Description**
Tour Planning Service Satisfied	Route planning	• The cycle computer should offer route planning. • Routing should consider the current weather forecast.
	Initial checkups	The cycle computer should offer an initial check-up to assess the drivers capabilities.
	Technical riding capabilities	• Frame quality level should be analysable and visible i.e., show the condition of the frame, interpret the frame condition by a coloured icon. • The quality level should be visualized by the time until the frame might break. • The cyclist should see the current speed of the cycle. • The cyclist should be informed when the oil in the shocks should be changed.
	Weather info	• The cyclist should see the current environmental temperature. • The temperature of the last 5 days should be analysable.
	Transferable to web	Track data should be transferred to a Web-portal to enable online competition / comparison.
	Tour details	• The cycle computer should provide complete details of the tours. • The cyclist should be informed about the current height (above sea level). A cumulative value should be shown by ascended and descended meters.
	Navigation	The cyclist should be able to navigate to a given location. The location could be a point of interest, e. g., a hotel. The cyclist should be informed about his global position on a map.
	Trip suggestions	The cycle computer should offer trip tips for professional sports cyclists e.g., gear change tips, speed tips based on the (known) route.

Figure 6.3: Partial Goal Model

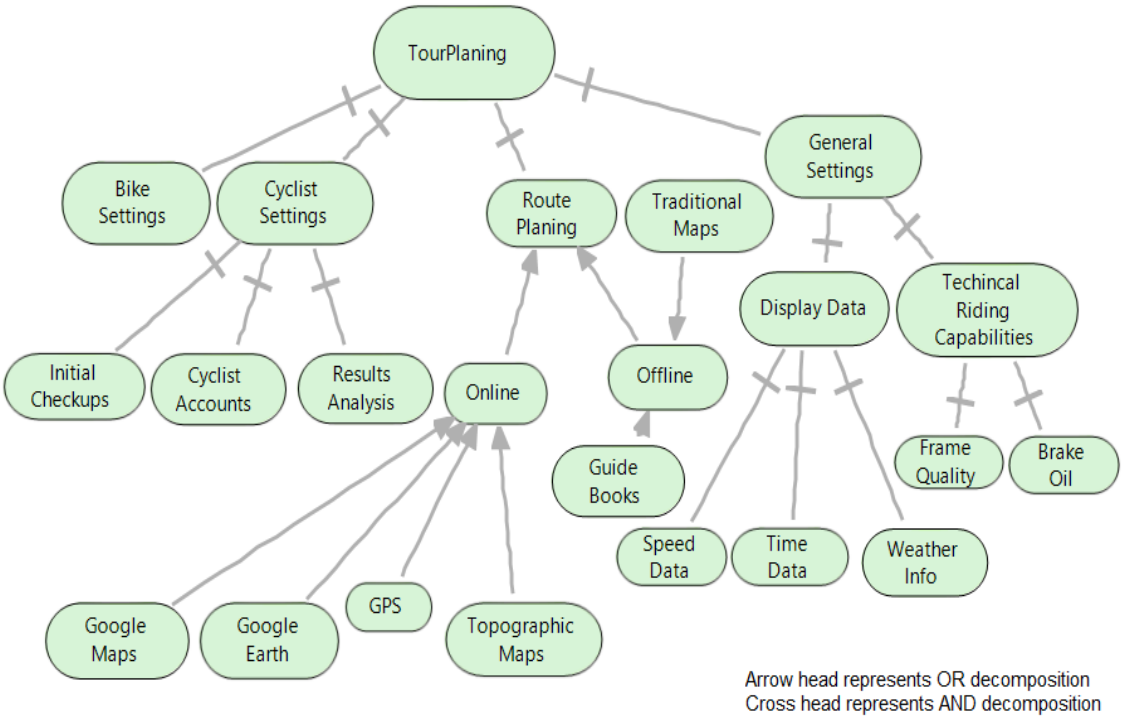


Table 6.2: Linguistic terms and their TFN values

Linguistic terms	**Representative TFN**
Very High	(0.9, 1.0, 1.0)
High	(0.7, 0.9, 1.0)
Medium	(0.3, 0.5, 0.7)
Low	(0, 0.1, 0.3)
Very Low	(0, 0, 0.1)

6.4.3.2 Stakeholders Opinions Accumulation

Three stakeholders are selected and these stakeholders are asked to give their judgments against functional goals in table 6.1. Their judgements are used to elicit the importance degree of each functional goal. To enhance the user-friendliness for interacting with stakeholders linguistic terms are used. Linguistic terms are used to describe complex and ill-defined situations which are difficult to be described in quantitative measure. These linguistic terms are represented using TFN. The TFN values for these linguistics terms are derived from [Che00]. Table 6.2 shows the linguistic terms and their representative TFN values. Table 6.3 shows stakeholders judgments against functional goals in table 6.1.

Table 6.3: Stakeholder judgements

High level goal	**Sub-goals till functional goals**	**SH1**	**SH2**	**SH3**
Tour Planning Service Satisfied	Route planning	Very High	High	Very High
	Initial checkups	Medium	High	High
	Technical riding capabilities	High	Medium	High
	Weather info	Low	High	Medium
	Transferable to web	Low	Low	Medium
	Tour details	High	Very High	High
	Navigation	Very High	High	Very High
	Trip suggestions	Medium	Medium	Medium

6.4.4 Aggregating the Importance Using TFN

The different importance degrees of each functional goal assigned by stakeholders is calculated using TFN. TFN is used to aggregate the subjective opinions of stakeholder using fuzzy set theory. Many methods based on mean, median, min, max, etc.; are available to aggregate the opinions. Among them average operation is most commonly used aggregation method [EK08]. The average operator is used as an aggregation methods to accumulate stakeholder opinions. Let's say there are 'k' number of stakeholders who assign linguistic term values according to table 6.2 to 'n' number of functional goals. The aggregated weight (importance) of each functional goal is calculated as [**?**]:

$$r_f = \frac{1}{n} \{L_f, M_f, H_f\} \tag{6.7}$$

where 'f' represents functional requirements from 1...n

$$L_f = \sum_{j=1}^{n} w_j L_{fj}, \quad M_f = \sum_{j=1}^{n} w_j M_{fj}, \quad H_f = \sum_{j=1}^{n} w_j H_{fj} \tag{6.8}$$

where 'j' represents number of stakeholders from 1...n. For 'n' number of stakeholders who use linguistic term according to table 6.2 to assign values to 'n' number of functional goals and 'w' represents weights of each stakeholder.

6.4.5 Apply Defuzzification Process on TFN

Defuzzification process is applied to convert calculated TFN values into quantifiable values (for crisp output). For its simplicity "2nd weighted mean" defuzzification method

Table 6.4: TFN, Defuzzification and Normalized Scores

Functional goals	**SH1**	**SH2**	**SH3**	**Triangular Fuzzy Numbers**	**Defuzzification**	**Normalized Values**
Route planning	Very High	High	Very High	(0.83, 0.96, 1.0)	0.93	0.17
Initial checkups	Medium	High	High	(0.56, 0.76, 0.9)	0.74	0.13
Technical riding capabilities	High	Medium	High	(0.56, 0.76, 0.9)	0.74	0.13
Weather info	Low	Medium	High	(0.33, 0.5, 0.66)	0.49	0.08
Transferable to web	Low	Low	Medium	(0.1, 0.23, 0.43)	0.24	0.04
Tour details	High	Very High	High	(0.76, 0.93, 1.0)	0.90	0.16
Navigation	Very High	High	Very High	(0.83, 0.96, 1.0)	0.93	0.17
Trip suggestions	Medium	Medium	Medium	(0.3, 0.5, 0.7)	0.5	0.09

is applied to convert a fuzzy number into crisp score. Defuzzification process is represented by the equation 6.9 and is adapted from [Opr11]:

$$DFN = (2M + L + H)/4 \tag{6.9}$$

L, M and H represents lower, middle and upper values of TFN.

6.4.6 Normalizing Values Obtained by Defuzzification Process

Although here all the fuzzy numbers are in interval [0,1] and therefore the calculation of normalization is not required, still the scores after the defuzzification process can be normalized by using the equation 6.10:

$$ND_i = D_i / \sum_{i=1}^{m} D_i \tag{6.10}$$

where 'm' represents number of functional goals.

Table 6.4 represents TFN, defuzzification and final normalized defuzzification values that give the importance of degrees of each functional goal. The defuzzification normalized values give the prioritized list of functional goals. Although here in the example, stakeholders are assigned same weight but it is possible to assign different weights to each stakeholders based on their importance in the project.

6.4.7 Functional and Quality Goal Impact Measurement

This process consist of three steps:

1. Determining project specific quality goals
2. Determining and evaluating the dependency among quality goals
3. Determining and evaluating the impact of quality goals and functional goals

6.4.7.1 Determining Project Specific Quality Goals

Quality models and NFR framework are useful for determining project based quality goals, that is, the quality goals related to high level system goals. Figure 5.8 provides widely used quality attributes in these models. The advantage of using these models is that they provide clear, detail definitions of quality attributes. The universality of these models, because of their acceptance all around the software community. The quality goals are then integrated to functional goal model. Figure 5.9 represents the conceptual model of quality goals integration to functional goal. Figure 6.4 shows two quality goals 'Safety' and 'Availability' for 'cyclecomputer' functional goal 'RoutePlanning'. These quality goals are represented as softgoals using openOME tool.

Figure 6.4: Quality Goals and Functional Goals

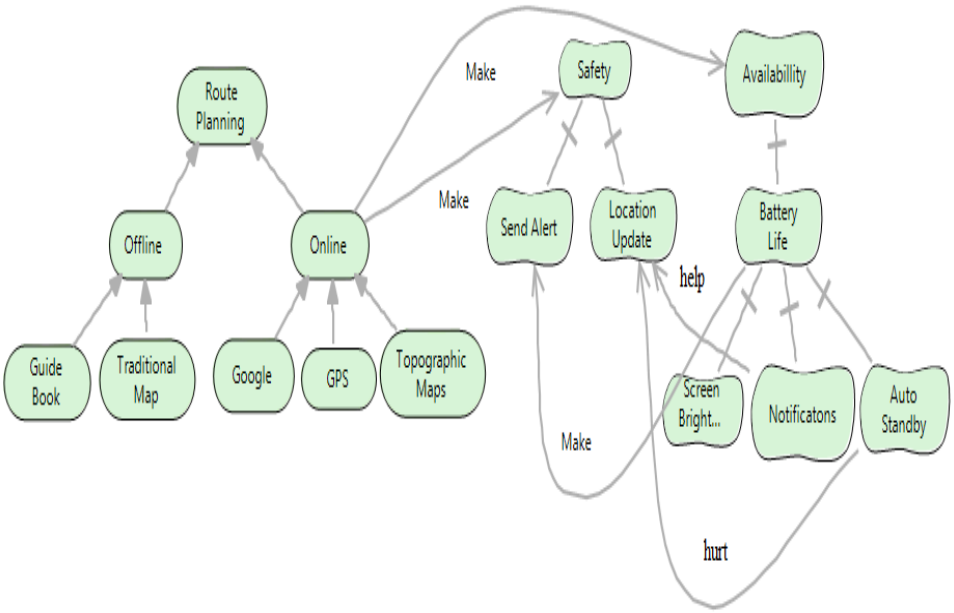


Table 6.5: Linguistic terms and their values for quality goals

Linguistic terms	Numerical Scale
Make	(0.4, 0.5, 0.5)
Help	(0.2, 0.4, 0.5)
Neutral	(-0.2, 0, 0.2)
Hurt	(-0.5, -0.4, -0.2)
Break	(-0.5, -0.5, -0.4)

Table 6.6: Quality Goals Impact and Measurement

	LQG21	LQG22	LQG23	LQG24	Triangular Fuzzy Numbers	Defuzz-ification	Norm-alized Values
LQG11	-	Make	Help	Hurt	(0.03, 0.16, 0.3)	0.15	0.18
LQG12	Make	-	Help	Make	(0.33, 0.46, 0.5)	0.43	0.51
LQG13	Hurt	Help	-	Help	(-0.03, 0.13, 0.26)	0.11	0.13
LQG14	Make	Help	Hurt	-	(0.03, 0.16, 0.3)	0.14	0.16

6.4.7.2 Determining and Evaluating the Dependency between Quality Goals

Quality goals are refined same as functional goals are refined in goal models. These lower level quality goals may influence other quality goals positively or negatively, for example, the fulfillment of one quality goal may hurt or help in the fulfillment of another quality goal. In this step, the importance of each individual quality goal identified in previous step (6.4.7.1) is measured using TFN (6.4.5) and get crisp values by applying the defuzzification process (6.4.6). The strength of relationships between quality goals can be measured. The linguistic terms and their numerical values used to get crisp values and to measure the relationship strengths are shown in figure 6.5. The real number interval which represents the direction and strength of relationships among quality goals is set [-0.5,0.5]. The range from negative number is chosen because the contribution 'hurt' or 'break' will have negative impact on other quality goals. These linguistic terms (make, help, hurt, break) are very common in GORE for their use as softgoals contribution. The same linguistic terms are used and then numerical values defined for these terms in the range [-0.5,0.5].

Let's say there are two quality goals (QG1, QG2) each is refined to four leaf level goals. Now leaf level goals of QG1 are influencing QG2 in positive and/or negative way. Table 6.6 shows their contributions, measurements and final column representing the priority of each leaf level goal of QG1.

The strength of relationships between two quality goals is measured and their val-

Table 6.7: Relationship Strength Values

	LQG21	**LQG22**	**LQG23**	**LQG24**
LQG11	-	1.0	0.8	-0.8
LQG12	1.0	-	0.8	1.0
LQG13	-0.8	0.8	-	0.8
LQG14	1.0	0.8	0.8	-

Table 6.8: Requirements Values after Quality Goals Interactions

Requirements	Score	Security	Safety	User friendly	Performance	Triangular Fuzzy Number	Defuzification	Normalized Values
Route planning	0.93	Neutral	Help	Neutral	Help	(0, 0.2, 0.35)	0.174	0.11
Navigation	0.93	Make	Help	Make	Help	(0.3, 0.45, 0.5)	0.395	0.25
Tour details	0.90	Neutral	Help	Help	Help	(0.15, 0.4, 0.42)	0.309	0.20
Initial check-ups	0.74	Make	Help	Help	Help	(0.25, 0.425, 0.5)	0.296	0.19
Technical riding capabilities	0.74	Make	Neutral	Neutral	Help	(0.05, 0.22, 0.35)	0.157	0.10
Trip suggestions	0.50	Neutral	Help	Help	Help	(0.1, 0.3, 0.425)	0.140	0.09
Weather info	0.49	Neutral	Neutral	Neutral	Help	(-0.4, 0.1, 0.275)	0.040	0.02
Transferable to web	0.24	Neutral	Help	Neutral	Help	(0, 0.2, 0.35)	0.045	0.03

ues for goals in table 6.6 are calculated in table 6.7, for example, relationship (LQG1, LQG2, 1.0) gives relationship value (1.0) between leaf level QG1 and leaf level QG2. Here first element LQG1 is impacting or contributing to second element LGQ2 (impacted by LGQ1).

6.4.7.3 Determining and Evaluating the Impact of Quality goals and Functional goals

In last part of this step the impact of quality goals and functional goals is determined and evaluated. Table 6.5 is used to assign the values, impacting goals are arranged vertically and impacted goals are arranged horizontally. Same steps as in 6.4.7.2 are repeated to measure the contributions and relationship strengths. This is the second step of the process and output is shown in table 6.8.

Table 6.9: Requirements Values after Development Factors

Requirements	Score	Risk	Time	Effort	Triangular Fuzzy Number	Defuzification	Normalized Values
Navigation	0.395	Medium	Medium	High	(0.43, 0.63, 0.8)	0.63	0.11
Tour details	0.309	Medium	High	High	(0.56, 0.76, 0.9)	0.41	0.25
Initial check-ups	0.296	High	High	High	(0.7, 0.9, 1.0)	0.25	0.20
Route planning	0.174	Medium	High	High	(0.56, 0.76, 0.9)	0.23	0.19
Technical riding capabilities	0.157	Low	Low	Medium	(0.1, 0.23, 0.43)	0.60	0.10
Trip suggestions	0.140	Medium	Medium	Medium	(0.3, 0.5, 0.7)	0.30	0.09
Transferable to web	0.045	Medium	High	High	(0.56, 0.76, 0.9)	0.060	0.02
Weather info	0.040	Medium	Medium	Medium	(0.3, 0.5, 0.7)	0.08	0.03

6.4.8 Development Factors Considerations

In the last step of the process, development constraints are involved in the prioritization process. The factors considered here are time, risk, and effort. In this ways developers opinions are taken into account. The output of final step is shown in table **??**.

Instead of selecting the highest priority requirements, the requirements are selected based on their importance in terms benefit over cost ratio. The requirements are selected until the specified upper limit of cost is reached. The density of each requirement calculated by priority per cost is given in 6.11.

$$D_i = P_i/C_i \tag{6.11}$$

After that requirements are sorted out in decreasing density order then in each iteration it is checked that total cost is reached to maximum specified limit or not. If it is less than the threshold value, it is selected from implementation. In this way algorithm ensures that maximum number of requirements are selected from prioritized list while not exceeding the cost limit. Algorithm for the selection of requirements is presented 1. In the example if there is a maximum cost of 70% to be spent then by 1 the requirements selected are shown in the 6.10. Only those requirements are selected which fulfil the cost constraints and for requirements when cost constraints reaches above 70% are not selected.

Table 6.10: Requirements Selected by Algorithm

Requirements	Priority (p)	Cost(c)	Density D= p/c	Total Cost	
Navigation	0.395	0.14	2.83	0.14	Selected
Tour details	0.309	0.12	2.57	0.26	Selected
Initial check-ups	0.296	0.12	2.46	0.38	Selected
Technical riding capabilities	0.157	0.14	1.12	0.52	Selected
Route planning	0.174	0.11	1.58	0.63	Selected
Trip suggestions	0.140	0.20	0.7	-	Not Selected
Weather info	0.040	0.06	0.66	0.69	Selected
Transferable to web	0.045	0.11	0.41	-	Not Selected

6.5 Comparison With Related Work

To measure the importance degree of each requirement many requirements prioritization methods are present in literature. Analytic Hierarchy Process (AHP) is one popular method for prioritization, it involves pair-wise comparison [Saa08]. All pair of requirements are compared to determine the priority level of one requirement over another requirement. Requirements are arranged in matrix form, that is, rows and columns. Then priority is specified to each pair of requirements by assigning a preference value between 1 and 9, where 1 expresses equal value while 9 indicates extreme value. AHP involves stakeholders opinions but pairwise comparison of all requirements make it cumbersome and difficult to use. This method also involves stakeholders opinions and take into consideration both functional and non-functional requirements. Comparisons are made only between the impacting requirements. Importance of both functional and quality goals is obtained using linguistic terms which are easy to deal from stakeholders point of view. These stakeholder opinions are then evaluated using fuzzy set concepts, weight for each functional goals and contribution/impact values are calculated.

In [Lam00b] [CKM02] qualitative approaches are used for measuring the contributions. These methods mainly focus on choosing the best alternative. They use temporal logic and label propagation algorithm. In this approach quantitative terms are used for measuring the strength of relationships. In [MD14] prioritizing process for software requirements is highlighted. It considers prioritization of both functional and non-functional requirements at the same level and as a result produces two separate prioritized lists: one of functional requirements and second for non-functional requirements. Like proposed approach their work also used the concepts from [LW92] but their work is only used for prioritization of functional and non-functional requirement

Algorithm 1 Requirements selection algorithm

```
1: procedure REQUIREMENTS-SELECTION(p, c, C)
2:     density D_i = P_i/C_i
3:     SortDecreasing (density)
4:     while i <= n do
5:         if c_i + TotalCost <= C then
6:             RequirementIsSelected
7:             TotalCost = c_i + TotalCost
8:             i = i+1
9:         else
10:            RequirementIsNotSelected
11:            i = i+1
12:        end if
13:    end while
14:    while n > i do
15:        if c_n + TotalCost <= C then
16:            RequirementIsSelected
17:            TotalCost = c_i + TotalCost
18:            n = n-1
19:        else
20:            RequirementIsNotSelected
21:        end if
22:    end while
23:    return TotalCost
24: end procedure
```

while proposed approach gives an integration model for functional and quality goals and it uses the prioritized requirements to measure their impact on each other.

Wiegers [Wie99] method is semi-quatitative method which focused on customer involvement. Requirements are prioritized based on four criteria defined as benefit, penalty, cost, and risk. The attributes (criteria) are assessed on a scale from 1 (minimum) to 9 (maximum). The customer determines the benefit and penalty values whereas the developers provides the cost and risk values associated with each requirement. Then, by using a formula, the relative importance value of each requirement is calculated by dividing the value of a requirement by the sum of the costs and technical risks associated with its implementation.

The work in [GSL14] focused on modeling the impact of non-functional requirements on functional requirements. For that matter, they investigate the relationships between functional and non-functional requirements. They advocate to define non-functional requirements at the highest level of abstraction like functional requirements. Their proposed approach uses and modifies the NFR framework concepts of contribution but there is nothing mentioned about how to measure the relationships (contributions, impacts) quantitatively.

The work of [YT97] was the initial attempt to use fuzzy concepts in requirements engineering. Their method deal with conflicting requirements and focus of their work is on prioritizing the conflicting requirements by finding some trade-off between these requirements. The conflicting requirements were represented using fuzzy logic and then they use reasoning scheme to infer the relationship between these conflicting requirements. Ito [Ito07] discussed the uncertainty of design decisions. This work suggests to use AHP and Quality Function Deployment (QFD) for prioritization and for conflict resolution. In [LHM+14] the distinction is made between functional goals and quality goals. They presented non-functional requirements as requirements over qualities i.e., non-functional requirements are modelled as quality goals. For quality goals they use ISO/IEC 25010 standard as reference. They distinguished between domain and co-domain of quality goals. The problem with their model is that functional goal(s) can not be refined into quality goal(s) and vice versa but in GORE there are situations where one encounter these refinements i.e., functional goal refinement results into quality goal and vice versa.

In [SPS+12] proposed the guidelines for the elicitation of trustworthy requirements. These guidelines are helpful in selection of project specific quality goals from goal models. Their model consist of three parts: decomposition tree, correlation matrix (CM) and priority vector. Their CM is also base based on fuzzy set theory but it is restricted to elicitation of trustworthy requirements.

This approach used the fuzzy set concepts to evaluate the importance of leaf level functional goal. Weight for each functional goal is calculated based on stakeholders opinions. These weights display stakeholders priorities for all functional requirements. The interaction of stakeholders at early phase of requirements engineering helps to capture the rational (by documenting the preferences) of each requirement and also helps to identify inconsistencies at the early phase of requirements engineering. Using the same method importance weight of quality goals is calculated. Quality goals are tailored using quality models and dependencies among quality goals and functional goals are modelled and measured using fuzzy concepts. The method gives a systematic structure to calculate the fuzzy weight of functional and quality goals. The subjective weights assigned by stakeholders are normalized into a comparable scale. The contributions and strength values are also determined and the strength of the relationships is measured using TFN and defuzzification process.

6.6 Summary

In this chapter an approach is presented to use the goal model of goal-oriented requirements engineering to establish the functional goals as criteria. These leaf levels

functional goals are prioritized according to stakeholders preferences. Triangular fuzzy numbers and defuzzification process is used for prioritization, the developers input and risk tolerance is dealt by defuzzification of TFN. After that, the process is used for specified quality goals which are tailored using quality models. In the final step, dependencies among quality goals and between functional goals are evaluated. Therefore, the proposed methodology was used to measure the strength of relationships.

The methodology was explained by 'cyclecomputer' case study where 8 functional goals were established and stakeholders opinions were collected for these functional goals. After calculating the importance value of each functional goal, the quality goals are integrated and prioritized them according to their dependencies. This approach is promising for ranking of both functional and quality goals because of stakeholders and developers involvement in the process.

Chapter 7

Extending the Approach for Alternatives Selection

The notion of goal and goal models is ideal for the alternative systems. Goal models provide us different alternatives during goal oriented requirements engineering. Once different alternatives are found, there is need to evaluate these alternatives to select the best one. The selection process consist of two parts. In first part of the selection process among alternatives an evaluation criteria is established. The evaluation criteria is based on leaf level goals as discussed in last chapter. Stakeholders are involved to contribute their opinions about the evaluation criteria. The input provided by various stakeholders is then converted into quantifiable numbers using fuzzy triangle numbers. After applying the defuzzification process on fuzzy triangle numbers the scores (weights) for each criteria are obtained. In second part these scores are used in the selection process to select the best alternative. The two step selection process helps to select the best alternative among many alternatives.

Decision making process is about the selection of best option among all the alternatives. In decision making problems there are multiple criteria for selection among the alternatives. The problems involving multiple criteria are called Multi Criteria Decision Making (MCDM) problems. Decision making can be challenging because of uncertainty and vagueness of selected criteria and also because of conflicting stakeholders interests. There might be different criteria but some are more important than others and tend to dominate the decision [EK08]. Fuzzy set theory is used to deal with multi criteria problems [Che00].

7.1 Selection Procedure

In Goal Oriented Requirements Engineering (GORE) there is great emphasis on alternative system proposals. Goal refinements help in finding alternatives and during requirements elaboration process many alternatives are considered. The qualitative and quantitative analysis of these alternatives helps to choose the best one. In alternative selection one have to decide about the best option according to stakeholders needs.

In the context of GORE, there is need to support the identification and managing of criteria for alternative's selection process. Finding the criteria based on GORE require high level goals to be analyzed till leaf goals are achieved i.e., requirements. As in the previous chapter, these leaf level goals help in establishing the criteria which are used in the selection process among alternatives. The criteria are based on stakeholders needs and preferences and therefore stakeholders opinions need to be involved in selection process. It helps to identify the importance of requirement according to stakeholders understandings and needs. Based on these criteria the qualitative and quantitative reasoning techniques are applied for the selection of alternative system proposals. It serves two purposes: first involving the stakeholders opinions in selection process and second finding the relative importance of these criteria.

The general procedure of selection among alternatives consists of the following steps:

1. Finding acceptance criteria
2. Involving stakeholders opinions
3. Finding scores of each criteria
4. Evaluating alternatives based on accepted criteria scores
5. Making a selection

7.2 Methodology

First of all different alternatives are explored during GORE and for this goal models obtained during GORE are used. AND/OR diagrams which are the essential output artefact of these goal models are used in the exploration phase of alternatives. Once different alternatives are selected, there is need to evaluate these alternatives to select the best one. The alternatives are compared based on the weighted criteria. The criteria are weighted using fuzzy numbers and stakeholders opinions are taken as input and then converted to fuzzy numbers. By using the fuzzy numbers the qualitative information of stakeholders is converted into quantitative one. The proposed methodology consist of following steps and is shown in figure 7.1.

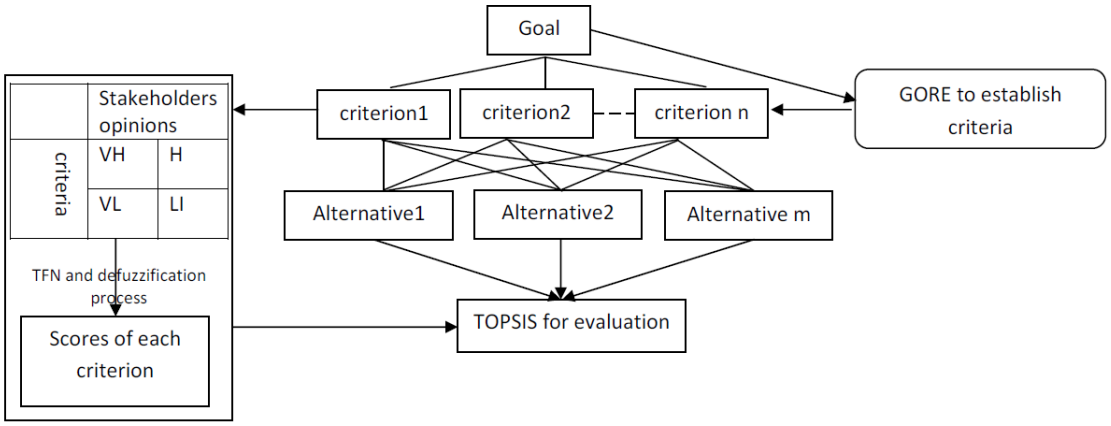
1. Establishing high level goal(s)
2. Establishing the criteria based on leaf level goals (directly assignable to agents: humans or system agents)
3. Identify relevant stakeholders and take their opinions for above established criteria as inputs
4. Calculate relative importance of each criterion by applying TFN and defuzzification process
5. Normalize the scores
6. Identifying the alternatives
7. Evaluate alternatives using TOPSIS based on scores of each criteria
8. Rank alternatives

7.2.1 TOPSIS Review

The Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) is a multi criteria decision analysis method. It is used to compare a set of alternatives based on weighted scores of each criterion. In this method two alternatives are hypothesized: positive ideal alternative and negative ideal alternative and then best alternative is selected which is closet to the positive ideal solution and farthest from negative ideal alternative [Gol13]. TOPSIS consist of following steps [Ols04]:

1. constructing a decision matrix
2. normalizing the decision matrix

Figure 7.1: Methodology Extension for Alternative Selection



3. finding the positive ideal and negative ideal alternatives
4. calculating the separation measures for each alternative
5. calculating the relative closeness to the ideal alternative

7.3 Cyclecomputer Example

As in the previous chapters, the 'cyclecomputer' system is used as case study.

7.3.1 Step 1 Establishing High level Goals

Though there are many goals related to 'cyclecomputer' but for space and simplicity considerations the following identified goals for high level 'cyclecomputer' goal are selected: Achieve[EntertainmentServiceSatisfied], Achieve[CompitionServiceSatisfied], Achieve[TrainingServiceSatisfied], Achieve[TourManagementServiceSatisfied].

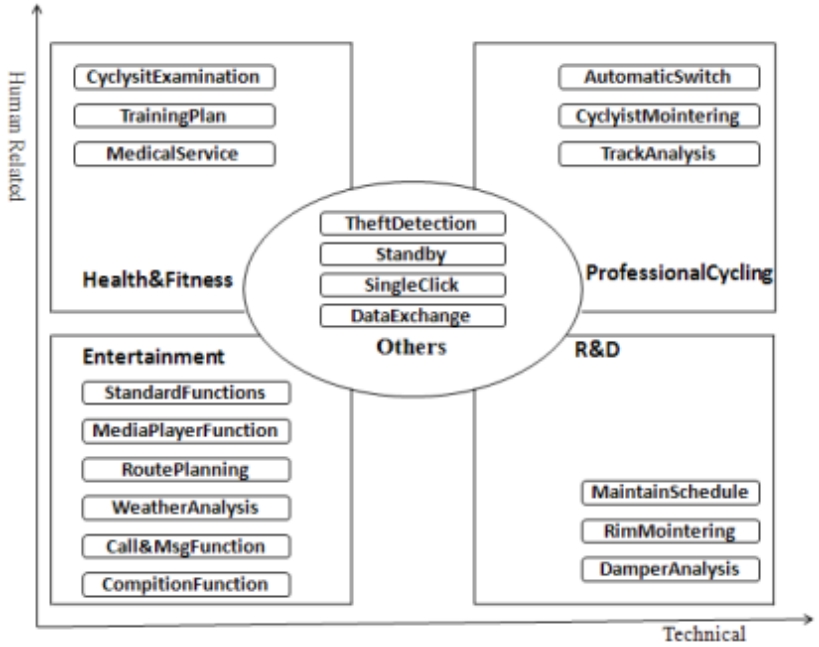
7.3.2 Refine Goals to Leaf Levels (establish criterion for each goal)

The above mentioned goals are refined using GORE until they are assignable to agents i.e., human agents or software agents. These leaf levels goals are used as criteria for alternative selection. Quality goals which include non-functional requirements and often serve selection criteria are also refined. The partial description of goals/subgoals was defined in last chapter in table 6.1 and is used here as well.

7.3.3 Identifying Stakeholders

Though there are number of stakeholders in 'cyclecomputer' but the relevant stakeholders for goals described in table 6.1 are shown in figure 7.2.

Figure 7.2: Relevant Stakeholders



1. Medical Cyclist: People who need a defined training / exercise due to any disease e.g., a heart disease. Medical cyclist can use pulse measurement, blood pressure, calory consumption by 'cyclecomputer' device.

2. Doctor (medical): The doctor will cooperate with a patient to set-up the correct training cycles. The cycles are dependant on the patients constitution.

3. Touring Cyclist: People who like to ride the bicycle for long trips (>100km) and they need specific services for their tours. The trips might take more than one day.

4. Trainer (sports): Create training plans, follow training plans, analyze the cyclist.

7.3.4 Stakeholders Opinions Accumulation

Three stakeholders, professional cyclist(SH1), fun cyclist (SH2), health and fitness cyclist (SH3) are selected. These stakeholders are asked to give their judgements against each criterion. Their judgements are used to elicit the importance degree of each criterion. To enhance the user-friendliness for interacting with stakeholders ordinal

Table 7.1: TFN, Defuzzification and Normalized Scores

Criterion	Triangular Fuzzy Number	Defuzzification	Normalized Values
Mic	(0.75, 0.82, 1)	0.84	0.067
Data storage	(0.75, 0.82, 1)	0.84	0.067
Audio service	(0.75, 0.82, 1)	0.84	0.067
User accounts	(0.75, 0.82, 1)	0.84	0.067
Transferable to web	(0.75, 0.82, 1)	0.84	0.067
Online modus	(0.75, 0.82, 1)	0.84	0.067
Offline modus	(0.75, 0.82, 1)	0.84	0.067
Initial checkups	(0.75, 0.82, 1)	0.84	0.067
Technical riding capabilities	(0.5, 0.79, 1)	0.771	0.062
Fitness level	(0.5, 0.721, 1)	0.735	0.059
Calories consumption	(0.5, 0.655, 0.75)	0.639	0.051
Route planning	(0.5, 0.721, 1)	0.735	0.059
Weather info	(0.75, 0.75, 0.75)	0.75	0.060
Tour details	(0.5, 0.572, 0.75)	0.598	0.048
Navigation	(0.75, 0.82, 1)	0.84	0.067
Trip suggestions	(0.25, 0.520, 1)	0.569	0.046

scale is used. The scale values are same as discussed in last chapter by using table 6.2. Next ordinal scale values are converted to actual numerical numbers to apply TFN.

7.3.5 Step 5 to 7

Steps 5 through 7 are performed using same equations defined in sections 6.4.4, 6.4.5, and 6.4.6. Table 7.1 represents TFN, defuzzification and final normalized defuzzification values that give the importance of degrees of each criterion. The defuzzification normalized values give the prioritized list of criteria which is used in TOPSIS to evaluated alternatives.

7.3.6 Cyclecomputer Alternatives

Four alternatives for evaluation: CM213C, CM404, HAC4Pro, Germin Edge 305 are selected. The preliminary analysis results of these selected alternatives are given in appendix C.

Table 7.2: Alternative fulfilling Criteria Scores

Description	Value
Alternative fulfilling criterion	9
Alternative partially fulfilling criterion	7
Alternative minimally fulfilling criterion	3
Alternative not fulfilling criterion	0.25

7.3.7 Evaluate Alternatives Using TOPSIS

7.3.7.1 Constructing Decision Matrix

For 'm' number of alternatives and 'n' number of criteria a m*n matrix is constructed. Values in the matrix are entered according to table 7.2. For four alternatives, four criteria are randomly selected along with their scores from table 7.1 and a decision matrix is constructed.

7.3.7.2 Normalizing Decision Matrix and Constructing Weighted Normalize Decision Matrix

The decision matrix is normalized according to equation 7.1:

$$r_{ij} = x_{ij} / \sqrt{(\sum_{i} x_{ij}^2)} \quad for\ i = 1, ..., m; j = 1, ..., n \tag{7.1}$$

and then multiplied with each criterion score to get the weighted normalized decision matrix. Figure 7.3 shows the resultant matrices.

Figure 7.3: Decision Matrices

	0.067	0.067	0.059	0.051
	C1	C2	C3	C4
A1	9	7	9	7
A2	7	9	7	7
A3	3	3	3	7
A4	9	3	7	3

(3a) decision matrix

r_{ij} =

	0.067	0.067	0.059	0.051
	C1	C2	C3	C4
A1	0.60	0.57	0.65	0.56
A2	0.47	0.74	0.51	0.56
A3	0.20	0.24	0.21	0.56
A4	0.60	0.24	0.51	0.24

(3b) normalized decision matrix

v_{ij} =

	C1	C2	C3	C4
A1	0.04	0.03	0.03	0.02
A2	0.03	0.05	0.03	0.02
A3	0.01	0.01	0.01	0.02
A4	0.04	0.01	0.03	0.01

(3c) weighted normalized decision

7.3.7.3 Determine the Positive Ideal and Negative Ideal Alternatives

Positive ideal and negative ideal alternatives are determined using the equations 7.2, 7.3 respectively:

$$A^* = (v_1^*, ..., v_n^*), \quad where\ v_j^* = max_i(v_{ij}) \tag{7.2}$$

positive ideal alternative: (0.04, 0.05, 0.03, 0.02)

$$A^{'} = (v_1^{'}, ..., v_n^{'}),\ where\ v_j^{'} = min_i(v_{ij}) \tag{7.3}$$

negative ideal alternative: (0.01, 0.01, 0.01, 0.01)

7.3.7.4 Calculating the Separation Measures

separation measures for both positive and negative ideal alternatives are measured using equations 7.4 and 7.5:

$$S_i^* = [\sum_j (v_j^* - v_{ij})2]^{1/2}, \quad i = 1, ..., m \tag{7.4}$$

$$S_i^{'} = [\sum_j (v_j^{'} - v_{ij})2]^{1/2}, \quad i = 1, ..., m \tag{7.5}$$

Figure 7.4 shows results for separation measure for positive ideal alternative and figure 7.5 shows results for negative ideal alternative.

Figure 7.4: Separation Measure for Positive Ideal Alternative

$$S_i^* = \begin{matrix} \\ A1 \\ A2 \\ A3 \\ A4 \end{matrix} \begin{pmatrix} (0.04\text{-}0.04)^2 + & (0.03\text{-}0.05)^2 + & (0.03\text{-}0.03)^2 + & (0.02\text{-}0.02)^2 \\ (0.03\text{-}0.04)^2 + & (0.05\text{-}0.05)^2 + & (0.03\text{-}0.03)^2 + & (0.02\text{-}0.02)^2 \\ (0.01\text{-}0.04)^2 + & (0.01\text{-}0.05)^2 + & (0.01\text{-}0.03)^2 + & (0.02\text{-}0.02)^2 \\ (0.04\text{-}0.04)^2 + & (0.01\text{-}0.05)^2 + & (0.03\text{-}0.03)^2 + & (0.01\text{-}0.02)^2 \end{pmatrix}^{1/2} \quad S_i^* = \begin{pmatrix} 0.02 \\ 0.01 \\ 0.06 \\ 0.14 \end{pmatrix}$$

Figure 7.5: Separation Measure for Negative Ideal Alternative

$$S_i^{'} = \begin{matrix} \\ A1 \\ A2 \\ A3 \\ A4 \end{matrix} \begin{pmatrix} (0.04\text{-}0.01)^2 + & (0.03\text{-}0.01)^2 + & (0.03\text{-}0.01)^2 + & (0.02\text{-}0.01)^2 \\ (0.03\text{-}0.01)^2 + & (0.05\text{-}0.01)^2 + & (0.03\text{-}0.01)^2 + & (0.02\text{-}0.01)^2 \\ (0.01\text{-}0.01)^2 + & (0.01\text{-}0.01)^2 + & (0.01\text{-}0.01)^2 + & (0.02\text{-}0.01)^2 \\ (0.04\text{-}0.01)^2 + & (0.01\text{-}0.01)^2 + & (0.03\text{-}0.01)^2 + & (0.01\text{-}0.01)^2 \end{pmatrix}^{1/2} \quad S_i^{'} = \begin{pmatrix} 0.042 \\ 0.078 \\ 0.02 \\ 0.037 \end{pmatrix}$$

7.3.7.5 Calculating Closeness to Ideal Solution

The relative closeness to the ideal solution is calculated using the equation 7.6:

$$C_i^* = S_i'/(S_i^* + S_i'), 0 < C_i^* < 1 \tag{7.6}$$

7.3.7.6 Ranking and Selection

Finally the ranking is done and the alternative closet to 1 is selected as the best alternative. Figure 7.6 gives results for selected alternatives and alternative A2 is selected as an ideal solution.

Figure 7.6: Relative Closeness to Ideal Solution

$C_i^* =$

A1	0.042/0.02+0.042 = 0.67
A2	0.078/0.01+0.078 = 0.88
A3	0.02/0.061+0.02 = 0.24
A4	0.037/0.104+0.037 = 0.26

7.4 Comparison With Related Work

Alternatives selection is ongoing research in the area of GORE. On the other hand methods like AHP [Saa08], TOPSIS [Gol13], Fuzzy AHP, Fuzzy TOPSIS [EK08] and VIKOR are used in classical Multi-Criteria Decision Making (MCDM) problems. Multi-criteria decision making (MCDM) has been widely used in selecting or ranking decision alternatives characterized by multiple and usually conflicting criteria [WL09]. The approach of these methods is useful and is adopted for alternatives selection and stakeholders involvement in GORE. [SAG] also emphasizes the importance of decision support in GORE but it differs as it uses Analytic Hierarchy Process (AHP) for prioritization and it deals with only softgoals.

AHP is more suitable for small number of stakeholders and if alternatives are increased to seven are more it becomes difficult to handle them with AHP because it involves pairwise comparison. In contrast proposed approach involves stakeholders opinions and take into consideration both functional and non-functional requirements. The importance of criteria is evaluated using fuzzy set concepts, weight for each criterion is calculated based on stakeholder opinions. When a new criterion is added it

is easy to extend, there is no need to change the previous calculations because newly added criterion is independent from others. These weights are then used in TOPSIS avoiding the cumbersome pair-wise comparisons of AHP.

AGORA [KHS02] is another quantitative approach for alternatives extending the goal oriented requirements analysis but the focus of AGORA is on requirements elicitation. The method focuses on alternative among subgoals, that is, selection of subgoal among many subgoals of same parent. Furthermore AGORA attaches a matrix called preference matrix to nodes of goal graph. It is suitable if number of stakeholders are small in number. When stakeholders are more (plus four) and have to select among many alternatives, this method becomes difficult to handle and goal graph becomes cumbersome.

Here Fuzzy set concepts are used to evaluate the importance of criteria for each goal. Weight for each criteria is calculated based on stakeholder opinions. These weights display stakeholder priorities for all requirements. The interaction of stakeholders at early phase of requirements engineering helps to capture the rational (by documenting the preferences) for the decisions and to identify inconsistencies at the early phase of requirements engineering. The method gives a systematic structure to calculate the fuzzy weight of each criterion. The subjective weights assigned by stakeholders are normalized into a comparable scale. The performance measures of all alternatives on criteria are visualized using TOPSIS which accounts for both the best and worst alternatives simultaneously.

7.5 Summary

In this chapter an approach was presented to use the goal model of goal-oriented requirements engineering to establish the acceptance criteria. After that TFN and defuzzification process is applied to get scores for each criterion. In the final step TOPSIS method is used to evaluate the alternatives and for selection of the best alternatives. TOPSIS method uses the score obtained by TFN and defuzzification process. The proposed methodology can be used against both the functional and non-functional requirements.

The methodology was explained by 'cyclecomputer' case study where 16 acceptance criteria are established and stakeholders opinions were collected for these criteria. After calculating the score of each criterion, four criteria (for simplicity considerations) were selected and based on these evaluated four alternatives. The approach is promising for ranking the criteria and using for ranking of alternative selection because the

stakeholders opinions as well as developers considerations and risk tolerance are taken into account for preference.

Chapter 8

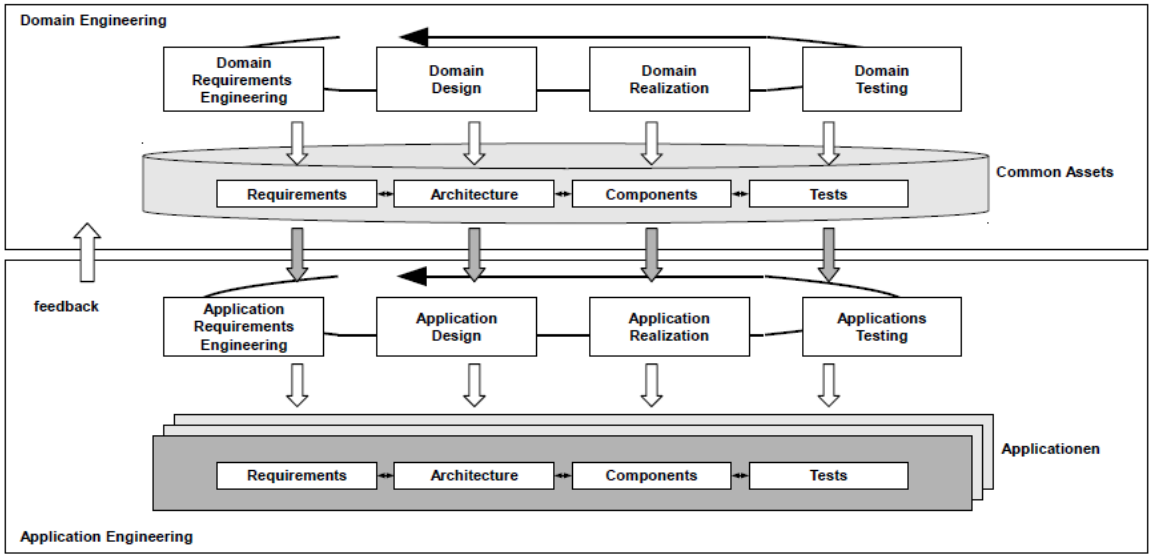
Goal Model Integration for Tailoring Product Line Development Processes

Many companies rely on the promised benefits of product lines, targeting systems between fully custom made software and mass products. Such customized mass products account for a large number of applications automatically derived from a product line. This results in the special importance of product lines for companies with a large part of their product portfolio based on their product line. The success of product line development efforts is highly dependent on tailoring the development process. This chapter presents an integrative model of influence factors to tailor product line development processes according to different project needs, organizational goals, individual goals of the developers or constraints of the environment. The model integrates goal models, SPEM models and requirements to tailor development processes.

Software systems developed based on the product line approach result in systems between custom made software and systems developed for a mass market. Thus, software product lines are customized mass products. The architecture of a product line consists of a core and diverse variable components. Any members of a product line are based on its core and one or more variable components. Core and variable components are pre-developed what results in the special usage of a product line. The customer simply selects and may parametrized the desired features of the future system. Based on the product line, the system (in more detail, the software application) will be automatically generated. The effort for the development of a product line core and its variable components will reach a break even point starting from four [WL99] up to five [LSR07] sold applications. This is mainly due to the large development efforts for the core of the product line, the product line training needed for the developers, the migration effort for companies to go towards the product line concept and the process maturity level needed for product line development [BC96]. The efforts for product line specific development processes are higher than the efforts for the development of standard systems and such development processes need to be tailored towards the project environment of the development team [SW00], [BR87]. The survey of 273 software projects in [Cla97] revealed a potential of reducing the development effort up to 21% by raising the CMM level by one. This shows the big potential of defined and tailored development processes. For the remainder of this chapter the terms method and process are used according to the Software & Systems Process Engineering Metamodel (SPEM) of the Object Management Group (OMG). A method is a reusable and goal oriented procedure made of several steps, referred to as tasks. A process is a sequence of tasks together with the timing information for the sequence. Thus, a process would contain all the timed steps needed to develop a product line. As an example, a review is taken from the method library and reused at different occasions in the process to validate the documents developed along the product line development process. Ten product line case studies have been analysed in [LSR07] out of the domains embedded, oil and gas, finances, mobile communications, telecommunications, multi-media, and the medical domain. All the case studies use a twofold development process, with a domain engineering (development of the product line itself) and an application engineering (development of applications based on the product line) phase, as shown in figure 8.1. Both phases are further subdivided in a requirements, a design, a realization and a testing phase. The common assets, managed in a repository, are in between both phases. They are developed in the domain engineering phase and used in the application engineering phase.

The challenges are the development methods and processes, which have been individually and manually defined by all case studies in [LSR07] as the project proceeded.

Figure 8.1: Product Line Development Process



Although guidelines for the development of product lines have been developed [LSR07], detailed recommendations for the tailoring step of a development process are still missing. It is not yet clear whether and to what degree a given development process will fit to its development environment. A structured approach to address this savings potential could be defined attributes together with a model to optimize the tailoring step of the development process for product lines. Therefore, a tailoring meta-model with a set of attributes to enhance the tailoring step with an optimization towards the presented attributes is presented.

8.1 The Need of Integration Model

The product line development method PuLSE as presented in [BFK+99] is equipped with the PuLSE Baselining and Customization (PuLSE-BC) [SW00] procedure to tailor PuLSE towards the needs of an organization. Any tailoring decisions are bound to the variable parts of the development process. The criteria for tailoring are based on organizational and project domain issues. Such manually elicited criteria result in the variability of the development process. Pulse-BC is managing this variability in an own model. A further refinement of tailoring product line development processes is presented in [AKM+09]. Here, a product line for development processes is proposed, referred to as process line. The requirements of the development processes in this process line are based on an analysis of current and future products, projects and processes. Thus, the processes are optimized towards the products and projects, to derive a tailored development process based on the process line. Tailoring is realized with prior-

itized attributes, with which the resulting elements of the product, process and project analysed are ranked. An automated analysis of the underlying models is not yet realized what also hinders the efficient analysis of different scenarios in different domains. The company specific strategy and the goals of groups as well as individual developers, referred to as soft attributes are also missing. Nevertheless such attributes are important since personal factors influence the success of development process changes to a larger degree than technological challenges [IF07], [SM98], [NWZ06]. As a result, a process line model based on products, processes and project data in relation to models of the company strategy and developer goals is needed. Here, the relations of the model elements and features of the process line are highly important to be able to realize its variability [BSRC10]. In addition, there is also need of a complete model of the attributes to enable an enhanced assessment of derived development processes.

Development process like the V-Model XT, SCRUM or OpenUP are targeting single system development efforts. Nonetheless parts of the methods are taken for the product line development. In [BH11] parts of an agile development process have been used for the product line development in a large company (SAP). Again, tailoring of development processes for product lines is an important success factor. As described in [BH11] but not yet accomplished, the strategic and business goals of an organization need to be part of the development process. The selection of process steps should be traceable to the business and strategic goals. Without such traces development processes cannot be fully analysed and tailored. Thus, the business goals need to be part of the above described process line.

In [Ter09] the tailorability of the V-Model XT towards product line development is analysed. Based on this work a process line was developed and a V-Model XT development processes could be derived based on the process line. Unfortunately, the selection of supporting tools for the development process is still left to the project manager and/or developer and the selection of tools is bound to the knowledge about their advantages and drawbacks, what is currently not part of the model of process lines. The analysis of product line approaches emphasizes the relevance of tools for the success of a product line development project.

All the presented approaches in this chapter are based on the product line development concept shown in figure 8.1 and offer ideas to relate the development process to the development environment. Although, none of the approaches is able to offer a complete model of a tailorable development process together with the elements/components of the development environment. Here, the analysis and assessment of development

processes need to include tools, since they strongly influence the expected effort of a product line development project.

The relation of decisions to the original goals can be realized with goal models [Lam09a]. Goal oriented business processes with variabilities are presented in [SCSP10]. Such models could be used as in [GW03] to analyse and assess the chances of success with the Goal-Question-Metric (GQM) method for product line development projects. For the tailoring step of a developers environment the in influential factors and attributes are still missing for process lines, but could be realized using a goal model. Thus, a comprehensive view onto product line development domain would be possible.

Finally an integrative model for the description of stakeholder needs and goals in relation to the development process artefacts and the development environment specifics is needed, to be able to analyse potential influences of changing goals early in the project development.

8.2 Tailoring Development Processes

As stated in the previous section the requirements on tailoring product line development processes are manifold. Here, these requirements are divided in two main parts

1. The goal model based requirements
2. The method model based requirements

The following categories and parts of the two models are based on own experiences in industrial projects and lessons learned within student software development projects.

First, the identification of influence factors that can be described by goal models contains soft factors, as shown in figure 8.2.

Based on experience it is estimated that about 70% of the challenges throughout the software development project can be traced back to such soft factors. Thus, addressing such factors can influence the success of a project by a large degree. As shown in figure 8.3, two top level factors are refined with a goal model.

The **strategy**of a company is very important when comes to the initial decision for or against a product line. Thus, the following sub-goals as refinement of the strategy are tightly connected to the product line development.

Figure 8.2: Goal and Method Models

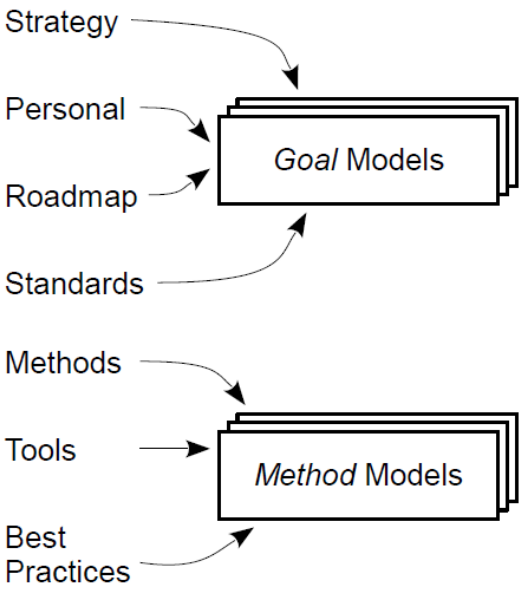
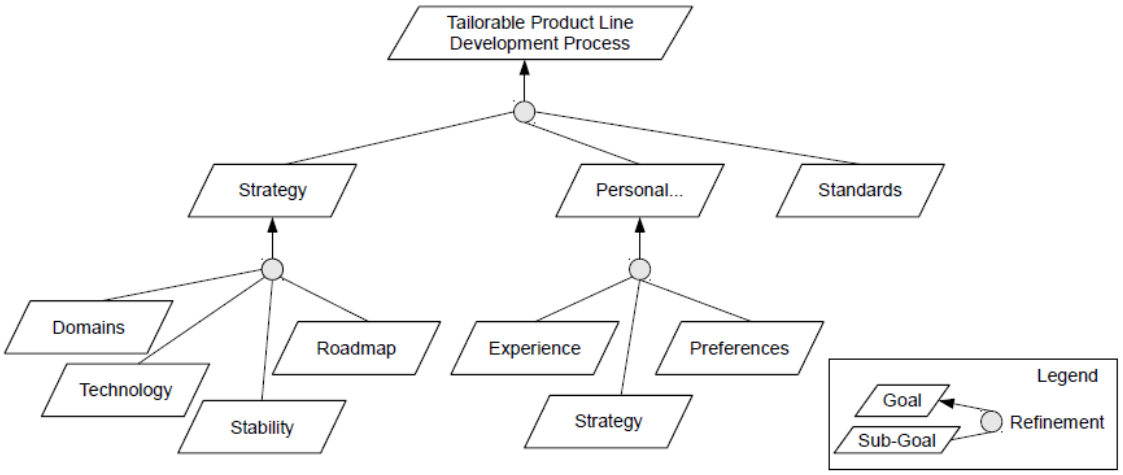


Figure 8.3: Integrated Goal Model



- The target **domain** or domains of the products that will be developed rule about the product line approach. New domains or domains that will be abandoned in the future need to be known and elicited in the requirements engineering phase. Of course, these requirements might have a large impact on the architecture of the product line, specifically to the core and the variabilities of the product line.
- Any strategic choice of the **technology** influences the future constraints (performance, memory, available development environment, available compilers) of the system and thus, constraints for the product line. For example, the realization of variabilities with the C language has reduced capabilities compared to C++.
- **Stability** of the strategy. For new companies this is highly relevant. The strat-

egy is subject of a high risk for changes. Thus, this goal influences the overall feasibility of the product line development.

- The **roadmap** includes the timing for the release of product features. For each release a set of features is identified. The length (way into the future) of the roadmap influences the technological choices and the re-development of the product line. Due to technological changes, fluctuation of employees (and with them the knowledge) and unforeseen requirements the implementation of the architecture of a product line needs to be adapted to this new environment. The roadmap needs to address these large and periodic updates.

The personal factors also have a large impact onto the other elements in the goal model. The personal goals are coupled with a stakeholder model of the involved persons in a software project. Each stakeholders should have an own personal goal model reflecting his/her position towards the product line development process. Since this is a very personal information it is recommended to keep this model private but use the information in correlation with the other models (strategy and standards) as well as use the results of a model analysis as input for the periodic discussions with the management within the company and/or of the respective project.

- Each stakeholders own experience should be related to the role descriptions of the basic development processes (e.g., OpenUp, SCRUM). Besides the potentials for further personal development, such an experience level should be related to the project roles (and their skills) which are attached to each development step. For exchangeable development steps, experiences set the rules on which step to take.

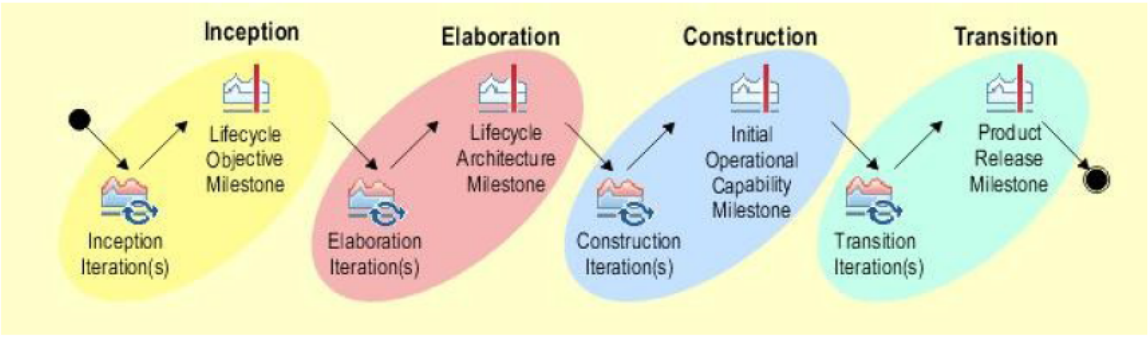
- Each stakeholder has preferences for application domains or technological choices. There are also preferences for methods used along the development process or for specific templates to be used for the deliverables of the development process. These preferences will influence the choices of the method and development process parts of the product line.

- Each stakeholder might (or should) have an own strategy in contrast to the company strategy. The alignment of the strategy of all different stakeholders is impossible, due to the private nature of this information. As with the experience, the awareness of the other goals and their correlation to the own strategy is an important step towards the integration into a developer group and a good starting point to develop an own roadmap. The individual analysis of the own strategy is a good point to think about the own position in the company and/or to better understand the own position.

Standards will influence the technology goals for the strategic planning and they recommend or require technologies and/or tools. For example, the safety standard IEC61508 recommends test case generation tools. Standards could also require a specific development process structure and give recommendations or require development methods.

The lower part of figure 8.2 shows the method models. Here, SPEM is used to describe all the needed parts of the methods, processes and best practices. As a SPEM implementation, OpenUP is shown in figure 8.4.

Figure 8.4: OpenUP Overview

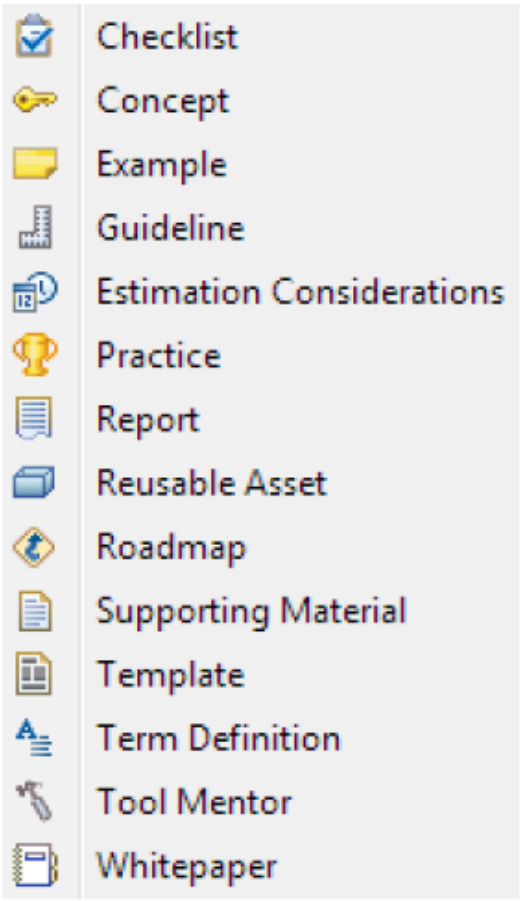


OpenUP is an open source development process for standard applications, the complete extension of OpenUP towards a product line is a future work package. Nevertheless this process is taken as tailoring example to address the above mentioned goals. The development process is split into four iterative phases. Compared to figure 8.1, the requirements is equivalent to the inception phase, the design is equivalent to the elaboration phase and the realization is equivalent to the construction phase. The testing steps are present in each iteration of the OpenUP process and at the first sight the testing phase in figure 8.1 does not match the OpenUP transition phase, but this testing phase is meant to be the final system test with an iterative testing approach as well and thus, the two models are comparable. For each of the development steps in figure 8.4 parts of the method steps of the OpenUP method library are taken and put together.

Each task has its responsible roles attached and each role has its tasks attached. As shown in figure 8.5 the developer role is required to perform the five given tasks and is also responsible for the four deliverables. The last of the SPEM elements relevant for the process tailoring step are the guidances. As shown in figure 8.6 there are 14 guidance types which can be used to support any SPEM element, e. g., a task.

Figure 8.5: Developer Role in OpenUp

Figure 8.6: OpenUP Guidance for SPEM elements

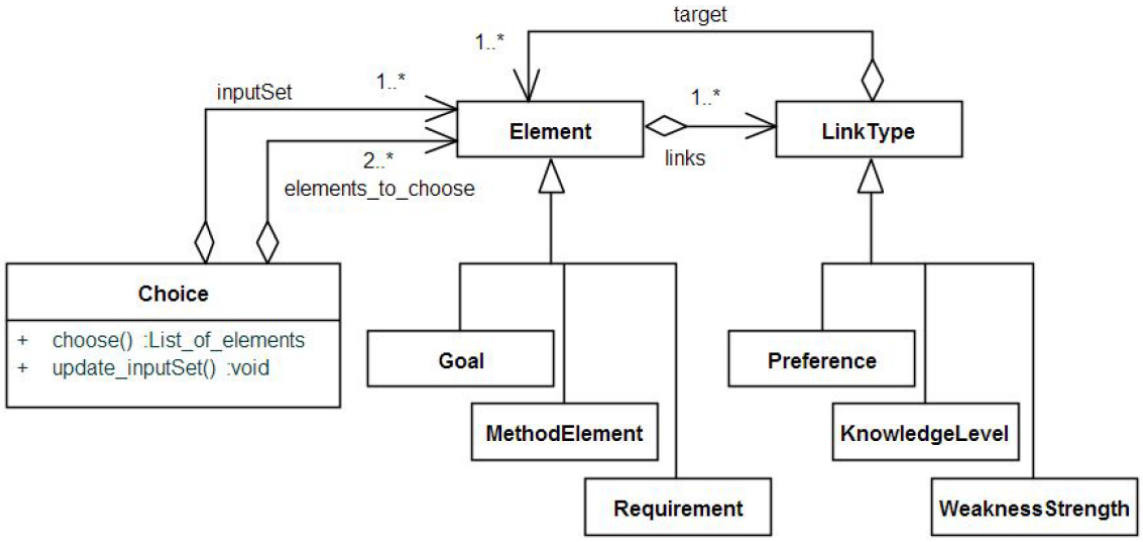


8.3 Tailoring Meta-model

Based on the above mentioned relations between goal models, method/process models and requirements, the proposed meta-model as shown in figure 8.7.

The **Element** abstracts the **Goal** model elements, the **MethodElements** of SPEM, and the **Requirement** elements found in most of the meta-models of requirements management tools like Polarion. The meta-model now allows to connect any element using links of the abstract **LinkType**. Currently the following link types are defined:

Figure 8.7: Meta-model for Development Process Tailoring



- Preferences - Are used to indicate a stakeholders preference for a given element (e. g., a developer might have a preference for a text editor which is part of the guidances of the process model). The preference link can have values between -100% (aversion against an element) up to +100% (this element is vitally important for a stakeholder)

- KnowledgeLevel - This link indicates the level of confidence a stakeholder might have with an element in model. The knowledge level link is divided in two categories. The knowledge as user of an element between 0% (the stakeholder knows nothing about an element) and 50% (the stakeholder knows everything to use and work with an element). The knowledge as teacher for an element my have values between 51% (the stakeholder has taught the use of an element at least once) and 100% (the stakeholder is an experienced teacher with more than 5 years of teaching experience).

- WeaknessStrength - Any element might weaken or strengthen another element. For example, the presence of a requirement for safety in the medical domain will result in high documentation demands what in consequence will strengthen the quality of the final product and at the same time weaken a fast delivery of the product. The weakness/strength link can have values between -100% (the source element will disable/weakens the target element) up to +100% (the source element requires/strengthens the target element. Thus, the target element be comes mandatory)

To work with the product line approach, variabilities are needed, as discussed in the first sections. The variability of the process is modelled with the SPEM content variability types (contributes, extends, replaces, extends and replaces) for the elements of a SPEM model. To trigger this variability of the process model, the **Choice** in the tailoring meta-model is introduced in figure 8.7. This has an input set of elements influencing the choice. This input set will be updated by the ***update_inputSet()*** method whenever the choices are going to be evaluated. This method will search for elements with target links present in the ***elements_to_choose*** list and will update the ***inputSet*** list accordingly. Once the ***update_inputSet()*** method has been executed the ***choose()*** method can follow with its execution to calculate the variant based on the given input elements.

The pseudo-code in figure 8.8 shows how to calculate the choice of elements. First a map of elements and its ranking is created. For all the elements in the list of input

Figure 8.8: choose Pseudo-code

```
1: ERMAP of elements_to_choose,rank
2: for all A in inputSet do
3:     linkSubSet =
4:     getLinkTypesFromTo(A, elements_to_choose)
5:     adjustRank(linkSubSet)
6: end for
7: select_SPEM_based(ermap)
```

elements, the elements which have links to elements in the ***elements_to_choose*** list are listed out. This is accomplished by the ***getLinkTypesFromTo*** method which stores its results in a list of links as subset of the original ***links*** list of the ***Element*** type. This list is then taken as input for the ***adjustRank*** method which in the current version simply adds the values for the preferences, knowledge level and weakness/strength values, to the ***ermap*** rankings discussed in the last section. Finally, a selection of choices based on the rankings and the SPEM models constraints is made. This meta-model can be extended in two ways:

1. First, any additional elements can be added to this meta-model to address future models which need to be integrated in the tailoring process.

2. Second, the link types can be extended by new links needed in the future.

8.4 Summary

This chapter discusses the current state of the product line development domain and the challenges when it comes to the development processes which need to be adapted to the specific needs of the development teams. Tailoring product line development processes has been identified to enable large savings for the domain engineering as well as application engineering phase of product line development projects. For an integrative approach to process line tailoring, a tailoring meta-model is proposed which includes goal models, SPEM process models as wells as requirements. With this model stakeholder specific goals can be used to support binding a variable part of the development process. This support addresses soft factors as well as concrete requirements. Future research work will be spent to further elicit attributes of different domains influencing the development process. In addition the enhancement of the few variable process steps in OpenUP towards a complete process line will also be subject of future research efforts.

Chapter 9

Evaluation of the Proposed Approach

This chapter presents evaluation of the approach. For the evaluation purpose, students experiment was performed. The proposed approach was evaluated by comparing it to other requirements prioritization approaches. The extensibility, usability, comprehensibility and understandability of the approach are analysed by the post experiment survey. Section 9.1 discusses the goals of the experiment, experiment steps are described in section 9.2. Next section 9.3 introduces the case study used in the experiment and requirements elicitation workshop results are presented in section 9.4. Execution of experiment performed in two rounds is explained in the section 9.5. Evaluation of results is presented in section 9.6 and in the end threats to validation of experiment are described in the section 9.7.

9.1 Goals of the Experiment

The goal of the experiment was to analyse the proposed approach against other requirements prioritization methods. The goals of the experiment were identified as:

- Evaluating the proposed approach
- Practical challenges of the approach
- Satisfaction of participants regarding the approach

  Ease of use

  Accuracy

  Understanding

  Reliability

  Comprehensiveness of the approach

  Recommendation of the approach

9.2 Steps of Experiment

The following steps were performed in the process of evaluation:

1. Presentation on requirements prioritization methods
2. Requirements elicitation workshop
3. Presentation on the proposed approach
4. Execution of the experiment
5. Results of the experiment
6. Post experiment survey

9.3 Case Study

A cyclecomputer is a device mounted on a bicycle that calculates display trip information, similar to the instruments in the dashboard of the car. The device with the display or head unit is attached to the handlebar for easy viewing. Some GPS watches can also be used as display. Important aspects of the cyclecomputer are the information it can offer. The information can be displayed differently with the widgets. Widgets are

components of an interface that enables a user to perform function or access a service. Most cycle computers don't display that much information. For cyclecomputer project the idea is to create the widgets that offer the user insightful information in an efficient manner. Users of the device can be any one who owns a bike, be it normal people who like to travel for fun, or the people who use the bike as a transportation medium and are curious about different facts and keep track of statistics like time and distance travelled. Users can also be professional cyclists who take part in competitions on their bikes and wish to improve their performance by taking different facts into account or keeping track of their progress while training.

9.4 Workshop Results

A complete guide of cyclecomputer and elicitation of requirements for cyclecomputer project was provided to students throughout the experiment.

9.4.1 Functional Requirements

After requirements elicitation workshop with students the following requirements were identified:

1. Show speed: The device should be able to keep track of the current speed. The bike is using KMH or MPH units.

2. Show travelled distance: The device should provide the information about the distance travelled. The distance is displayed in KMH or MPH units. It should keep track of total distance and one time distance travelled by the user.

3. Show date and time: Current date and time should be displayed in widget in 12 or 24 hour format.

4. Show stopwatch and countdown: User can keep time with the Stopwatch and the Countdown. Stopwatch time starts from 0 and keeps track of the time till the user pauses the timer or resets it. For the Countdown the user can choose a time and the time will go in reverse order till it reaches 0 and then it will launch an alarm sound.

5. Show temperature: The temperature will also be measured and displayed in a widget. Temperature can be displayed in Fahrenheit or Celsius units.

6. Show humidity: The device measure and displays the current humidity. The humidity is displayed as relative humidity (as %) and absolute humidity (g/m3)

7. Show wind speed: The device should display the speed and direction of the wind in form as of a compass showing from where the wind is blowing and showing the speed in center.

8. Show brake disk temperature: The device will keep track of brake disks temperature.

9. Show wheel RPM: the device should display rotations per minute of the wheel.

10. Show user direction: the device should be able to display the direction the user is heading on a compass.

11. User accounts: The device should have a user management. Many cyclists should be able to use the same device and user specific data needs to be password protected.

12. Transferable to web: For online competition/comparison data should be transferable to a web portal.

13. Online modus: The competition mode should be used 'online' (while riding the bike).

14. Route planning: The device should offer route planning. The planning should be done based on topographic maps. Routing should consider the current weather forecast.

9.4.2 Non-functional Requirements

The following non-functional requirements were considered for the elicited functional requirements by the participants of experiment:

1. Security: Security was considered to be an important factor for following functional requirements User account, Route planning, Online modus, Transferable to web.

2. Safety: Safety requirement was further refined to Send alert and Location update.

   Send alert: It is considered to be interacting with the following functional requirements: Show speed, Route planning, Show travelled distance, Show brake disk temperature, Show temperature, Show wheel RPM, Show wind speed.

   Location update: It is considered to be interacting with the following functional requirements: Route planning, Show travelled distance, Show brake disk temperature, Show humidity, Transferable to web, Show temperature, Show user direction, Online modus.

3. Availability: The device should be available in working condition in/for long routes. Therefore availability is important for Route planning and even if something bad happens (weather update failure etc.,) it should display at least speed, time.

4. User friendliness: User can display four widgets at same time on same page. Three small widgets displaying information such as time, speed, temperature etc., and one bigger widget displaying information such as graphs.

5. Performance: Speed should be updated at every second. Travelled distance should be updated by every meter and weather related information should be updated every minute.

6. Accuracy: Accuracy of distance should be +-5 meter, speed should be within +-2 KMH, temperature should be with +-2 C then accuracy should also consider accurate weather predictions.

9.5 Execution of Experiment

On the experiment day participants were presented with the following information:

- Objective of the experiment and case study (although they were already explained a week prior to the experiment).
- Functional requirements document which contains requirements to be prioritized by the participants (requirements elicited after workshop).
- Non-functional requirements document (requirements elicited after workshop) explaining the dependencies among requirements.
- Development factors (time, effort, risk) to be included in the prioritization process were explained.
- Directly after the experiment participants were asked to rate the approach by using the survey.

9.5.1 Sample Population

Eleven master students participated in the experiment. Students are enrolled in university master degree course" Research in Computer & Systems Engineering. The students in the experiment have a comparable educational background. A week before the actual experiment, presentations were given to the students to introduce case study and also

to explain the proposed approach and other prioritization methods. Therefore these students had same level of expertise on the approaches and of the case study used in the experiment. After that a workshop was organized to elicit the requirements for case study and students prioritize the elicited requirements. Student involvement as sample population consists of the following steps:

- Convincing the students about the need of prioritizing the requirements
- Training of the students participating in the process of prioritizing
- Working with the students to prioritize the requirements

9.5.2 Research Question of Experiment

The following research questions were identified for the experiment:
RQ1 Which approach is taking less time?
RQ2 Which approach is easier to use?
RQ3 Which approach is giving more accurate results?
RQ4 Are ranks produced similar?

9.5.3 First Round

The experiment was conducted in two rounds: Table 9.1 represents the design used in the experiment. In first round one group of six students were asked to prioritize the requirements according to the proposed approach. Each member was given the scales and requirements document to prioritize the requirements. The scale used for functional requirements is given in table6.2. The other group was asked to choose the approach they like from the presented approaches. The students selected Analytic Hierarchy Process (AHP). This reason of this approach might be because of its popularity in literature. The six students in the first group on the average took 8 minutes to complete the first step of proposed approach and in total took 14 minutes to complete all three steps while second group performing AHP took 25 minutes. The time was higher because of large number of comparisons they had to make; for 14 requirements total of 91 comparisons were made. Therefore the time of the second group was considerably higher as compared to first group. Table9.2 represents the participants judgements of the first group of students while B.1 gives the second group participants pair-wise comparisons results using AHP.

Table 9.3 gives the ranks based on AHP pairwise comparisons and Table 9.4 gives the value of distance matrix of AHP while table 9.5 gives the results of proposed approach for prioritization of functional requirements.

Table 9.1: Design Used

Group	Round 1	Round 2
G1	Proposed approach	AHP, 100 dollar, Top ten, Bubble sort, Numerical assignment/Priority group
G2	AHP	Proposed Approach

Table 9.2: Participants Judgements

Sr	Requirements	P1	P2	P3	P4	P5	P6
1	Show speed	High	High	Very High	High	High	High
2	Show travelled distance	High	Medium	Medium	High	Medium	High
3	Show date and time	Medium	Medium	Medium	Medium	Medium	High
4	Show stopwatch and countdown	Medium	Medium	High	Medium	Medium	Medium
5	Show temperature	Medium	Medium	Medium	Medium	Medium	Medium
6	Show humidity	Medium	Medium	Medium	High	High	Medium
7	Show wind speed	Medium	Medium	Low	Low	Low	High
8	Show brake disk temperature	High	Medium	Medium	Medium	High	High
9	Show wheel RPM	Medium	Medium	Medium	Medium	Medium	Medium
10	Show user direction	Medium	Low	Low	Medium	Low	Medium
11	User accounts	Medium	Medium	Medium	High	High	High
12	Transferable to web	Medium	Medium	Medium	High	Low	High
13	Online modus	Medium	Low	Medium	Medium	Low	Low
14	Route planning	High	High	High	Medium	Medium	High

After this non-functional requirements document was given to both groups and they were explained the interactions among requirements. Both groups were asked to prioritize the functional requirements based on these dependencies represented in 9.6.

AHP became difficult to handle as the requirements increased to double digit. Another issue observed with AHP was the consistency ratio: the ideal value of which should be below than 20% but in first attempt when participants made pairwise comparisons, it was 23% and participants were asked to re-arrange their priorities. Rearranging priorities mean they made another round of pairwise comparisons and this time consistency ratio was 9.0% and the total time taken was 22 minutes. Participants using AHP had difficulties regarding handling of interdependencies among requirements

Table 9.3: Ranks based on AHP Comparisons

Requirements	Priority	Rank
Show speed	33.7%	1
Show travelled distance	14.6%	2
Show date and time	11.4%	3
Show stopwatch and countdown	5.7%	4
Show temperature	7.1%	5
Show wind speed	4.9%	6
Show brake disk temperature	4.3%	7
Show humidity	4.2%	8
Show wheel RPM	2.9%	9
Show user direction	2.8%	10
User accounts	2.4%	11
Transferable to web	2.3%	12
Online modus	2.1%	13
Route planning	1.7%	14

Table 9.4: AHP Distance Matrix

Show speed	Show travelled distance	Show date and time	Show stopwatch and countdown	Show temperature	Show humidity	Show wind speed	Show brake disk temperature	Show wheel RPM	Show user direction	User accounts	Transferable to web	Online modus	Route planning
1	8	8	8	8	8	8	6	8	6	6	6	6	6
0.125	1	4	4	4	4	4	4	4	4	4	4	4	7
0.125	0.25	1	4	4	4	4	4	4	4	4	4	4	4
0.125	0.25	0.25	1	2	2	2	2	2	2	3	2	2	3
0.125	0.25	0.25	0.5	1	3	3	3	3	3	4	4	4	4
0.125	0.25	0.25	0.5	0.333333	1	2	2	2	2	2	2	2	2
0.125	0.25	0.25	0.5	0.333333	0.5	1	3	3	3	3	3	3	3
0.166667	0.25	0.25	0.5	0.333333	0.5	0.333333	1	3	3	3	3	3	3
0.125	0.25	0.25	0.5	0.333333	0.5	0.333333	0.333333	1	2	2	2	2	2
0.166667	0.25	0.25	0.5	0.333333	0.5	0.333333	0.333333	0.5	1	2	2	2	2
0.166667	0.25	0.25	0.333333	0.25	0.5	0.333333	0.333333	0.5	0.5	1	2	2	2
0.166667	0.25	0.25	0.5	0.25	0.5	0.333333	0.333333	0.5	0.5	0.5	1	2	2
0.166667	0.25	0.25	0.5	0.25	0.5	0.333333	0.333333	0.5	0.5	0.5	0.5	1	2
0.166667	0.142857	0.25	0.333333	0.25	0.5	0.333333	0.333333	0.5	0.5	0.5	0.5	0.5	1

and also they did not considered development constraints like cost, effort and time in considering prioritizing requirements. Based on participants opinions the AHP method did not provide any information on how to include development constraints (like cost, effort, time) into prioritization. The interactions among requirements were quantified according to 6.5 and table 9.7 represents the output of the first group participants priorities while table 9.8 gives the non-functional requirements priority list.

The participants were asked to only enter the values for both AHP and proposed approach and all the calculations were done by the author of the experiment. One advantage of the re-arraigned priorities in proposed approach is that if there are major differences between two priority lists (functional requirements priority list and list after

Table 9.5: Prioritization after First Step

Requirements	**Normalized Value**	**Rank**
Show speed	0.111	1
Route planning	0.093	2
Show travelled distance	0.085	3
Show brake disk temperature	0.085	4
User accounts	0.085	5
Show humidity	0.078	6
Show date and time	0.070	7
Show stopwatch and countdown	0.070	8
Transferable to web	0.070	9
Show temperature	0.063	10
Show wheel RPM	0.063	11
Show wind speed	0.050	12
Show user direction	0.039	13
Online modus	0.039	14

requirements interactions) they can be revised accordingly.

In last step, participants of first group prioritized requirements incorporating the development constraints time, effort and risk. Table 9.9 gives the final priority list of the first groups participants.

9.5.4 Second Round

In second round of the experiment, the second group was given the task of prioritizing the requirements based on proposed approach and first group was asked to use five approaches of their own choice to prioritize the requirements. The selected approaches were: AHP, 100 dollar test, numerical assignment, Bubble sort and Top-ten requirements. As in first round of experiment, AHP took more time as compared to other approaches and was difficult for students to accommodate dependencies and development constraints for prioritizing using AHP. 100 dollar test was simple approach to use but it is more suitable when numbers of participants are three to five and when there are strict timing constraints. It is not suitable when there are large numbers of requirements and participants are more in numbers. Though it was simple but as in AHP it did not handle dependencies well and it also has scalability issue. Table 9.10 gives the priority list of requirements based on 100 dollar test.

Numerical assignment was another technique used but it had the problem of assigning priorities into groups: High, Medium, Low and each requirement was assigned priority into one of these groups. Individual priority to each requirement was an issue

Table 9.6: Requirements Interaction

Requirements	Score	Security	Send alert	Location update	User friendly	Performance	Accuracy	Availability
Show speed	0.88	Neutral	Help	Neutral	Help	Help	Help	Help
Route planning	0.74	Make	Help	Make	Help	Help	Make	Help
Show travelled distance	0.68	Neutral	Help	Help	Help	Help	Help	Neutral
Show brake disk temperature	0.68	Make	Help	Help	Help	Help	Help	Help
User accounts	0.68	Make	Neutral	Neutral	Help	Help	Neutral	Neutral
Show humidity	0.62	Neutral	Help	Help	Help	Help	Neutral	Neutral
Show date and time	0.55	Neutral	Neutral	Neutral	Help	Help	Neutral	Neutral
Show stopwatch and countdown	0.55	Neutral	Help	Neutral	Help	Neutral	Neutral	Neutral
Transferable to web	0.55	Make	Help	Help	Help	Neutral	Neutral	Neutral
Show temperature	0.50	Neutral	Help	Help	Help	Help	Help	Neutral
Show wheel RPM	0.50	Make	Help	Neutral	Help	Help	Help	Help
Show wind speed	0.37	Neutral	Help	Neutral	Help	Help	Help	Neutral
Show user direction	0.31	Neutral	Help	Help	Help	Help	Neutral	Neutral
Online modus	0.31	Make	Make	Make	Help	Help	Neutral	Neutral

in this approach. Bubble sort was fourth technique used by participants but like AHP it had the pairwise comparisons issue. In bubble sort each requirement was assigned weights and then comparisons were made between the two requirements to prioritize them and the process continued till all requirements were in order. Bubble sort compared two requirements unlike AHP where the relativity was also determined by assigning values from 1 to 9. At the end Top-ten requirements technique was used by the students. In this simply 10 most important requirements were selected but benefit or value of each requirement was not measured (the same issue with other approaches: numerical assignment and bubble sort). In practice this approach is too simple too be selected for prioritization and is also not applicable where different weights are assigned to different stakeholders in the process.

Table 9.7: Requirements Priorities after Interactions

Sr	Requirements	Triangular Fuzzy Number(TFN)	Defuziffication	Normalized Value
1	Online modus	(0.053, 0.101, 0.110)	0.365	0.159
2	Route planning	(0.211, 0.327, 0.37)	0.308	0.134
3	Show brake disk temperature	(0.155, 0.281, 0.34)	0.264	0.115
4	Show speed	(0.075, 0.25, 0.36)	0.233	0.101
5	Show travelled distance	(0.058, 0.194, 0.281)	0.181	0.079
6	Show wheel RPM	(0.085, 0.178, 0.228)	0.167	0.072
7	Show temperature	(0.042, 0.142, 0.207)	0.133	0.058
8	Show humidity	(0.017, 0.141, 0.230)	0.132	0.057
9	Transferable to web	(0.031, 0.133, 0.204)	0.125	0.054
10	User accounts	(0, 0.126, 0.223)	0.118	0.051
11	Show wind speed	(0.010, 0.084, 0.137)	0.078	0.034
12	Show user direction	(0.008, 0.070, 0.115)	0.065	0.028
13	Show date and time	(-0.04, 0.063, 0.157)	0.060	0.026
14	Show stopwatch and countdown	(-0.04, 0.063, 0.157)	0.060	0.026

Table 9.8: Non-functional Requirements Priorities

Sr	Non-functional Requirements	Triangular Fuzzy Number(TFN)	Defuziffication	Normalized Value
1	User friendliness	(0.2, 0.4, 0.5)	0.375	0.212
2	Send alert (Safety)	(0.164, 0.35, 0.457)	0.330	0.186
3	Performance	(0.171, 0.342, 0.457)	0.328	0.185
4	Location update (Safety)	(0.057, 0.242, 0.371)	0.228	0.129
5	Security	(0.057, 0.214, 0.328)	0.203	0.115
6	Accuracy	(0.014, 0.207, 0.35)	0.194	0.109
7	Availability	(-0.085, 0.114, 0.285)	0.107	0.060

Table 9.9: Final Priority List

Sr	Requirements	Triangular Fuzzy Number(TFN)	Defuziffication	Normalized Value
1	Route planning	(0.3, 0.5, 0.7)	0.50	0.180
2	Online modus	(0.2, 0.366, 0.566)	0.425	0.155
3	Show brake disk temperature	(0.2, 0.366, 0.566)	0.307	0.111
4	Show speed	(0.2, 0.366, 0.566)	0.270	0.097
5	Show travelled distance	(0.2, 0.366, 0.566)	0.211	0.076
6	Show wheel RPM	(0.2, 0.366, 0.566)	0.192	0.069
7	Show temperature	(0.2, 0.366, 0.566)	0.155	0.056
8	Show humidity	(0.2, 0.366, 0.566)	0.152	0.055
9	User accounts	(0.2, 0.366, 0.566)	0.136	0.049
10	Transferable to web	(0.3, 0.5, 0.7)	0.108	0.039
11	Show stopwatch and countdown	(0.1, 0.233, 0.433)	0.104	0.037
12	Show wind speed	(0.2, 0.366, 0.566)	0.090	0.032
13	Show date and time	(0.2, 0.366, 0.566)	0.069	0.024
14	Show user direction	(0.43, 0.633, 0.8)	0.044	0.015

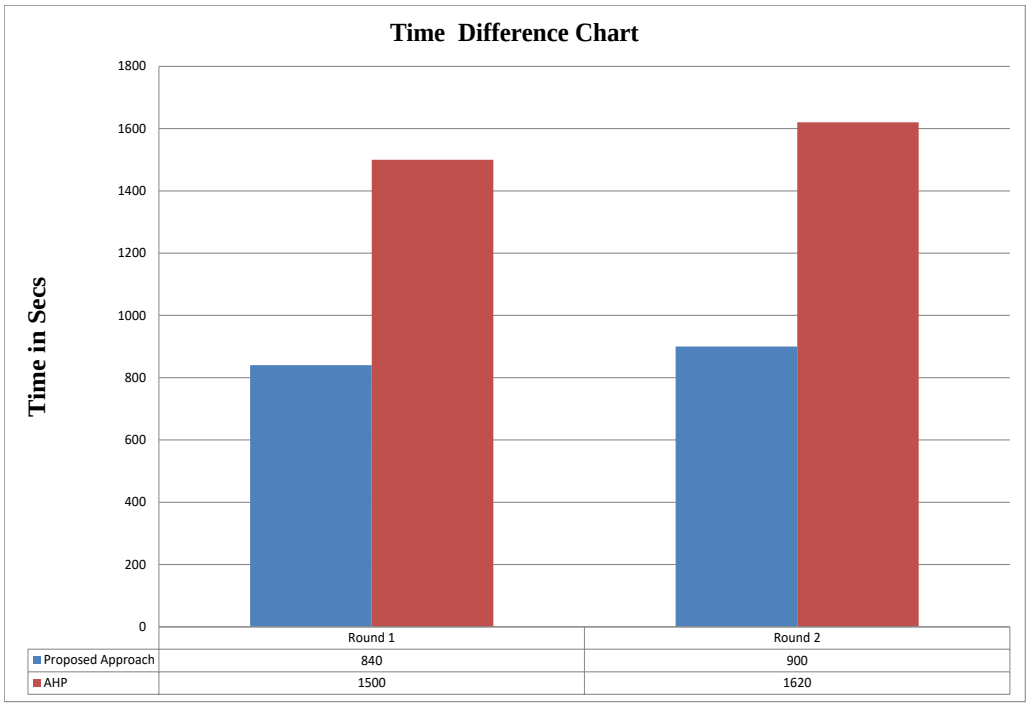
Table 9.10: 100 Dollar Test

Sr	Requirements	$
1	Show speed	6
2	Show travelled distance	5
3	Show date and time	5
4	Show stopwatch and countdown	4
5	Show temperature	4
6	Show humidity	2
7	Show wind speed	2
8	Show brake disk temperature	5
9	Show wheel RPM	4
10	Show user direction	3
11	User accounts	5
12	Transferable to web	6
13	Online modus	6
14	Route planning	5
	Non-functional requirements	
15	User friendliness	7
16	Send alert (Safety)	6
17	Performance	5
18	Location update (Safety)	5
19	Security	4
20	Accuracy	6
21	Availability	5

9.6 Evaluation of Results

The results were examined by answering research questions of the experiment defined in 9.5.2. Regarding first **RQ1**, graph in figure 9.1 gives the time difference of both rounds in seconds between proposed approach and AHP while graph in figure 9.2 gives the time difference of approaches used in second round. As both figures show that proposed approach performed better than AHP, bubble sort in terms of time consumption. Top ten and priority group are better because they do not handle dependencies among the requirements and are too simple to be used in practical applications when there are large number of requirements and large number of stakeholders. The difference in final rankings of these approaches is because of proposed approach take dependencies and development constraints into consideration while other approaches either do not take them into account or they become too complex to handle them.

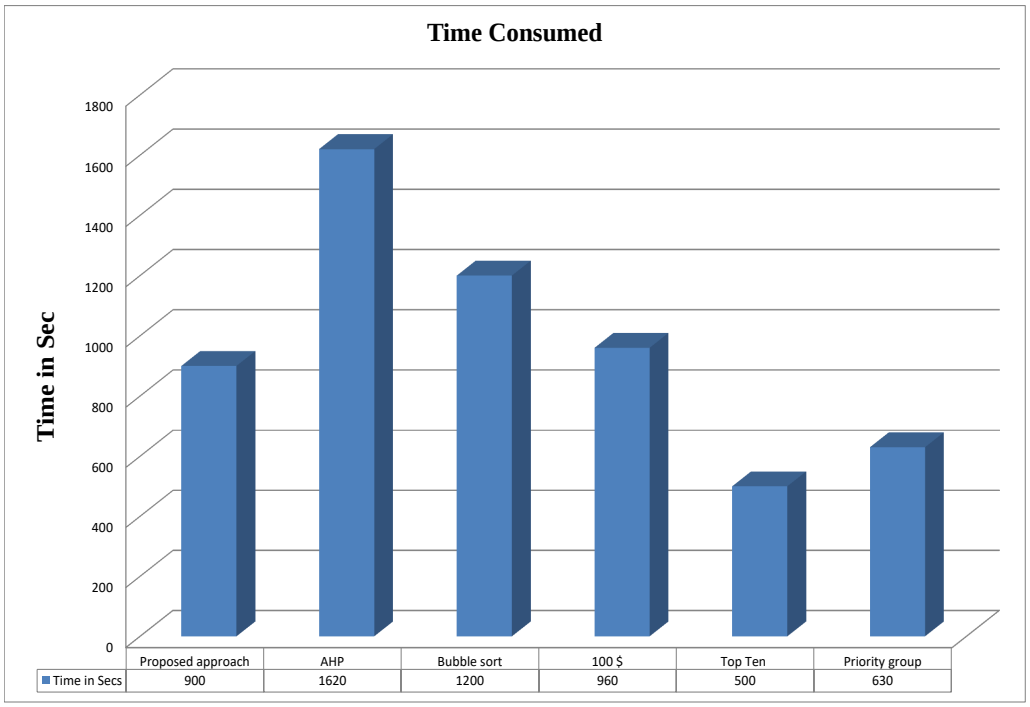
Figure 9.1: Time Difference between AHP and Proposed Approach



Regarding **RQ2** and **RQ3** defined in 9.5.2, a post experiment questionnaire was given to the participants. Graph in figure 9.3 clearly shows that participants trusted the proposed approach over other methods regarding these questions.

To answer **RQ4** the output ranks of both rounds for proposed approach and AHP

Figure 9.2: Time Difference of Used Approaches



were compared. The ranks of AHP for both rounds are given in table 9.11 while change in ranks by proposed approach are given in table 9.12 and table 9.13 respectively for functional and non-functional requirements.

Graph in figure 9.4 represents the changes of both ranks. The change is given in % and in absolute terms. Although the changes in both ranks are subjective in nature but they also represent the understanding difficulty of AHP.

Figure 9.5 gives the ranks produced in both rounds by proposed approach. The ranking of both rounds are more consistent as compared to rankings of AHP and in addition it also gives the ranking of non-functional requirements which are also consistent for both rounds and are shown in figure 9.6.

Further results were examined by evaluating the responses from questionnaire given to the participants who had used the proposed approach and also other approaches. The assessment scale used to verify the participants responses was referred to as Very High: means participant is strongly satisfied to the outcome generated after using the approach; High means participant is satisfied to the outcome; Medium means the participant is satisfied to certain extent about the effectiveness of the approach; Low means the participant is satisfied to some extent and Very Low means the participant

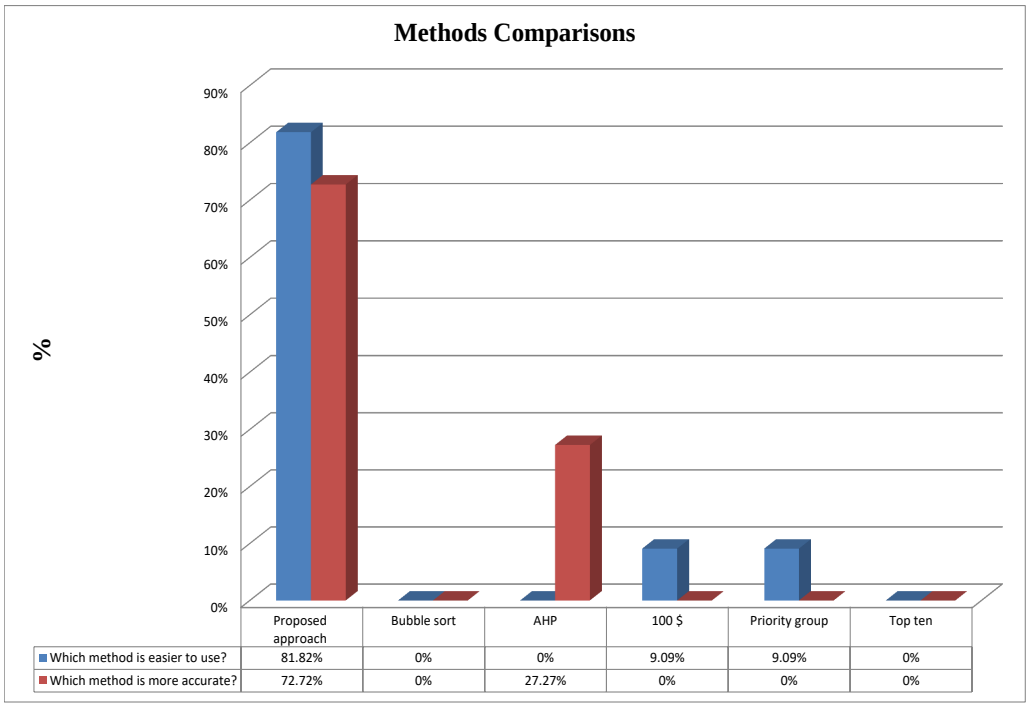
Table 9.11: AHP Ranks of Both Rounds

Sr	**Round 1**	**Sr**	**Round 2**	**Change %**	**Absolute Change**
1	Show speed	1	Show speed	0.00%	0
2	Show travelled distance	2	Route planning	80.00%	12
3	Show date and time	3	Show travelled distance	-6.66%	-1
4	Show stopwatch and countdown	4	Show brake disk temperature	26.66%	4
5	Show temperature	5	User accounts	40.00%	6
6	Show humidity	6	Show humidity	0.00%	0
7	Show wind speed	7	Show date and time	-26.66%	-4
8	Show brake disk temperature	8	Show stopwatch and countdown	-26.66%	-4
9	Show wheel RPM	9	Transferable to web	20.00%	3
10	Show user direction	10	Show temperature	-33.33%	-5
11	User accounts	11	Show wheel RPM	-13.33%	-2
12	Transferable to web	12	Show wind speed	-33.33%	-5
13	Online modus	13	Show user direction	-20.00%	-3
14	Route planning	14	Online modus	-6.66%	-1

Table 9.12: Proposed Approach Ranks of Both Rounds for Functional Requirements

Sr	**Round 1**	**Sr**	**Round 2**	**Change %**	**Absolute Change**
1	Route planning	1	Route planning	0.00%	0
2	Online modus	2	Online modus	0.00%	0
3	Show brake disk temperature	3	Show brake disk temperature	0.00%	0
4	Show speed	4	Show speed	0.00%	0
5	Show travelled distance	5	Show temperature	13.33%	2
6	Show wheel RPM	6	Show travelled distance	-6.66%	-1
7	Show temperature	7	Show wheel RPM	-6.66%	-1
8	Show humidity	8	Show humidity	0.00%	0
9	User accounts	9	User accounts	0.00%	0
10	Transferable to web	10	Transferable to web	0.00%	0
11	Show stopwatch and countdown	11	Show stopwatch and countdown	0.00%	0
12	Show wind speed	12	Show wind speed	0.00%	0
13	Show date and time	13	Show user direction	6.66%	1
14	Show user direction	14	Show date and time	-6.66%	-1

Figure 9.3: Methods Comparisons



is not satisfied to the effectiveness of proposed approach. The outcome of this activity clearly dominated the results of first and second category (Very High and High) that proved the effectiveness of the approach. Table 9.14 shows the survey questionnaire results.

Figure 9.7 gives the results of participants knowledge on requirements engineering and about requirements prioritization techniques. The results depicts that participants have good understanding of requirements engineering and prioritization approaches. Figure 9.8 results depict the satisfaction of participants in terms of ease of use, extensibility, reliability, difficulty, and overall use of the approach. As it is evident from figure 9.8 that participants rated the proposed approach either very high or high for these factors and that shows the applicability of approach. The results in figure 9.9 show the representation of evaluation against questions in terms of improved performance; higher level of customer satisfaction; handling dependencies; higher level of developers involvement and recommendations on using the approach to others. The participants answers show high interest for the approach as they mostly are satisfied to recommend it to others and for handling the mentioned criteria (higher level customer involvement, higher level developers involvement and handling dependencies) in the experiment.

Table 9.13: Proposed Approach Ranks of Both Rounds for Non-functional Requirements

Sr	Round 1	Sr	Round 2	Change %	Absolute Change
1	User friendliness	1	User friendliness	0.00%	0
2	Send alert (Safety)	2	Send alert (Safety)	0.00%	0
3	Performance	3	Performance	0.00%	0
4	Location update (Safety)	4	Security	12.50%	1
5	Security	5	Location update (Safety)	-12.50%	-1
6	Accuracy	6	Accuracy	0.00%	0
7	Availability	7	Availability	0.00%	0

Table 9.14: Survey Questionnaire Results

Sr	Questions	Very High	High	Medium	Low	Very Low
1	How would you rate yourself experience wise in requirements engineering?	36.36%	45.45%	18.18%	0%	0%
2	How would you rate yourself knowledge wise in the requirements prioritisation?	27.27%	45.45%	27.27%	0%	0%
3	How satisfied are you with the approach?	18.18%	63.64%	18.18%	0%	0%
4	How satisfied are you with the ease of use?	27.27%	54.54%	18.18%	0%	0%
5	How satisfied are you with the extensibility of the approach?	18.18%	81.82%	0.00%	0%	0%
6	How satisfied are you with the understanding difficulty of the approach?	9.09%	54.55%	36.36%	0%	0%
7	How satisfied are you with the reliability of the approach?	9.09%	63.64%	27.27%	0%	0%
8	How satisfied are you with the approach in terms of handling desired prioritisation?	18.18%	63.64%	18.18%	0%	0%
9	What are the chances that you will use this approach?	27.27%	54.55%	18.18%	0%	0%
10	What are the chances that you will recommend this approach to friend or colleague?	36.36%	63.64%	0.00%	0%	0%

Figure 9.4: AHP Ranks of Both Rounds

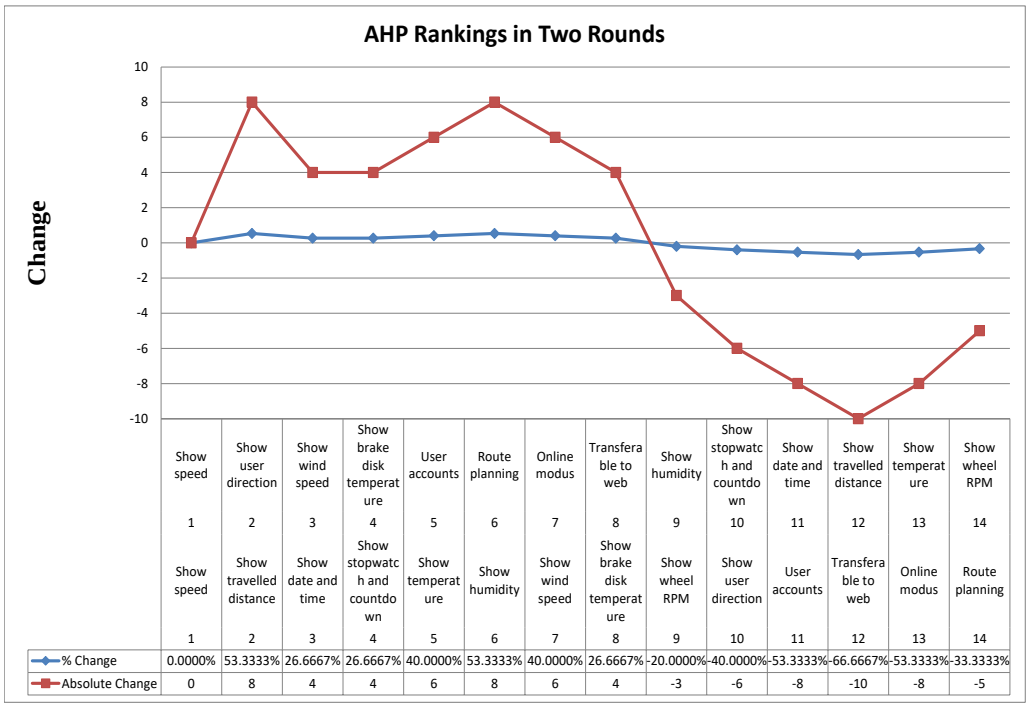


Figure 9.5: Proposed Approach Ranks of Both Rounds

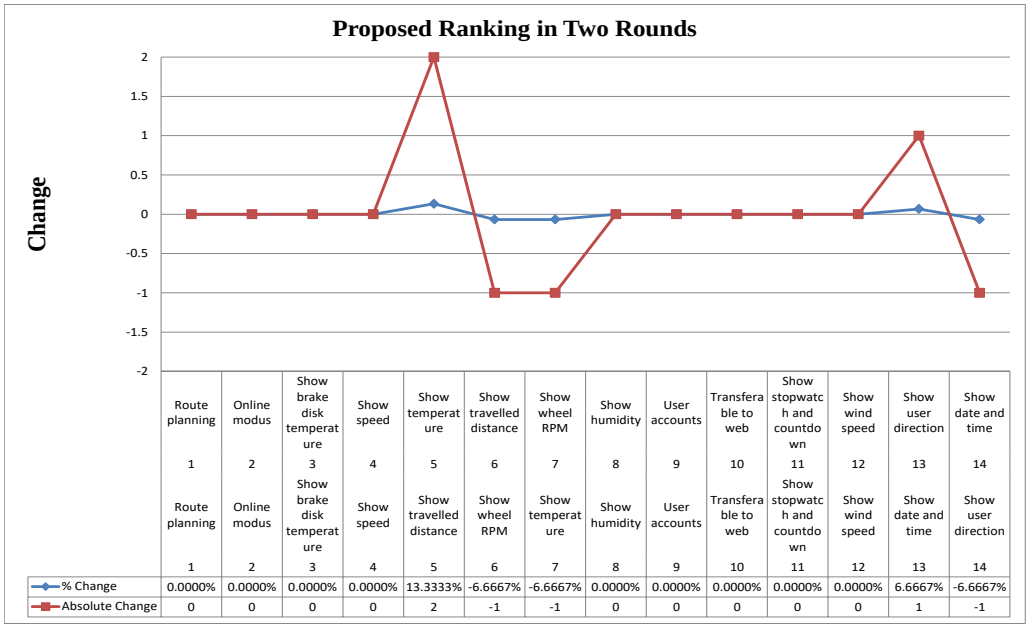


Figure 9.6: NFR Ranks of Both Rounds

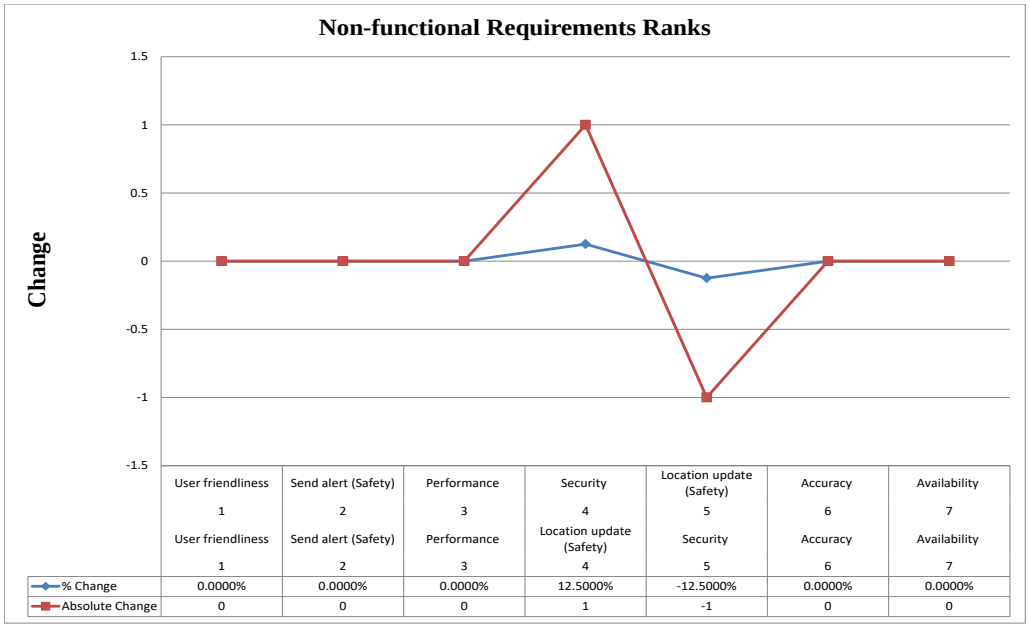


Figure 9.7: Participants Expertise in RE

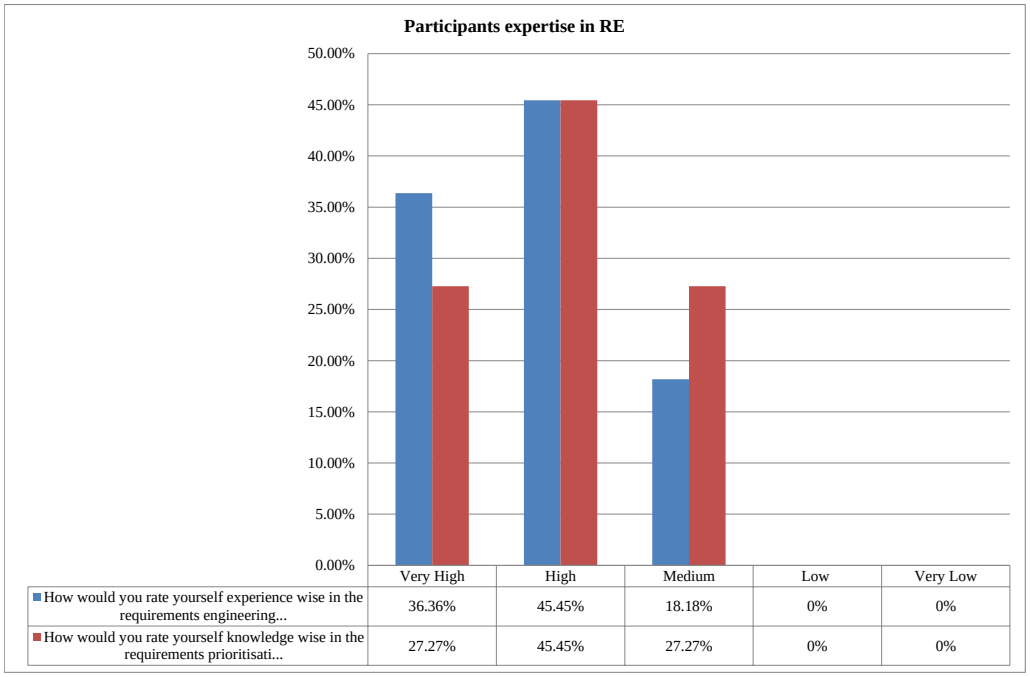
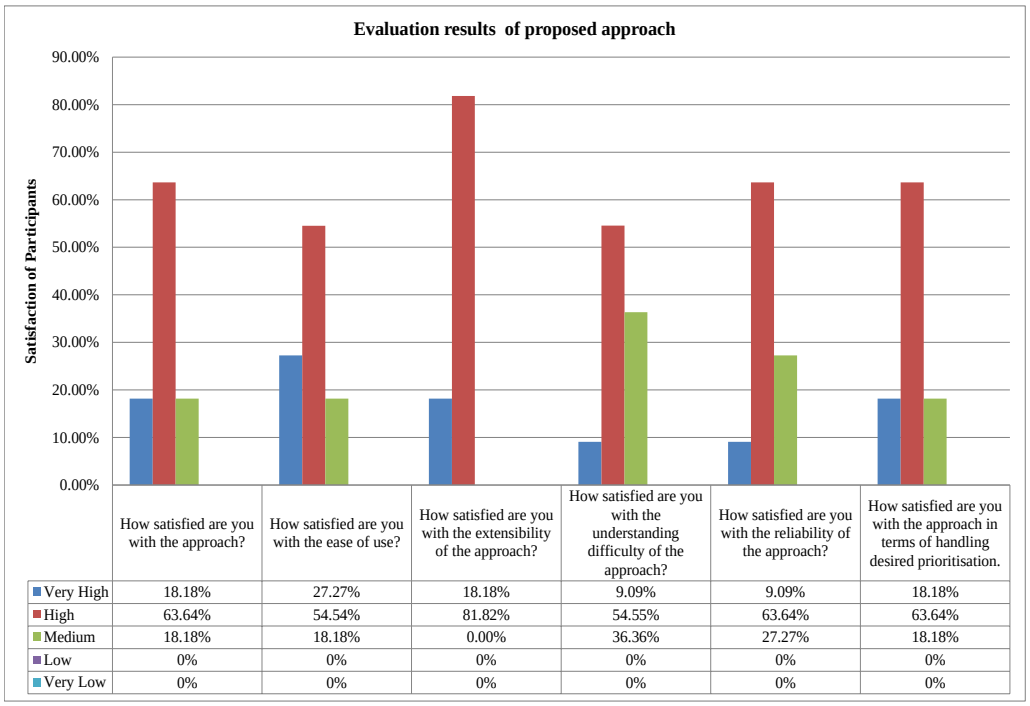
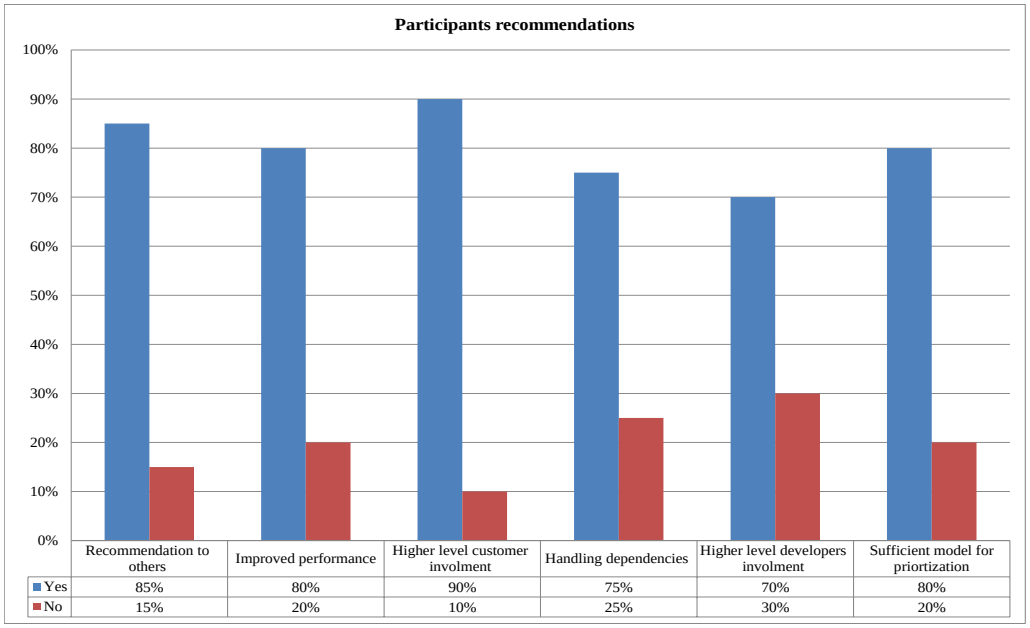


Figure 9.8: Evaluation Results



	How satisfied are you with the approach?	How satisfied are you with the ease of use?	How satisfied are you with the extensibility of the approach?	How satisfied are you with the understanding difficulty of the approach?	How satisfied are you with the reliability of the approach?	How satisfied are you with the approach in terms of handling desired prioritisation.
Very High	18.18%	27.27%	18.18%	9.09%	9.09%	18.18%
High	63.64%	54.54%	81.82%	54.55%	63.64%	63.64%
Medium	18.18%	18.18%	0.00%	36.36%	27.27%	18.18%
Low	0%	0%	0%	0%	0%	0%
Very Low	0%	0%	0%	0%	0%	0%

Figure 9.9: Participants Recommendations About Approach



	Recommendation to others	Improved performance	Higher level customer involment	Handling dependencies	Higher level developers involment	Sufficient model for priortization
Yes	85%	80%	90%	75%	70%	80%
No	15%	20%	10%	25%	30%	20%

9.7 Validation of Experiment

It is always vital to analyse the possible threats in an experiment in order to validate the results obtained. Validity has been a key challenge in scientific research and it means to check that experiment fulfils its intended purposes. The objective of the experiment here was to illustrate the application of the proposed approach and to make comparisons to other prioritization methods. Following potential threats should be kept in mind when analysing the results of this experiment:

9.7.1 Conclusion Validity

A threat to conclusion validity can lead to reach an incorrect conclusion about relationships in the experiment. However, small sample size and low (or medium) numbers of requirements were issues in this experiment to highlight any significant relationships. Although the participants have good understanding on requirements prioritization approaches and results are significant but this significance is limited because of these issues and it is still regarded as a partial threat to the evaluation. Usefulness of the approach signifies the effectiveness and future experimentation practice.

9.7.2 Internal Validity

Internal validity is about finding the causal relationship in the experiments that is where one can define certain outcome is because of certain treatment. To minimize any causal relationships in the experiment, it was performed in two rounds. To eliminate any biasses in the experiment all participants in both rounds used the same material and shared the same presentations on the prioritization approaches.

9.7.3 Construct Validity

Construct validity is about how well the operationalization of experiment is performed. In experiment several variables were selected to measure the ”satisfaction” of the approach for example, ease of use, understanding, accuracy, extensibility and reliability. With regard to these variables, participants were asked open questions in likert scale (qualitative manner) rather than a quantitative to make the participants more comfortable for expressing their opinions freely. However, the experiments are performed independently from a software project. This a potential threat in the experimentation. However the main objective of the experiment was to know the applicability of the approach and understanding the differences and problems of other prioritizing approaches.

9.7.4 External Validity

External validity is about how well the results of the experiment can be generalized. The external validity is always an important issue in the student experiments and the same threat is in this experiment but since the students were well prepared for the experiment therefore the results can not be dismissed just for this reason. The results obtained in the experiment about the applicability of the approach are promising but still they cannot be generalized for other applications in other environments. Never the less evaluation results does provide valuable understandings to advantages and disadvantages of the different methods and the practical challenges one may face during the prioritization of requirements.

9.8 Summary

This chapter discussed the evaluation of the proposed approach based on student experiments where the proposed approach was compared against other requirements prioritization methods. The variables used to check the effectiveness of the approach are: ease of use, reliability, understandability and extensibility of the approach compared to other approaches. The results extracted from the experiments are presented and in the end possible implication to validity of results are presented.

Chapter 10

Conclusions and Outlook

This chapter presents the conclusion of this thesis. Section 10.1 discusses how the goals identified in section 1.2 are achieved in this thesis. Then the chapter concludes with an outlook into future research directions based on the results of this thesis.

10.1 Thesis Goals and Acquirement

In section 1.2 research goals of this thesis were presented. For each of these goals, short answers based on the work presented in this thesis are given.

Goal 1: Evaluating goal methods/frameworks for requirements analysis.
Chapter 3 presented the GORE frameworks from literature and a complete goal oriented analysis of requirements is described. The findings from these frameworks that are relevant for this work are highlighted in table 3.3.

Goal 2 and 3: Establishing the means for stakeholders representation into GORE and providing a prioritization scheme for high level goals based on stakeholder preferences.
Once goal analysis has been performed and the functional level goals are obtained, stakeholder opinions are gathered using linguistic terms. Their opinions are then quantified using the fuzzy numbers and crisp values are obtained by defuzzification process defined in section 6.4.5. This gives the first priority list of operational functional goals based on stakeholders representations.

Goal 4: Providing an integration of quality models into goal models.
Chapter 5 presents classification of quality model and comparison of these quality models. An integration of goal models to quality models is presented. The quality

models and goal-quality model integration helps to identify quality requirements related to goals and to find the dependencies among these requirements.

Goal 5 and 6: Establishing the influences (positive or negative) of quality goals and measuring the impact of quality goals on each other and of high level goals.
To establish the influences, contributions links of goal models are used and section 6.4.7.2 describes how to measure the impact of them. The same defuzzifications process 6.4.5 is used to quantify the measure. After that the prioritization obtained as an answer to research goal 2 and 3 is updated based on dependencies. At that point two prioritize lists are obtained one for functional requirements and one for non-functional requirements.

Goal 7: Involving development factors into prioritization and selection of requirements based on cost constraints.
Section 6.4.8 describes to update the prioritization list based on development factors and algorithm 6.10 gives the final selection of requirements while fulfilling the cost constraints.

Goal 8: Alternative selection of system solutions based on proposed approach.
Chapter 7 provides the extension of the proposed approach for alternative selection. TOPSIS method was integrated into proposed approach to evaluate all the alternative solutions. Score obtained by TFN and defuzzification process 6.4.5 are used as weighing criteria by TOPSIS to rank best alternative.

Goal 9: Provide the tailoring of product line development process based on goals.
For tailoring product line development process, a tailoring meta-model is presented in section 8.2. This meta-models integrates goal models, SPEM process models as well as requirements. With this model stakeholder specific goals can be used to support binding a variable part of the development process.

10.2 Future Work

Regarding the approach different aspects of improvements can be discussed for future work. Though the approach was applied on a case study where different quality goals are integrated to functional goals but still an integration of complete set of quality goals could be a future work.

Though the approach is tested to handle the dependencies among functional and non-functional requirements but it still needs an evaluation when the interactions among non-functional requirements are too many or they are increased in depth. For practitioner acceptance of this approach further evaluation for the industry projects is required.

The goal based tailoring of product line development process presented in this thesis could further be enhanced to feature oriented modelling. The goal structures could be mapped to feature models where feature models are parametrized according to initial goals. By this feature models will be enhanced with new structures, and will introduce another level of variability. Each level of variability will results in a design supporting the road map of product with its future changes without breaking the software architecture.

Appendices

Appendix A

Implementation and Modelling

A.1 Implementation of case Study

Cycle Computer Requirements Analysis

Performing the requirements analysis for Cycle Computer project consist of:

- Finding requirements, analyzing them
- Incorporating stakeholder choices, developers opinions
- Using fuzzy logic to prioritize these requirements

Basic settings

```
echo = TRUE  # make code visible
library(ggplot2)
library(gplots)
library(lattice)
library(RColorBrewer)
library(corrplot)
library(FactoMineR)
library(data.table)
library(plyr)
library(reshape)
library(scales)
require(gridExtra)
setwd("C:/Users/dell pc/Desktop/course/Implementation/implementation")
```

Data Processing

```
data<- "SH.csv"
data<-read.csv(data)

requirements<- data[,1:2]
shInputs<- as.data.frame(data[,3:5])
dim(data)
```

```
## [1] 16  5
```

```
head(data)
```

```
##                          Goals Sub.goals.till.leaf.level.goals SHO1 SHO2
## 1 EntertainmentServiceSatisfied                             Mic    H   VH
## 2                                                  Data storage    H   VH
## 3                                                 Audio service   VL   VH
## 4      CompitionServieSatisifed                  User accounts   VH    H
## 5                                            Transferable to web   VH    H
## 6                                                   Online modus   VH    H
##   SHO3
## 1    H
## 2    H
## 3   VL
## 4    H
## 5    H
## 6    H
```

```
summary(data)
```

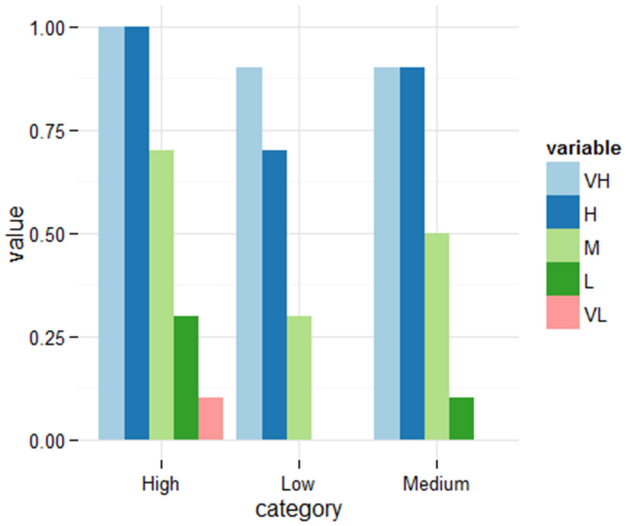
```
##                          Goals        Sub.goals.till.leaf.level.goals
##                            :12   Audio service       : 1
##  CompitionServieSatisifed  : 1   Calories consumption: 1
##  EntertainmentServiceSatisfied : 1   Data storage        : 1
##  TourManagementServiceSatisfied: 1   Fitness level       : 1
##  TrainingServiceSatisfied  : 1   Initial checkups    : 1
##                                  Mic                 : 1
##                                  (Other)             :10
##  SHO1   SHO2   SHO3
```

```
##  VL:1   VL:5   VL:3
##
##
##
##
```

Summary of data shows there are four high level goals which are further refined till leaf level goals are achieved i.e., the goals that are directly assignable to agents. Sixteen leaf level goals against are obtained from four high level goals. These goals are presented to three stakeholders and get their opinions are recorded against each leaf level goal.

Stakeholder opinions are obtained in linguistic terms for simplicity reasons. Their opinions are then converted into fuzzy numbers.

```
VH<-c(0.9,0.9,1.0)
H<-c(0.7,0.9,1.0)
M<-c(0.3,0.5,0.7)
L<-c(0.0,0.1,0.3)
VL<-c(0.0,0.0,0.1)
likertScale<- data.frame(VH,H,M,L,VL)
attr(likertScale, "row.names")<-c("Low","Medium","High")
likertScale$category <- row.names(likertScale)
mdfr <- melt(likertScale, id.vars = "category")
(p <- ggplot(mdfr, aes(category, value, fill = variable)) +
geom_bar(stat='identity',position=position_dodge()) +
scale_fill_brewer(palette="Paired")+theme_minimal())
```



Map Stakeholder opinions to numeric values

The stakeholder opinions from above table are mapped to fuzzy number.

```
mapOpinions<-list(VH=VH, H=H, M=M, L=L, VL=VL)
```

The following function is used to extract the stakeholder opinions from data and replace them to numeric values.

```
    relevel <- function(shInputs, levelmap) {
                    shInputs[] <- lapply(shInputs, function(x)
levelmap[as.numeric(x)]);shInputs
}

newValues<-relevel(shInputs, mapOpinions)
newValues
```

```
##             SHO1          SHO2          SHO3
## 1  0.9, 0.9, 1.0 0.7, 0.9, 1.0 0.9, 0.9, 1.0
## 2  0.9, 0.9, 1.0 0.7, 0.9, 1.0 0.9, 0.9, 1.0
## 3  0.3, 0.5, 0.7 0.7, 0.9, 1.0 0.3, 0.5, 0.7
## 4  0.7, 0.9, 1.0 0.9, 0.9, 1.0 0.9, 0.9, 1.0
## 5  0.7, 0.9, 1.0 0.9, 0.9, 1.0 0.9, 0.9, 1.0
## 6  0.7, 0.9, 1.0 0.9, 0.9, 1.0 0.9, 0.9, 1.0
## 7  0.7, 0.9, 1.0 0.9, 0.9, 1.0 0.9, 0.9, 1.0
## 8  0.9, 0.9, 1.0 0.9, 0.9, 1.0 0.7, 0.9, 1.0
## 9  0.7, 0.9, 1.0 0.3, 0.5, 0.7 0.7, 0.9, 1.0
## 10 0.9, 0.9, 1.0 0.3, 0.5, 0.7 0.7, 0.9, 1.0
## 11 0.9, 0.9, 1.0 0.3, 0.5, 0.7 0.7, 0.9, 1.0
## 12 0.7, 0.9, 1.0 0.9, 0.9, 1.0 0.3, 0.5, 0.7
## 13 0.9, 0.9, 1.0 0.9, 0.9, 1.0 0.9, 0.9, 1.0
## 14 0.9, 0.9, 1.0 0.3, 0.5, 0.7 0.3, 0.5, 0.7
## 15 0.7, 0.9, 1.0 0.9, 0.9, 1.0 0.9, 0.9, 1.0
## 16 0.9, 0.9, 1.0 0.3, 0.5, 0.7 0.9, 0.9, 1.0
```

Calculating TFN

Triangular Fuzzy Number (TFN) is calculated for each leaf level goal against stakeholder opinions and the results are saved in new data set with new TFN column.

```
TFN <- Reduce('+', lapply(newValues, function(x) do.call(rbind,
x)))/ncol(newValues)
newTFN<-newValues
newTFN$TFN <- do.call(paste, c(as.data.frame(TFN), sep=", "))
newTFN
```

```
##             SHO1          SHO2          SHO3
## 1  0.9, 0.9, 1.0 0.7, 0.9, 1.0 0.9, 0.9, 1.0
## 2  0.9, 0.9, 1.0 0.7, 0.9, 1.0 0.9, 0.9, 1.0
## 3  0.3, 0.5, 0.7 0.7, 0.9, 1.0 0.3, 0.5, 0.7
## 4  0.7, 0.9, 1.0 0.9, 0.9, 1.0 0.9, 0.9, 1.0
## 5  0.7, 0.9, 1.0 0.9, 0.9, 1.0 0.9, 0.9, 1.0
## 6  0.7, 0.9, 1.0 0.9, 0.9, 1.0 0.9, 0.9, 1.0
## 7  0.7, 0.9, 1.0 0.9, 0.9, 1.0 0.9, 0.9, 1.0
## 8  0.9, 0.9, 1.0 0.9, 0.9, 1.0 0.7, 0.9, 1.0
## 9  0.7, 0.9, 1.0 0.3, 0.5, 0.7 0.7, 0.9, 1.0
## 10 0.9, 0.9, 1.0 0.3, 0.5, 0.7 0.7, 0.9, 1.0
## 11 0.9, 0.9, 1.0 0.3, 0.5, 0.7 0.7, 0.9, 1.0
## 12 0.7, 0.9, 1.0 0.9, 0.9, 1.0 0.3, 0.5, 0.7
## 13 0.9, 0.9, 1.0 0.9, 0.9, 1.0 0.9, 0.9, 1.0
## 14 0.9, 0.9, 1.0 0.3, 0.5, 0.7 0.3, 0.5, 0.7
## 15 0.7, 0.9, 1.0 0.9, 0.9, 1.0 0.9, 0.9, 1.0
## 16 0.9, 0.9, 1.0 0.3, 0.5, 0.7 0.9, 0.9, 1.0
##                                          TFN
## 1                  0.833333333333333, 0.9, 1
## 2                  0.833333333333333, 0.9, 1
## 3  0.433333333333333, 0.633333333333333, 0.8
## 4                  0.833333333333333, 0.9, 1
## 5                  0.833333333333333, 0.9, 1
## 6                  0.833333333333333, 0.9, 1
## 7                  0.833333333333333, 0.9, 1
## 8                  0.833333333333333, 0.9, 1
## 9  0.566666666666667, 0.766666666666667, 0.9
## 10 0.633333333333333, 0.766666666666667, 0.9
## 11 0.633333333333333, 0.766666666666667, 0.9
## 12 0.633333333333333, 0.766666666666667, 0.9
## 13                                0.9, 0.9, 1
## 14               0.5, 0.633333333333333, 0.8
## 15                 0.833333333333333, 0.9, 1
## 16               0.7, 0.766666666666667, 0.9
```

```
dtp<-data.table(TFN)
```

Defuzzification process

The defuzzification process is performed to convert TFN values to crisp value. These values are stored in csv file and are extracted below:

```
defuzi<- read.csv("SH1.csv")
alpha<-defuzi[,6]
beta<-defuzi[,7]

alpha<-data.table(alpha)
alpha

##     alpha
##  1:   0.3
##  2:   0.6
##  3:   0.7
##  4:   0.8
##  5:   0.9
##  6:   0.3
##  7:   0.2
##  8:   0.6
##  9:   0.9
## 10:   0.5
## 11:   0.8
## 12:   0.4
## 13:   0.1
## 14:   0.3
## 15:   0.4
## 16:   0.8

beta<-data.table(beta)
beta

##     beta
##  1:  0.5
##  2:  0.8
##  3:  0.1
##  4:  0.2
##  5:  0.8
##  6:  0.7
##  7:  0.7
##  8:  0.9
##  9:  0.9
## 10:  0.5
## 11:  0.2
## 12:  0.9
## 13:  0.5
## 14:  0.7
## 15:  0.7
## 16:  0.4

dtp[,DFN:={beta;alpha; lb<-V1+(V2-V1)*beta; rb<-V3+(V2-V3)*beta;
(alpha*rb+(1-alpha)*lb)}]
```

Normalizing the values

The normalization is performed on scores obtained after the defuzzification process. The normalization is performed by following equation:

```
sumDFN<-sum(dtp[,dtp$DFN])
dtp[,NDFN:=DFN/sumDFN]
```

Writing results as external file

```
 write.table(dtp, file = "interactive.csv", row.names = FALSE)
```

After that subgoals, their defuzzification and normalized defuzzification values are combained.

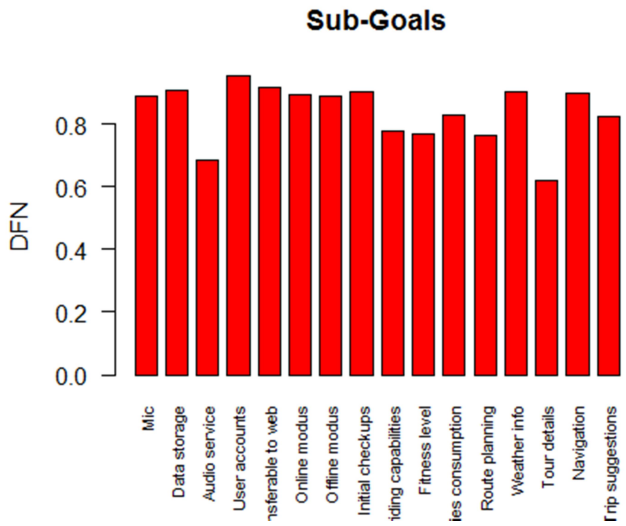
```
ndfn<-read.csv("interactive.csv")
data<-read.csv("SH.csv")
ndfn1<-cbind(data[,2], ndfn)
names(ndfn1)[1]<-paste("subgoals")
write.table(ndfn1, file = "subgoal-values.csv", row.names = FALSE)
```

Graph for the above results:

```
figs2<- read.csv("interactiveCombined.csv")
figs2
```

```
##                                     DFN       NDFN
## Mic                           0.8916667 0.06640848
## Data storage                  0.9066667 0.06752563
## Audio service                 0.6843333 0.05096696
## User accounts                 0.9533333 0.07100122
## Transferable to web           0.9166667 0.06827040
## Online modus                  0.8950000 0.06665674
## Offline modus                 0.8900000 0.06628435
## Initial checkups              0.9033333 0.06727738
## Technical riding capabilities 0.7766667 0.05784365
## Fitness level                 0.7666667 0.05709888
## Calories consumption          0.8306667 0.06186540
## Route planning                0.7640000 0.05690028
## Weather info                  0.9050000 0.06740150
## Tour details                  0.6203333 0.04620044
## Navigation                    0.9000000 0.06702912
## Trip suggestions              0.8226667 0.06126958
```

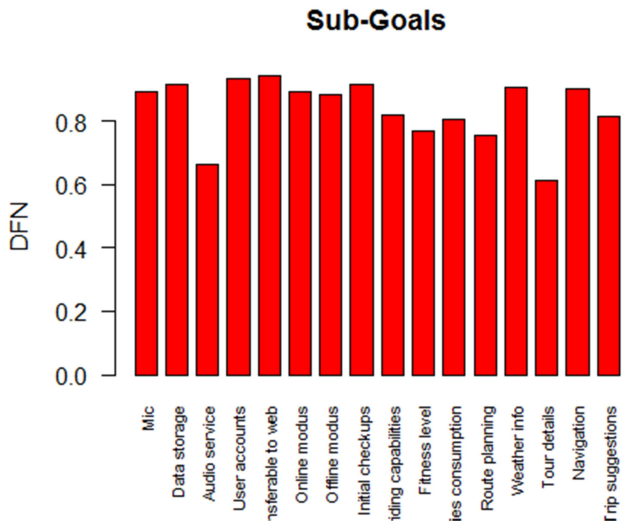
```
figs1<-data.matrix(figs2)

barplot(figs1[-17,1],ylab=colnames(figs1)[1],col=rainbow(1),main="Sub-
Goals",las=2, cex.names=.7,  space = 0.4)
```



If one considers the preference values (i.e., developers opinion) and ignore the risk tolerance value, then following equation is used for defuzzification:

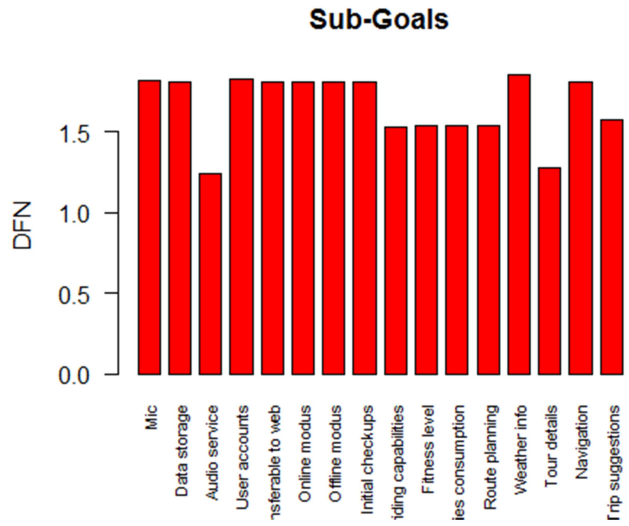
```
preferenceValue<-data.table(TFN)
preferenceValue[,DFN:={(alpha*V3+V2+(1-alpha)*V1)*0.5}]
sumDFN<-sum(preferenceValue[,preferenceValue$DFN])
preferenceValue[,NDFN:=DFN/sumDFN]
write.table(preferenceValue, file = "preference.csv", row.names = FALSE)
ndfn<-read.csv("preference.csv")
data<-read.csv("SH.csv")
ndfn1<-cbind(data[,2], ndfn)
```

```
names(ndfn1)[1]<-paste("subgoals")
write.table(ndfn1, file = "preferenceCombined.csv", row.names = FALSE)
figs2<- read.csv("prefCombined.csv")
figs1<-data.matrix(figs2)
barplot(figs1[-17,1],ylab=colnames(figs1)[1],col=rainbow(1),main="Sub-
Goals",las=2, cex.names=.7,  space = 0.4)
```



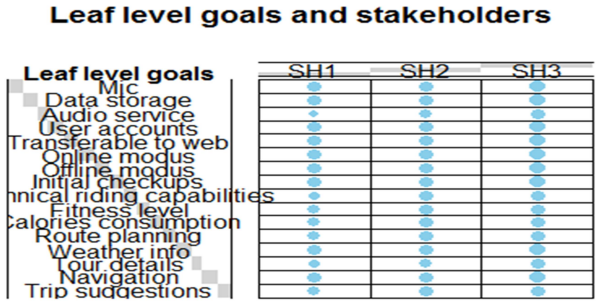
If only the risk tolerance values are considered and ignore the developers opinion, then defuzzification process is applied by following equation:

```
toleranceValue<-data.table(TFN)
toleranceValue[,DFN:={(V3+(V2-V3)*beta)+(V1+(V2-V1)*beta)}]
sumDFN<-sum(toleranceValue[,toleranceValue$DFN])
toleranceValue[,NDFN:=DFN/sumDFN]
write.table(toleranceValue, file = "tolerance.csv", row.names = FALSE)
ndfn<-read.csv("tolerance.csv")
data<-read.csv("SH.csv")
ndfn1<-cbind(data[,2], ndfn)
names(ndfn1)[1]<-paste("subgoals")
write.table(ndfn1, file = "toleranceCombined.csv", row.names = FALSE)
figs2<- read.csv("tolCombined.csv")
figs1<-data.matrix(figs2)
barplot(figs1[-17,1],ylab=colnames(figs1)[1],col=rainbow(1),main="Sub-
Goals",las=2, cex.names=.7,  space = 0.4)
```



Visualizing leaf level goals and stakeholders using graphical matrix

```
lgoalSh<- read.csv("SH_Input.csv")
dt <- as.table(as.matrix(lgoalSh))
balloonplot(t(dt), main ="Leaf level goals and stakeholders",label.digits=2,
xlab ="", ylab="Leaf level goals",label = FALSE, show.margins = FALSE)
```



Calculating distance matrix

The distance matrix (a kind of correlation or dissimilarity matrix). the distance matrix is used to compare the requirements. To get the final distance matrix, one have to calculate row margins and column margins.

Row margins and column margins are calculated by following methods respectively:

```
dist<- read.csv("dist.csv")
row.sum <- apply(dist, 1, sum)
head(row.sum)
```

```
##                 Mic        Data storage       Audio service
##            3.533333            4.133333            2.666667
##       User accounts Transferable to web        Online modus
##            3.733333            4.433333            3.733333
```

```
col.sum <- apply(dist, 2, sum)
head(col.sum)
```

```
##               SH1               SH2               SH3 Developer.opinion
##          11.66667          13.20000          15.10000           8.60000
##    Risk.tolerance
##           9.50000
```

```
#grand total
n <- sum(dist)
n
```

```
## [1] 58.06667
```

Row profile

Since the requirements are arranged as rows and to compare requirements, the row profile is calculated by taking each row point and dividing by the sum of all row points.

```
row.profile <- dist/row.sum
head(row.profile)
```

```
##                          SH1       SH2       SH3 Developer.opinion
## Mic                0.2358491 0.2547170 0.2830189        0.08490566
## Data storage       0.2016129 0.2177419 0.2419355        0.14516129
## Audio service      0.1625000 0.2375000 0.3000000        0.26250000
## User accounts      0.2232143 0.2410714 0.2678571        0.21428571
## Transferable to web 0.1879699 0.2030075 0.2255639        0.20300752
## Online modus       0.2232143 0.2410714 0.2678571        0.08035714
##                    Risk.tolerance
## Mic                     0.14150943
## Data storage            0.19354839
## Audio service           0.03750000
## User accounts           0.05357143
## Transferable to web     0.18045113
## Online modus            0.18750000
```

The average row profile is computed by dividing the column sum to grand total. Column sum and grand total are already calculated above.

```
average.rp <- col.sum/n
average.rp
```

```
##               SH1               SH2               SH3 Developer.opinion
##         0.2009185         0.2273249         0.2600459         0.1481056
##    Risk.tolerance
##         0.1636051
```

Distance (or similarity) between requirements

To compare 2 requirements, one need to compute the squared distance between their profiles e.g., the distance between mic and data storage is calculated as follow:

```
Mic.p <- row.profile["Mic",]
DStorage.p <- row.profile["Data storage",]
# Distance between Mic and Data storage
d2 <- sum(((Mic.p - DStorage.p)^2) / average.rp)
d2
```

```
## [1] 0.05940535
```

The requirements with less distance between them are closer to each other as compared to requirements with more distance value. The distance from the average profile for all the requirements (rows) is given below.

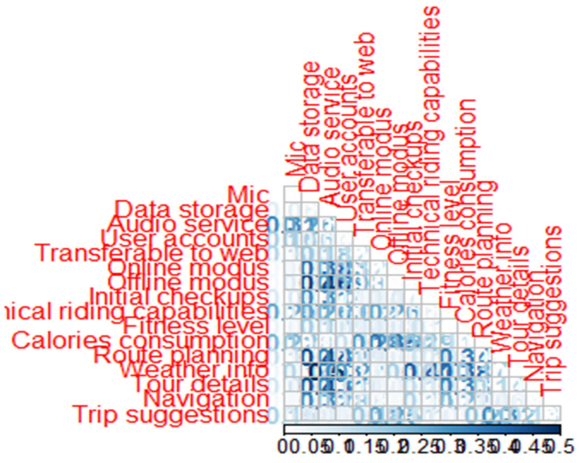
```
d2.row <- apply(row.profile, 1,function(row.p, av.p){sum(((row.p -
av.p)^2)/av.p)}, average.rp)
as.matrix(round(d2.row,3))
```

```
##                              [,1]
## Mic                         0.041
## Data storage                0.007
## Audio service               0.199
```

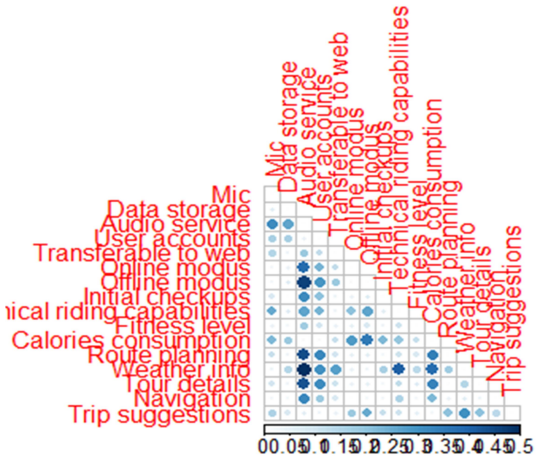
```
## User accounts                  0.107
## Transferable to web            0.030
## Online modus                   0.038
## Offline modus                  0.070
## Initial checkups               0.018
## Technical riding capabilities 0.089
## Fitness level                  0.002
## Calories consumption           0.126
## Route planning                 0.059
## Weather info                   0.128
## Tour details                   0.054
## Navigation                     0.017
## Trip suggestions               0.056
```

To get the distance matrix, squared distance is computed between each row profile and the other rows.

```
## average.profile: average profile
dist.matrix <- function(data, average.profile){
           mat <- as.matrix(t(data))
               n <- ncol(mat)
               dist.mat<- matrix(NA, n, n)
               diag(dist.mat) <- 0
               for (i in 1:(n - 1)) {
                    for (j in (i + 1):n) {
                        d2 <- sum(((mat[, i] - mat[, j])^2) /
average.profile)
                        dist.mat[i, j] <- dist.mat[j, i] <- d2
                    }
               }
        colnames(dist.mat) <- rownames(dist.mat) <- colnames(mat)
        dist.mat
}

# Distance matrix
dist.mat <- dist.matrix(row.profile, average.rp)
dist.mat <-round(dist.mat, 2)
# Visualizing the matrix
corrplot(dist.mat, type="lower", method = "number",  is.corr = FALSE)
```




```
corrplot(dist.mat, type="lower", method = "circle",  is.corr = FALSE)
```



Stakeholders analysis

Since the stakeholders are arranged in columns, the Column profile is used for stakeholders analysis in the same way as the row profiles.

```
col.profile <- t(dist)/col.sum
col.profile <- as.data.frame(t(col.profile))
head(col.profile)
```

```
##                            SH1        SH2        SH3 Developer.opinion
## Mic                 0.07142857 0.06818182 0.06622517        0.03488372
## Data storage        0.07142857 0.06818182 0.06622517        0.06976744
## Audio service       0.03714286 0.04797980 0.05298013        0.08139535
## User accounts       0.07142857 0.06818182 0.06622517        0.09302326
## Transferable to web 0.07142857 0.06818182 0.06622517        0.10465116
## Online modus        0.07142857 0.06818182 0.06622517        0.03488372
##                     Risk.tolerance
## Mic                      0.05263158
## Data storage             0.08421053
## Audio service            0.01052632
## User accounts            0.02105263
## Transferable to web      0.08421053
## Online modus             0.07368421
```

After that average column profile is calculated as follow:

```
row.sum <- apply(dist, 1, sum)
# average column profile= row sums/grand total
average.cp <- row.sum/n
head(average.cp)
```

```
##                 Mic        Data storage       Audio service
##          0.06084960          0.07118255          0.04592423
##       User accounts Transferable to web        Online modus
##          0.06429392          0.07634902          0.06429392
```

Distance (similarity) between stakeholders

To compare stakeholders, the squared distance between their column profiles e.g., is computed the distance between stakeholder 1 and stakeholder 2 is calculated as follow:

```
SH1.p <- col.profile[, "SH1"]
SH2.p <- col.profile[, "SH2"]
d2 <- sum(((SH1.p - SH2.p)^2) / average.cp)
d2
```

```
## [1] 0.00780916
```

The average profile for all stakeholders is computed:

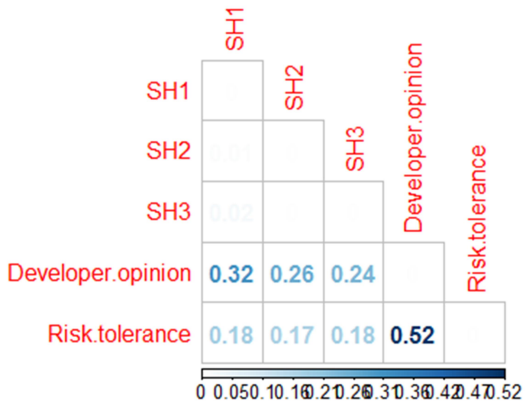
```
d2.col <- apply(col.profile, 2, function(col.p, av.p){sum(((col.p -
av.p)^2)/av.p)},         average.cp)
round(d2.col,3)
```

```
##               SH1               SH2               SH3 Developer.opinion
##             0.022             0.007             0.007             0.209
##    Risk.tolerance
##             0.138
```

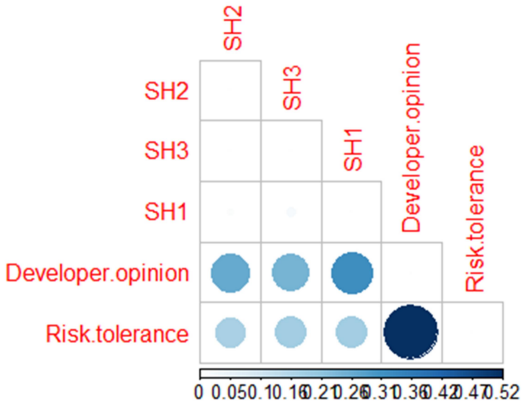
```
# Distance matrix
dist.mat <- dist.matrix(t(col.profile), average.cp)
dist.mat <-round(dist.mat, 2)
dist.mat
```

```
##                    SH1  SH2  SH3 Developer.opinion Risk.tolerance
## SH1               0.00 0.01 0.02              0.32           0.18
## SH2               0.01 0.00 0.00              0.26           0.17
## SH3               0.02 0.00 0.00              0.24           0.18
## Developer.opinion 0.32 0.26 0.24              0.00           0.52
## Risk.tolerance    0.18 0.17 0.18              0.52           0.00
```

```
# Visualize the matrix
corrplot(dist.mat, type="lower", method = "number", is.corr = FALSE)
```




```
corrplot(dist.mat, type="lower", order="hclust", is.corr = FALSE)
```


SH2
SH3
SH1
Developer.opinion
Risk.tolerance
0 0.05 0.1 0.16 0.21 0.26 0.31 0.36 0.42 0.47 0.52

A.2 Alternatives Selection Using a variant of TOPSIS

The Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) is a multi criteria decision analysis method. It is used to compare a set of alternatives based on weighted scores of each criterion. We already had computed the scores of each criterion and we will use these score for the selection of best alternative. We randomly selected four criteria and four alternatives to compare against these criteria. The selected criteria are: Navigation, audio service, route planning, user accounts.

```
# criteria
c1<-"Navigation";c2<- "audioService";c3<-"routePlanning";c4<-"userAccounts"
criteria<-c(c1,c2,c3,c4)
# weights against each criterion, should be equal to length(criteria)
weight<-c(0.9,0.68,0.76,0.95)

#set of alternatives
Al1<-"a1";Al2<-"a2";Al3<-"a3";Al4<-"a4"
alternative<-c(Al1,Al2,Al3,Al4)

#scores values to be assigned
fulfilled<-0.9;partiallyFulfilled<-0.7;minimunFulfilled<-0.3;notFulfilled<-
0.01

# score vector of length= length(criteria)*length(alternative)
sc<-c(fulfilled,notFulfilled,notFulfilled,minimunFulfilled,minimunFulfilled,
      partiallyFulfilled,fulfilled,fulfilled,notFulfilled,partiallyFulfilled,
      minimunFulfilled,partiallyFulfilled,notFulfilled,minimunFulfilled,
      partiallyFulfilled,fulfilled)
```

Topsis variant

Topsis variant implementation to select best alternative according to given criteria.

```
topsisVariant<-function (criteria = NULL, critweights=NULL,
alternativesId=NULL, scores = NULL)
{
        if (missing(criteria))
                stop("'criteria' should be in a vector")
        if (missing(critweights))
                stop("'criteria' should be in a vector")
        if(length(criteria)!= length(critweights))
                stopr("Each criteria must have a weight")
        if (missing(alternativesId))
                stop("'alternativesId' should be in a vector")
        if (missing(scores))
                stop("'Score Values' are missing")
        if (!( is.vector(scores)))
                stop("'Score Values' should be in a vector")
        if(length(scores)!= length(alternativesId)*length(criteria))
                 stop("Scores are not entered for each alternative against
```

```
each criteria")

        ## filter the scores into decision matrix according to the given
alternatives and criteria

        if (!is.null(alternativesId))
                if (!is.null(criteria))     {
                        decision<- matrix(scores, length(alternativesId),
byrow=TRUE)
                        rownames(decision)<-alternativesId
                        colnames(decision)<-criteria
                }

        # create Rij matrix
        R <- matrix(nrow = nrow(decision), ncol = ncol(decision))
        for (i in 1:nrow(decision)) {
                for (j in 1:ncol(decision)) {
                        R[i, j] <- decision[i, j]/sqrt(sum(decision[, j]^2))
                }
        }
        rownames(R)<-alternativesId
        colnames(R)<-criteria

        ## create Vij matrix
        w<-diag(critweights)
        V<- R %*%w

        ## make positivel ideal solution
        psMax<- apply(V,2,max)

        ##subtracting max from matrix column wise
        sp<-sweep(V[,],2,psMax)

        ##square, sum, square root
        sp<-sp^2
        sp<-rowSums(sp)
        sp<-sqrt(sp)

        ## buid negative ideal solution
        # select min from each col
        nsMin<-apply(V,2, min)

        #subtracting min from matrix column wise
        sn<-sweep(V[,],2,nsMin)

        #square, sum, square root
        sn<-sn^2
        sn<-rowSums(sn)
        sn<-sqrt(sn)
```

```
        ## closet to ideal solution
        id<-sp+sn
        ideal<- sn/id
        ideal<- as.data.frame(ideal)
        print("Relative Closness to Ideal Solution")
        print(ideal)
        # best solution
        best<-max(ideal)
        print("Best Solution is:")
        print(best)
}
# Results

d<-topsisVariant(criteria,weight,alternative,sc)
```

```
## [1] "Relative Closness to Ideal Solution"
##        ideal
## a1 0.5054113
## a2 0.6047311
## a3 0.3712308
## a4 0.4088185
## [1] "Best Solution is:"
## [1] 0.6047311
```

A.3 Regression Modelling

Experimental Set-up

```
## load required packages
library(dplyr)
library(data.table)
library(triangle)
library(ggplot2)
library(manipulate)
library(UsingR)
```

1. From our model equation, we require alpha and beta values crosspoding to stakeholder opinions and risk tolerance respectively.

```
set.seed(1000)
# Produce Alpha and Beta values

alpha<- rtriangle(1000, 0, 1)
alpha<-data.table(alpha)
beta<- rtriangle(1000, 0, 1)
beta<-data.table(beta)
```

2. Producing data for set of requirements

```
Requ <- data.table("Requ"=rep(letters[1:10],100))
```

3. Assigning the sacles used to obtain stakeholder opinions on requirements.

```
VH <- c(0.9, 1.0, 1.0)
H <- c(0.7, 0.9 ,1.0)
M <- c(0.3, 0.5, 0.7)
L <- c(0.0, 0.1, 0.3)
VL <- c(0.0, 0.0, 0.1)
```

4. Populating stakeholders opinion against the set of requirements. We selected 10 stakeholders for requirements opinion accumulation.

```
SH1<- data.table("SH1"= sample(rep(list(VH,H,M,L,VL),200),replace=TRUE))
SH2<- data.table("SH2"= sample(rep(list(H,VH,L,M,VL),200),replace=TRUE))
SH3<- data.table("SH3"= sample(rep(list(H,H,M,L,L),200),replace=TRUE))
SH4<- data.table("SH4"= sample(rep(list(H,H,VH,M,L),200),replace=TRUE))
SH5<- data.table("SH5"= sample(rep(list(H,H,VH,M,L),200),replace=TRUE))
SH6<- data.table("SH6"= sample(rep(list(VH,VH,H,M,M),200),replace=TRUE))
SH7<- data.table("SH7"= sample(rep(list(H,H,M,M,M),200),replace=TRUE))
SH8<- data.table("SH8"= sample(rep(list(M,M,VL,VL,L),200),replace=TRUE))
SH9<- data.table("SH9"= sample(rep(list(M,H,L,VL,VL),200),replace=TRUE))
SH10<- data.table("SH10"= sample(rep(list(M,H,L,VL,L),200),replace=TRUE))
```

5. Combining requirements and stakeholder opinions.

```
total <- cbind(Requ,SH1,SH2,SH3,SH4,SH5,SH6,SH7,SH8,SH9,SH10)
head(total)
```

```
##    Requ         SH1         SH2         SH3         SH4         SH5
## 1:    a 0.7,0.9,1.0 0.9,1.0,1.0 0.0,0.1,0.3 0.3,0.5,0.7 0.7,0.9,1.0
## 2:    b 0.3,0.5,0.7 0.0,0.1,0.3 0.7,0.9,1.0 0.7,0.9,1.0 0.7,0.9,1.0
## 3:    c 0.7,0.9,1.0 0.7,0.9,1.0 0.7,0.9,1.0 0.0,0.1,0.3 0.3,0.5,0.7
## 4:    d 0.0,0.1,0.3 0.0,0.1,0.3 0.3,0.5,0.7 0.7,0.9,1.0 0.7,0.9,1.0
## 5:    e 0.9,1.0,1.0 0.9,1.0,1.0 0.3,0.5,0.7 0.9,1.0,1.0 0.3,0.5,0.7
## 6:    f 0.7,0.9,1.0 0.0,0.1,0.3 0.3,0.5,0.7 0.9,1.0,1.0 0.9,1.0,1.0
##            SH6         SH7         SH8         SH9        SH10
## 1: 0.9,1.0,1.0 0.3,0.5,0.7 0.3,0.5,0.7 0.7,0.9,1.0 0.0,0.1,0.3
## 2: 0.9,1.0,1.0 0.3,0.5,0.7 0.0,0.0,0.1 0.0,0.0,0.1 0.0,0.1,0.3
## 3: 0.9,1.0,1.0 0.7,0.9,1.0 0.3,0.5,0.7 0.3,0.5,0.7 0.0,0.1,0.3
## 4: 0.9,1.0,1.0 0.7,0.9,1.0 0.3,0.5,0.7 0.3,0.5,0.7 0.3,0.5,0.7
## 5: 0.3,0.5,0.7 0.3,0.5,0.7 0.0,0.1,0.3 0.3,0.5,0.7 0.0,0.1,0.3
## 6: 0.3,0.5,0.7 0.7,0.9,1.0 0.3,0.5,0.7 0.0,0.0,0.1 0.3,0.5,0.7
```

6. Performing TFN analysis

```
data<-total[,2:11, with=FALSE]

TFN <- Reduce('+', lapply(data, function(x) do.call(rbind, x)))/ncol(data)
data$TFN <- do.call(paste, c(as.data.table(TFN), sep=", "))
head(data)
```

```
##            SH1         SH2         SH3         SH4         SH5         SH6
## 1: 0.7,0.9,1.0 0.9,1.0,1.0 0.0,0.1,0.3 0.3,0.5,0.7 0.7,0.9,1.0 0.9,1.0,1.0
## 2: 0.3,0.5,0.7 0.0,0.1,0.3 0.7,0.9,1.0 0.7,0.9,1.0 0.7,0.9,1.0 0.9,1.0,1.0
## 3: 0.7,0.9,1.0 0.7,0.9,1.0 0.7,0.9,1.0 0.0,0.1,0.3 0.3,0.5,0.7 0.9,1.0,1.0
## 4: 0.0,0.1,0.3 0.0,0.1,0.3 0.3,0.5,0.7 0.7,0.9,1.0 0.7,0.9,1.0 0.9,1.0,1.0
## 5: 0.9,1.0,1.0 0.9,1.0,1.0 0.3,0.5,0.7 0.9,1.0,1.0 0.3,0.5,0.7 0.3,0.5,0.7
## 6: 0.7,0.9,1.0 0.0,0.1,0.3 0.3,0.5,0.7 0.9,1.0,1.0 0.9,1.0,1.0 0.3,0.5,0.7
##            SH7         SH8         SH9        SH10              TFN
## 1: 0.3,0.5,0.7 0.3,0.5,0.7 0.7,0.9,1.0 0.0,0.1,0.3 0.48, 0.64, 0.77
## 2: 0.3,0.5,0.7 0.0,0.0,0.1 0.0,0.0,0.1 0.0,0.1,0.3 0.36, 0.49, 0.62
## 3: 0.7,0.9,1.0 0.3,0.5,0.7 0.3,0.5,0.7 0.0,0.1,0.3 0.46, 0.63, 0.77
## 4: 0.7,0.9,1.0 0.3,0.5,0.7 0.3,0.5,0.7 0.3,0.5,0.7 0.42, 0.59, 0.74
## 5: 0.3,0.5,0.7 0.0,0.1,0.3 0.3,0.5,0.7 0.0,0.1,0.3 0.42, 0.57, 0.71
## 6: 0.7,0.9,1.0 0.3,0.5,0.7 0.0,0.0,0.1 0.3,0.5,0.7 0.44, 0.59, 0.72
```

```
dtp<-data.table(TFN)
V2<-dtp$V2 # middle of TFN
```

7. Calculating the fuzziness

```
# formula is H-L/2M (H, M, L stands for right, middle and left values)

dtp[,fn:={(V3-V1)/2*V2}]
fn<-dtp$fn
```

8. Defuzzifying TFN to produce the ranks of requirements. Alpha and beta values are created above and crossponds to developers opinions and risk tolerance values.

```
dtp[,DFN:={beta;alpha; lb<-V1+(V2-V1)*beta; rb<-V3+(V2-V3)*beta;
(alpha*rb+(1-alpha)*lb)}]
head(dtp)
```

```
##      V1   V2   V3      fn       DFN
## 1: 0.48 0.64 0.77 0.09280 0.6251102
## 2: 0.36 0.49 0.62 0.06370 0.4931981
## 3: 0.46 0.63 0.77 0.09765 0.5899158
## 4: 0.42 0.59 0.74 0.09440 0.6045936
## 5: 0.42 0.57 0.71 0.08265 0.5686491
## 6: 0.44 0.59 0.72 0.08260 0.5544840
```

```
DFN<-dtp$DFN
```

9. Normalization process

```
sumDFN<-sum(dtp[,dtp$DFN])
dtp[,NDFN:=DFN/sumDFN]
NDFN<-dtp$NDFN
head(dtp)
```

```
##      V1   V2   V3      fn       DFN         NDFN
## 1: 0.48 0.64 0.77 0.09280 0.6251102 0.0012165886
## 2: 0.36 0.49 0.62 0.06370 0.4931981 0.0009598614
## 3: 0.46 0.63 0.77 0.09765 0.5899158 0.0011480932
## 4: 0.42 0.59 0.74 0.09440 0.6045936 0.0011766593
## 5: 0.42 0.57 0.71 0.08265 0.5686491 0.0011067041
## 6: 0.44 0.59 0.72 0.08260 0.5544840 0.0010791360
```

10. Producing list of priortized requirements

```
data1<-total[,1 ,with=FALSE]
ndfn1<-cbind(data1, dtp)
head(ndfn1)
```

```
##    Requ   V1   V2   V3      fn       DFN         NDFN
## 1:    a 0.48 0.64 0.77 0.09280 0.6251102 0.0012165886
## 2:    b 0.36 0.49 0.62 0.06370 0.4931981 0.0009598614
## 3:    c 0.46 0.63 0.77 0.09765 0.5899158 0.0011480932
## 4:    d 0.42 0.59 0.74 0.09440 0.6045936 0.0011766593
## 5:    e 0.42 0.57 0.71 0.08265 0.5686491 0.0011067041
## 6:    f 0.44 0.59 0.72 0.08260 0.5544840 0.0010791360
```

```
fig<-ndfn1[,.(fn, DFN)]
fi<-ndfn1[,.(V2, DFN)]
fig<-as.data.table(fig)
fi<-as.data.table(fi)
```

Now, we have a data set where each requirement is ranked based on stakeholders opinion.

Regression modeling

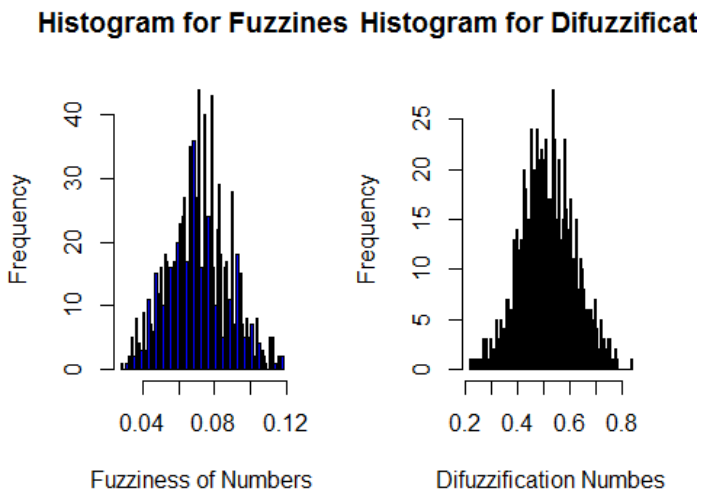
Regression analysis is used to perform estimations amnong variables. The relationship among the variables is of dependent-independent nature. It helps to understand how the value of dependent variable changes when one or more indepedent vaiables values are changed.

In our setup, we have following kind of questions to be answered:

- How fuzziness of TFN predicts the rank of requirements
- What is the mean relationship between fuzinees of numbers and rank of requirements
- Investigation the variations in ranks of requirements
- Quatifying the impact
- Predicting the rank of requirements based on our experiment results

Let's first have a look at data. To perform regression analysis, one important assumption is that data should be normally distributed. We checked this assumption with a histogram.

```
par(mfrow=c(1,2) )
hist(fig$fn,col="blue",main="Histogram for Fuzziness",xlab = "Fuzziness of
Numbers",breaks=100 )
hist(fig$DFN,col="blue",main = "Histogram for
Difuzzification",xlab="Difuzzification Numbes",breaks=100)
```



Regression analysis assumption let x and y represents fuzinees value of TFN and DFN i.e., ranks of requirements respectively. clearly we can see both graphs follow the normal distribution and therefore normality assumption of regression analysis holds for our data.

Explaining the ranks using Fn

We want to develop a model that allows us to make prediction about what value of y (rank) will be for any given value of x (Fn).

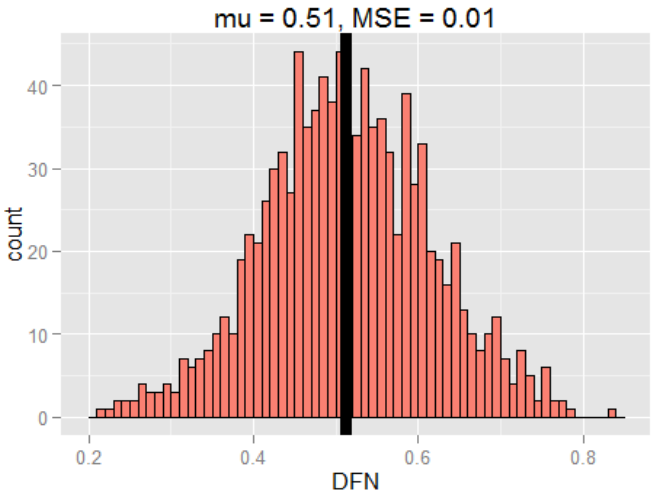
To find pattern in the data, we used linear regression modeling. For prediction purpose, the least square method is used. In linear regression modeling, we use all of the data to calculate a straight line which is used to predict ranks based on fuzziness of numbers (Fn values). Since Fn value is used to predict rank (DFN) value of requirements, Fn is called an 'Explanatory Variable' while DFN is called a 'Response Variable' in our model.

Finding the middle To determine the physical center of the histogram, we will find the middle of distribution by Least squares (LS) method by using following formula:

$$\sum_{i=1}^{n} (Y_i - \mu)^2$$

where Y_i is the rank of each requirement for $i = 1, \ldots, n$ and μ is the mean of sample data $\overline{Y}$.

```
meanDfn <- mean(fig$DFN)
mse <- mean((fig$DFN - meanDfn)^2)
    g <- ggplot(fig, aes(x = DFN)) + geom_histogram(fill = "salmon",
        colour = "black", binwidth =0.01 )
    g <- g + geom_vline(xintercept = meanDfn, size = 3)
    g <- g + ggtitle(paste("mu = ", round(meanDfn,2), ", MSE = ",
        round(mse, 2), sep = ""))
    g
```



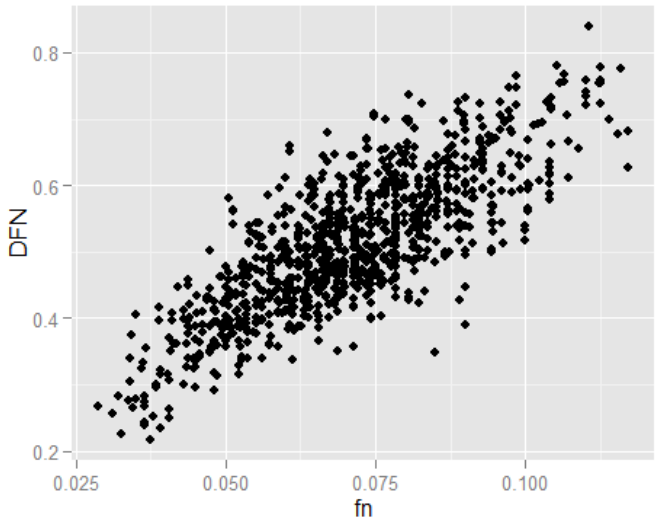
LS method was used because the data holds the following properties:

- Relationship between x and y is linear
- The distribution of x and y is normal
- Variance of x is much less than the variance of y
- Y is the respone to x i.e., the value of y is a function of x; y = f(x)
- we need to show the trends on scatter plot
- The sample correlation is high

Note, here x and y represents Fn and DFN values respectively.

Now, we will show the comparison of ranks against Fn values by using the scatter plot. Next figure shows a scatterplot of DFN (on the Y-axis) versus Fn (on the X-axis):

```
g <- ggplot(fig, aes(x= fn, y=DFN))+geom_point()
g
```



Regression analysis assumption The points seem to fall around a certain pattern, sloping upwards, suggesting linear relationship between independent and dependent variable. Therefore, Linearity assumption of regression analysis holds for our data.

Regression to origin

To draw conclusion using LS method, we need to fit the line to data and that line should minimize the Sum of Square Error/residuals (SSE). SSE is simply the square of residual/error

values. Residual or error value is defined as the difference between the best fit line and the observed value. The formula for the line is:

$$\sum_{i=1}^{n} (Y_i - X_i\beta)^2$$

which is interpreted as, for particular value of X_i, the slope β should minimizes the sum of the squared vertical distances of the points to the line which in fact are the residuals.

Plotting the line to data

Fitting the best line The best fit line must pass through the centroid i.e., the mean of each vairable x and y. In the code below, we subtracted the means so that the origin is the mean of the DFN and fn values. Since we are interested in the slop, we subtracted 1 from lm function to get rid of the intercept.

```
lm1 <- lm(I(DFN - mean(DFN))~ I(fn - mean(fn)) - 1, data = fig)
```

Applying LS to regression line Let Y_i be the i^{th} DFN value and X_i be the i^{th} fn value. Consider finding the best line

$$\text{DFNvalue} = \beta_0 + \text{fnvalue}\beta_1$$

By LS formula to minimize following equation over β_0 and β_1

$$\sum_{i=1}^{n} Y_i - (\beta_0 + \beta_1 X_i)^2$$

Now, minimizing the above equation will minimize the sum of the squared distances between the fitted line at the fn value ($\beta_1 X_i$) and the observed DFN value (Y_i).This is actually least square equation of line $Y = \beta_0 + \beta_1 X$. For data points (X_i, Y_i) where Y_i is the outcome obtains the line $Y = \hat{\beta}_0 + \hat{\beta}_1 X$ where

$$\hat{\beta}_1 = \text{Cor}(Y, X)\frac{\text{Sd}(Y)}{\text{Sd}(X)} \quad \hat{\beta}_0 = \overline{Y} - \hat{\beta}_1\overline{X}$$

$\hat{\beta}_1$ has the units of Y/X, $\hat{\beta}_0$ has the units of Y. The line passes through the point $(\overline{X}, \overline{Y})$. The slope of the regression line with X as the outcome and Y as the predictor is $\text{Cor}(Y, X)\text{Sd}(X)/\text{Sd}(Y)$. When data is normalized (centered and scaled), $\frac{X_i - \overline{X}}{\text{Sd}(X)}, \frac{Y_i - \overline{Y}}{\text{Sd}(Y)}$, the slope is $\text{Cor}(Y, X)$ because $\text{Sd}(X)$ and $\text{Sd}(Y)$ is 1.

```
# Double Check all the calculations

y <- fig$DFN
x <- fig$fn
beta1 <- cor(y, x) *  sd(y) / sd(x)
```

```
beta0 <- mean(y) - beta1 * mean(x)
rbind(c(beta0, beta1), coef(lm(y ~ x)))
```

```
##      (Intercept)        x
## [1,]   0.1643706 4.912688
## [2,]   0.1643706 4.912688
```

```
# Reversing the relationship
beta1 <- cor(y, x) *  sd(x) / sd(y)
beta0 <- mean(x) - beta1 * mean(y)
rbind(c(beta0, beta1), coef(lm(x ~ y)))
```

```
##      (Intercept)         y
## [1,] 0.005253362 0.1282138
## [2,] 0.005253362 0.1282138
```

Regression through the origin should yield an equivalent slope when data is centered.

```
yc <- y - mean(y)
xc <- x - mean(x)
beta1 <- sum(yc * xc) / sum(xc ^ 2)
c(beta1, coef(lm(y ~ x))[2])
```

```
##                 x
## 4.912688 4.912688
```

Normalization should result in slope being the same as correlation.

```
# By normalization SD should be 1 and mean should be 0
yn <- (y - mean(y))/sd(y)
sd(yn)
```

```
## [1] 1
```

```
mean(yn)
```

```
## [1] -1.430954e-16
```

```
xn <- (x - mean(x))/sd(x)
sd(xn)
```

```
## [1] 1
```

```
mean(xn)
```

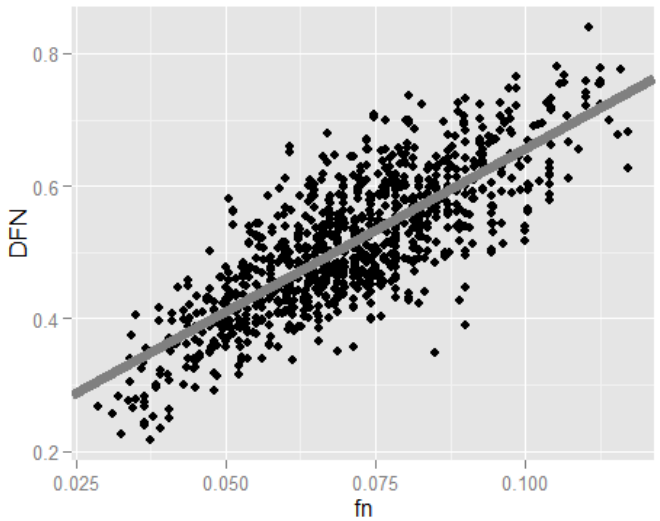
```
## [1] 4.199744e-17
```

```
c(cor(y, x), cor(yn, xn), coef(lm(yn ~ xn))[2])
```

```
##                             xn
## 0.7936462 0.7936462 0.7936462
```

```
fit <- lm(DFN ~ fn, data = fig)

coef(fit)
```

```
## (Intercept)          fn
##   0.1643706   4.9126881

summary(fit)

##
## Call:
## lm(formula = DFN ~ fn, data = fig)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.233096 -0.042987 -0.004952  0.041125  0.195712
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.164371   0.008705   18.88   <2e-16 ***
## fn          4.912688   0.119207   41.21   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06231 on 998 degrees of freedom
## Multiple R-squared:  0.6299, Adjusted R-squared:  0.6295
## F-statistic:  1698 on 1 and 998 DF,  p-value: < 2.2e-16

g <- g + geom_abline(intercept = coef(fit)[1], slope = coef(fit)[2],
                     size = 3, colour = grey(.5))
g
```



Interpreting results The slope of the line is 4.91 and intercept i.e., y-intercept is 0.16. It is interpretted as for each change in one unit of x the average change in the mean of y is about 4.91 units. Remember that x and y represents fuzziness and DFN values respectively. For example, we can see from fitted line graph, when x is 0.0375 y might be between 0.3 and 0.4 (0.34 excatly for this case).

Predicting the new values of y

```
newx<-c(0.035, 0.062, 0.087)
coef(fit)[1] + coef(fit)[2] * newx
```

```
## [1] 0.3363147 0.4689573 0.5917745
```

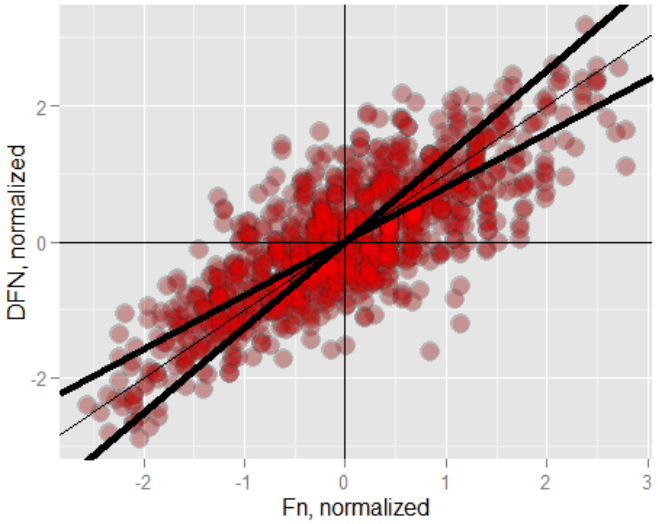
Regression to the mean

In last step (fitting the best line), we normalized X (fn value) and Y (DFN value) so that they both have mean 0 and standard deviations 1 and regression line passes through the (0,0) (the mean of the X and Y). In that case, the slope of the regression line is $\text{Cor}(Y, X)$, regardless of which variable is the outcome because standard deviations of both variables are 1. Now, if X is the outcome and we want to create a plot where X is on the horizontal axis, the slope of the least squares line of that plot is $1/\text{Cor}(Y, X)$.

```
rho<-cor(xn,yn)
rho
```

```
## [1] 0.7936462
```

```
g = ggplot(data.frame(xn, yn), aes(x = xn, y = yn))
g = g + geom_point(size = 5, alpha = .2, colour = "black")
g = g + geom_point(size = 4, alpha = .2, colour = "red")
g = g + geom_vline(xintercept = 0)
g = g + geom_hline(yintercept = 0)
g = g + geom_abline(position = "identity")
g = g + geom_abline(intercept = 0, slope = rho, size = 2)
g = g + geom_abline(intercept = 0, slope = 1 / rho, size = 2)
g = g + xlab("Fn, normalized")
g = g + ylab("DFN, normalized")
g
```



Interpreting results From the above figure, we can conclude:

- If we had to predict a y (DFN) normalized value, it would be $\text{Cor}(Y, X) * X_i$
- If we had to predict a x (fn) normalized value, it would be $\text{Cor}(Y, X) * Y_i$
- Multiplication by correlation shrinks toward 0 i.e., regression toward the mean
- If the correlation is 1 there is no regression to the mean

Explaining variations in DFN values

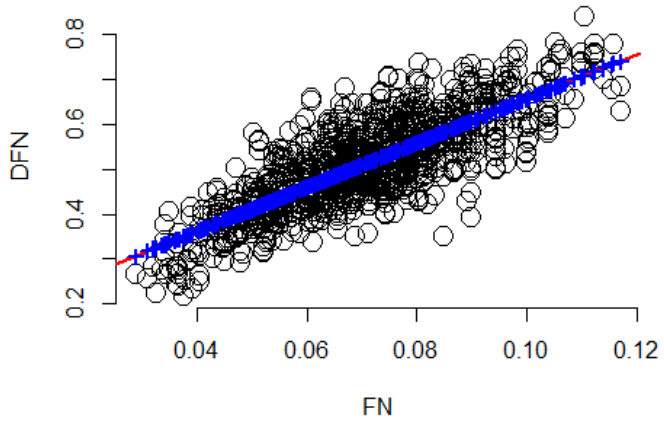
Variations arround the regression line is explained using the residuals, which is the vertical distance between the observed data point and fitted data point. We used residuals to explain variations in DFN values.

```
n<-length(y)
fit <- lm(DFN ~ fn, data = fig)

# residual/errors
e<-resid(fit)

#sum of all residuals must be 0
sum(e)
```

```
## [1] -5.312591e-17
```

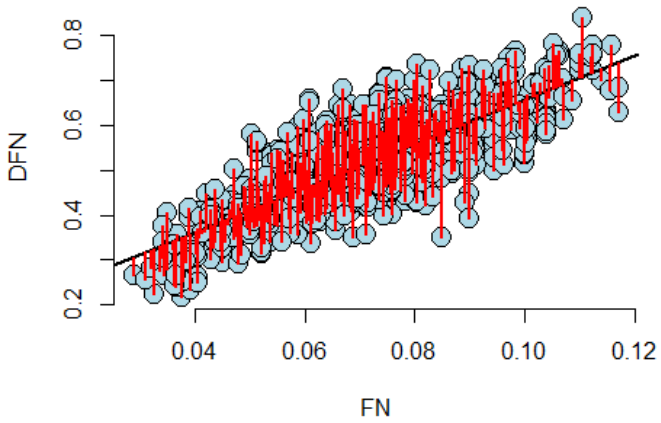
```
# residual * x must be 0
sum(e * x)
```

```
## [1] 1.196688e-17

#estimated or predicted value of y
yhat <- predict(fit)
max(abs(e - (y - coef(fit)[1] - coef(fit)[2] * x)))

## [1] 1.247613e-14

plot(fig$fn, fig$DFN,xlab = "FN",ylab = "DFN",
     col = "black", cex = 2, pch = 21,frame = FALSE)
abline(fit,col="red",lwd=2)
points(fig$fn,yhat,col="blue",lwd=2,pch=3)
```



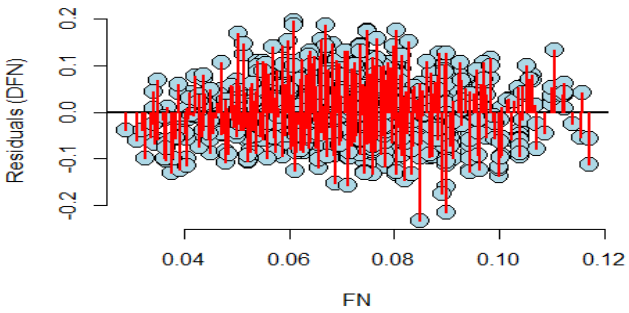
Plotting residuals

```
plot(fig$fn, fig$DFN,
     xlab = "FN",
     ylab = "DFN",
     bg = "lightblue",
     col = "black", cex = 2, pch = 21,frame = FALSE)
abline(fit, lwd = 2)

for (i in 1 : n)
  lines(c(x[i], x[i]), c(y[i], yhat[i]), col = "red" , lwd = 2)
```



plottng residuals on vertical axis Fn on horizontal axis

```
plot(x, e,
     xlab = "FN",
     ylab = "Residuals (DFN)",
     bg = "lightblue",
     col = "black", cex = 2, pch = 21,frame = FALSE)
abline(h = 0, lwd = 2)
for (i in 1 : n)
  lines(c(x[i], x[i]), c(e[i], 0), col = "red" , lwd = 2)
```



In above figure, residuals are the signed length of the red lines. We dont observe any specific pattern in the residuals, proving our model a good fit.

Residual standard error

Our model equation is

$$Y_i = \beta_0 + \beta_1 X_i + \epsilon_i$$

where $\epsilon_i \sim N(0, \sigma^2)$

The average squared residual is calculated by σ^2.

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n} e_i^2$$

```
# residual standard error or residual variation value
sqrt(sum(resid(fit)^2) / (n - 2))
```

```
## [1] 0.06230787
```

The total variability in DFN value is the variability around an intercept.

$$\text{Totalvariablity} = \sum_{i=1}^{n} (Y_i - \overline{Y})^2$$

The regression variability is the variability around the regression line explained by the fn

$$\text{Regressionvariablity} = \sum_{i=1}^{n} (\hat{Y}_i - \overline{Y})^2$$

The error variability (residual variablity) is what's leftover around the regression line

$$\text{Residualvariablity} = \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2$$

Note that error and regression vaiablities add up to toal variablity explained by the model.

$$\sum_{i=1}^{n} (Y_i - \overline{Y})^2 = \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2 + \sum_{i=1}^{n} (\hat{Y}_i - \overline{Y})^2$$

```
e = c(resid(lm(DFN ~ 1, data = fig)),
      resid(lm(DFN ~ fn, data = fig)))
fit = factor(c(rep("Itc", nrow(fig)),
               rep("Itc, slope", nrow(fig))))
g = ggplot(data.frame(e = e, fit = fit), aes(y = e, x = fit, fill = fit))
g = g + geom_dotplot(binaxis = "y", size = 0.001,stackdir = "center")
g = g + xlab("Fitting approach")
```

```
g = g + ylab("Residual")
g
```

R squared R squared is the percentage of the total variability in the dependent variable (DFN in our case) accounted for independent variable (fn in our case).

$$R^2 = \frac{\sum_{i=1}^{n}(\hat{Y}_i - \overline{Y})^2}{\sum_{i=1}^{n}(Y_i - \overline{Y})^2}$$

For a model to be good fit, the difference between R^2 and adjusted R^2 should be minimum and in our model values of R^2 and adjusted R^2 are 0.6299 and 0.6295 respectively which indicates a good model fit.

Inference by regression model

For inferencing we use statistical fomula

$$\frac{\hat{\theta} - \theta}{\hat{\sigma}_{\hat{\theta}}}$$

where $\hat{\theta}$ is an estimate of interest and θ is the estimand of interest and $\hat{\sigma}_{\hat{\theta}}$ is the standard error of $\hat{\theta}$. This formula is used to create confidence internval.

Now consider our model

$$Y_i = \beta_0 + \beta_1 X_i + \epsilon_i$$

where $\epsilon \sim N(0, \sigma^2)$. The model is refined to

$$\hat{\beta}_0 = \overline{Y} - \hat{\beta}_1 \overline{X}$$

whereas we already had computed

$$\hat{\beta}_1 = Cor(Y, X)\frac{Sd(Y)}{Sd(X)}$$

By substituting these values and standard error of our regression model in statistical formula, we will get:

$$\sigma^2_{\hat{\beta}_1} = Var(\hat{\beta}_1) = \sigma^2 / \sum_{i=1}^{n}(X_i - \overline{X})^2$$

$$\sigma^2_{\hat{\beta}_0} = Var(\hat{\beta}_0) = \left(\frac{1}{n} + \frac{\overline{X}^2}{\sum_{i=1}^{n}(X_i - \overline{X})^2}\right)\sigma^2$$

σ is replaced by its estimate we had already computed. Final form is of formula is

$$\frac{\hat{\beta}_j - \beta_j}{\hat{\sigma}_{\hat{\beta}_j}}$$

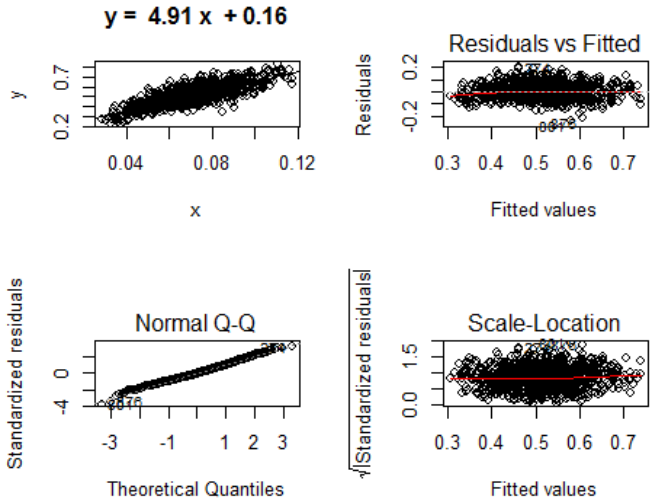
Code for inference formula

```
y <- fig$DFN; x <- fig$fn; n <- length(y)
beta1 <- cor(y, x) * sd(y) / sd(x)
beta0 <- mean(y) - beta1 * mean(x)
e <- y - beta0 - beta1 * x
sigma <- sqrt(sum(e^2) / (n-2))
ssx <- sum((x - mean(x))^2)

# standard errors for our regression coefficients and the t statistic
seBeta0 <- (1 / n + mean(x) ^ 2 / ssx) ^ .5 * sigma
seBeta1 <- sigma / sqrt(ssx)
tBeta0 <- beta0 / seBeta0
tBeta1 <- beta1 / seBeta1

# P-values
pBeta0 <- 2 * pt(abs(tBeta0), df = n - 2, lower.tail = FALSE)
pBeta1 <- 2 * pt(abs(tBeta1), df = n - 2, lower.tail = FALSE)
coefTable <- rbind(c(beta0, seBeta0, tBeta0, pBeta0), c(beta1, seBeta1,
tBeta1, pBeta1))
colnames(coefTable) <- c("Estimate", "Std. Error", "t value", "P(>|t|)")
rownames(coefTable) <- c("(Intercept)", "x")
coefTable
```

```
##              Estimate  Std. Error  t value       P(>|t|)
## (Intercept) 0.1643706 0.008705394 18.88147  3.145887e-68
## x           4.9126881 0.119206940 41.21143 1.284160e-217
```

The first column are the actual estimates, second column shows standard errors, the third is the t value (the first divided by the second).

```
fit <- lm(y ~ x)
par(mfrow=c(2,2))
lm.result=simple.lm(x,y)
plot(lm.result)
```




```
sumCoef <- summary(fit)$coefficients

## confidence interval for intercept
sumCoef[1,1] + c(-1, 1) * qt(.975, df = fit$df) * sumCoef[1, 2]

## [1] 0.1472877 0.1814536

## confidence interval for slope
sumCoef[2,1] + c(-1, 1) * qt(.975, df = fit$df) * sumCoef[2, 2]

## [1] 4.678763 5.146613
```

Results With 95% confidence, we estimate that a unit increase in Fn value results in a 4.67 to 5.14 increase in DFN value.

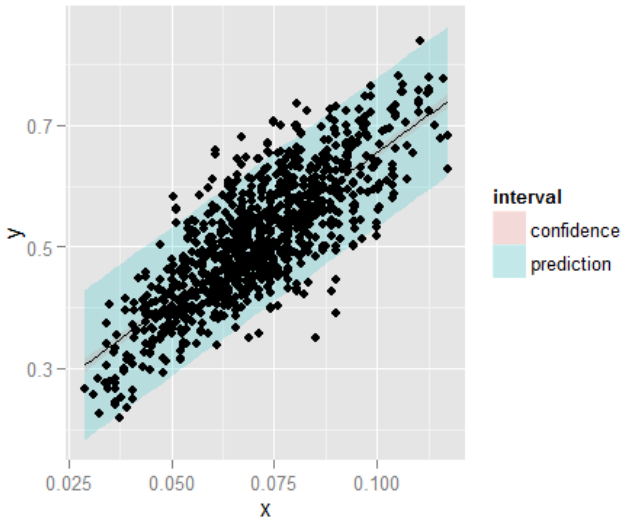
Prediction interval Prediction interval is used to estimate the uncertainity in prediction. Consider the problem of predicting Y at a value of X. In our example, this is predicting the value of DFN given the fn value. The estimate for prediction at point x_0 is:

$$\hat{\beta}_0 + \hat{\beta}_1 x_0$$

standard error is needed for prediction interval.

```
## Prediction interval
newx = data.frame(x = seq(min(x), max(x), length = 50))
p1 = data.frame(predict(fit, newdata= newx,interval = ("confidence")))
p2 = data.frame(predict(fit, newdata = newx,interval = ("prediction")))
p1$interval = "confidence"
```

```
p2$interval = "prediction"
p1$x = newx$x
p2$x = newx$x
dat = rbind(p1, p2)
names(dat)[1] = "y"

g = ggplot(dat, aes(x = x, y = y))
g = g + geom_ribbon(aes(ymin = lwr, ymax = upr, fill = interval), alpha =
0.2)
g = g + geom_line()
g = g + geom_point(data = data.frame(x = x, y=y), aes(x = x, y = y), size =
2)
g
```

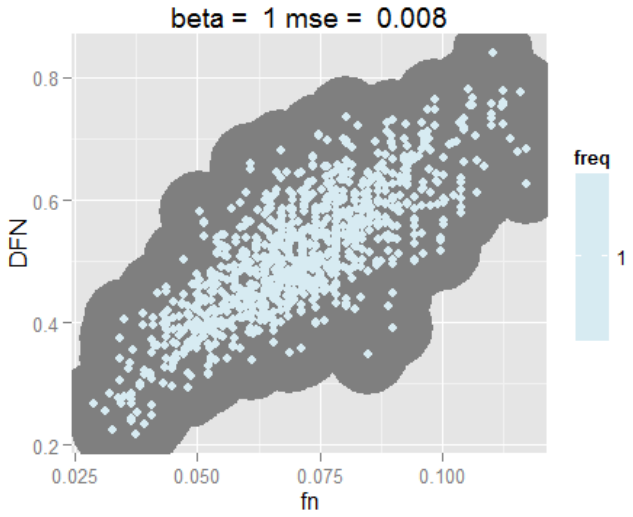


Note that the confidence interval is much narrow as compared to prediction interval , because it is prediction of line at those particular values of x.

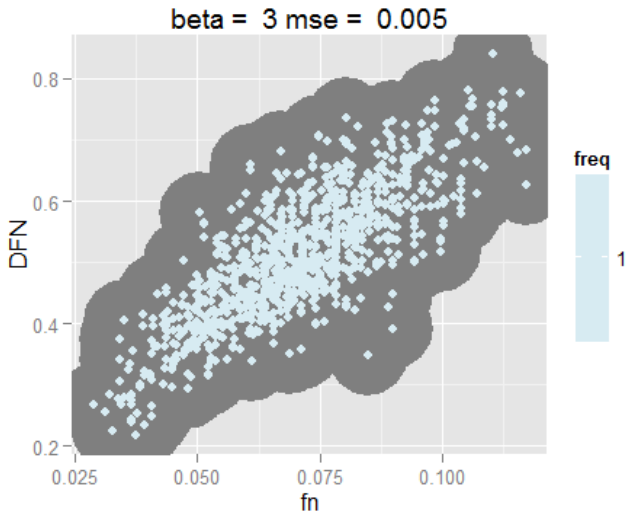
Plot for best mse value:

```
y <- fig$DFN - mean(fig$DFN)
x <- fig$fn - mean(fig$fn)
freqData <- as.data.frame(table(x, y))
names(freqData) <- c("child", "parent", "freq")
fig$DFN <- as.numeric(as.character(fig$DFN))
fig$fn <- as.numeric(as.character(fig$fn))
msePlot <- function(beta){
    g <- ggplot(filter(freqData, freq > 0), aes(x = fn, y = DFN))
```

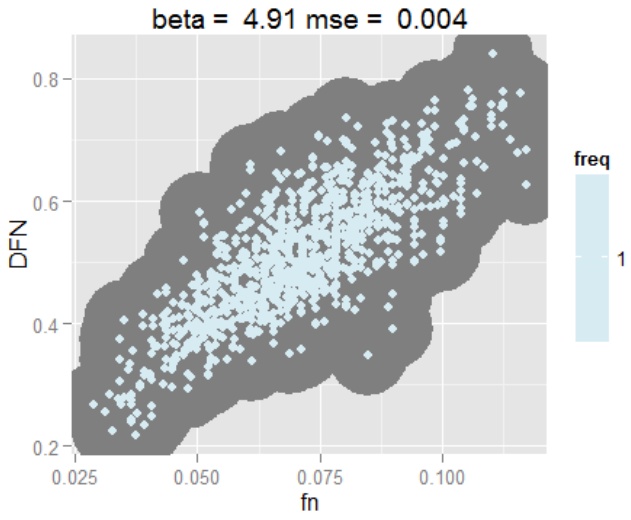
```
    g <- g  + scale_size(range = c(2, 20), guide = "none" )
    g <- g + geom_point(colour="grey50", aes(size = freq+5, show_guide =
FALSE))
    g <- g + geom_point(aes(colour=freq, size = freq))
    g <- g + scale_colour_gradient(low = "lightblue", high="white")
    mse <- mean( (y - beta * x) ^2 )
    g <- g + ggtitle(paste("beta = ", beta, "mse = ", round(mse, 3)))
    g
}
msePlot(1)
```



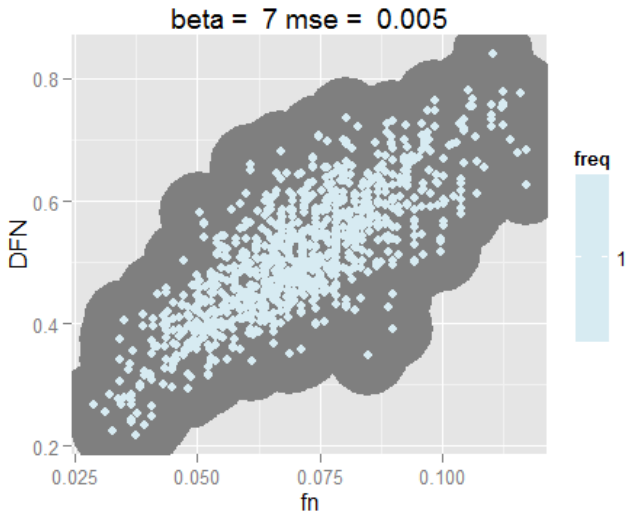

```
msePlot(3)
```



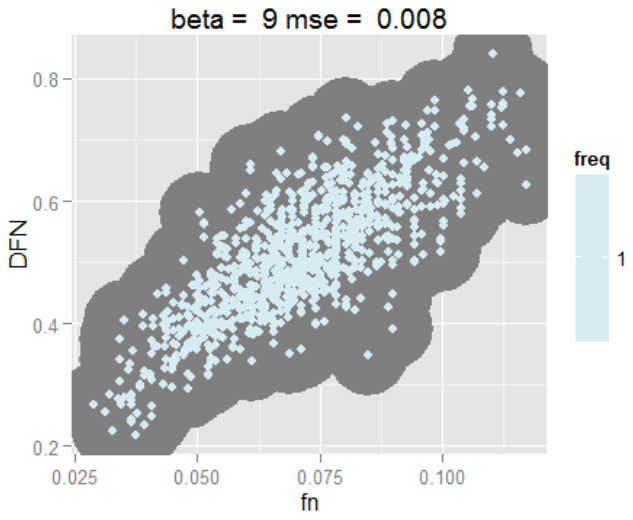

```
msePlot(4.91)
```




```
msePlot(7)
```


beta = 7 mse = 0.005
DFN
fn
freq
1
0.2
0.4
0.6
0.8
0.025
0.050
0.075
0.100

```
msePlot(9)
```

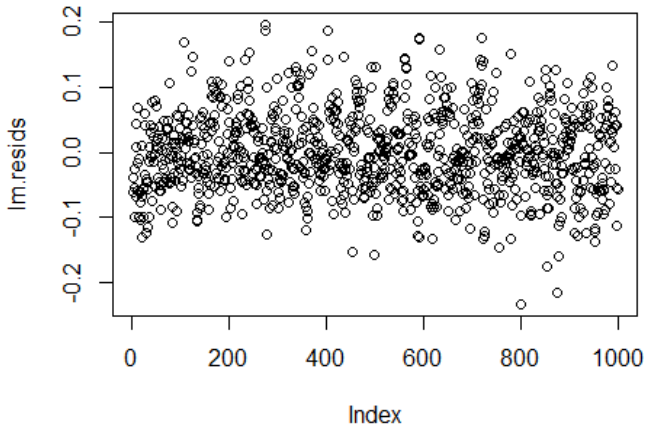

beta = 9 mse = 0.008
DFN
fn
freq
1
0.2
0.4
0.6
0.8
0.025
0.050
0.075
0.100

Note that the minimum mse value is against value 4.91, which was the slope value calculated by our model. Any beta value more or less than 4.91 results in increase of mse value.

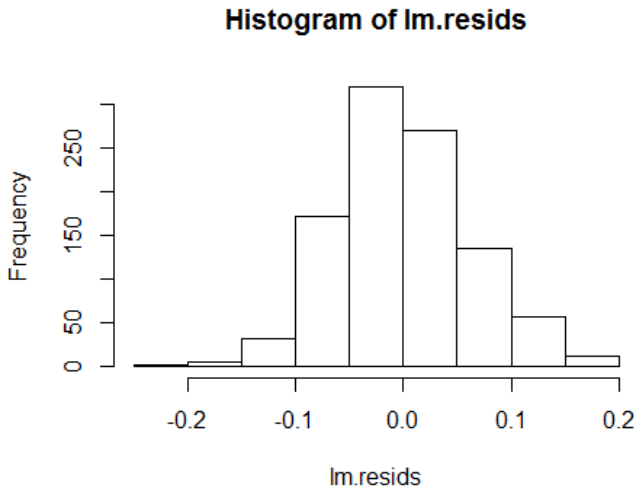
Testing regression assumptions

For statistical inferences about the regression line, we first have to make sure that the assumptions of the model are appropriate. In this case, we will check that residuals have no trends, and are normally distributed

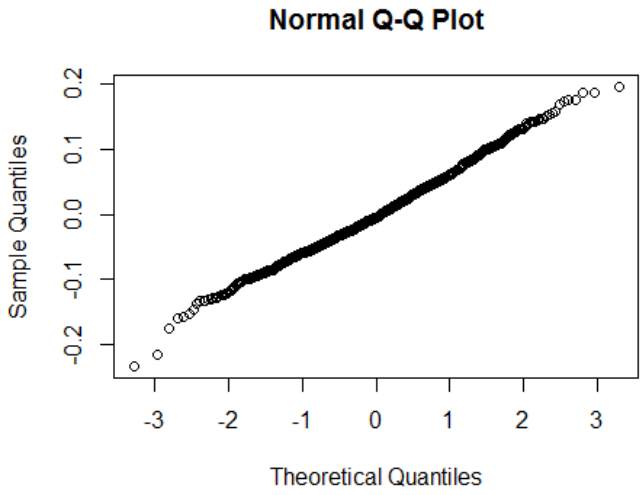
```
# residuals as a vector
lm.resids = resid(lm.result)

# change in spread
plot(lm.resids)
```




```
# data is bell shaped
hist(lm.resids)
```




```
# data on straight line
qqnorm(lm.resids)
```



Appendix B

Cycle Computer Goals

Figure B.1: High Level Goal Model

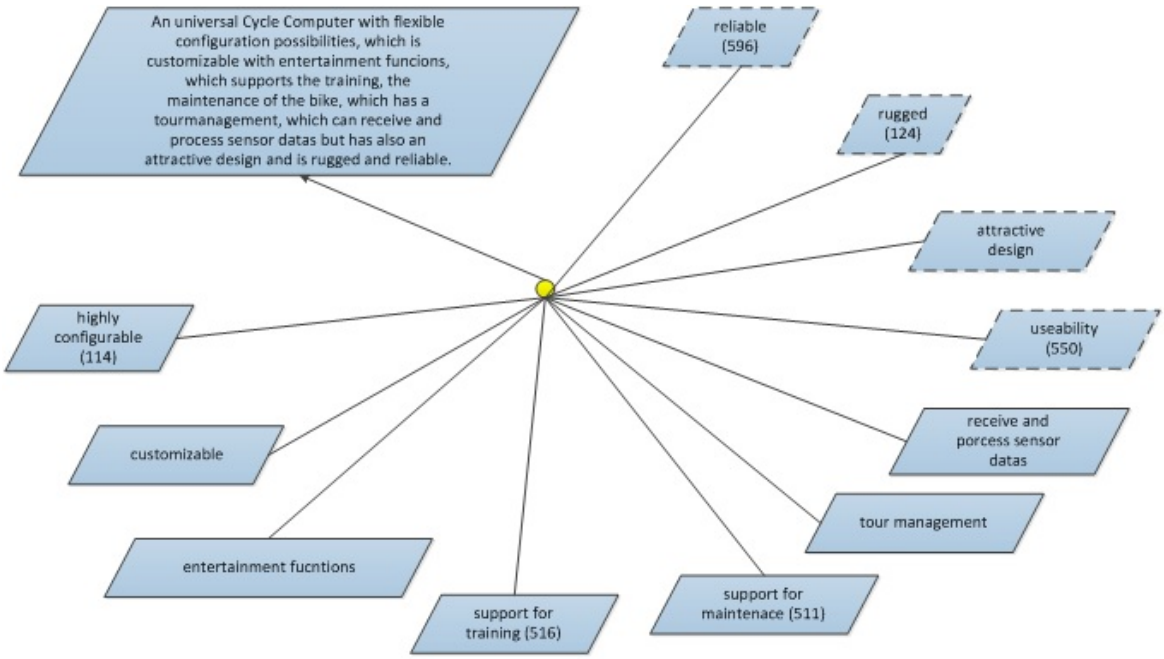


Figure B.2: Flexible Configuration

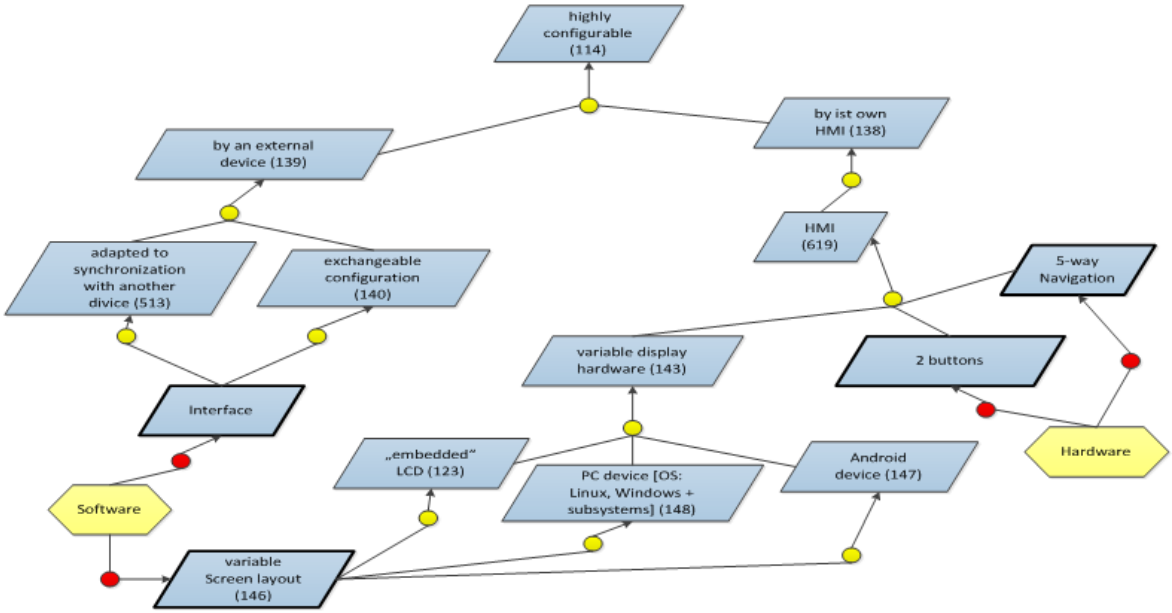


Figure B.3: Customization

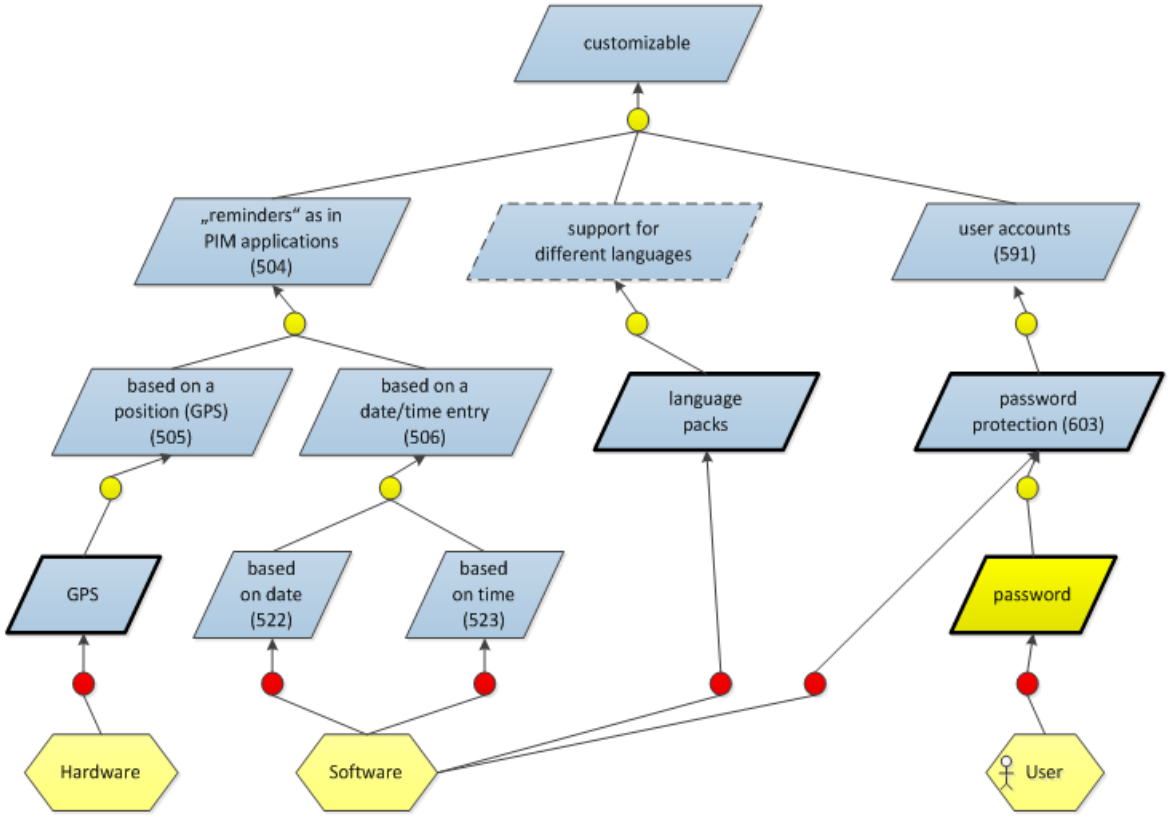


Figure B.4: Attractiveness

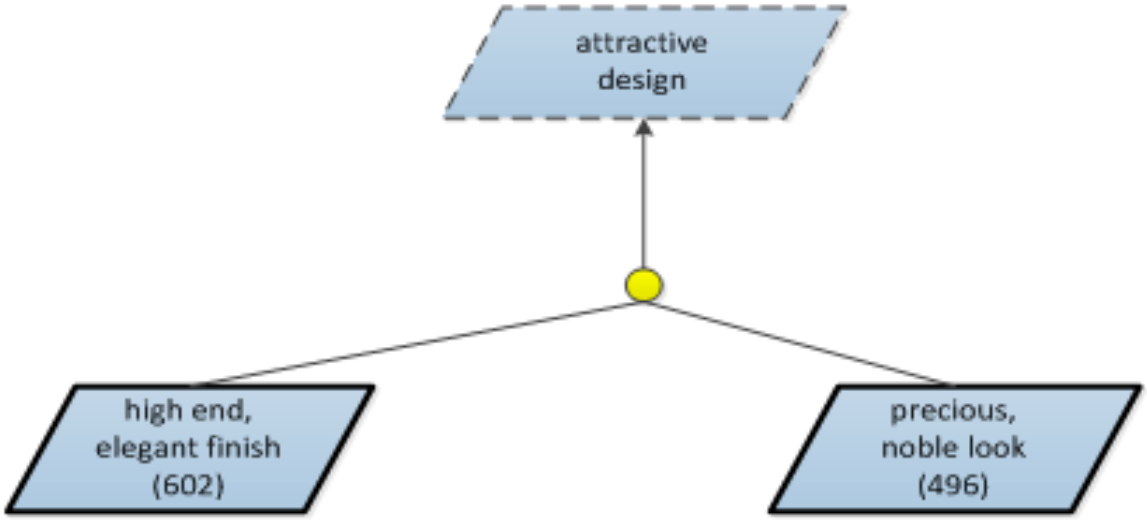


Figure B.5: Entertainment

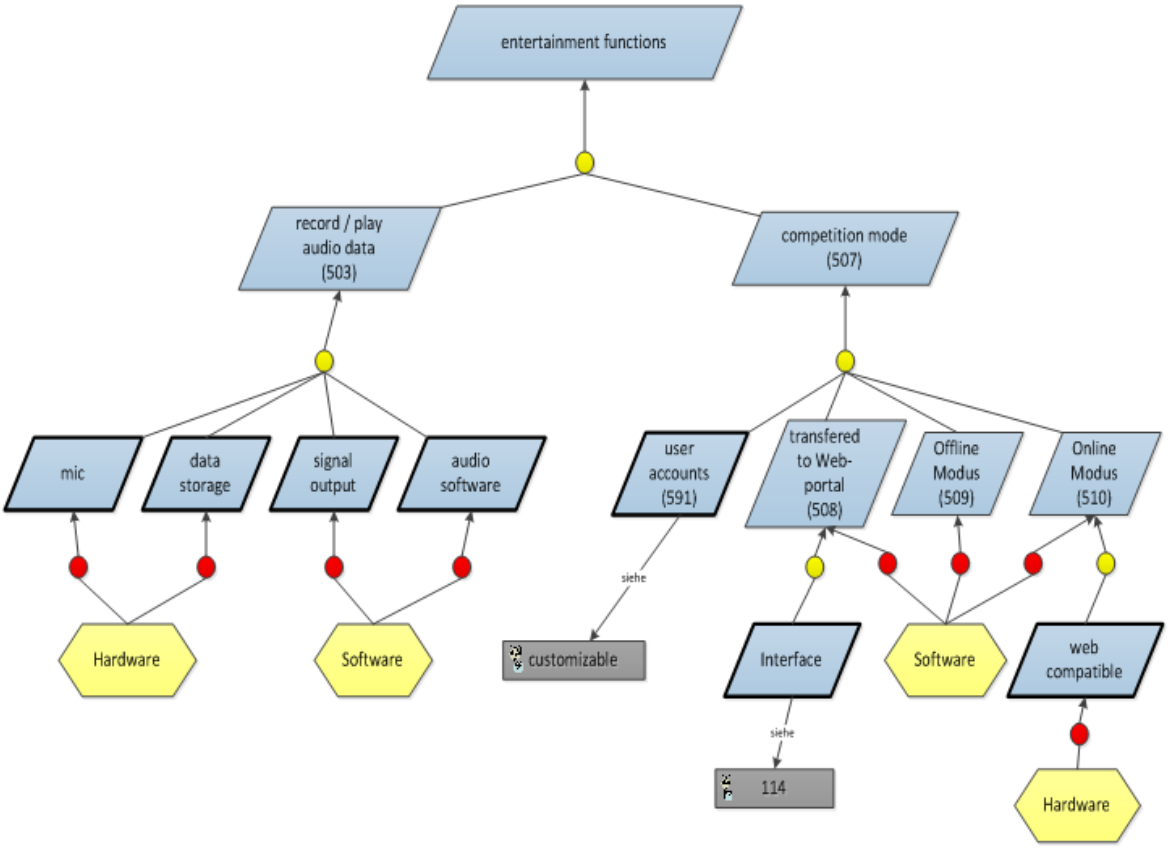


Figure B.6: Usability

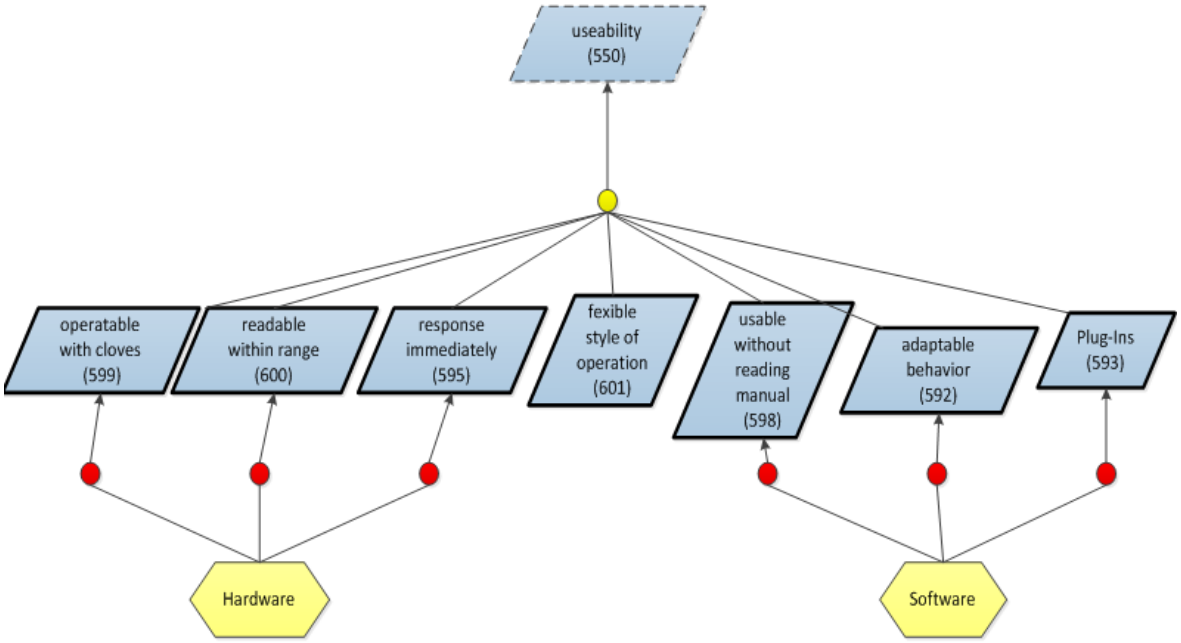


Figure B.7: Training Support

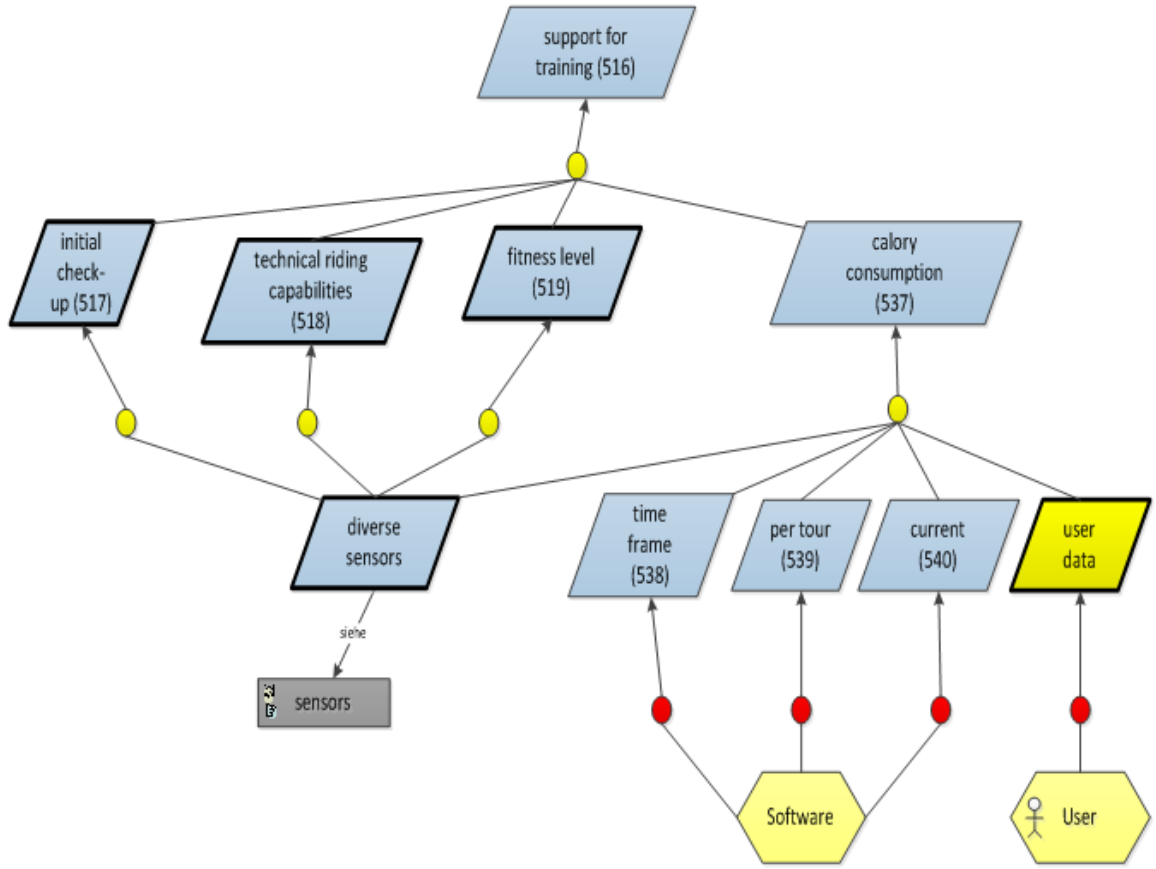


Figure B.8: Maintenances

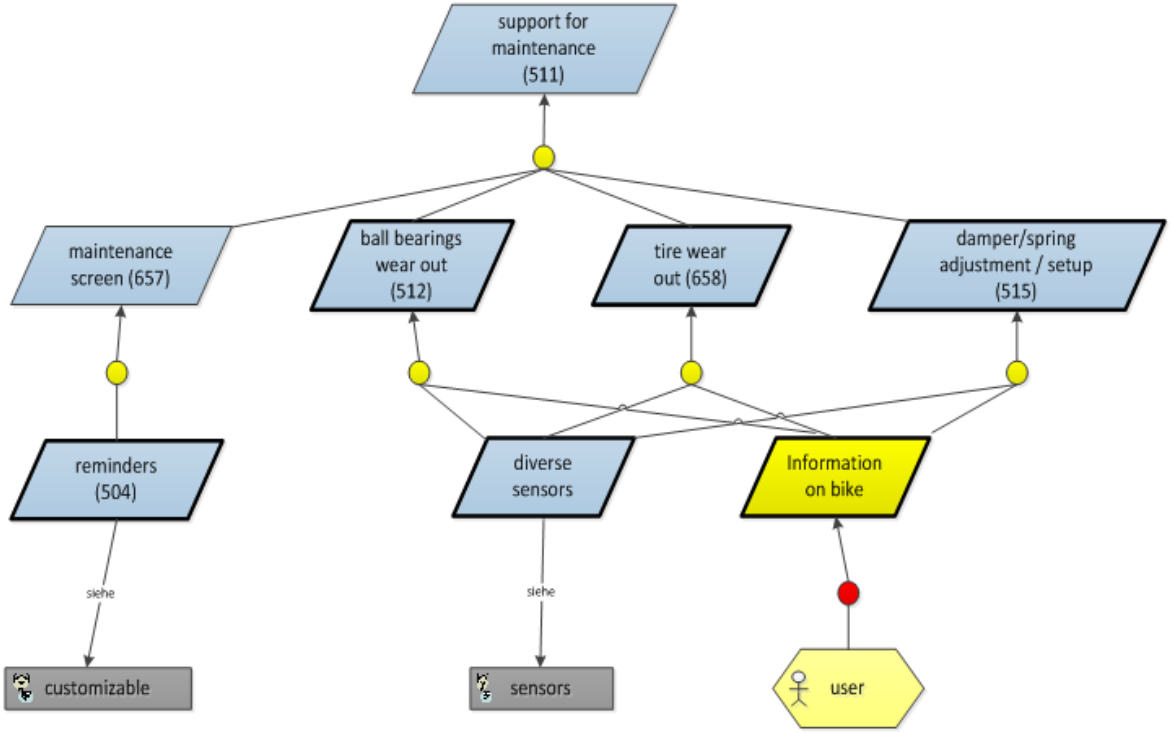


Figure B.9: Tour Management

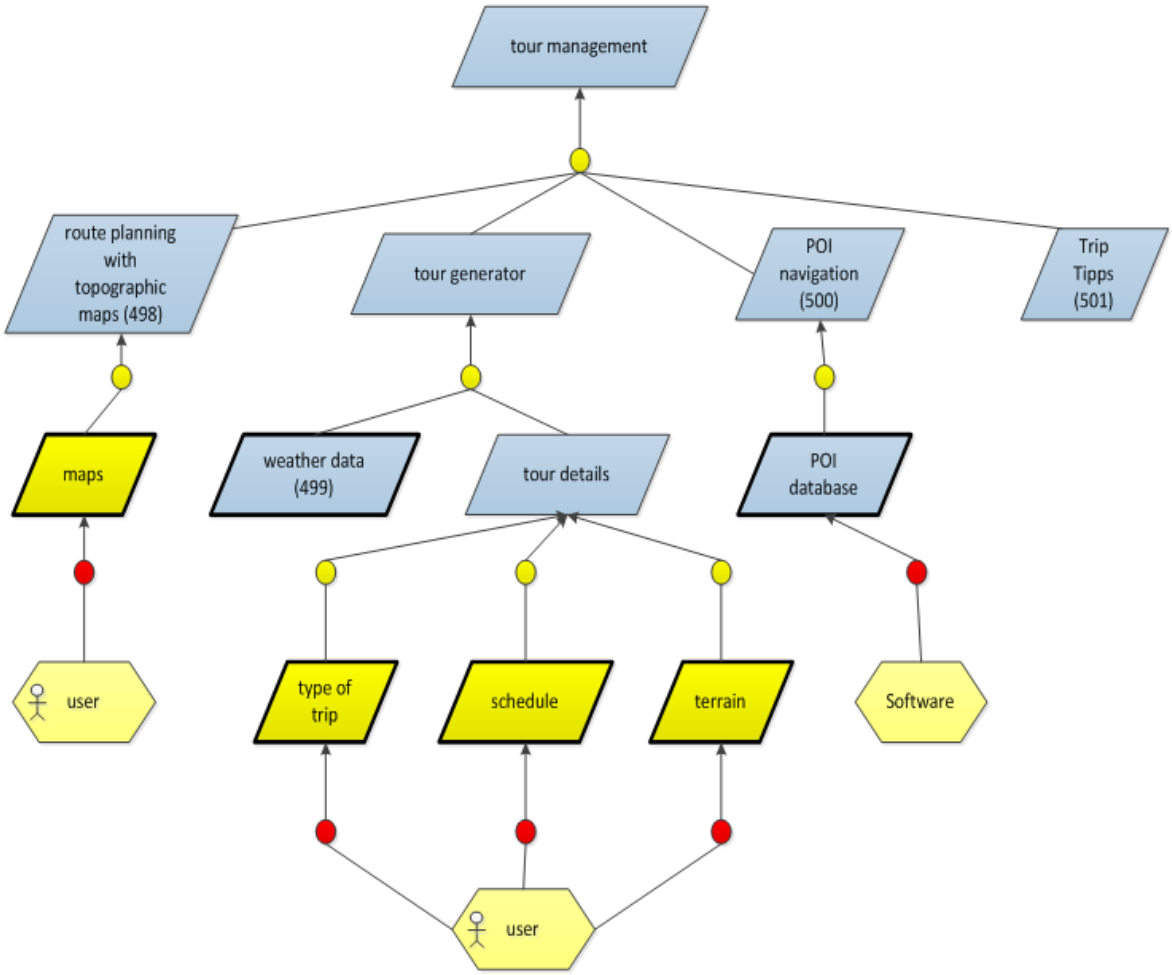


Figure B.10: Reliability

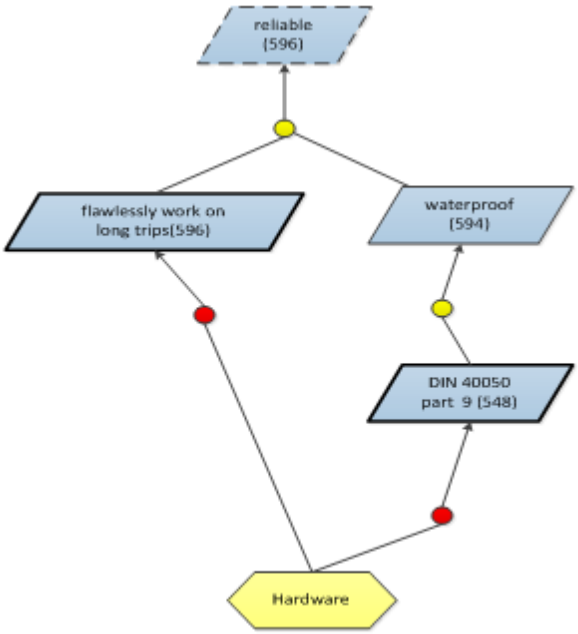


Figure B.11: Sensor Data

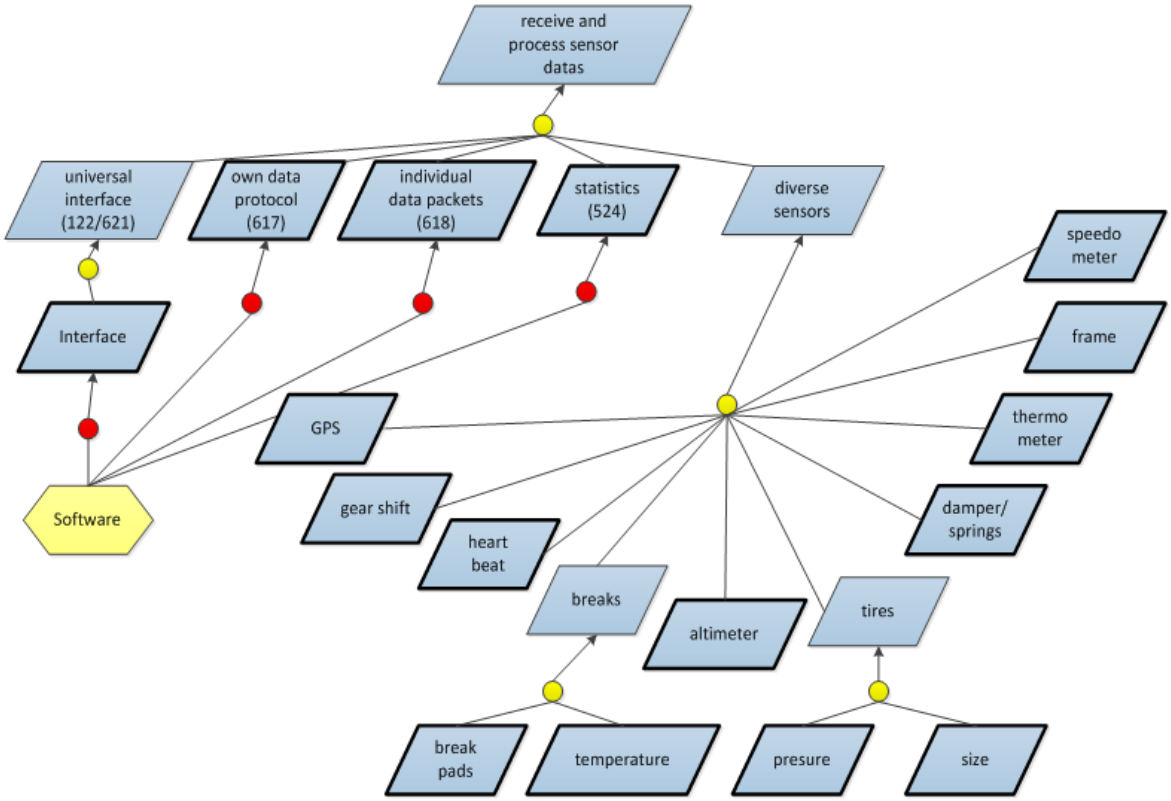
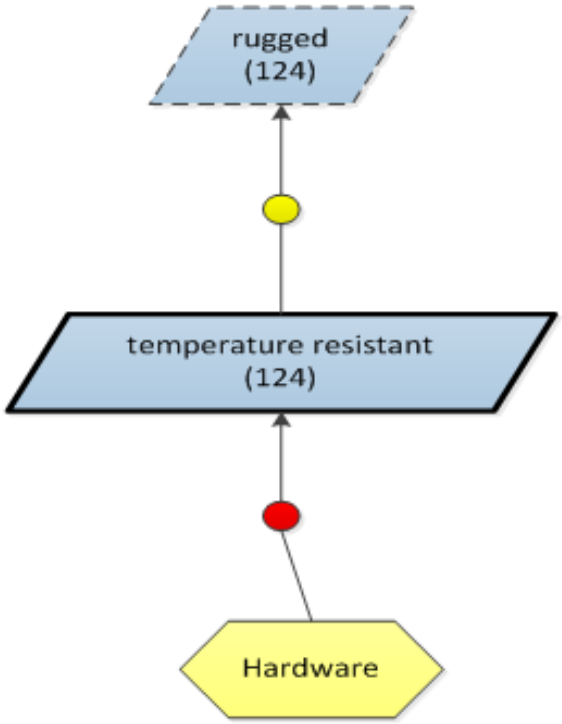


Figure B.12: Robustness



B.1 AHP Pairwise Comparisons

	A - Importance - or B?			Equal	How much more?
1	Show speed	or	Show traveled distance	1	2 3 4 5 6 7 8 9
2	Show speed	or	Show date and time	1	2 3 4 5 6 7 8 9
3	Show speed	or	Show stopwatch and countdown	1	2 3 4 5 6 7 8 9
4	Show speed	or	Show temperature	1	2 3 4 5 6 7 8 9
5	Show speed	or	Show humidity	1	2 3 4 5 6 7 8 9
6	Show speed	or	Show wind speed	1	2 3 4 5 6 7 8 9
7	Show speed	or	Show brake disk temperature	1	2 3 4 5 6 7 8 9
8	Show speed	or	Show wheel RPM	1	2 3 4 5 6 7 8 9
9	Show speed	or	Show user direction	1	2 3 4 5 6 7 8 9
10	Show speed	or	User accounts	1	2 3 4 5 6 7 8 9
11	Show speed	or	Transferable to web	1	2 3 4 5 6 7 8 9
12	Show speed	or	Online modus	1	2 3 4 5 6 7 8 9
13	Show speed	or	Route planning	1	2 3 4 5 6 7 8 9
14	Show traveled distance	or	Show date and time	1	2 3 4 5 6 7 8 9
15	Show traveled distance	or	Show stopwatch and countdown	1	2 3 4 5 6 7 8 9
16	Show traveled distance	or	Show temperature	1	2 3 4 5 6 7 8 9
17	Show traveled distance	or	Show humidity	1	2 3 4 5 6 7 8 9
18	Show traveled distance	or	Show wind speed	1	2 3 4 5 6 7 8 9
19	Show traveled distance	or	Show brake disk temperature	1	2 3 4 5 6 7 8 9
20	Show traveled distance	or	Show wheel RPM	1	2 3 4 5 6 7 8 9
21	Show traveled distance	or	Show user direction	1	2 3 4 5 6 7 8 9
22	Show traveled distance	or	User accounts	1	2 3 4 5 6 7 8 9
23	Show traveled distance	or	Transferable to web	1	2 3 4 5 6 7 8 9
24	Show traveled distance	or	Online modus	1	2 3 4 5 6 7 8 9
25	Show traveled distance	or	Route planning	1	2 3 4 5 6 7 8 9
26	Show date and time	or	Show stopwatch and countdown	1	2 3 4 5 6 7 8 9
27	Show date and time	or	Show temperature	1	2 3 4 5 6 7 8 9
28	Show date and time	or	Show humidity	1	2 3 4 5 6 7 8 9
29	Show date and time	or	Show wind speed	1	2 3 4 5 6 7 8 9
30	Show date and time	or	Show brake disk temperature	1	2 3 4 5 6 7 8 9
31	Show date and time	or	Show wheel RPM	1	2 3 4 5 6 7 8 9
32	Show date and time	or	Show user direction	1	2 3 4 5 6 7 8 9
33	Show date and time	or	User accounts	1	2 3 4 5 6 7 8 9
34	Show date and time	or	Transferable to web	1	2 3 4 5 6 7 8 9
35	Show date and time	or	Online modus	1	2 3 4 5 6 7 8 9
36	Show date and time	or	Route planning	1	2 3 4 5 6 7 8 9

37	Show stopwatch and countdown	or	Show temperature	1	2	3	4	5	6	7	8	9
38	Show stopwatch and countdown	or	Show humidity	1	2	3	4	5	6	7	8	9
39	Show stopwatch and countdown	or	Show wind speed	1	2	3	4	5	6	7	8	9
40	Show stopwatch and countdown	or	Show brake disk temperature	1	2	3	4	5	6	7	8	9
41	Show stopwatch and countdown	or	Show wheel RPM	1	2	3	4	5	6	7	8	9
42	Show stopwatch and countdown	or	Show user direction	1	2	3	4	5	6	7	8	9
43	Show stopwatch and countdown	or	User accounts	1	2	3	4	5	6	7	8	9
44	Show stopwatch and countdown	or	Transferable to web	1	2	3	4	5	6	7	8	9
45	Show stopwatch and countdown	or	Online modus	1	2	3	4	5	6	7	8	9
46	Show stopwatch and countdown	or	Route planning	1	2	3	4	5	6	7	8	9
47	Show temperature	or	Show humidity	1	2	3	4	5	6	7	8	9
48	Show temperature	or	Show wind speed	1	2	3	4	5	6	7	8	9
49	Show temperature	or	Show brake disk temperature	1	2	3	4	5	6	7	8	9
50	Show temperature	or	Show wheel RPM	1	2	3	4	5	6	7	8	9
51	Show temperature	or	Show user direction	1	2	3	4	5	6	7	8	9
52	Show temperature	or	User accounts	1	2	3	4	5	6	7	8	9
53	Show temperature	or	Transferable to web	1	2	3	4	5	6	7	8	9
54	Show temperature	or	Online modus	1	2	3	4	5	6	7	8	9
55	Show temperature	or	Route planning	1	2	3	4	5	6	7	8	9
56	Show humidity	or	Show wind speed	1	2	3	4	5	6	7	8	9
57	Show humidity	or	Show brake disk temperature	1	2	3	4	5	6	7	8	9
58	Show humidity	or	Show wheel RPM	1	2	3	4	5	6	7	8	9
59	Show humidity	or	Show user direction	1	2	3	4	5	6	7	8	9
60	Show humidity	or	User accounts	1	2	3	4	5	6	7	8	9
61	Show humidity	or	Transferable to web	1	2	3	4	5	6	7	8	9
62	Show humidity	or	Online modus	1	2	3	4	5	6	7	8	9
63	Show humidity	or	Route planning	1	2	3	4	5	6	7	8	9
64	Show wind speed	or	Show brake disk temperature	1	2	3	4	5	6	7	8	9
65	Show wind speed	or	Show wheel RPM	1	2	3	4	5	6	7	8	9
66	Show wind speed	or	Show user direction	1	2	3	4	5	6	7	8	9
67	Show wind speed	or	User accounts	1	2	3	4	5	6	7	8	9
68	Show wind speed	or	Transferable to web	1	2	3	4	5	6	7	8	9
69	Show wind speed	or	Online modus	1	2	3	4	5	6	7	8	9
70	Show wind speed	or	Route planning	1	2	3	4	5	6	7	8	9

71	Show brake disk temperature	or Show wheel RPM	1	2 **3** 4 **5** 6 **7** 8 **9**
72	Show brake disk temperature	or Show user direction	1	2 **3** 4 **5** 6 **7** 8 **9**
73	Show brake disk temperature	or User accounts	1	2 **3** 4 **5** 6 **7** 8 **9**
74	Show brake disk temperature	or Transferable to web	1	2 **3** 4 **5** 6 **7** 8 **9**
75	Show brake disk temperature	or Online modus	1	2 **3** 4 **5** 6 **7** 8 **9**
76	Show brake disk temperature	or Route planning	1	2 **3** 4 **5** 6 **7** 8 **9**
77	Show wheel RPM	or Show user direction	1	2 **3** 4 **5** 6 **7** 8 **9**
78	Show wheel RPM	or User accounts	1	2 **3** 4 **5** 6 **7** 8 **9**
79	Show wheel RPM	or Transferable to web	1	2 **3** 4 **5** 6 **7** 8 **9**
80	Show wheel RPM	or Online modus	1	2 **3** 4 **5** 6 **7** 8 **9**
81	Show wheel RPM	or Route planning	1	2 **3** 4 **5** 6 **7** 8 **9**
82	Show user direction	or User accounts	1	2 **3** 4 **5** 6 **7** 8 **9**
83	Show user direction	or Transferable to web	1	2 **3** 4 **5** 6 **7** 8 **9**
84	Show user direction	or Online modus	1	2 **3** 4 **5** 6 **7** 8 **9**
85	Show user direction	or Route planning	1	2 **3** 4 **5** 6 **7** 8 **9**
86	User accounts	or Transferable to web	1	2 **3** 4 **5** 6 **7** 8 **9**
87	User accounts	or Online modus	1	2 **3** 4 **5** 6 **7** 8 **9**
88	User accounts	or Route planning	1	2 **3** 4 **5** 6 **7** 8 **9**
89	Transferable to web	or Online modus	1	2 **3** 4 **5** 6 **7** 8 **9**
90	Transferable to web	or Route planning	1	2 **3** 4 **5** 6 **7** 8 **9**
91	Online modus	or Route planning	1	2 **3** 4 **5** 6 **7** 8 **9**

Appendix C

Cycle Computer comparisons

Feature	**CM213C**	**CM404**	**HAC4Pro**	**Germin Edge 305**
Price [€]		12	70	250
Speed [Miles]	yes	yes	yes	yes
Speed [KM]	yes	yes	yes	yes
Speed digits [xxx,]	3	3	3	3
Speed digits [,xxx]	1	1	1	1
Average speed			yes	yes
Wireless Speed Sensor	no	no	yes	n/a
Daytime AM/PM	yes	yes	yes	yes
Daytime 24h	yes	yes	yes	yes
Date day/month/year	no	no	yes	yes
Alarm clock			yes	
Stopwatch			yes	
Tire1 Size	yes	yes	yes	yes
Tire2 Size	yes	no	yes	yes
Sum-up Tire1 and Tire2	yes	no	yes	
Tire Size digits	4	4	4	
Tire Size min [mm]			500	
Tire Size max [mm]			3000	
Overall distance	5	5	5	
Overall distance digits [xxx,]	5	5	5	
Overall distance digits [,xxx]	1	1	1	
Overall riding time			yes	
Set overall distance	no	no	yes	
Daily distance	yes	yes	yes	

Daily distance digits [xxx,]	3	3	3
Daily distance digits [,xxx]	2	2	2
Daily distance reset after [h]	12	12	
Daily riding time	no	no	yes
Distance digits	5	5	
Distance [Miles]	yes	yes	yes
Distance [KM]	yes	yes	yes
Dist backup, batt change	yes	no	no
Max batt. Change time [sec]	15	0	
Low battery warning	no	no	yes
Battery life [months]			10
PedalFreq	yes	no	yes
Max. PedalFreq	no	no	yes
Min. PedalFreq	no	no	yes
Auto Turn off after [sec]	300	300	300
Auto Turn on, on tire turn	yes	yes	yes
Heartbeat Sensor	no	no	yes
No of Buttons	2	4	5
Height Sensor	no	no	yes
Height min [m]			-200
Height max [m]			9000
Height in m			yes
Height in feet			yes
Daily height			yes
Daily ascend			yes
Daily descend			yes
Set overall height			yes
Show gradient (up/down)			yes
Set Gradient min			0
Set Gradient max			0.99
Show average gradient			yes
Show max gradient			yes
Show min gradient			yes
Variometer ...			
Current ascend value			yes
Current descend value			yes
Max ascend			yes
Max descend			yes
Average ascend			yes

Average descend			yes	
No of ascends			yes	
No of descens			yes	
GPS	no	no	no	yes
Auto Lap	no	no	no	yes
Virtual partner	no	no	no	yes
Temp Sensor			yes	
Temp Celsius			yes	
Temp Fahrenheit			yes	
Max Temp			yes	
Min Temp			yes	
PC-Connection	no	no	yes	
PC Analysis SW	no	no	yes	
Fitness				
Sex	no	no	yes	
Body weight	no	no	yes	
Complete weight	no	no	yes	
Age	no	no	yes	
Set heartbeat1 min. level	no	no	yes	
Set heartbeat1 max. level	no	no	yes	
Set heartbeat2 min. level	no	no	yes	
Set heartbeat2 max. level	no	no	yes	
Ride by heartbeat zone	no	no	yes	
Heartbeat alarm (outside zone)	no	no	yes	
Check cool down heartbeat	no	no	yes	
Time in riding zone	no	no	yes	
Time above riding zone	no	no	yes	
Time below riding zone	no	no	yes	
Fitness level	no	no	yes	
Current calory consumption	no	no	yes	
Overall calory consumption	no	no	yes	
Current performance in Watts	no	no	yes	
Average performance	no	no	yes	
Max. performance	no	no	yes	
Compare training sessions	no	no	yes	
Countdown timer 1	no	no	yes	
Countdown timer 1 max [min:sec]	no	no	99:59	
Countdown timer 2	no	no	yes	
Countdown timer 2 max [min:sec]	no	no	99:59	

Firmware upgradeable	no	no	yes	
Sleep mode	no	no	yes	
Ski mode (use device for skiing)	no	no	yes	
Backlight	no	no	yes	
Display size				128x160 4-level-grayscale
Waterproof [m]			30	
Operation temp min [°C]			-20	
Operation temp max [°C]			+60	
Algorithms				
Calculate heartbeat zone by Sex, age, fitness level			yes	
Measure „Ruhepuls“			yes	

Appendix D

Abbreviations

AHP	Analytic Hierarchy Process
UML	Unified Modeling Language
RE	Requirements Engineering
GORE	Goal Oriented Requirements Engineering
NFR	Non-functional requirements
OMG	Object Management Group
SPEM	Systems Process Engineering Metamodel
CMM	Capability Maturity Model
GRL	Goal-oriented Requirement Language
GR	Goal Reasoning
SIG	Softgoal Interdependency Graph
GCT	Goal Centric Traceability
GQM	Goal Question Metrics
URN	User Requirements Notation
QFD	Quality Function Deployment
GBRAM	Goal Based Requirements Analysis Method
HOQ	House of Quality
KAOS	Knowledge Acquisitions in automated Specification
RSD	Requirements Specification Document
SD	Strategic Dependency
SR	Strategic Rationale
T-Tools	Formal Tropos tool
FURPS	Functionality Usability Reliability Performance Supportability
ISO	International Organization for Standardization
TOPSIS	Technique for Order of Preference by Similarity to Ideal Solution

Bibliography

[AKM+09] Armbrust, Ove ; Katahira, Masafumi ; Miyamoto, Yuko ; Münch, Jürgen ; Nakao, Haruka ; Ocampo, Alexis: Scoping Software Process Lines. In: *Softw. Process* 14 (2009), Mai, Nr. 3, 181–197. http://dx.doi.org/10.1002/spip.v14:3. – DOI 10.1002/spip.v14:3. – ISSN 1077–4866

[Ant96] Anton, Annie I.: Goal-Based Requirements Analysis. In: *Proceedings of the 2Nd International Conference on Requirements Engineering (ICRE '96)*. Washington, DC, USA : IEEE Computer Society, 1996 (ICRE '96). – ISBN 0–8186–7252–8, 136–

[AP98] Anton, Annie I. ; Potts, Colin: The Use of Goals to Surface Requirements for Evolving Systems. In: *Proceedings of the 20th International Conference on Software Engineering*. Washington, DC, USA : IEEE Computer Society, 1998 (ICSE '98). – ISBN 0–8186–8368–6, 157–166

[AW03] Aurum, Aybüke ; Wohlin, Claes: The fundamental nature of requirements engineering activities as a decision-making process. In: *Information & Software Technology* 45 (2003), Nr. 14, 945–954. http://dx.doi.org/10.1016/S0950-5849(03)00096-X. – DOI 10.1016/S0950–5849(03)00096–X

[BC96] Brownsword, Lisa ; Clements, Paul: A Case Study in Successful Product Line Development / Software Engineering Institute, Carnegie Mellon University. Version: 1996. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=12587. Pittsburgh, PA, 1996 (CMU/SEI-96-TR-016). – Forschungsbericht

[BFK+99] Bayer, Joachim ; Flege, Oliver ; Knauber, Peter ; Laqua, Roland ; Muthig, Dirk ; Schmid, Klaus ; Widen, Tanya ; DeBaud, Jean-Marc: PuLSE: A Methodology to Develop Software Product Lines. In: *Proceedings of the 1999 Symposium on Software Reusability*. New York, NY, USA : ACM, 1999 (SSR '99). – ISBN 1–58113–101–1, 122–131

[BH11] Blau, B. ; Hildenbrand, T.: Product Line Engineering in Large-Scale Lean and Agile Software Product Development Environments - Towards a Hybrid Approach to Decentral Control and Managed Reuse. In: *Availability, Reliability and Security (ARES), 2011 Sixth International Conference on*, 2011, S. 404–408

[Boe76] Boehm, B. W.: Software Engineering. In: *IEEE Trans. Comput.* 25 (1976), Dezember, Nr. 12, 1226–1241. http://dx.doi.org/10.1109/TC.1976.1674590. – DOI 10.1109/TC.1976.1674590. – ISSN 0018–9340

[BR87] Basili, V. R. ; Rombach, H. D.: Tailoring the Software Process to Project Goals and Environments. In: *Proceedings of the 9th International Conference*

on Software Engineering. Los Alamitos, CA, USA : IEEE Computer Society Press, 1987 (ICSE '87). – ISBN 0–89791–216–0, 345–357

[BSRC10] Benavides, David ; Segura, Sergio ; Ruiz-Cortés, Antonio: Automated Analysis of Feature Models 20 Years Later: A Literature Review. In: *Inf. Syst.* 35 (2010), September, Nr. 6, 615–636. http://dx.doi.org/10.1016/j.is.2010.01.001. – DOI 10.1016/j.is.2010.01.001. – ISSN 0306–4379

[CCL99] Chung, Lawrence ; Cesar, Julio ; Leite, Sampaio P.: *Non-functional requirements in software engineering*. 1999

[Cha] http://www.umsl.edu/~sir3b/Sean_Isserman_Requirements_Elicitation_Home.html

[Che00] Chen, Chen-Tung: Extensions of the TOPSIS for Group Decision-making Under Fuzzy Environment. In: *Fuzzy Sets Syst.* 114 (2000), August, Nr. 1, 1–9. http://dx.doi.org/10.1016/S0165-0114(97)00377-1. – DOI 10.1016/S0165–0114(97)00377–1. – ISSN 0165–0114

[Chr92] Christel, Kyo Michael; & K. Michael; & Kang: Issues in Requirements Elicitation (CMU/SEI-92-TR-012). / Software Engineering Institute, Carnegie Mellon University,. Version: 1992. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=12553. 1992.. – Forschungsbericht

[CKM02] Castro, Jaelson ; Kolp, Manuel ; Mylopoulos, John: Towards Requirements-driven Information Systems Engineering: The Tropos Project. In: *Inf. Syst.* 27 (2002), September, Nr. 6, 365–389. http://dx.doi.org/10.1016/S0306-4379(02)00012-1. – DOI 10.1016/S0306–4379(02)00012–1. – ISSN 0306–4379

[Cla97] Clark, Bradford K.: *The Effects of Software Process Maturity on Software Development Effort*. 1997

[CPL] Cysneiros, Luiz M. ; Prado Leite, Julio César S.: *Using the Language Extended Lexicon to Support Non-Functional Requirements Elicitation*

[CPL01] Cysneiros, Luiz M. ; Prado Leite, Julio César S.: Using UML to Reflect Non-functional Requirements. In: *Proceedings of the 2001 Conference of the Centre for Advanced Studies on Collaborative Research*, IBM Press, 2001 (CASCON '01), 2–

[Dav03] Davis, Alan M.: The Art of Requirements Triage. In: *Computer* 36 (2003), März, Nr. 3, 42–49. http://dx.doi.org/10.1109/MC.2003.1185216. – DOI 10.1109/MC.2003.1185216. – ISSN 0018–9162

[DFL91] Dardenne, A. ; Fickas, S. ; Lamsweerde, A. van: Goal-directed concept acquisition in requirements elicitation. In: *IWSSD '91: Proceedings of the 6th international workshop on Software specification and design*. Los Alamitos, CA, USA : IEEE Computer Society Press, 1991. – ISBN 0–8186–2320–9 (PAPER), S. 14–21

[DL96] Darimont, Robert ; Lamsweerde, Axel van: Formal Refinement Patterns for Goal-Driven Requirements Elaboration. In: *SIGSOFT FSE*, 1996, S. 179–190

[DLF93] Dardenne, Anne ; Lamsweerde, Axel van ; Fickas, Stephen: Goal-directed Requirements Acquisition. In: *Selected Papers of the Sixth International Workshop on Software Specification and Design*. Amsterdam, The Netherlands, The Netherlands : Elsevier Science Publishers B. V., 1993 (6IWSSD), 3–50

[Don04] Donzelli, Paolo: A Goal-driven and Agent-based Requirements Engineering Framework. In: *Requir. Eng.* 9 (2004), Februar, Nr. 1, 16–39. http://dx.doi.org/10.1007/s00766-003-0170-4. – DOI 10.1007/s00766–003–0170–4. – ISSN 0947–3602

[Dro95] Dromey, R. G.: A Model for Software Product Quality. In: *IEEE Trans. Softw. Eng.* 21 (1995), Februar, Nr. 2, 146–162. http://dx.doi.org/10.1109/32.345830. – DOI 10.1109/32.345830. – ISSN 0098–5589

[EK08] Ertuğrul, İrfan ; Karakaşoğlu, Nilsen: Comparison of fuzzy AHP and fuzzy TOPSIS methods for facility location selection. In: *The International Journal of Advanced Manufacturing Technology* 39 (2008), Nr. 7-8, 783-795. http://dx.doi.org/10.1007/s00170-007-1249-8. – DOI 10.1007/s00170–007–1249–8. – ISSN 0268–3768

[Fea87] Feather, Martin S.: Language Support for the Specification and Development of Composite Systems. In: *ACM Trans. Program. Lang. Syst.* 9 (1987), März, Nr. 2, 198–234. http://dx.doi.org/10.1145/22719.22947. – DOI 10.1145/22719.22947. – ISSN 0164–0925

[Fra98] Franch, Xavier: Systematic Formulation of Non-Functional Characteristics of Software. In: *Proceedings of the 3rd International Conference on Requirements Engineering: Putting Requirements Engineering to Practice*. Washington, DC, USA : IEEE Computer Society, 1998 (ICRE '98). – ISBN 0–8186–8356–2, 174–181

[FS97] Fowler, Martin ; Scott, Kendall: *UML Distilled: Applying the Standard Object Modeling Language*. Essex, UK, UK : Addison-Wesley Longman Ltd., 1997. – ISBN 0–201–32563–2

[Gol13] Goli, Davoud: GROUP FUZZY TOPSIS METHODOLOGY IN COMPUTER SECURITY SOFTWARE SELECTION. In: *International Journal of Fuzzy Logic Systems;* Vol. 3 (2013), April, Nr. Issue 2, p29

[Gra92] Grady, Robert B.: *Practical Software Metrics for Project Management and Process Improvement*. Upper Saddle River, NJ, USA : Prentice-Hall, Inc., 1992. – ISBN 0–13–720384–5

[GRL08] GRL: *Goal-oriented Requirement Language*. 2008

[GSL14] Gnaho, Christophe ; Semmak, Farida ; Laleau, Regine: Modeling the Impact of Non-functional Requirements on Functional Requirements. Version: 2014. http://dx.doi.org/10.1007/978-3-319-14139-8_8. In: Parsons, Jeffrey (Hrsg.) ; Chiu, Dickson (Hrsg.): *Advances in Conceptual Modeling* Bd. 8697. Springer International Publishing, 2014. – DOI 10.1007/978–3–319–14139–88. – ISBN 978–3–319–14138–1, 59-67

[GW03] Geppert, B. ; Weiss, David M.: Goal-oriented assessment of product-line domains. In: *Software Metrics Symposium, 2003. Proceedings. Ninth International*, 2003. – ISSN 1530–1435, S. 180–188

[IEE98] IEEE: IEEE Guide for Software Requirements Specifications. In: *IEEE Std 830-1998* (1998)

[IF07] Inoki, Mari ; Fukazawa, Yoshiaki: Software Product Line Evolution Method Based on Kaizen Approach. In: *Proceedings of the 2007 ACM Symposium on Applied Computing*. New York, NY, USA : ACM, 2007 (SAC '07). – ISBN 1–59593–480–4, 1207–1214

[Ito07] Ito, Teruaki: Dealing with uncertainty in design and decision support applications. In: *International Journal of Soft Computing Applications* Vol.1 (2007), Nr. No.1, S. pp.5–16,

[JFS08] Jureta, Ivan ; Faulkner, Stéphane ; Schobbens, Pierre-Yves: Clear justification of modeling decisions for goal-oriented requirements engineering. In: *Requir. Eng.* 13 (2008), Nr. 2, 87–115. `http://dx.doi.org/10.1007/s00766-007-0056-y`. – DOI 10.1007/s00766–007–0056–y

[Jorsh] Jorg: *Elicitation of a Complete Set of Non-Functional Requirements.*, Diss., English

[Kav02] Kavakli, Evangelia: Goal oriented requirements engineering: a unifying framework. In: *Requirements Engineering Journal, Springer-Verlag London* 6 (2002), S. 237–251

[KHS02] Kaiya, H. ; Horai, H. ; Saeki, M.: AGORA: attributed goal-oriented requirements analysis method. In: *Requirements Engineering, 2002. Proceedings. IEEE Joint International Conference on*, 2002. – ISSN 1090–705X, S. 13–22

[KLM11] Klas, M. ; Lampasona, C. ; Munch, J.: Adapting Software Quality Models: Practical Challenges, Approach, and First Empirical Results. In: *Software Engineering and Advanced Applications (SEAA), 2011 37th EUROMICRO Conference on*, 2011, S. 341–348

[KR97] Karlsson, J. ; Ryan, K.: A cost-value approach for prioritizing requirements. In: *Software, IEEE* 14 (1997), Sep, Nr. 5, S. 67–74. `http://dx.doi.org/10.1109/52.605933`. – DOI 10.1109/52.605933. – ISSN 0740–7459

[KS98] Kotonya, Gerald ; Sommerville, Ian: *Requirements Engineering - Processes and Techniques*. John Wiley & Sons, 1998 `http://www.comp.lancs.ac.uk/computing/resources/re/`

[Lam00a] Lamsweerde, Axel van: Requirements Engineering in the Year 00: A Research Perspective. In: *Proceedings of the 22Nd International Conference on Software Engineering*. New York, NY, USA : ACM, 2000 (ICSE '00). – ISBN 1–58113–206–9, 5–19

[Lam00b] Lamsweerde, Axel van: Requirements engineering in the year 00: a research perspective. In: *ICSE*, 2000, S. 5–19

[Lam04] Lamsweerde, Axel v.: Goal-Oriented Requirements Enginering: A Roundtrip from Research to Practice. In: *Proceedings of the Requirements Engineering Conference, 12th IEEE International*. Washington, DC, USA : IEEE Computer Society, 2004 (RE '04). – ISBN 0–7695–2174–6, 4–7

[Lam09a] Lamsweerde, A. van: *Requirements Engineering: From System Goals to UML Models to Software Specifications*. Wiley, 2009 http://books.google.de/books?id=AYk_AQAAIAAJ. – ISBN 9780470012703

[Lam09b] Lamsweerde, Axel: Conceptual Modeling: Foundations and Applications. Version: 2009. http://dx.doi.org/10.1007/978-3-642-02463-4_20. Berlin, Heidelberg : Springer-Verlag, 2009. – ISBN 978–3–642–02462–7, Kapitel Reasoning About Alternative Requirements Options, 380–397

[Lap05] Lapouchnian, Alexei: Goal-Oriented Requirements Engineering: An Overview of the Current Research. In: *RE*, 2005

[Leh05] Lehto, Marttiin P. Jari A. A. Jari A.: Decision-Based Requirements Engineering Process. In: *Workshop on Collaborative (embedded) Systems Development t, 6th International Conference on Product Focused Software Process Improvement, Profes* (2005)

[Let01] Letier, Emmanuel: *Reasoning about Agents in Goal-Oriented Requirements Engineering*. 2001

[LF91] Lamsweerde, Axel van (Hrsg.) ; Fugetta, Alfonso (Hrsg.): *ESEC '91, 3rd European Software Engineering Conference, Milan, Italy, October 21-24, 1991, Proceedings*. Bd. *550*. Springer, 1991 (Lecture Notes in Computer Science). – ISBN 3–540–54742–8

[LHM+14] Li, Feng-Lin ; Horkoff, J. ; Mylopoulos, J. ; Guizzardi, R.S.S. ; Guizzardi, G. ; Borgida, A. ; Liu, Lin: Non-functional requirements as qualities, with a spice of ontology. In: *Requirements Engineering Conference (RE), 2014 IEEE 22nd International*, 2014, S. 293–302

[LL00] Lamsweerde, Axel van ; Letier, Emmanuel: Handling Obstacles in Goal-Oriented Requirements Engineering. In: *IEEE Trans. Softw. Eng.* 26 (2000), Oktober, Nr. 10, 978–1005. http://dx.doi.org/10.1109/32.879820. – DOI 10.1109/32.879820. – ISSN 0098–5589

[LL02] Lamsweerde, Axel van ; Letier, Emmanuel: From Object Orientation to Goal Orientation: A Paradigm Shift for Requirements Engineering. In: *RISSEF*, 2002, S. 325–340

[LLb98] Lamsweerde, Axel V. ; Letier, Emmanuel ; (belgium, B-Louvain la-neuve: *Integrating Obstacles in Goal-Driven Requirements Engineering*. 1998

[LSR07] Linden, Frank J. van d. ; Schmid, Klaus ; Rommes, Eelco: *Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering*. Secaucus, NJ, USA : Springer-Verlag New York, Inc., 2007. – ISBN 3540714367

[LW92] Liou, Tian-Shy ; Wang, Mao-Jiun J.: Ranking Fuzzy Numbers with Integral Value. In: *Fuzzy Sets Syst.* 50 (1992), September, Nr. 3, 247–255. http://dx.doi.org/10.1016/0165-0114(92)90223-Q. – DOI 10.1016/0165–0114(92)90223–Q. – ISSN 0165–0114

[LW00] Leffingwell, Dean ; Widrig, Don: *Managing Software Requirements: A Unified Approach*. Boston, MA, USA : Addison-Wesley Longman Publishing Co., Inc., 2000. – ISBN 0–201–61593–2

[MB01] Malan, Ruth ; Bredemeyer, Dana: Defining Non-Functional Requirements / Bredemeyer Consulting. 2001. – White Paper

[MCL+01] Mylopoulos, John ; Chung, Lawrence ; Liao, Stephen ; Wang, Huaiqing ; Yu, Eric: Exploring Alternatives During Requirements Analysis. In: *IEEE Softw.* 18 (2001), Januar, Nr. 1, 92–96. http://dx.doi.org/10.1109/52.903174. – DOI 10.1109/52.903174. – ISSN 0740–7459

[MCN92] Mylopoulos, J. ; Chung, L. ; Nixon, B.: Representing and Using Nonfunctional Requirements: A Process-Oriented Approach. In: *IEEE Trans. Softw. Eng.* 18 (1992), Juni, Nr. 6, 483–497. http://dx.doi.org/10.1109/32.142871. – DOI 10.1109/32.142871. – ISSN 0098–5589

[MD14] Mohammad Dabbagh, Sai Peck L.: An Approach for Integrating the Prioritization of Functional and Nonfunctional Requirements. In: *The Scientific World Journal* Volume 2014 (2014) (2014), Nr. Article ID 737626, S. 13 pages

[MS11] Mansoor, Arfan ; Streitferdt, Detlef: On the Impact of Goals on Long-Living Systems. In: *Software Engineering (Workshops)*, 2011, S. 133–138

[Myl06] Mylopoulos, John: Goal-Oriented Requirements Engineering, Part II. In: *RE*, 2006, S. 4

[MZN10] Mairiza, Dewi ; Zowghi, Didar ; Nurmuliani, Nurie: An Investigation into the Notion of Non-functional Requirements. In: *Proceedings of the 2010 ACM Symposium on Applied Computing*. New York, NY, USA : ACM, 2010 (SAC '10). – ISBN 978–1–60558–639–7, 311–317

[NWZ06] Niazi, Mahmood ; Wilson, David ; Zowghi, Didar: Critical success factors for software process improvement implementation: an empirical study. In: *Software Process: Improvement and Practice* 11 (2006), Nr. 2, 193–211. http://dx.doi.org/10.1002/spip.261. – DOI 10.1002/spip.261. – ISSN 1099–1670

[Ols04] Olson, D. L.: Comparison of Weights in TOPSIS Models. In: *Math. Comput. Model.* 40 (2004), Oktober, Nr. 7-8, 721–727. http://dx.doi.org/10.1016/j.mcm.2004.10.003. – DOI 10.1016/j.mcm.2004.10.003. – ISSN 0895–7177

[Opr11] Opricovic, Serafim: Fuzzy VIKOR with an Application to Water Resources Planning. In: *Expert Syst. Appl.* 38 (2011), September, Nr. 10, 12983–12990. http://dx.doi.org/10.1016/j.eswa.2011.04.097. – DOI 10.1016/j.eswa.2011.04.097. – ISSN 0957–4174

[PF] http://www.bcs.org/content/conwebdoc/19584

[Poh10a] Pohl, Klaus: *Requirements Engineering: Fundamentals, Principles, and Techniques*. 1st. Springer Publishing Company, Incorporated, 2010. – ISBN 3642125778, 9783642125775

[Poh10b] Pohl, Klaus: *Requirements Engineering: Fundamentals, Principles, and Techniques*. 1st. Springer Publishing Company, Incorporated, 2010. – ISBN 3642125778, 9783642125775

[Reg01] Regnell, et a. B.: Requirements Mean Decisions Research issues for Understanding and Supporting Decision Making in Requirement Engineering. In: *In First Conference on Software Engineering Research and Practice (SERPS01),* (2001)

[Rom85] Roman, G. C.: A Taxonomy of Current Issues in Requirements Engineering. In: *Computer* 18 (1985), April, Nr. 4, 14–23. http://dx.doi.org/10.1109/MC.1985.1662861. – DOI 10.1109/MC.1985.1662861. – ISSN 0018–9162

[RS79] Ross, D. T. ; Schoman, K. E. Jr.: Classics in Software Engineering. Version: 1979. http://dl.acm.org/citation.cfm?id=1241515.1241537. Upper Saddle River, NJ, USA : Yourdon Press, 1979. – ISBN 0–917072–14–6, Kapitel Structured Analysis for Requirements Definition, 363–386

[Saa08] Saaty, Thomas L.: Decision making with the analytic hierarchy process. In: *Int. J. of Services Sciences,* Vol.1, (2008), Nr. No.1, S. pp.83 – 98. http://dx.doi.org/10.1504/IJSSCI.2008.017590. – DOI 10.1504/IJSSCI.2008.017590

[SAG] S, Vinay ; Aithal, Shridhar ; G, Sudhakara: *Integrating TOPSIS and AHP into GORE Decision Support System*

[SCSP10] Santos, Emanuel ; Castro, Jaelson ; Sánchez, Juan ; Pastor, Oscar: A Goal-Oriented Approach for Variability in BPMN. In: *Anais do WER10 - Workshop em Engenharia de Requisitos, Cuenca, Ecuador, April 12-13, 2010*, 2010

[SM98] Stelzer, Dirk ; Mellis, Werner: Success factors of organizational change in software process improvement. In: *Software Process: Improvement and Practice* 4 (1998), Nr. 4, 227–250. http://dx.doi.org/10.1002/(SICI)1099-1670(199812)4:4<227::AID-SPIP106>3.0.CO;2-1. – DOI 10.1002/(SICI)1099–1670(199812)4:4<227::AID–SPIP106>3.0.CO;2–1. – ISSN 1099–1670

[SMMM98] Sutcliffe, Alistair G. ; Maiden, Neil A. M. ; Minocha, Shailey ; Manuel, Darrel: Supporting Scenario-Based Requirements Engineering. In: *IEEE Trans. Softw. Eng.* 24 (1998), Dezember, Nr. 12, 1072–1088. http://dx.doi.org/10.1109/32.738340. – DOI 10.1109/32.738340. – ISSN 0098–5589

[Som95] Sommerville, Ian: *Software Engineering (5th Ed.)*. Redwood City, CA, USA : Addison Wesley Longman Publishing Co., Inc., 1995. – ISBN 0–201–42765–6

[SPS+12] Shao, Fei ; Peng, Rong ; Sun, Dong ; Lai, Han ; Liu, You-Song: An attribute-driven model for trustworthy requirements elicitation. In: *International Journal of Digital Content Technology and its Applications* 6 (2012), Nr. 23, S. 531–540

[Sta04] Standardization, International O.: *ISO/IEC TR 9126-4: Software engineering - product quality - Part 4 : Quality in use metrics.* ISO, 2004 http://books.google.de/books?id=LM5xkQEACAAJ

[SW00] Schmid, Klaus ; Widen, Tanya: Customizing the PuLSE^{TM} Product Line Approach to the Demands of an Organization. In: *Software Process Technology, 7th European Workshop, EWSPT 2000, Kaprun, Austria, February 21-25, 2000, Proceedings*, 2000, 221–238

[Ter09] Ternite, T.: Process Lines: A Product Line Approach Designed for Process Model Development. In: *Software Engineering and Advanced Applications, 2009. SEAA '09. 35th Euromicro Conference on*, 2009. – ISSN 1089–6503, S. 173–180

[VL01] Van Lamsweerde, Axel: Goal-Oriented Requirements Engineering: A Guided Tour. In: *Proceedings of the Fifth IEEE International Symposium on Requirements Engineering*. Washington, DC, USA : IEEE Computer Society, 2001 (RE '01), 249–

[Wie99] Wiegers, K.: First Things First: Prioritizing Requirements. In: *Software Development Online* 7 (1999), September, S. 48–53

[WL99] Weiss, David M. ; Lai, Chi Tau R.: *Software Product-line Engineering: A Family-based Software Development Process*. Boston, MA, USA : Addison-Wesley Longman Publishing Co., Inc., 1999. – ISBN 0–201–69438–7

[WL09] Wang, Tien-Chin ; Lee, Hsien-Da: Developing a Fuzzy TOPSIS Approach Based on Subjective Weights and Objective Weights. In: *Expert Syst. Appl.* 36 (2009), Juli, Nr. 5, 8980–8985. http://dx.doi.org/10.1016/j.eswa.2008.11.035. – DOI 10.1016/j.eswa.2008.11.035. – ISSN 0957–4174

[WPAOPL09] Werneck, Vera Maria B. ; Pádua Albuquerque Oliveira, Antonio de ; Prado Leite, Julio Cesar S.: Comparing GORE Frameworks: i-star and KAOS. In: *WER*, 2009

[WS03] Witold Suryn, Alain A.: ISO/IEC SQuaRE. The second generation of standards for software product quality. In: *IASTED 2003 - SEA 2003 November 3-5, 2003 Marinadel Rey, CA, USA* (2003)

[YM98] Yu, Eric ; Mylopoulos, John: Why Goal-Oriented Requirements Engineering. In: *REFSQ98* (1998)

[YT97] Yen, John ; Tiao, W. A.: A Systematic Tradeoff Analysis for Conflicting Imprecise Requirements. In: *Proceedings of the 3rd IEEE International Symposium on Requirements Engineering*. Washington, DC, USA : IEEE Computer Society, 1997 (RE '97). – ISBN 0–8186–7740–6, 87–

[Yu96] Yu, Eric Siu-Kwong: *Modelling Strategic Relationships for Process Reengineering*. Toronto, Ont., Canada, Canada, Diss., 1996. – UMI Order No. GAXNN-02887 (Canadian dissertation)

[Yue87] Yue, K.: What Does It Mean to Say that a Specification is Complete, 1987

[Zad99] Zadeh, Lotfi A.: Some Reflections on the Anniversary of Fuzzy Sets and Systems. In: *Fuzzy Sets Syst.* 100 (1999), April, 1–3. http://dl.acm.org/citation.cfm?id=310817.310818. – ISSN 0165–0114

[Zav97] Zave, Pamela: Classification of Research Efforts in Requirements Engineering. In: *ACM Comput. Surv.* 29 (1997), Dezember, Nr. 4, 315–321. http://dx.doi.org/10.1145/267580.267581. – DOI 10.1145/267580.267581. – ISSN 0360–0300

Erklärung

Ich bestätige, dass diese These kein Material enthält, das für die Vergabe eines anderen Abschlusses oder ein Diplom in meinem Namen angenommen wurde, in jeder Universität und zu meinem besten Bewusstsein, ich habe Referenzen verwendet, auf die ausdrücklich im Text referenziert wurde.

Ich bestätige, dass das Urheberrecht der im Rahmen dieser Arbeit veröffentlichten Werke mit dem Urheberrecht-Inhaber dieser Werke bleibt.

Ich gebe auch die Erlaubnis dafür, dass eine digitale Version meiner Dissertation im Internet zur Verfügung gestellt wird, über die digitale Forschungsrepository der Universität, die Bibliothekssuche und auch über Web-Suchmaschinen, es sei denn eine Zustimmung der Universität wurde gewährt, mit der den Zugang für einen Zeitraum beschränken wurde.

Mir wurde mitgeteilt, dass jede Unrichtigkeit der vorgelegten oben genannten Erklärung als ein Betrugsversuch bewertet wird, und nach § 7 Abs. 10 der Promotionsordnung, führt dies zu einem Abbruch des Promotionsverfahrens.