Comprehensive Methodology for Complex Systems' Requirements Engineering and Decision Making

Vikas Schukla

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Institut National des Sciences Appliquées de Toulouse (INSA de Toulouse)*

Présentée et soutenue le *06/01/2014* **par :**

VIKAS SHUKLA

Comprehensive Methodology for Complex Systems' Requirements Engineering and Decision Making

JURY

CAMILLE SALINESI	Professeur, Université de Paris 1 Panthéon-Sorbonne, France	Président du Jury
DANIEL KROB	Professeur, Ecole Polytechnique, Paris, France	Membre du Jury
GUILLAUME AURIOL	Maître de Conférences, INSA de Toulouse, Toulouse, France	Membre du Jury
HAMID DEMMOU	Maître de Conférences, Université Paul Sabatier, Toulouse, France	Membre du Jury
DOMINIQUE SÉGUÉLA	Expert Senior en Système Spatiaux Sol-Bord, Centre National d'Études Spatiales, Toulouse, France	Membre du Jury
PHILIPPE THUILLIER	Technical Director, Systems Engineering, Intelligent Systems/Altran (group solution) Toulouse, France	Membre du Jury

École doctorale et spécialité :
EDSYS : Informatique et Génie Industriel
Unité de Recherche :
Laboratoire d'Analyse et d'Architecture des Systèmes(8001)
Directeur de Thèse :
Guillaume AURIOL
Rapporteurs :
Camille SALINESI et *Daniel KROB*

UNIVERSITÉ DE TOULOUSE

ÉCOLE DOCTORALE EDSYS

T H È S E

en vue de l'obtention du

Doctorat de l'Université de Toulouse

delivré par l'Institut National des Sciences Appliquées de Toulouse

Mention : Informatiques et Génie Industriel

Présentée et soutenue par

Vikas SHUKLA

Comprehensive Methodology for Complex Systems' Requirements Engineering and Decision Making

Directeur de thèse : Guillaume Auriol

Préparée au LAAS-CNRS Toulouse

soutenue le 06 Janvier 2014

Jury

Rapporteurs :	Camille SALINESI, Professeur, Université Paris-1, Panthéon Sorbonne, Paris, France
	Daniel KROB, Professeur, Ecole Polytechnique, Paris, France
Directeur :	Guillaume AURIOL, Maître de Conférences, Institut National des Sciences Appliquées de Toulouse, Toulouse, France
Examinateur :	Hamid DEMMOU, Maître de Conférences, Université Toulouse-III, Paul Sabatier, Toulouse, France
Invités :	Dominique SÉGUÉLA, Expert senior en Systèmes Spatiaux Sol-Bord, Centre National d'Études Spatiales, Toulouse, France
	Philippe THUILLIER, Technical Director, Systems Engineering, Intelligent Systems / Altran (group solution) Toulouse, France

Acknowledgements

Once I read somewhere that, "*journey itself is more important than the destination*". I could not really understand what it meant that time, but now I know the subtle and salient message hidden within that simple looking phrase. The journey of this thesis did not started the day doctoral school selected my candidature, but long before ... Every single person who pointed me towards the right directions deserves my sincere thanks. Whenever, I walked through the right direction, I was provided by them with the signs and symptoms of right path.

First thanks to my father, who is my first guru, and source of all spirit and inspirations. He only taught me about good and bad, about righteousness and wickedness. Next, my sincere thanks to my mother who blessed me with the courage, patience and energy necessary to overcome various hurdles. My siblings for providing me with the just necessary environment, equilibrium, and space that I needed, and for taking on their shoulders completely the responsibility that I owed to my family during all these years of my stay in France.

I would like to express my immense gratitude to Yves and Mireille Karceles, my French family, who did not only hosted me for the first critical months of my stay in France, but provided me with the confidence of learning and accepting the new language, and adapting to the completely new culture, society and values.

I would like to thank Dr. Jean Arlat, director LAAS-CNRS, for providing me with all the resources, equipments and other necessary conditions to carry out the research activities with ease.

I would like to express my regards and gratitude to Dr. Camille Salinesi, Professor, University de Paris-1, Panthéon-Sorbonne, and Dr. Daniel Krob, Professor, Ecole Polytechnique, Paris, for honoring me by accepting, reporting and recommending my research work. I would equally like to pay my regards to other members of jury, Dr. Hamid Demmou, Associate Professor, Université Paul Sabatier, Toulouse, Mrs. Dominique Séguéla, Senior System Expert, CNES, Toulouse and Mr. Philippe Thuillier, Technical Director, Systems Engineering, Altran group, for their valuable recommendations, insights and feedbacks.

I would equally take opportunity to thank my Ph.D. advisor Dr. Guillaume Auriol, who selected me as his student, poured his invaluable efforts to refine me to become a better researcher and teacher, provided his full attention and stood by me in times of need, praised and criticized me for my ultimate good.

Dr. Vincent Albert and Lucie for maintaining the continuous flow of motivation, enthusiasm and musical interpretations of various issues of not only research, but of other crucial axes of life throughout the Vee-Cycle of PhD project. I would take the opportunity to equally thank Dr. Claude Baron for the encouragement and timely help she provided, and for the empathy she bore in difficult times of thesis.

Special thanks to Dr. Pierre-Emmanuel Hladik for providing me with the valuable guidance, motivation, support and encouragement to take final decision of opting for doctoral research and for providing me with the necessary feedbacks during masters degree-thesis and doctoral research work.

I would also like to pay my gratitude to senior members of Systems and Integration Engineering Team, Dr. Alexandre Nkesta, Dr. Abd-El-Kader Sahraoui, Dr. Philippe Esteban and Dr. Jean-Claude Pascal for wholeheartedly welcoming me in the team.

I would like to thank our Canadian collaborators at University of Waterloo, Dr. Keith William Hipel, for the fruitful summer research stay at System Design Department and for such a multi-perspective enriching experience.

Sincere thanks to Dr. Christophe Chassot, Dr. Didier Le Botlan, Mrs. Gwendoline Le Corre, Dr. Theirry Monteil and Dr. Slim Abdellatif for providing guidance, support and convivial integration into the faculty team at DGEI. I would equally like to thank other members of faculty team at DGEI and LAAS, Denis Carvin, Miguel Nuñez Del Prado Cortez, Maxime Cheramy, Johan Mazel, Code Diop, Abdelaziz El Fatni, Florian Perget, and Mathieu Claeys. I would like to take the opportunity to thank Mrs. Hélène Thirion, Secretary, EDSYS, Mrs. Joëlle Breau, Secretary, DGEI, INSA Toulouse, and Mrs. Sonia de Sousa, Secretary, ISI team, for promptly providing their support in the numerous tedious administrative tasks and procedures.

I feel greatly indebted to my colleagues of A-126, LAAS for their ceaseless flow of motivation, encouragement, discussions, feedbacks, translations, questions and for numerous subtle explanations of things coming from different walks of life. I would like to thank Damien Foures, Saurabh Indra, Sylvain Noblecourt, Roberto Pasqua, Romaric Guillerm, Rui Xue and Xinwei Zhang for sharing and filling the numerous coffee breaks at LAAS premises. Thanks to Lucille, Marion, Sven, Crock, Boubou, Max and Tony for numerous sweet and unforgettable moments.

Many thanks to Amit, Sonal, Gurunatha Kargal, Chandu, Thilak, Vikram, Rajesh, Shiva & Pushpa, Akshath, Mujahid, Nagendra, Tajuddin for numerous things. First, for sharing their precious moments and company during this journey. Second, for making me feel at home during this exceptionally long stay in France. Third and most important, for the timely breaks to make life look good and realize that the happiness was so affordable. Special thanks to my friend and roommate Naveen Kumar Channaiah for bearing so much of patience, providing moral support, and for sharing the stressful moments of the final year. Numerous thanks to Boris-Wilfried Nyasse, Claire Dufour, Maxime Corbion, Fan Wei, Yuxing Zhang, and Miao Jing for their help and collaboration during initial years of my stay in Toulouse. My friends of engineering degree with whom I shared different types of fate and time: difficult, cherishable and decisive — Kamal Dalakoti, Nishith Gupta, Anoop Painuly, Abhishek Kapruwan, Amit Joshi, Nishant Pandey, Aalap Maithani, Rachit Lohani, Rahul Jain, Vijay Bhaskar Semwal, Ankur Binwal, Vivek Shaily, Vipul Jain, Virendra Singh Dharamshaktu, Surender Deopa, Garima Singh, sarthak Nautiyal, Arshad Ali, Sachin Gupta, and Ranjan Saxena.

Finally, I would also take the opportunity to thank my old friends back home, with whom I grew up and shared my childhood, and have barely seen and met them during these last few years: Vivek Tiwari, Vivek Kukreja, Yuvraj Manish Bhatt, Dheeraj Gunwant, and Deepanshu Bhatt for making schooling worth remembering.

Mukam karoti vachalam
pangum langhayate girim
yat-kripa tam aham vande
shri-gurum dina-taranam

~~~~~~

To the supreme soul who created this most complex system called Universe...to my motherland India and beautiful country of France...to my Family & my Friends
~~~~~~

Contents

General Introduction **1**

1 Context and Problem Formulation **7**
1.1 Introduction . . . 7
1.2 Problem Context . . . 8
1.2.1 Complex Systems . . . 8
1.2.2 Systems Engineering . . . 8
1.3 Problem Focus . . . 13
1.3.1 Technical Processes . . . 13
1.3.2 Decision Management Processes . . . 15
1.3.3 Vee-Model of Project Development Life-Cycle . . . 15
1.4 Research Focus . . . 17
1.4.1 Requirements Engineering . . . 18
1.4.2 Requirements Management & Traceability . . . 21
1.4.3 Decision-making . . . 23
1.5 Conclusion . . . 28

2 Requirements Engineering **29**
2.1 Introduction . . . 29
2.2 What are actually requirements ? . . . 30
2.2.1 State of Art Requirements Engineering Techniques . . . 36
2.2.2 Proposed Formulations on Requirements . . . 40
2.2.3 Proposed Comprehensive Requirements Modeling Language . . . 41
2.3 Writing Natural Language Requirements . . . 47
2.3.1 State of Art of NLRs Writing Techniques . . . 47
2.3.2 Proposed Approach for Writing Requirement . . . 53
2.3.3 Experiment and Empirical Findings . . . 55
2.3.4 Using Negation to Negotiate the Requirements . . . 58
2.4 Discussion . . . 58
2.5 Conclusion . . . 59

3 Requirements Traceability **61**
3.1 Introduction . . . 61
3.2 Requirements Traceability . . . 62
3.2.1 Requirements Traceability Processes and Problems . . . 63
3.2.2 Traceability Recovery Challenges . . . 68
3.3 State of Art of Requirement Traceability . . . 70
3.3.1 Information Retrieval Based Techniques . . . 70
3.3.2 Structurally Rule Based Techniques . . . 74
3.3.3 Linguistically Rule Based . . . 75

3.3.4 Transformation Rule Based . . . 75
3.3.5 Other Miscellaneous Based . . . 76
3.3.6 Works on Traceability Maintenance . . . 82
3.3.7 Traceability For Systems Engineering . . . 83
3.4 Proposed Solution for Traceability Problems . . . 84
3.4.1 Semantics of Relationships for Requirement Traceability . . . 85
3.4.2 Planning and Managing Traceability Strategy . . . 87
3.4.3 Trace Creation Process . . . 89
3.4.4 Trace Maintenance Process . . . 93
3.4.5 Trace Usage . . . 96
3.4.6 Using Traceability Information for SE Activities . . . 96
3.4.7 Comprehensive Traceability During Project Development . . . 98
3.5 Discussion . . . 101
3.6 Conclusion . . . 102

4 Decision-Making in SE 105
4.1 Introduction . . . 105
4.2 Decision Analysis in SE . . . 106
4.2.1 Issues with Decision Making . . . 109
4.2.2 Criteria Weighting Problem in Systems Engineering . . . 110
4.3 State of Art of Decision Making . . . 112
4.3.1 Multi Criteria Decision Making Methods . . . 112
4.3.2 Game Theory Based Conflict Resolution and Negotiation . . . 117
4.3.3 State of art of Criteria Weighting Techniques . . . 118
4.4 Proposed Methodology . . . 123
4.4.1 Roles in Decision making for SE . . . 124
4.4.2 Prerequisite to technique . . . 125
4.4.3 Methodology . . . 125
4.4.4 Optimality Check . . . 131
4.5 Simple Example . . . 131
4.6 Analysis and Comparison With Other Technique . . . 137
4.6.1 Optimality Check . . . 137
4.6.2 With Entropy . . . 137
4.6.3 With Rank Order Centroid . . . 138
4.6.4 With Eigen-vector from AHP . . . 138
4.7 Discussion . . . 139
4.8 Conclusion . . . 140

5 Integrating Requirements Engineering and Decision Making 143
5.1 Introduction . . . 143
5.2 Comprehensive Methodology: Integrating Concepts . . . 144
5.2.1 Methodology . . . 144
5.2.2 Tool Support: SysEngLab . . . 147
5.3 Case Study: Iron Bird Integrated Simulator . . . 152

5.3.1 Assumptions . . . 154
5.3.2 IBIS Stakeholders Needs Elicitation . . . 154
5.3.3 IBIS System Requirements Definition . . . 165
5.3.4 IBIS Architecture Design and Analysis . . . 171
5.3.5 Landing Gear Detail Design . . . 173
5.3.6 Deciding Specifications using our Technique . . . 175
5.3.7 Deciding Design Components . . . 177
5.4 Requirements Traceability . . . 182
5.4.1 Purposed Traceability . . . 183
5.4.2 Cost-effective Traceability . . . 183
5.4.3 Pre-requirement traceability . . . 183
5.4.4 Post-requirement traceability . . . 186
5.5 Limitations and Conclusions . . . 188

Conclusion and Future Perspectives **189**

A Tools Developed **197**
A.1 SysEngLab . . . 197
A.2 RequirementLab . . . 197
A.3 DecisionLab . . . 198

B Résume en Français: Approche Globale d'Ingénierie Systèmes **199**
B.1 Introduction . . . 199
B.2 Systèmes Complexes . . . 200
B.3 Ingénierie Systèmes . . . 200
B.4 Ingénierie des Exigences . . . 202
B.5 Gestion des Exigences et de la Traçabilité . . . 204
B.6 La Prise de Décision et Résolution de Conflit . . . 205
B.7 Approche Globale . . . 207
B.8 Conclusion . . . 209

C List of Publications and Reports **211**

Bibliography **213**

Index **234**

List of Figures

1 Thesis Goals & Objectives . . . 3
2 Thesis Outline . . . 5

1.1 System Life-Cycle Processes (ISO 15288:2008) . . . 11
1.2 Generic Life Cycle (ISO 15288:2008) . . . 12
1.3 Vee-Model of Development Life-Cycle . . . 16

2.1 Taxonomy of Requirements . . . 34
2.2 Requirements Relationships . . . 35
2.3 Relating Requirements, Rationales and Viewpoints . . . 40
2.4 CReML Goal Meta-model . . . 42
2.5 CReML Responsibility Meta-model . . . 43
2.6 CReML Strategy Meta-model . . . 43
2.7 Taxonomy of Ambiguity Types . . . 51
2.8 Restricted Interpretation . . . 54
2.9 Gross Percentage of Affirmative Phrases vs Negative Phrases . . . 56
2.10 Individual Percentage of Affirmative Phrases vs Negative Phrases . . . 57

3.1 Current Traceability Process . . . 64
3.2 Traceability Relationships and Infrastructure . . . 85
3.3 Trace Information Model . . . 86
3.4 Proposed Traceability Process . . . 88
3.5 Proposed Traceability Maintenance Trace . . . 94
3.6 Trace Addition . . . 94
3.7 Trace Suspension . . . 95
3.8 Proposed Traceability Usage . . . 96

4.1 Interest Vs. Power Grid . . . 110
4.2 Taxonomy of MCDM Methods . . . 113
4.3 Decision Makers Classification . . . 124
4.4 Proposed Decision Methodology . . . 126
4.5 Interest vs. Influence Grid . . . 127
4.6 Decreasing Utility Function . . . 128
4.7 Taking in Account Differences in Utilities of Categories . . . 130
4.8 Categorization Using Interest vs. Influence Grid . . . 133
4.9 Score vs. Rank . . . 135
4.10 Criteria vs. Weight . . . 138

5.1 Comprehensive Methodology . . . 145
5.2 Example of Solution Components . . . 150
5.3 Solution According to Business Engineer . . . 151

5.4 Life Cycle of Simulation in SE . . . 153
5.5 IBIS Context Diagram . . . 155
5.6 IBIS Rationale Map . . . 158
5.7 IBIS Goal Diagram . . . 159
5.8 IBIS Viewpoint Map . . . 159
5.9 IBIS Constraints . . . 160
5.10 IBIS Viewpoint Map . . . 160
5.11 IBIS Objectives . . . 161
5.12 Stakeholder Weighting . . . 162
5.13 Criteria Weighting . . . 163
5.14 Test Case for Customer Requirements . . . 165
5.15 Architecture of IBIS Platform . . . 166
5.16 IBIS Simulation Infrastructure Platform . . . 167
5.17 IBIS Landing Gear . . . 168
5.18 IBIS Landing Gear System . . . 168
5.19 IBIS Landing Gear System Requirements . . . 169
5.20 Landing Gear Derived System Requirements . . . 169
5.21 IBIS Landing Gear MiTL Requirements . . . 170
5.22 Validation Test cases for Landing gear of IBIS . . . 171
5.23 IBIS Landing Gear User Interface Requirements . . . 172
5.24 IBIS Landing Set Detail Design . . . 173
5.25 Landing Gear Signals and Interfaces Specifications . . . 174
5.26 Landing Gear Signals and Interfaces Specifications . . . 175
5.27 Categorizing Stakeholder . . . 175
5.28 Categorizing Stakeholder . . . 176
5.29 Generating Criteria Weights . . . 176
5.30 Stakeholder Categorizing and Weighting . . . 178

A.1 SysEngLab Console . . . 197
A.2 DecLab module in SysEngLab Console . . . 198

List of Tables

1.1 Increasing Complexity of Aircraft Simulators (1989-2002) 10
1.2 Stages and their Purpose in Life Cycle ISO/IEC 15288:2008 12
1.3 Technical Process 13
1.4 Keeney's Classification of Decision Theories 25

2.1 Terminologies Used in Requirements Engineering 31
2.2 Requirement Definition by SE Standards 33
2.3 Comparing popular Gore RMLs 38
2.4 Requirement Artifacts definition in CReML 43
2.5 Comparing Different Approaches for Ambiguity Problem in NLRs . . 50
2.6 Table of Requirements Using and Without Using Negation 55

3.1 Comparative Analysis of Various Traceability Techniques 79

4.1 Comparing Alternative Focused and Value focused thinking 107
4.2 Decision Contexts 107
4.3 MCDM Techniques 116
4.4 Subjective Criteria Weighting Techniques 122
4.5 Multi Criteria Alternative Evaluation 131
4.6 Design Criteria of a Hybrid Car 132
4.7 Hybrid Car 133
4.8 Design Criteria categorization 133
4.9 Design Criteria Scores 135
4.10 Normalized Design Criteria Weight 136
4.11 Multi Criteria Alternative Evaluation 136

5.1 User Stories 156
5.2 Prioritizing Goals 162
5.3 Goal Categorization 162
5.4 Goal Weights 163
5.5 Writing Requirements with Negation 164
5.6 Design Criteria for Specification Selection 175
5.7 Stakeholder Weight 176
5.8 Criteria Weights 176
5.9 Normalized Criteria Weights 176
5.10 Evaluation Degree of Redundancy 177
5.11 Design Criteria for Landing Gear Component Selection 177
5.12 Design Criteria Categorization 179
5.13 Design Criteria Scores 180
5.14 Normalized Design Criteria Weight 180

5.15 Door Actuator Alternatives 181
5.16 Extension and Retraction Actuators 182
5.17 Stakeholder Rationale Goal Traceability Matrix 184
5.18 Objectives Stakeholders Goals Traceability Matrix 184
5.19 Pre-requirement Traceability 185
5.20 Post-requirement Traceability 187

List of Abbreviations

AFIS Association Française d'ingénierie Système

AHP Analytical Hierarchy Process

AIAA American Institute of Aeronautics and Astronautics

AIT Arrow's Impossibility Theorem

ANP Analytical Network Process

CBA Cost-Benefit Analysis

CFG Customer Focus Group

CM Configuration Management

CMMI Capability Maturity Model Integration

DFD Data Flow Diagram

DM Decision Maker

DSS Decision Support System

EIA Electronics Industries Alliance

FFBD Functional Flow Block Diagram

H/W Hardware

HoQ House of Quality

IBIS Iron Bird Integrated Simulator

IEEE The Institute of Electrical and Electronics Engineers

IEEE/EIA 12207 IEEE/EIA Standard 12207, Software Life Cycle Processes

INCOSE International Council of Systems Engineering

M&S Modeling and Simulation

MAUT Multi Attribute Utility Theory

MBSE Model Based Systems Engineering

MCDA Multi Criteria Decision Analysis

MCDM Multi Criteria Decision Making

MNS Mission Need Statement

MoA Matrix of Alternatives

MoE Measure of Effectiveness

MoP Measure of Performance

OMG Object Management Group

OOSEM Object Oriented Systems Engineering Method

PERT Program Evaluation and Review Technique

RE Requirements Engineering

RM Requirements Management

RML Requirement Modeling Language

RT Requirement Traceability

S/W Software

SADT Structured Analysis and Design Technique

SD System Design

SE Systems Engineering

SEBoK Systems Engineering Body of Knowledge

SIP Stakeholder Identification Process

SME Subject Matter Expert

SoS System of Systems

SoSE System of Systems Engineering

StdV Standards Viewpoint

SV Systems Viewpoint

SysML System Modeling Language

TD Technology Development

TRL Technology Readiness Level

UML Unified Modeling Language

V&V Verification and Validation

VV&A Verification Validation and Accreditation

WP Work Package

General Introduction

This chapter briefly introduces the research work accomplished during the thesis. The research works presented in this thesis were carried out in Laboratoire d'Analyse et d'Architecture des Systèmes of Centre National de la Recherche Scientifique (CNRS-LAAS), Toulouse, France, with the research funding from Ministry of Higher Education and Research (MESR). The research was carried out with Ingénierie Système et Intégration (ISI) group, i.e., Systems and Integration Engineering group at LAAS-CNRS. ISI team works in the context of design of complex systems by improving development life-cycle processes, particularly requirement engineering & management, modeling & simulation, verification & validation, and safety engineering, in a method, process and tool vision.

First the background and motivation of this thesis are introduced. The primary research goal of the thesis, i.e., comprehensive methodology for the design in systems engineering is presented. Next, in order to achieve the primary goal the primary research goal is divided into a set of three sub-goals: requirements engineering, requirements management & traceability, and decision making. The necessary technical objectives to achieve the sub-goals are identified and research themes necessary to achieve these objectives are determined. Next, a brief insight into the actual research themes is presented and their significance to the systems engineering is high-lightened. The research problems are identified and difficulty to address them is also discussed briefly. Finally, structure of the thesis and the links between the various chapters are presented.

Background and Motivation

Recently, there has been a surge in the complexity of our immediate environment in which we live. We find ourselves surrounded with numerous technologies, to carry out our normal day to day tasks. The rate of induction of these new technologies to our environment has took us with surprise. Our daily routine as compared to our immediate predecessors has changed enormously. At any instant, we can access information easily, and also we are becoming both consumer and producer of the information. With the latest innovations in the technologies nothing seems to be impossible. With these completely new radical innovative technologies, we can achieve what was once thought to be impossible. These new technologies have given birth to new necessities for complex systems, which are safer, more capable, more robust... Contrary to this, our existing product development processes are not yet well equipped to harness completely the benefits of these new techniques and technologies to deliver the organized complexity demanded into the products.

Over the years, the Standish group's Chaos reports [Eveleens 2010, Hull 2010] have reportedly shown the numbers of challenged projects nearly fixed at 46%, while the percentage of failed projects varying between 18%–40%. In fact recent Chaos

report shows 24% of project as failures. Also, with rise in the complexity of the products demanded, the traditional techniques used to develop them are often found to be unsuitable or lacking in multiple aspects. The techniques, tools, and human resources used to develop them remain under tremendous pressure. It is evident that, our means to develop such highly complex systems are far from being ready or suitable to develop them. We need completely new set of updated competent methodologies, processes and tools to develop such new highly complex systems and technologies, that are demanded nowadays and in near future. There is an immediate need to completely overhaul our existing product design methodologies, to take benefit from new technologies and deliver much more, better and faster.

The above mentioned concerns provide the motivation and fundamental necessity of research to address them properly. We formulate our research goals keeping in mind the previously identified concerns of system design industry. The primary goal identified for the research was *to improve systems engineering activities comprehensively to deliver with ease correct system with quality.* In this thesis, to achieve this primary goal, we divided it into three sub-goals: *to be able to deliver correct system demanded*, *to be able to deliver the quality of system demanded* and *to be able to develop system with low cognitive load.* To address the identified sub-goals relevant research themes were identified with the various axes of systems engineering (design and development of complex systems falls under a multidisciplinary domain called *Systems Engineering*). To achieve the three subgoals, a goal based requirements modeling language is introduced, certain guidelines to write natural language requirement are developed, a requirement traceability mechanism is introduced, a multi criteria-multi participant decision making method is developed, which can also be used for conflict resolution and negotiation. A few of the modifications are proposed for system modeling techniques.

For the first sub-goal, *to be able to deliver correct system demanded*, requirements engineering and requirements management were identified as major themes. For the second goal, *to be able to deliver the quality of system demanded*, three themes were identified requirements management, requirements traceability and decision making. Finally for third sub-goal i.e., *to be able to develop system with low cognitive load*, all four themes previously identified were recognized as necessary for its accomplishment. It is clear that, the issues with the three sub-goals are linked to each other. As, the three sub-goals are holistically linked and they need to be addressed holistically. Figure 1, shows the goals, subgoals and objectives of this thesis. Finally, the major contributions of this thesis can be classified in the following themes of research.

— Requirement Engineering,

— Requirement Management & Traceability,

— Decision Making/Engineering.

Following, we briefly present the importance of each theme to the problem and what were the outcome of research from each of the themes, and show how they are linked together to our primary goal.

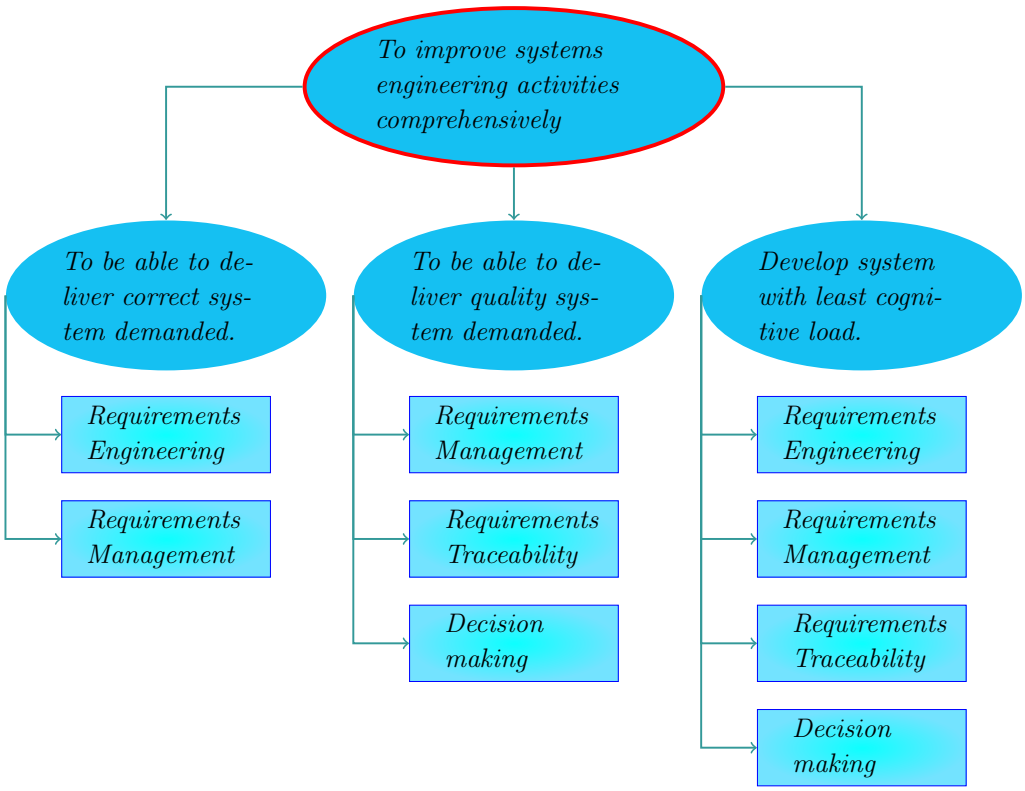


Figure 1: Thesis Goals & Objectives

Requirement Engineering, Management & Traceability

Requirements Engineering activities start as soon as the projects sets out. It is carried out in an early phase of a system development program to understand the client or end-users' needs clearly. Requirements Engineering involves a set of activities, which allow to understand the end-users' actual need and transform them into a contractual set of user-needs also called user requirements/customer requirements. It is the most important phase of a project, as if it is sure that a requirements engineering team has elicited all the right set of user requirements in the very beginning, and if given to proper resources in forthcoming phases, the project is destined to succeed. While, on the contrary, if even a trivial requirement is missed or misunderstood at this stage, it's sure to cost dearly to the stakeholders involved in the later phases. The later one discovers a problem, the higher it costs to them.

Although requirements engineering literature is largest among all the three themes concerned with the thesis, there are still some of the problems and hurdles left, which need to be solved. There are substantially large number of techniques proposed to carry out requirements engineering, but a few of them are really popular. The majority of problems linked with the requirements arises during the transition of the requirements, i.e., when requirements are passed from one person to other. This aspect may seem to be trivial for someone, who has never worked

in the systems engineering industry, but nowadays in the era of globalization the development is carried out round the globe and round the clock. Which leaves a big space for blunders about the interpretation of requirement.

Requirements management & traceability can be visualize together as one management type activity, but still there is subtle boundary between them. Requirements management & traceability activities involve all the activities which are linked to their handling, i.e., prioritization, implementation, negotiation and traceability of requirements throughout the life-cycle. The requirement management is all about how requirements are handled and carried out throughout the life-cycle of the project. Requirements management activities are directly responsible for the quality of the end product. A proven requirements management process will lead to greater user confidence in the product. It can help to reduce the uncertainty of the project success or failure. Requirements engineering should provide requirement traceability, which is one of the recommended activities for the project development process. Requirement traceability in systems engineering projects is much more sought-after owing to their long lifespans. We show in the later chapters that, our approach can provide the right traceability for right engineering activity with least effort. More issues and details about requirements engineering, management & traceability are presented in Chapter 2 & 3.

Decision Making and Conflict Resolution

In a systems engineering project, decision making and conflicts are pervasive phenomenon, decisions are made and needed, as soon as the stakeholder identification process starts. As, systems engineering project development involves interaction among multiple disciplines, each having more or less vague understanding of others, this often leads to emergence of conflicts during project as it progresses. Some conflicts need tacit efforts and are resolved mutually, but a few of them are severe conflicts. Such severe conflicts need to be resolved in order to go through the project development. Similarly, evaluating and choosing a right architecture or component requires efforts from multiple decision makers or stakeholders. User needs are elicited and transformed into the system requirements and later into system architecture, components, subcomponents,... Often, the organizations following the SE principles model more than one alternative solution to the systems architectures, which are later compared and evaluated over a few criteria to choose the best one. Various stakeholders have differences regarding the priority of their requirements often arrive at conflicts. These conflicts among the stakeholders, may emerge at any point of the life cycle of product. System design activity begins after the system requirements are fixed. Once the system requirements are fixed various system behavior models are proposed. The follow-up process of these behavior models is physical allocations to subsystems and components. During this step of system-modeling various problems arise: how to select the best set of subsystems and components to implement the behavior? As there are numerous set of solutions proposed, it becomes tedious task to analyze the various available possible configurations for system design. Of-

ten, these numerous solutions lead to conflicts among the decision-makers. These conflicts can only be resolved by properly evaluating the proposed various solutions, but these evaluations depend on the preference of the various system stakeholders which are also decision-makers. An appropriate decision strategy can help to avoid lot of wastage of time and resources.

Decision making is one of the area which can help to augment the quality of the product and cut short the delivery time of the product by resolving the conflicts and providing solutions to the evaluations of the requirements, architecture solutions, design components,... More details about decision-making are presented in Chapter 4.

Structure of the thesis

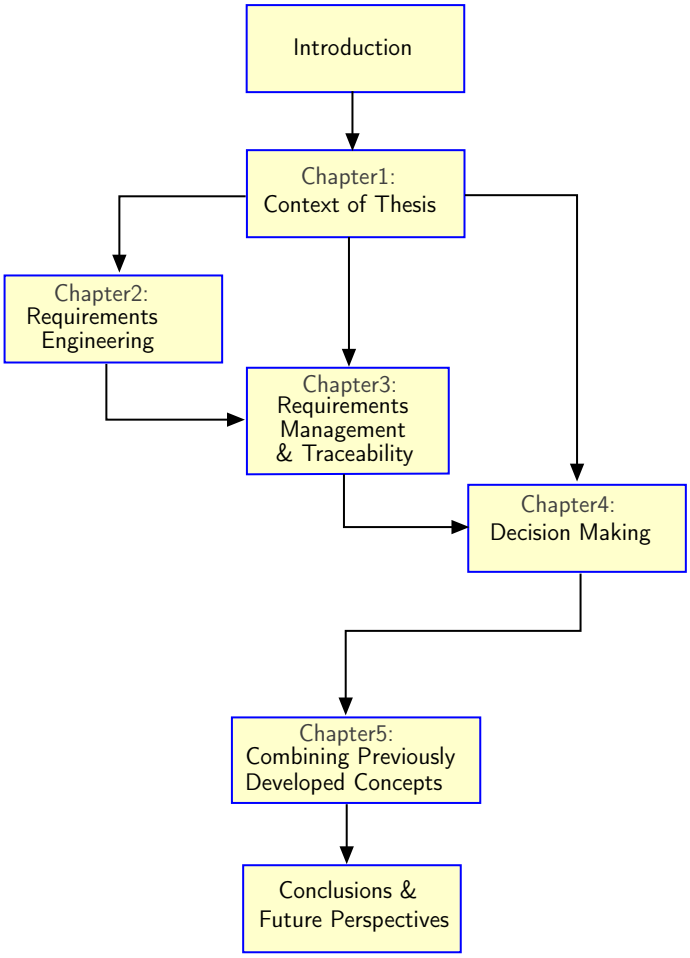


Figure 2: Thesis Outline

Figure 2, shows the outline of this thesis and how its different chapters are linked together giving solution to the subgoals and themes shown in Figure 1. Following

to this general introduction, Chapter 1, presents the context of the problem i.e., the fundamental notions of systems engineering and its activities, with respect to a few of standards EIA-630 [EIA632 2005], INCOSE handbook of systems engineering [Haskins 2011], IEEE 1220 [IEEE 2005]. It presents the problem focus, i.e., the product life cycle and product development cycles. Finally, research foci of this thesis are presented, i.e., the RE & RM, system modeling, and aspects related to decision-engineering, it provides the links between the previously mentioned activities. Major problems for each foci are identified to carry out the research work.

Chapter 2, presents the state of art of requirement engineering and management. The problems in the themes of requirement engineering and management are exposed. The proposed solution are also presented, together with results and brief discussion over each solution brought forward.

Chapter 3, presents the issues with the requirements traceability in the life cycle. The state of art if requirements traceability with different objectives is presented, various techniques are compared and then our solutions are presented to a few of the problems.

Chapter 4, presents the state of art of decision making involved in SE project, using the multi criteria group decision making, and conflict resolution. various problematics of decision making and conflict resolution are presented. Finally, our proposed solutions are presented and discussed how they contribute to the associated sub-goals.

Chapter 5, presents how all the previously introduced solutions and modifications brought to the various themes can be holistically integrated together to provide a comprehensive approach for the system design of complex systems. It also presents an application of our comprehensive approach, on a real case study of a system design: Iron Bird Integrated Simulator (IBIS) system. IBIS shows how our techniques can be used to create a complex system.

Final chapter presents the general conclusion of the research work carried out during the thesis and the future perspectives.

Chapter 1

Context and Problem Formulation

Contents

1.1 Introduction 7
1.2 Problem Context 8
1.2.1 Complex Systems 8
1.2.2 Systems Engineering 8
1.3 Problem Focus 13
1.3.1 Technical Processes 13
1.3.2 Decision Management Processes 15
1.3.3 Vee-Model of Project Development Life-Cycle 15
1.4 Research Focus 17
1.4.1 Requirements Engineering 18
1.4.2 Requirements Management & Traceability 21
1.4.3 Decision-making 23
1.5 Conclusion 28

1.1 Introduction

Over thousands of years of human civilization, humans have evolved to learn how to survive. Humans have surpassed other species by mastering the art of resolving their problems by using tools and techniques, which they developed from their immediate environment. Often, they created entities and structures which they used in their day-to-day life to increase their quality of life and decrease the amount of effort. These primary entities or structures were called simple *machines or tools*. Over the time, humans learned to use a set of these machines together to solve their problems, this set of interacting simple machines is called a *system*. Going forward with the evolution humans continued learning to use these set of primitive systems together to do relatively more difficult tasks. The moment humans interwove these simple systems in such a manner that resultant system was no more comprehensible, complex systems came into existence.

1.2 Problem Context

1.2.1 Complex Systems

As, it is clear from the introduction the term complex system is relative. A system may seem to be complex to one observer and simple to another. But in this thesis we should formally define the boundary of interpretation of the term complex system. The term complex system is previously defined in numerous references [Sage 2000, Haskins 2011, IEEE 2005, EIA632 2005, ISO 2008, IEEE 1998, IEEE 2008].

Definition 1.1. Combination of interacting elements organized to achieve one or more stated purposes [ISO 2008]. Complexity of a system is relative. A complex system often involves multiple interdependent agents, a purposeful behavior, it often learns and adapts and manifests an emergent behavior.

Definition 1.2. A combination of interacting elements organized to achieve one or more stated purposes [Haskins 2011]. An integrated set of elements, subsystems, or assemblies that accomplish a defined objective. These elements include products (hardware, software,firmware), processes, people, information, techniques, facilities, services, and other support elements.

Definition 1.3. A complex system is one that by design or function or both is difficult to understand and verify [Weng 1999].

Elements of a complex system

A complex system has many diverse components, these components are often interdependent, and there are usually nonlinear interactions, i.e., small changes can bring big effects and big changes may lead to small effects. A complex system acts on local knowledge and conditions, i.e., its not centrally controlled. It has a purposeful and autonomous behavior. Complex systems tend to have hierarchical organization.

Behaviors of complex systems

A complex systems solves complex problems, it continually reshapes its collective future, exhibits novelty and adapts itself, it learns from its experience and displays emergent and possibly unexpected and unpredictable behavior. Complex systems are both organized and varied.

1.2.2 Systems Engineering

The multidisciplinary domain concerned with the whole life-cycle of systems is often associated to systems engineering. SE is recognized as a preferred mechanism to establish agreement for the creation of products and services to be traded between the two or more organizations. It can be applied for any kind of system development: for a home appliance, an aircraft, a nuclear power plant, etc. Proper application of SE principles and methods can maximize the chances of success for a project.

Definition of Systems Engineering

There are various institutions which have previously tried to provide a formal definition to Systems Engineering. INCOSE, NASA, USDoD, USDoT, AFIS, have coined different definitions for SE.

Definition 1.4. INCOSE SE handbook [Haskins 2011] defines it as "*Systems engineering is an interdisciplinary approach and means to enable the realization of successful systems.It focuses on defining customer needs and required functionality early in the development cycle, documenting requirements, and then proceeding with design synthesis and system validation while considering the complete problem: operations, cost and schedule, performance, training and support, test, manufacturing, and disposal. SE considers both the business and the technical needs of all customers with the goal of providing a quality product that meets the user needs.*"

Definition 1.5. NASA SE handbook defines it as [NASA 2007] "*Systems engineering is the art and science of developing an operable system capable of meeting requirements within often opposed constraints.*"

SE focuses on defining customer needs and required functionality early in the development cycle, documenting requirements, and then proceeding with design synthesis and system validation while considering the complete problem: operations, cost and schedule, performance, training and support, test, manufacturing, and disposal. SE considers both the business and the technical needs of all customers with the goal of providing a quality product that meets the user needs. In simple terms, the SE approach consists of:

- Identification and quantification of system goals,
- Creation of alternative system design concepts,
- Performance of design trades,
- Selection and implementation of the best design,
- Verification that the design is properly built and integrated, and
- Post implementation assessment of how well the system meets (or met) the goals

In this thesis term SE or engineering project is used for representing hardware intensive systems. The term software engineering or software project is used to represent software intensive system. However, it is unquestionable that, the term SE is applicable for both hardware and software based systems. We consider a project to be of SE, when it has a majority of part in non-software subsystems. We use the term non-software intensive system for a SE project and a software intensive system for software engineering project.

1.2.2.1 Historical Developments

The term "*Systems Engineering*" is comparatively new and can be traced to early 1930s to Bell Telephone laboratories. The discipline of "*Systems Engineering*" was first time taught in 1950 at MIT [INCOSE 2013]. Over the years multiple SE standards have evolved, from MIL-STD 499 in 1969 to MIL-STD 499B in 1994, EIA-632 in 1998, IEEE-1220 in 1994 to IEEE-1220-2004, ISO/IEC 15288:2002 to ISO/IEC 15288:2008. SE as a discipline was formally recognized with the introduction of international standard ISO/IEC 15288 in 2002 [Haskins 2011]. SE is continuously evolving to the level *System of Systems Engineering* (SoSE).

First major applications of SE principles in modern age dates to World War II, when a British multi-disciplined team was formed in 1937 to analyze air defense system, and when Bell laboratories supported the development of project Nike during 1939-45 [Arunski 1999, Haskins 2011]. Space shuttle stands among one of the most complex systems ever developed using SE. Nuclear reactors, satellite launch vehicles, long-haul passenger aircrafts, aircraft carrier, supertankers, rapid-transit passenger transport system are few of the popularly known man made complex systems. The Table 1.1 provides an idea of increasing complexity of the systems under usage and development in recent times [Albert 2012]. Table 1.1 compares the complexity of various Airbus Industries aircrafts simulators through the span of 1989-2002.

Table 1.1: Increasing Complexity of Aircraft Simulators (1989-2002)

Metrics\Aircraft	**A-320 (1989)**	**A-330 (1992)**	**A-340 (1998)**	**A-380 (2002)**
Components	30	50	68	100
Line of Code	- -	- -	1,000,000	4,500,000
Manual Coding	- -	- -	47%	19%
Average CPU Load (models and exchanged data)	1	2	3	30
Avionics Signals	2000	4000	5000	9000

It is interesting to note that with the time the size of aircrafts have grown bigger so has the complexity of the system.

1.2.2.2 Systems Engineering Project Life-Cycle Processes

Like any man made engineering system, SE project has a life-cycle. Every SE project can be assumed to have two parts: product and project (management). Product part is the expected end-product and enabling products which are expected to satisfy or solve the problems of the client. Project part includes all the other necessary activities and resources which allow to develop the product itself. ISO/IEC 15288 standard divides the project life-cycle into four stages as shown in Fig.1.1, it also identifies four system life-cycle processes that are followed during the various stages of the project:

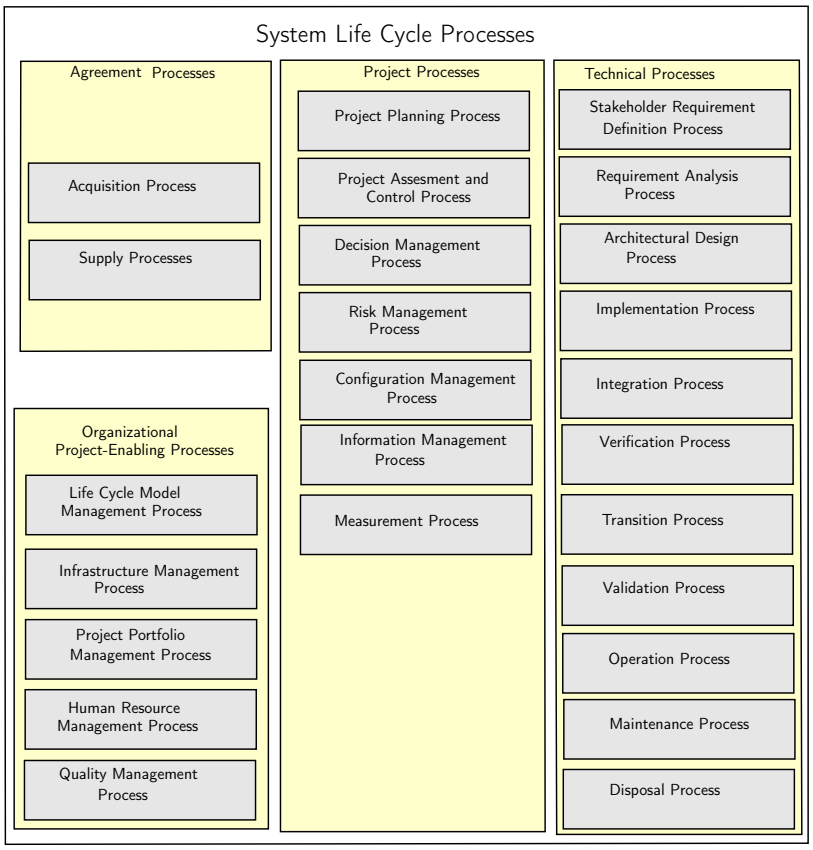


Figure 1.1: System Life-Cycle Processes (ISO 15288:2008) [ISO 2008]

1. Agreement Processes,
2. Organizational project-enabling processes,
3. Project Processes,
4. Technical processes,

The *Agreement Processes* are used to create the formal contracts of acquisition and supply between two organizations or between two teams of a same organization. They are employed to agree upon the various respective responsibilities of the organizations involved in the project.

The *Organizational project-enabling processes* are concerned with ensuring that the resources needed to enable the project to meet the needs and expectations of the organization's interested parties are met. They are typically the strategic policy process of the organization concern with remaining competitive and profitable in the business.

The *Project Processes* are concerned with managing the resources and assets allocated by enterprise management and with applying them to fulfill the agreements into which that organization enters. They relate to the management of projects, in particular to planning in terms of cost, time scales and achievements, to the checking

of actions to ensure that they comply with plans and performance criteria, and to the identification and selection of corrective actions that recover shortfalls in progress and achievement.

The *Technical Processes* are concerned with technical actions throughout the life cycle. They transform the needs of stakeholders first into a product and then, by applying that product, provide a sustainable service, when and where needed in order to achieve customer satisfaction. The Technical Processes are applied in order to create and use a system, whether it is in the form of a model or is a finished product, and they apply at any level in a hierarchy of system structure.

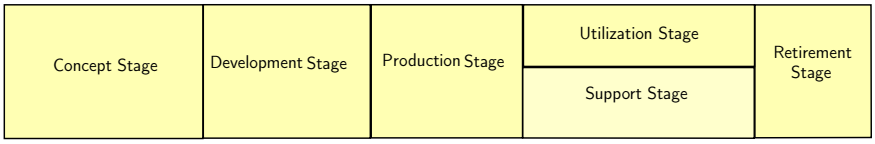


Figure 1.2: Generic Life Cycle (ISO 15288:2008)

A project life-cycle englobes all the activities surrounding the project from its inception to its retirement. Fig.1.2, shows a generic project life-cycle as shown in ISO/IEC/IEEE 15288:2008. A typical system engineering projects goes through various stages: concept stage, development stage, production stage, utilization stage, support stage and finally retirement stage. Table 1.2, presents the significance of

Table 1.2: Stages and their Purpose in Life Cycle ISO/IEC 15288:2008

Life Cycle Stages	**Purpose**	**Decision Point**
Concept	Identify stakeholders' needs Explore concepts Propose viable solutions	Decision Options: –Execute next stage, –Continue this stage, –Go to a preceding stage, –Hold project activity, –Terminate project
Development	Refine system requirements Create solution description Build system Verify and validate system	
Production	Produce systems Inspect and test	
Utilization	Operate system to satisfy users' needs	
Support	Provide sustained system capability	
Retirement	Store, archive or dispose of the system	

various stages in a generic life-cycle and the decision options available at each stage. Another term popularly used within systems engineering is '*development cycle*', which consists of only the phases which are linked to concept and development stages as shown in Fig 1.2 and Table 1.2. This thesis is primarily concerned with the Technical processes involved during the *development cycle*. A few aspects of the project processes which are directly linked to the Technical process are also studied.

1.3 Problem Focus

Already, we have presented the global context of this thesis, now we present the problem focus of the thesis, i.e., the precise scope of this thesis where solutions have been brought up, which concerns to our problems. Our problem focus lies in the Technical processes of the system life cycle processes and particularly in the development cycle of the product. Next sections will discuss the Technical processes and the development life cycle.

1.3.1 Technical Processes

Technical processes under ISO/IEC 15288 are further classified into eleven sub-processes as shown in Table 1.3. The technical processes concerns with the technical activities throughout the life cycle of the product from conception to retirements. But in this thesis work we limit ourselves to the development life cycle of the product, i.e., we limit ourselves to the process from stakeholder requirement analysis till architectural Design Process. The implementation process, integration process, verification process, transition process, validation process, operation process, maintenance process and disposal process are not in scope of our problem focus, we present them briefly but they would not find any mention in rest of this thesis.

Table 1.3: Technical Process

Process	**Definition**
Stakeholder Requirement Definition process	The purpose of the Stakeholder Requirements Definition Process is to define the requirements for a system that can provide the services needed by users and other stakeholders in a defined environment. It identifies stakeholders, or stakeholder classes, involved with the system throughout its life cycle, and their needs, expectations, and desires. It analyzes and transforms these into a common set of stakeholder requirements that express the intended interaction the system will have with its operational environment and that are the reference against which each resulting operational service is validated.
Requirements Analysis Process	The purpose of the Requirements Analysis Process is to transform the stakeholder, requirement-driven view of desired services into a technical view of a required product that could deliver those services. This process builds a representation of a future system that will meet stakeholder requirements and that, as far as constraints permit, does not imply any specific implementation. It results in measurable system requirements that specify, from the supplier's perspective, what characteristics it is to possess and with what magnitude in order to satisfy stakeholder requirements.

Table 1.3 – Technical Processes continued

Process	**Definition**
Architectural Design Process	The purpose of the Architectural Design Process is to synthesize a solution that satisfies system requirements. This process encapsulates and defines areas of solution expressed as a set of separate problems of manageable, conceptual and, ultimately, realizable proportions. It identifies and explores one or more implementation strategies at a level of detail consistent with the system's technical and commercial requirements and risks. From this, an architectural design solution is defined in terms of the requirements for the set of system elements from which the system is configured. The specified design requirements resulting from this process are the basis for verifying the realized system and for devising an assembly and verification strategy.
Implementation Process	The purpose of the Implementation Process is to realize a specified system element. This process transforms specified behavior, interfaces and implementation constraints into fabrication actions that create a system element according to the practices of the selected implementation technology.
Integration Process	The purpose of the Integration Process is to assemble a system that is consistent with the architectural design. This process combines system elements to form complete or partial system configurations in order to create a product specified in the system requirements.
Verification Process	The purpose of the Verification Process is to confirm that the specified design requirements are fulfilled by the system. This process provides the information required to effect the remedial actions that correct non-conformances in the realized system or the processes that act on it.
Transition Process	The purpose of the Transition Process is to establish a capability to provide services specified by stakeholder requirements in the operational environment. This process installs a verified system, together with relevant enabling systems, e.g., operating system, support system, operator training system, user training system, as defined in agreements.
Validation Process	The purpose of the Validation Process is to provide objective evidence that the services provided by a system when in use comply with stakeholders' requirements, achieving its intended use in its intended operational environment. This process performs a comparative assessment and confirms that the stakeholders' requirements are correctly defined.

Table 1.3 – Technical Processes continued

Process	Definition
Operation Process	The purpose of the Operation Process is to use the system in order to deliver its services. This process assigns personnel to operate the system, and monitors the services and operator-system performance.
Maintenance Process	The purpose of the Maintenance Process is to sustain the capability of the system to provide a service. This process monitors the system's capability to deliver services, records problems for analysis, takes corrective, adaptive, perfective and preventive actions and confirms restored capability.
Disposal Process	The purpose of the Disposal Process is to end the existence of a system entity. This process deactivates, disassembles and removes the system and any waste products, consigning them to a final condition and returning the environment to its original or an acceptable condition.

1.3.2 Decision Management Processes

As decisions need to be made during the architectural design process. ISO/IEC 15288:2008 mentions a decision management process specifically to the project processes. Our interests lies in the decisions which are made during the analysis of alternatives for component selections. For these reasons decision management process lies in our areas of interest.

ISO/IEC 15288:2008 states that: "the purpose of the decision management process is to select the most beneficial course of project action where alternatives exist. This process responds to a request for a decision encountered during the system life cycle, whatever its nature or source, in order to reach specified, desirable or optimized outcomes. Alternative actions are analyzed and a course of action selected and directed. Decisions and their rational are recorded to support future decision-making."

1.3.3 Vee-Model of Project Development Life-Cycle

There are various project development life-cycle models, such as Waterfall model, Spiral, Vee, and Agile Development models like XP, yPBL which are used to mark the various stages of life-cycle: definition, design, development, production, deployment, and withdrawal of the system. In this thesis, we considered the Vee-model of life-cycle owing to the ease of visualization it provides for the various Technical processes and its popularity and acceptance in SE community. We assume that the product development cycle follows Vee-model as shown in Figure 1.3. The development cycle is the focus of our research problems. We researched throughout the literature to determine the key activities which influence the majority of product and their success or failure.

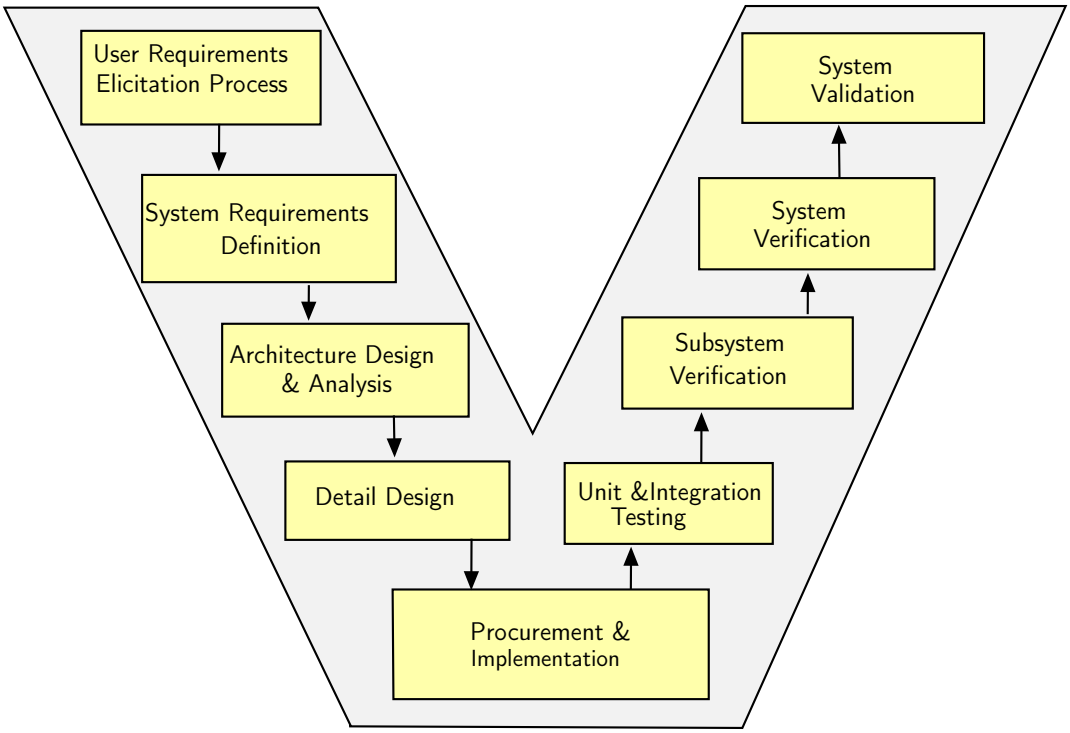


Figure 1.3: Vee-Model of Development Life-Cycle

The Vee-model of system development cycle shown in Fig.1.3, constitutes only the two premier stages of generic life cycle: concept stage and the development stage, as shown in Fig.1.2 [Forsberg 1995, Haskins 2011]. Vee-model shows the systematic flow of various processes in the development cycle of the product. The presented Vee-model comprises till the validation process of the presented Technical processes, hence it is only the project development cycle. An extended Vee-model covers all of the Technical processes of the project life cycle, from conception process till disposal process mentioned in [ISO 2008, Forsberg 1995].

User requirements elicitation phase's purpose is to elicit all the relevant user needs, desires, wants, rationales and determine exhaustively all the potential stakeholder and their corresponding roles, which can influence the success of the project. End result of this phase should provide exhaustive list of all the user/stakeholder needs with their origins.

System requirements definition phase takes input from the previous phase and transforms them into set of system requirements. Abstract models of the functional and structural architectures are developed to understand better the system requirements & vice-versa.

Architecture design & analysis phase involves development of concepts of structural and behavioral architectures to satisfy the system requirements. Multiple logical and structural architectures are proposed to satisfy the system requirements. System requirements are allocated to the logical and structural architectures and

architecture specifications are developed. Evaluations of architectures is carried out by a group of developing team and stakeholders, Once a particular architecture is accepted detail design is carried out.

Detail design phase involves transforming the architecture specifications into specific design specifications. The abstract architectures are rendered more precise. The goal of this phase this phase is a set of detailed design specifications that should result in a useful system product. Another product of this phase is a refined set of specifications for the operational deployment and evaluation phases of the life cycle. Again, design alternatives are evaluated and a final choice is made, which can be developed with detail design testing and at least preliminary operational implementation.

During the *Procurement and implementation* phase all the equipments and design components are procured from the respective organization and system is implemented as described in the detail design phase.

Unit & integration testing phase involves with every design equipment to be tested unit wise and with completely integrated to its proximate environment. The testing is carried out with respect to the detail design specifications previously developed.

Sub-system verification phase involves the test cases for the subsystem to verify that are in conformance with the subsystem architectural specifications previously developed in the architectural analysis phase.

System verification phase put to tests the system developed to verify that the system corresponds well and is in conformance with the system requirements. It is verified that all the system requirements are implemented into the system.

System validation phase involves putting system to series of test that were accepted and contracted with client during the stakeholder requirement elicitation process. These tests validate that system satisfies well all the the contracted needs of the stakeholders. Our problem focus remains in the development cycle of the product.

1.4 Research Focus

Currently, there are numerous issues in the development cycle of the product. This thesis tried to focus on the issues which seemed more influencing on the end-result. It is clear that the errors caused during the left hand side of the Vee-model of development cycle imply huge cost escalations and other unnecessary wasteful expenditures. If the solutions to the problems of the processes to the left hand side of Vee-cycle could be brought, then results of the processes to the right hand side of Vee-cycle would improve significantly. We have classified some of the problems of the left hand side into three broad categories: requirements engineering, requirements management & traceability and decision engineering. In this thesis, we present requirements engineering, requirements management & traceability separately owing to the extent of contribution in them, but in literature they are treated in single domain of requirements engineering.

1.4.1 Requirements Engineering

Before discussing about requirements engineering & management it is essential to know, what is a 'requirement'? In literature, there are various definitions of requirement, and its very hard to say which one is most appropriate. An appropriate definition of requirements [MITRE 2012] is "*a requirement is a singular documented need—what a particular product or service should be or how it should perform. It is a statement that identifies a necessary attribute, capability, characteristic, or quality of a system in order for it to have value and utility to a user.*"

INCOSE SE handbook [Haskins 2011] defines it as "*A statement that identifies a system, product or process' characteristic or constraint, which is unambiguous, clear, unique, consistent, stand-alone (not grouped), and verifiable, and is deemed necessary for stakeholder acceptability.*"

Requirements engineering & management are concerned with the elicitation, evaluation, specification, analysis, and evolution of the objectives, functionalities, qualities and constraints to be achieved by a system within an organizational or physical environment [van Lamsweerde 2009].

Requirements engineering is the discipline concerned with establishing requirements. It consists of requirements elicitation, analysis, specification, validation and verification methods (V&V), communicating, documenting and managing requirements. Requirements engineering is a set of interdisciplinary functions that mediates between the domains of the acquirer (customer) and the supplier (developer) to establish and maintain the requirements to be met by the system, software or service of interest [IEEE 2011].

Sommerville and Sawyer [Sommerville 1997], define it as "*a requirements engineering process is a structured set of activities which are followed to derive, validate, and maintain a systems requirements document.*"

Out of the eleven technical processes defined in ISO/IEC 15288:2008, stakeholder requirement definition process and requirement analysis process fall under the domain of requirements engineering, previously mentioned in Table 1.3.

Stakeholder is a popularly used term in SE issues and processes, understanding stakeholders is necessary in the context of project development, the term stakeholder can be defined as [ISO 2008] "*a party having a right, share or claim in a system or in its possession of characteristics that meet that party's needs and expectations.*"

"*A stakeholder in an organization is any group or individual who can affect or is affected by the achievement of the organization's objectives.*"[Freeman 2010]

The definition of stakeholder in the domain of systems & software engineering changes slightly. In the context of information systems, stakeholders may be defined as: "*A stakeholder is any individual, group, or organization that can affect or be affected (positively or negatively) by the system under study and that have direct or indirect influence on the systems.*"[Ballejos 2008], INCOSE SE handbook defines it as "*A stakeholder is any entity (individual or organization) with a legitimate interest in the system.*"

Requirements can be categorized in two categories: Stakeholders requirements &

System requirements. Stakeholders requirement are also known as user requirements or user needs, they are defined as abstract statements describing the system services which people need or desire to have in their system for use and integration with their organization [Sommerville 1997].

System requirements are detailed specifications of the system facilities which should be implemented and the constraints on that implementation. This description may be the basis of a contract for a system development so it should be a complete description of the behavior of the system [Sommerville 1997].

Stakeholder requirements are treated and transformed to system requirements, which represent more specifically what system should do in order to achieve the stakeholder requirements. A system requirement is more technically enriched form of stakeholder requirement.

The system specifications are derived following to system requirements, system specification are formalized and quantified system requirements; the system specifications typically include the following [Buede 2011, Sage 2000]:

1. A definition of the system as an entity,
2. The major characteristics of the system,
3. Related technical performance measures for system evaluation,
4. Some general criteria for design and assembly of the system,
5. Major data requirements for the system,
6. Logistics and producibility considerations,
7. Test and evaluation requirements,
8. Quality assurance provisions.

Stakeholder requirements are largely independent of any particular product team; they are not specific to a particular architecture or concept, Whereas specifications are more specific towards with a particular architecture or concept in focus [Ulrich 2008].

1.4.1.1 Requirements Elicitation, Modeling & Analysis

Requirement elicitation is the term used to designate the process of deriving requirements from the potential end-users, clients, and their environment. A substantial part of requirement elicitation is dedicated to uncovering, extracting, and surfacing the wants of the potential stakeholders. The major activities of the requirement elicitation involve, understanding the application domain, identifying the sources of requirements, analyzing the stakeholders, selecting the techniques, approaches and tools to use and eliciting the requirements from stakeholders and other resources [Zowghi 2005, Goguen 1993]. The process starts with the very basic need of product or system. The product need is followed up by stakeholder identification process (SIP). The SIP leads to various roles of the stakeholders influenced by the various rationales and beliefs. After identification of various stakeholders,

their needs are taken into account. This iteration is carried out suitable number of time, until the development team is satisfied with the SIP that all the stakeholders have been taken into account. These gathered needs are the result of first phase of product development life cycle. The gathered needs are also known as customer requirements or stakeholder requirements.

There are many techniques in the literature which describe to carry out the elicitation processes such as: interviews (closed and open), questionnaires, task analysis, domain analysis, introspection, repertory grids, card sorting, laddering, group work, brainstorming, joint application development, workshops, observation, protocol analysis, apprenticing, prototyping, goal based approaches, direct elicitation, ethnography, viewpoints, scenario generation, use-cases, Mind Maps, etc.

Requirement Modeling refers to techniques involved with the comprehension and representation of the context and environmental problem domain of the system under development. Requirement modeling allows to analyze and understand the problem and propose the pertinent solutions, and their representation using a formal or semi-formal technique. There are many Requirement Modeling Languages (RML) and techniques proposed in the literature such as: i* framework, KAOS framework, Use-cases modeling from UML & SysML, Z-language, B-language, Tropos based methodologies etc.

Although, there are numerous RMLs proposed in the literature Natural languages are usually the one which are used to capture the requirements formally or informally first-hand from the stakeholders, as they allow to document or write any type of need, wants or desire owing to the huge vocabulary it boosts. The process of writing down the requirements is also called requirement documentation in requirements engineering. Natural language remains the only viable solution by through which the requirements can be communicated and validated by the end-users, clients or stakeholders emerging from the various disciplines. Formal languages allow to represent the requirements in a unambiguous way, and are often sought by the developers as blue prints for development process. But, they are often source of irritation to customers as they can't understand it, which causes issues in making them contractual.

1.4.1.2 Research questions for Requirements Engineering

In literature, we have identified a few of the problems in requirement engineering and propose a few of pertinent solutions to them. State of art of relevant requirements engineering activities and the proposed solution for each challenge is presented in Chapter 2. Identified research question for the requirements engineering activities are listed as below:

Question 1.1. What are the relationships between the various requirement artifacts, customer requirements and system requirements ? what is their significance to clients and other stakeholders ?

Question 1.2. How to improve the early phase requirements modeling with ade-

quate level of scalability of graphical diagrams ?

Question 1.3. How to improve the modeling of the requirements such that their core and optional features of the system under development are clear to designer from the beginning of RE phase ?

Question 1.4. How to model the preference of the various stakeholders and weight the various goals and objectives of the system under the study during the requirement modeling phase?

Question 1.5. How to write requirements in a more uniguous manner, which are also easy to understand?

1.4.2 Requirements Management & Traceability

Requirements management is the process of managing existing requirements and requirements related artifact [Glinz 2011]. Requirements management en-globes activities that ensure requirements are identified, documented, maintained, communicated and traced throughout the life cycle of a system, product, or service [IEEE 2011, Pyster 2012]. Requirements management ensures that each of the desired requirement is built into the system according to their priority; if a requirement is abandoned, then provides the reasons for its seclusion.

1.4.2.1 Requirements Negotiation & Prioritization

Once the stakeholder requirements are elicited and analyzed by the developing team, before passing to the next stage, they need to be negotiated with the various stakeholders, designers and developers. There are many reasons to carry out this activity such as: exhaustive implementation of the all the requirements may not be feasible with the given resources, client stakeholders may not agree upon requirement, the level of requirements' priority or criticality. Or simply it may not be possible to implement the given requirement with given resources for the enterprise. All such requirements are put through the negotiation process. This process is known as requirements negotiation . The result of the negotiation process leads to an acceptable description or suitable modification of the requirement such that the stakeholders agree upon the decision and next stage can be carried on.

For developing a system, it is obvious that all the requirements or needs do not hold the same importance for stakeholders. The needs and thus leading requirements need to be given priority in order of their perceived importance or value to the stakeholder. This process is popularly known as requirements prioritization. There exist numerous techniques in the literatures but still RE community phases multiple problems while carrying out these activities [Parnell 2011, Jureta 2006].

1.4.2.2 Requirements Traceability

In systems development process, requirements traceability is the one of the anticipated attributes for the quality control. In EIA-632 standard [EIA632 2005] require-

ments traceability is described as an important task of outcomes management under control process requirements procedure. It is defined as, *requirements traceability is instituted for tracking requirements from the identification of acquirer and other stakeholder requirements to the system technical requirements, logical solution representations, physical solution representations, derived technical requirements, and specified requirements.*

Requirements traceability is seen as an index of quality in systems and is one of the anticipated attributes throughout the product life-cycle. Various norms like EIA-632, IEEE-1220 [IEEE 2005], ISO/IEC 15288 [ISO 2008], and CMMI [SEI 1994], recommend it as 'best practice' and strongly suggest its usage.

The recent research works of Gotel *et al.* [Gotel 2012a, Gotel 2012b] have led to recent developments in requirements traceability, by providing the much necessary vision for the future course of research. In [Gotel 2012b, Gotel 2012a] they explored in detail seven key challenges of traceability: *purposed*, *cost-effective*, *configurable*, *trusted*, *scalable*, *portable*, and *valued*. They identify the *grand challenge of traceability* (GCT) called *ubiquity*, which could only be achieved by achieving the seven other in a systematic manner.

Requirements traceability is always associated with artifacts. In literature, an artifact is defined as something that is created or shaped by humans, either directly or indirectly via automation [Gotel 2012b]. In other words a product, which may have originated during the course of development process or is utilized during the development process or later, and is important for the success of project. Requirements traceability process consists of tracing artifacts to the user needs, requirements, rationales, stakeholders, design, use cases, test cases, and constraints or to final disposal plan. This process consists of four major parts: creation, generation, maintenance and usage of traces. Requirements traceability finds vast usage in many aspects of product life-cycle: quality management, product verification and validation, testing, change impact analysis, maintenance plan, re-usability plan etc. It leads to better client satisfaction and a stronger confidence in product from customer side.

As previously mentioned, requirements traceability process implies four activities: trace creation, trace generation, trace maintenance, and trace usage. Trace creation can be reactive or proactive, usually it is reactive, i.e., traces are created as a response to user demand of trace capture, in proactive tracing the system proposes to capture traces. Traceability generation process can be manual, semi-automatic and fully automatic. Trace maintenance is most demanding activity and can be completely manual or semi-automated. Manual traceability generation techniques are often plagued with scalability issues, whereas the automatic traceability recovery techniques lack in precision and recall. The recovery process recovers the traces from the various existing artifacts. The trace maintenance process keeps track of evolution of the various requirements and various artifacts. The trace utilization process deploys the available information from the traces to various systems engineering activities like: configuration management, conflict resolution, system comprehension, project scheduling, etc.

The scope of requirements traceability can be different with different traceability policies or strategies. A good requirements traceability process in a project provides the accountability of every artifact from its very fundamental reason of existence. There are primarily three types of requirement traceability generation techniques: manual traceability, automatic traceability, semi-automatic traceability. Recent trends have shown that stakeholders are more and more aware of values provided by the requirements traceability. Value based traceability emerged as new paradigm in the domain of requirements traceability, it becomes evident to have an estimate of cost of traceability scheme, which an organization is supposed to use. This thesis tried to provide a semi-automatic approach for requirements traceability with taking in account important questions of value and cost of traceability .

1.4.2.3 Research questions for Requirements Management

A few of the research problems in requirements management which provide the basis of research work are presented below:

Question 1.6. How to negotiate, prioritize and weight the requirements with the stakeholders?

Question 1.7. How to engineer requirements such that requisite amount of trace requirements are implemented throughout the SE of life cycle of product?

Question 1.8. How to do requirements traceability in SE projects?

Question 1.9. How to maintain requirements traceability in SE project ?

Question 1.10. How to provide the cost effective requirements traceability?

Question 1.11. How to provide purposeful requirements traceability to stakeholders?

Question 1.12. Can we estimate the cost of requirements traceability? If yes how? if no why?

Solutions to above questions are presented and discussed in Chapter 3.

1.4.3 Decision-making

Decision making is a pervasive phenomenon of nature, from microscopic level to macroscopic levels, from plant kingdom to animal kingdom, every individual takes decisions at some point. Humankind is also busy in making decisions at one or another point of time. Every moment of a human life is surrounded by decisions, to do something or to not to do something. Sometimes these decisions are trivial and sometimes these decisions hold great importance. Often, the decisions made in the daily-life are based on single criterion and are made by a single decision maker. The responsibility of such a decision depends upon the sole single decision maker, but

at some instants some decisions involve more than one decision maker, thus adding fabric of complexity to the decision problem.

In this thesis, we often use two terms *decision-maker* and *stakeholder*. The term decision-makers refers to the entities, who are responsible for choosing a course of action or option. All decision-makers are stakeholders but the not all stakeholders are decision makers. Decision making becomes more complex, when multiple criteria get involved in the problem. Such problems, where multiple criteria and multiple decision makers are involved in a problem are called multi criteria decision making or analysis (MCDM/MCDA) problems [Triantaphyllou 2000]. Often, such is the case of systems engineering projects. Decisions are needed in a product development cycle, although they are made much before the project even starts, i.e., they are not only parts of the technical processes but also of the other three processes of the systems life-cycle and particularly of project processes. In this thesis, we concentrate on the decision making applicable to the product development cycle. Decisions are made whenever there are choices to be made. In systems engineering often a series of decision are made, such as series of decision can be termed decision-strategy. Decisions are also needed whenever conflicts broke out among stakeholders during the product development cycle. Often, in a SE project, decisions are made while taking in account multiple criteria and multiple decision makers.

A systems engineering project involves multiple stakeholders and multiple criteria during decision analysis [Sage 2000]. Such problems need attention and a formal methodology, to provide pertinent solutions. Variety of methodologies exist in the literature which provide more or less acceptable solutions, such as the famous AHP technique [Saaty 1990], Multi-attribute utility theory [Keeney 1993], ELECTRE Methods [Roy 1968, Roy 1978, Roy 1973], PROMETHEE [Brans 2005], TOPSIS [Hwang 1981], etc. In this thesis, we are concerned with a systematic decision making methodology for complex decisions involving MCDA/MCDM.

A systems engineering project typically involves a crowd of multi-disciplinary stakeholders. Success of a project depends upon the decisions and choices made during the analysis of architectures & alternatives, and during the selection of components during the detail design phase [Ulrich 2008]. Systems engineering is about making the right decisions to achieve the development of a product which is exactly demanded by their clients and stakeholders. These right decisions do not come automatically from a thin air, rather precise metrics are needed to evaluate the appropriate alternatives and right design components. As these decisions are based on these multiple criteria, they need to be weighted to measure their impact on the final decision choice. This task of providing weights to the criteria may seem trivial for a single decision maker, but when multiple stakeholders are involved this task becomes fairly difficult. As the different stakeholders differ upon the weights for various criteria, while each stakeholder being correct in his own view.

It is interesting to see that industries seldom use techniques which demand high cognitive loads on DMs. Even if a techniques is more correct technically but leads to high cognitive load, it will hardly find usage in industry. Industries prefer simple techniques to pacify majority of its non-technical stakeholders.

1.4.3.1 Approaches of Decision making

In literature, there are popularly three types of decision theories mentioned: *normative*, *descriptive* and *prescriptive*. *Normative theories* describe how the rational decisions should be made in an organization. They are also known as theory of rational choice. They assume that a rational decision-maker (DM), will choose an alternative with highest expected value. Most popular theory of the rational behavior under risk and uncertainty is *expected utility theory* [Von Neumann 1937], where the objective is to maximize the expected utility. In literature, there are multiple utility theories and there is no consensus on a single *utility theory*. In economics, *Utility* can be defined as "*the pleasure or satisfaction obtained from a good or service.*"

Decision making, is not new to the human being, thousands of years ago ancient civilizations have shown engineering and design skill evident from the monuments. Such designs needed some sort of decision making approach, but unfortunately they was never published. The earliest mentions of in *utility* in modern times can be traced to Gabriel Cramer (1728) [Cramer 1728] and Daniel Bernoulli (1738) [Bernoulli 1738] theory can be traced to Daniel Bernoulli in 1738, with his solutions of St. Petersburg Paradox. Table 1.4, presents the Keeney's classification of the decision theory [Keeney 1992].

Table 1.4: Classification of the Three Categories of Decision Theories [Keeney 1992]

Theories	**Domain**	**Criterion**	**Judges of Theories**
Normative	All decisions	Correctness	Theoretical sage
Descriptive	Classes of Decisions	Empirical Validity	Experimental Researcher
Prescriptive	Specific Decisions	Usefulness	Applied Analysts

Other well known works in utility theory are from Keeney & Raiffa [Keeney 1993], Fishburn [Fishburn 1970], subjective utility theory [Savage 1954], introduced by Savage in 1954. Normative theories assume that decision makers are able to make the required assessments of probabilities and utilities accurately. However, over the years experiments have shown that the axioms of the expected utility theory are often not held valid during the actual decision making. It has been argued that the decisions cannot be made by just analyzing the probabilities, which are often the base of utility theories, as they represent most of the time decision making problem by a lottery, which are obviously based on some probability function.

Descriptive theories describe how people actually make decisions. Descriptive theories use psychological and social behavior models to reason how decisions are actually made. One of the popular *descriptive theory* is the *prospect theory* proposed

by Kahneman and Tversky in 1979 [Kahneman 1979]. The descriptive theories argue that basic assumption of normative theories of complete human rationality is not only impractical but unrealistic. The judged *expected utility* of an alternative cannot be reproduced same by different rational stakeholders. Judging of utility depends upon other subconscious factors arising during the interaction process.

The discrepancy between theory and real behavior is the very heart of prescriptive interventions and decision analysis as a more pragmatic approach than what the normative theories suggest [Bell 1988]. In 1966, Howard coined the term *decision analysis* as a formal procedure for the analysis of decision problems.

Prescriptive decision theories are usually domain specific and they are judged my their usefulness or ease of usability, French *et. al.* [French 2000] suggested that a decision analysis theory should be able to perform reasonably on following clauses:

- *Axiomatic basis.* The axiomatic analysis should be acceptable to users and they should want their decision making to reflect the ideal behavior encoded in the set of axioms used for analysis.
- *Feasibility.* The techniques and methods should be easily usable, suggesting that input data elicitation process should be feasible.
- *Transparency to users.* The analysis process should be transparent to the various participating stakeholders and they should be able to understand the analysis process.
- *Robustness.* The sensitivity to the variations in input must be understood, e.g., the heavy dependency on a particular data should be taken in account during the analysis process.

1.4.3.2 Conflict Resolution & Decision making

Occurrence of conflicts in SE projects is part of reality in which projects are developed. During a SE project, often some stakeholders do not agree upon some decision, or some direction of flow of process and lead to a state of fix or indecisiveness. Such state of fix or indecisiveness over some issue can be called a *conflict*. Conflicts in case of SE projects cannot be resolved using the traditional conflict mitigation approaches. In literature [Parnell 2011], the conflicts during the SE projects has received attention recently with the increase in the number of challenged projects, which have been somehow to poor conflict resolution approaches deployed by the organizations. Even when all the decisions are carefully carried out, conflicts do occur and seek attention. Decision making processes are accountable for conflict resolution in a SE project.

Project development process often encounters conflicts, which can emerge during any of the project cycle phase: user/customer requirements elicitation, system requirement definition, architecture analysis, detailed design, implementation, testing, verification & validation, etc. During the user/customer requirements elicitation phase conflicts occur among stakeholders for prioritization of the needs, similarly

in system requirements definition phase, conflicts may occur during finalizing the system requirements metrics precision.

Choosing the appropriate architecture concept may pose a conflict among the stakeholders during architecture analysis and selecting the right sub-system components for implementing the final system may also lead to a conflict. It is very clear that, whenever, there is a choice to be made among the available solutions, with one or more DMs involved, there could be a potential for a conflict or an indecisiveness.

1.4.3.3 Preference Modeling and Negotiation

Preference modeling is the process of evaluating concepts with respect to customer needs and other criteria, comparing the relative strengths and weaknesses of the concepts, and selecting one or more for further investigation and development.

The decision-making techniques can be classified into two types: formal and informal techniques. The various formal existing approaches for modeling and evaluating the preferences are as follows [Triantaphyllou 2000, Greco 2004]: weighted sum model (WSM), weighted product model (WPM), analytic hierarchy process (AHP) and its improvements like analytic network process (ANP), the three ELECTRE methods and the TOPSIS method. Then there are other categories of decision-making tools, which allow imprecision in to the preference structure like Fuzzy sets and Fuzzy multi criteria decision-making. Utility theories [Fishburn 1970, Arrow 1963] and multi-attribute utility theory [Keeney 1993], are also used for decision-making. The other significant approaches are as follows: PROMETHEE -I & II, multi objective linear programming [Greco 2004]. The question, "*which decision making method, should be used to choose the best method for decision making?*" is yet to be answered by the scientific community.

The informal techniques are the group decision-making techniques based on voting involving cardinal or ordinal social welfare functions. The graph model for conflict resolution can be categorized into informal techniques for decision making. The graph model for decision making has been used earlier for disputes and conflicts [Fang 1993], for example application of the resolution of the environmental conflict of Garrison diversion unit (GDU) [Fang 1993], and Canadian-US softwood lumber dispute. The other works on graph model have tried to answer the stability analysis of states with uncertain preferences and also with the measure of strength of preferences. Issues like status quo analysis and policy stability of states have been discussed in previous works [Li 2004a, Hamouda 2004, Zeng 2004]. The graph model uses the preference structures defined in literature [Fishburn 1970, Öztürké 2005]. Roy *et al.* [Roy 1984] tried to define what is called *fundamental relational system of preference* and *combined relational system of preference*. Recent research has shown that, the traditional game theoretic approaches i.e., non-cooperative games are not very beneficial in system design activities [Mahajan 2004].

Similarly, the conflict resolution and negotiation in requirement engineering as previously introduced five generations of WinWin negotiation models [Boehm 1998, Boehm 2001, Jureta 2009, Kukreja 2012, Robinson 1990, Boehm 1995]. Machine

learning based approaches for software prioritization [Perini 2012] [Egyed 1997] analyzed the negotiation patterns, but still most of them lack to provide a *systems theory* of how the requirements are compared and given rank within the cognition of the stakeholders.

1.4.3.4 Research Questions for Decision Making

In literature, we have evaluated the state of art of decision-making techniques for various phases of development life-cycle. State of art of Decision-making is presented in Chapter 4. The major issues in decision-making are listed below:

Question 1.13. How to generate weights for a set of criteria for group decision making?

Question 1.14. How to remain transparent to all the stakeholders or decision makers in decision process?

Question 1.15. How to do group decision making and negotiate?

Question 1.16. How to resolve a conflict?

Question 1.17. How to provide a requisite level of transparency to the various decision makers?

Solution to the above mentioned issues are presented in the Chapter 4.

1.5 Conclusion

Previously, we have presented problem context, followed by problem focus and then research focus. Although, we have divided the problems found in the literature in three subcategories: requirements engineering & management and decision engineering. The identified problems are linked to each other in an holistic manner, small errors arising in the requirements engineering phase may jeopardize the whole project. It would be interesting to note that an incomplete solution to any problem may create another problems instead of solving it. Great efforts have to be poured into the requirements engineering and management, during their elicitation and prioritization, and equally great efforts have to be poured in for their traceability to develop a quality product. A product should satisfy its end-users and clients as much as possible. To make this happen, appropriate components and design elements should be included in the system according to their perceived utility by the end-users. This can only be achieved through efficient and transparent decision-making.

Chapter 2

Requirements Engineering

Contents

2.1 Introduction 29
2.2 What are actually requirements ? 30
2.2.1 State of Art Requirements Engineering Techniques 36
2.2.2 Proposed Formulations on Requirements 40
2.2.3 Proposed Comprehensive Requirements Modeling Language . 41
2.3 Writing Natural Language Requirements 47
2.3.1 State of Art of NLRs Writing Techniques 47
2.3.2 Proposed Approach for Writing Requirement 53
2.3.3 Experiment and Empirical Findings 55
2.3.4 Using Negation to Negotiate the Requirements 58
2.4 Discussion 58
2.5 Conclusion 59

2.1 Introduction

Requirements are backbone of a SE project. The early phase of requirements engineering deals with elicitation of stakeholder needs, expectations, requirements, system environment, operational capabilities, constraints, points of interactions with the system, etc., of the system under development and determine the measure of effectiveness and measure of performance. The needs and requirements should be traced to the various rationales of stakeholder with their preferences. Understanding stakeholder needs, requirements, architecture specifications, and design specifications for a complex product holds up-most importance in a SE project. An improperly carried out RE activity can cost dearly to the project and cause dents into the image of developing enterprise. There are issues with properly eliciting requirements and their representations. Writing requirements is a critical task for future system development. An ambiguous requirement can jeopardize, delay or may lead to faulty design solution to the problem, whereas a properly written set of requirements can swift the way for the rapid development of the right product demanded. In this Chapter, we explore a few of the problems of the RE phase that were raised in Section 1.4.1, we provide their corresponding state of art and propose some solutions to the identified problems.

A SE project involves multiple stakeholders, in their various forms of roles and actors, during and after the various SE project phases. These stakeholders and their various roles and responsibilities lead to various needs and requirements. *Requirements engineering* (RE) activities englobe activities like stakeholder identification, requirements elicitation, modeling, measurement, analysis, documentation, verification and validation, negotiation, improvement, prioritization, etc. All these activities require certain understanding of the system under development and hence their requirements itself. Previously, in Section 1.4.1, a few of the questions were raised relevant to the early phase RE.

Question 1.1: *what are the relationships between the various requirement artifacts, customer requirements and system requirements ? what is their significance to clients and other stakeholders ?*

Question 1.2: *how to improve the early phase requirements modeling with adequate level of scalability of graphical diagrams ?*

Question 1.3: *how to improve the modeling of the requirements such that their core and optional features of the system under development are clear to designer from the beginning of RE phase ?*

Question 1.4: *how to model the preference of the various stakeholders and weight the various goals and objectives of the system under the study during the requirement modeling phase ?*

Question 1.5: *how to write requirements in a more uniguous (unambiguous) manner, which are also easy to understand ?*

Section 2.2 presents the theoretical aspects of the requirements and simplified interpretations of the various RE artifacts, their significance to the stakeholders and to the project itself. It also presents the requirements modeling and state of art of requirements modeling. In Subsection 2.2.2 proposed formulations on requirements are presented, which provided solution to the Question 1.1. Subsection 2.2.3 presents the proposed goal based requirement language called Comprehensive Requirements Modeling language (CReML), as a solution to Questions 1.2, 1.3 and 1.4. Section 2.3 presents the issues linked with writing natural language requirements. Section 2.3.2 presents the method for writing requirements unambiguously and a simple experiment and empirical study and thus solution to Question 1.5. Section 2.4 discusses the issues linked with the proposed solution their potential benefits and challenges. The extent of their pertinence of the proposed solutions are discussed and other open questions left. Section 2.5 draws the conclusions.

2.2 What are actually requirements ?

It is essential to understand requirements first, to be able to work upon them and realize a successful product. As the literature of RE is vast, there are already numerous definitions of requirements and requirement artifacts. The various popularly used requirements artifacts terminologies are: rationale, needs/stakeholder requirements, views, viewpoints, desires, beliefs, constraints, etc. Different terminologies

used in RE are selected and presented in Table 2.1. The definitions of the requirements and specifications as mentioned in different standards and norms are briefed in Table 2.2. Given that there are multiple definitions of requirements in literature, it still remains today a tedious task to extract requirements properly from the user-stories owing to the confusion that term 'requirements' semantically mean to a particular requirement engineer.

Table 2.1: Terminologies Used in Requirements Engineering

Terminologies	**Definitions**
Rationale	Rationale can be defined as the fundamental justifications or reasons that lead to needs, desires, wants, constraints or requirements. The rationale of a requirement is information which summarizes why that requirement has been specified [Sommerville 1997].
Needs/Stakeholder requirements	Stakeholders' statements that describe the capabilities, expectations they want the system to fulfill [ISO 2008]. Needs are result of formulation by stakeholder analyst.
View	A view is a representation of whole system from the perspective of a single view-points. An excerpt from an artifact, containing only those parts one is currently interested in [Glinz 2011].
View-Point	A view-point represents an encapsulation of partial information about a system's requirements from a particular perspective [Sommerville 1997].
Desires/Wants	Desires represent the motivational state of an stakeholder. The state they want to achieve [Rao 1995]. They form the basis of needs. They are the stakeholder statements as a results of his beliefs which represent his perception of problem and often problem solution (what might solve the problem).
Belief	A belief represents the perception that a stakeholder holds about world, a belief may not necessarily be true [Rao 1995].
Intentions	Word 'intention' is often used in RE, it is used as synonym to desires [Rao 1995].
Constraints	A requirement that limits the solution space beyond what is necessary for meeting the given functional requirements and quality requirements [Glinz 2011].
Assumption	An assumption is an indicative property intended to be relevant to the system development project [Zave 1997].
Stakeholder	An individual, group of people, organization or other entity that has a direct or indirect interest (or stake) in a system [Hull 2010].

Continued on next page

Table 2.1 – continued from previous page

Terminologies	**Definitions**
Actor/Agent	A person, a system or a technical device in the context of a system that interacts with the system [Glinz 2011]. A Person or a technical device that act and process information in order to achieve some goals [Van Lamsweerde 2001].
Role	A role is an abstract characterization of the behavior of a social actor within some specialized context or domain of endeavor [Yu 1995].
Scenario	A description of a potential sequence of events that lead to a desired (or unwanted) result [Glinz 2011].
Goal	A goal is a condition or desired set of affairs that a stakeholder wants to achieve [Yu 1995, Bolchini 2003].
Objectives	Objectives are specific, measurable steps that can be taken to meet the goals.

In order to understand requirements, it is essential to know from where they do come. Requirements do come from the stakeholders, some times from standards, norms or laws. Stakeholders often present their requirements based upon the beliefs they hold about their environment. These beliefs may be right or wrong. In Table 2.1 desires, intentions and beliefs are taken in account from the literature and their subtle difference are presented. In Table 2.1, *needs* can be differentiated by other in a manner that they are the feasible or achievable set of statements as they are often the result of formulation by a requirement analyst. Other terms, such as wants, desires, intentions they might represent infeasible solutions, or useless description, which does not actually solves their problem. Sometime they express more than what they actually need to solve their problem. Requirements for stakeholder come from their desires, beliefs, intentions and rationales. Constraints are externally imposed limitation on system requirements, design, or implementation or on the process used to develop or modify a system [IEEE 2011]. It is imposed on the solution by force or compulsion and may limit or modify the design changes or solution space.

Requirements can be classified in numerous ways, but the one popularly accepted classification of requirements classify them into functional and non-functional requirements [IEEE 2011, Sommerville 2010, Hull 2010].

Table 2.2: Requirement Definition by SE Standards

Standards	**Stakeholder requirements**	**System requirements**	**Specification/Specified requirements**
EIA-632 [EIA632 2005]	A requirement that represents what stakeholders of a system need or expect of the system products.	A requirement derived from one or more stakeholder requirements and stated in technical terms.	A document that contains specified requirements for a product and the means to be used to determine that the product satisfies these requirements.
ISO15288 [ISO 2008]	Stakeholder requirements describe the needs, wants, desires, expectations and perceived constraints of identified stakeholders.	System requirements specify, from the developer's perspective, what characteristics it is to possess and with what magnitude in order to satisfy stakeholder requirements.	Specifications are the basis of the system solution and an origin for system element acquisition agreements, including acceptance criteria. They may be in the form of sketches, drawings or other descriptions appropriate to the maturity of the development effort, e.g. feasibility design, conceptual design, prefabrication design.
IEEE1220 [IEEE 2005]	What the stakeholder wants the system to accomplish. How well each function is to be accomplished. The natural and induced environments in which the product of the system operates or may be used.Constraints.	A statement that identifies a product or process operational, functional, or design characteristic or constraint, which is unambiguous, testable or measurable, and necessary for product or process acceptability.	A document that fully describes a design element or its interfaces in terms of requirements (functional, performance, constraints, and design characteristics) and the qualification conditions and procedures for each requirement.

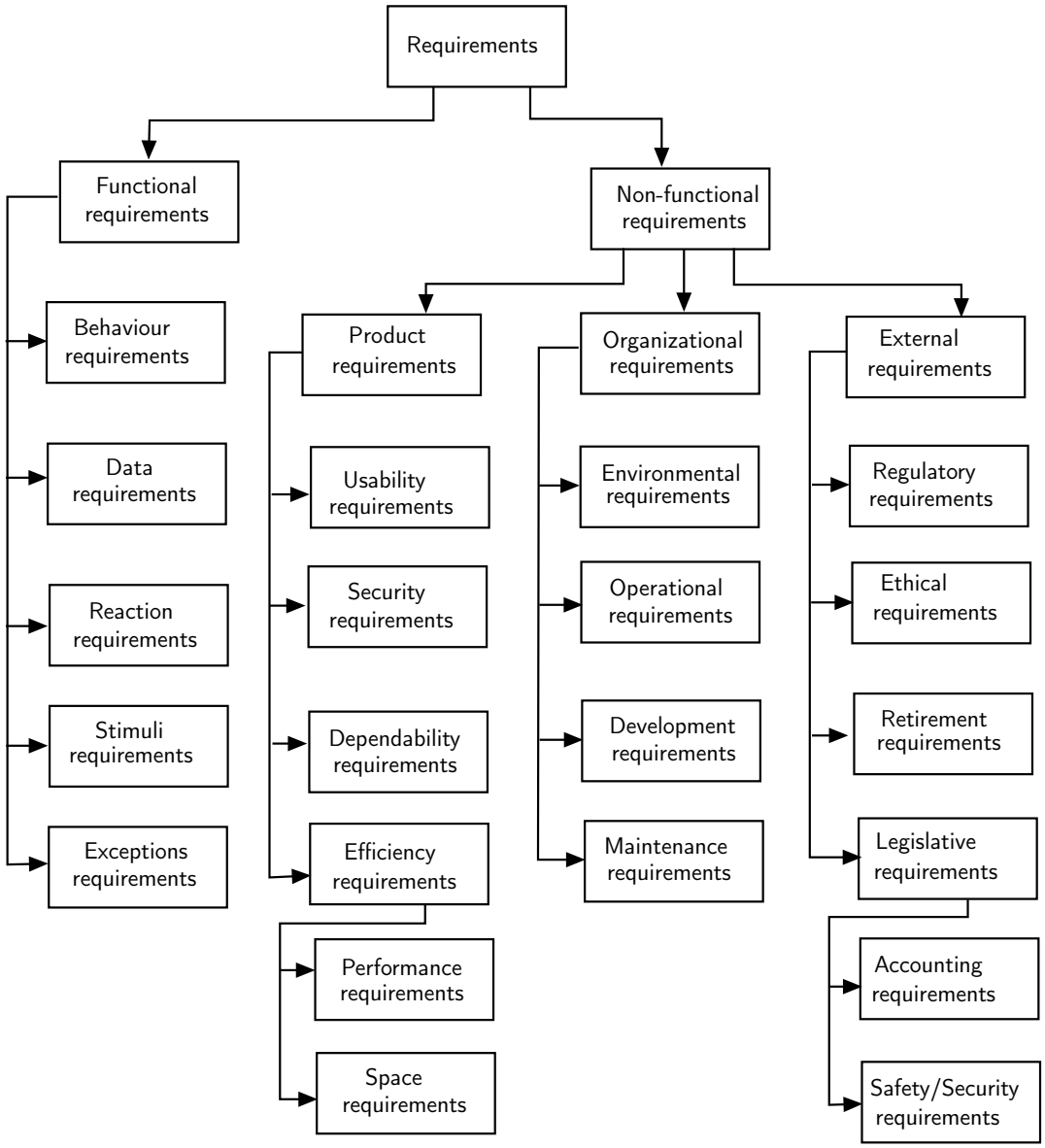


Figure 2.1: Taxonomy of Requirements

Functional requirements can be further classified as: behavior, data, reaction, stimuli and exceptions requirements (comes from software engineering background). Non-functional requirements can be classified into three types: product requirement, organizational and external requirements. Product requirements can be divided into usability requirements, security requirements, dependability requirements and efficiency requirements. Efficiency requirements can be divided again into performance requirements and space requirements. Organizational requirements can be divided into environmental requirements, operational requirements, development and maintenance requirements. Finally, external requirements can be divided into regulatory requirements, ethical requirements, retirement requirements, and legislative require-

ments. Legislative requirements can be divided into accounting requirements and safety/security requirements. Figure 2.1, shows the taxonomy of requirements as summarized from the literature. Functional requirements for a system describe what the system should do. Specific functional requirements define the input-output to the system the, reaction of the system upon the given input, system services and functions and exceptions (which come from software based systems). Non-functional requirements are all other requirements which are not concerned with any of the services or functionalities that the system should provide. All the quality requirements, specific constraints, standards compliance requirements fall under the non-functional requirements.

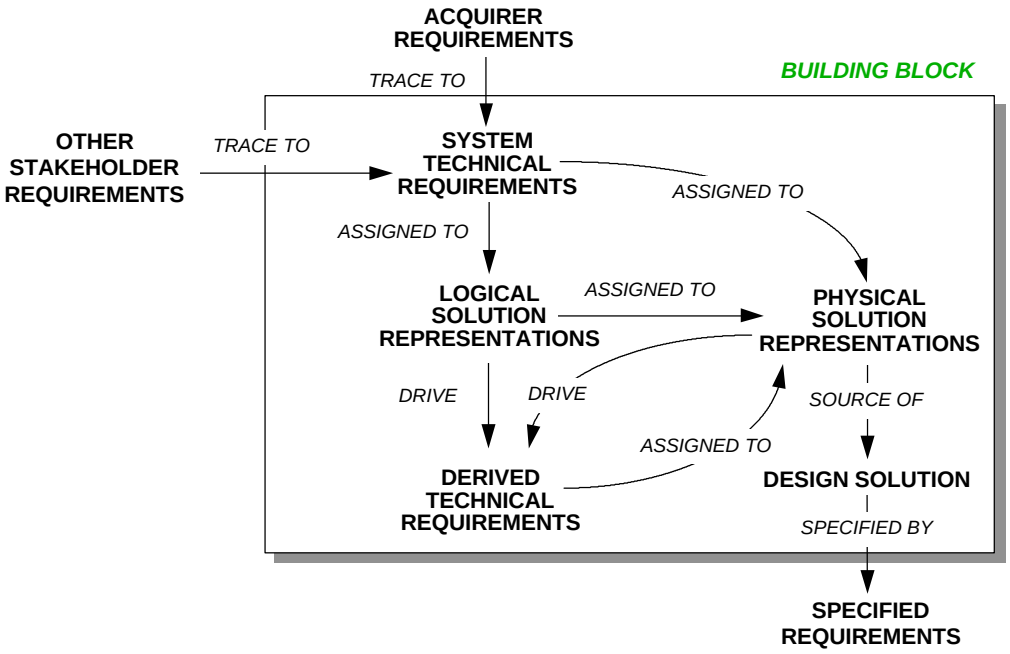


Figure 2.2: Requirements Relationships [EIA632 2005]

In literature, [EIA632 2005] mention specifically the existing relationships between the various types of requirements as shown in Figure 2.2. Acquirer requirements and other stakeholder requirements have relationship *trace to* with the system technical requirements. System technical requirements are *assigned to* logical solution representation and physical solution representations. Logical solutions are also *assigned to* physical solutions. Logical and physical solution representations *drive* the technical requirements. The derived technical requirements are in turn *assigned to* physical solutions representations. The physical solutions are *source of* design solutions and *specified by* specified requirements. The relationships presented by [EIA632 2005] can provide reasonable vocabulary for modeling traceability relationships, once the stakeholder requirements are known. But certainly lacks some other semantical relations which actually exists between the artifacts.

Other complementing relationships (other than one's defined in [EIA632 2005]) can be found in literature of requirements traceability. These relationships emerged from different types of development methods and techniques. Relationships are just a noun to quantify the semantics of particular dependency. Some other relationships are introduced in Section 2.2.3 to further provide specific semantical meaning to relationships existing between the artifacts. Stakeholders statements are stored as user-stories, source of all requirements from which all requirements are *derived*. More details about the semantics of relationships is provided in Section 3.4 of next chapter.

2.2.1 State of Art Requirements Engineering Techniques

Poor RE is the reason for majority of all the challenged and failed SE projects. Many empirical studies previously carried out have indicated that poor RE leads to poor requirements, faulty design, poor requirement traceability, rework and cost/budget overflows [Van Lamsweerde 2000, Eveleens 2010, Hull 2011, Tonnellier 2012]. Requirements modeling is carried out during early phase of the RE. There are a few of approaches like: Goal-Oriented RE (GORE) [Van Lamsweerde 2000], Scenario-Based RE (SBRE) [Carroll 1995], Aspect-Oriented RE (AORE) [Grundy 1999], and Service-Oriented RE (SORE) [Lichtenstein 2007]. The two most popular and referenced modeling methodology are GORE and SBRE, owing to the benefits and ease it provides during the requirement modeling phases. Often, GORE and SBRE are used together in industries for empowering better RE, often one being the input to other and vice-versa. It has been argued and shown that the GORE and SBRE complement each other during the requirement modeling phase of RE [Misra 2005]. However in literature, GORE and SBRE have been criticized for their inadequacy to handle non-functional requirements in use cases or viewpoint boundaries [Grundy 1999]. Some other issues like their inaptness to represent the business rules across the requirement documentation [Rashid 2008]. AORE distinguishes itself by explicitly separating the composition specifications. SORE has its roots in service oriented computing and service oriented software engineering, with their main goal of improving the development and delivery process for complex inter-organizational software applications. SORE emerged as a methodology for developing software based systems with special stress on service performance metrics, exhaustive service specifications, socio-technical issues while dealing with customers and service providers [Flores 2010].

In our work, the two referenced and used methodologies are GORE and SBRE owing to their popularity in the practice as empirical studies have shown. There are tools built upon popularly known GORE methodologies: i* and KAOS, and have reported lots of success in industrial application because of ease of understanding it offers to both technical and non-technical stakeholders. But still there is some scope of improvement and certain difficulties and problems to be improved as discussed later in Section 2.2.1.1. RE research has focused on goals as a way of providing the rationales (why) for a system under development [Van Lamsweerde 2001].

Literature of GORE based methodologies is vast if we take in account all the RMLs mentioned. There are a few notable GORE frameworks such as i*, Tropos, KAOS, GBRAM, NFR, etc., but still there are various aspects to be improved upon as mentioned in Section 2.2.1.1. As previously mentioned, our approach refers to i* and KAOS owing to the proximity of our approach. i* and KAOS frameworks are apparently the two most popular GORE methodologies. The seminal work of Erik Yu [Yu 1997] introduced the i* framework. It is hard to provide a fair comparison between them, as both of them have certain benefits and disadvantages when compared to each other [Werneck 2009]. Underlying principles of GORE were reinforced by Regev *et. al.* [Regev 2005] by bringing together the various concepts from various methodologies. Recent works involving goals, preferences, and inconsistency have led to development of an abstract requirement modeling language called Techne [Jureta 2010]. Recent work have tried to address the optionality and preference of the requirements during the modeling [Liaskos 2010]. The preference of the goals are marked on the the goal notation thus allowing to evaluate the importance of one goal with respect to another. Goal argumentation methods (GAM) were introduced to make it explicit the reasons of selecting a goal [Jureta 2006].

Tropos [Castro 2002] framework was founded on social and intentional aspects of information system, used requirement modeling based on their operational environment. Tropos introduced textual syntax to allow the later phase formal analysis of the early requirements models done using i* to represent their social milieu. The major advantage given by i* based frameworks is they allow to represent the strategic relationships existing between the various stakeholders of the project. But many empirical studies [Easterbrook 2005, Werneck 2009] have shown that the readability of the models designed using i* are greatly marred when the number of participants is sufficiently large. This problem comes due to layout used by the i* models, on the contrary it can be argued that the tree based layout of KAOS models are much better in this aspect of the goal modeling. One of the disadvantage of the KAOS models is its deficiency in modeling the strategic relationships of the stakeholders. Previously mentioned techniques for integrating preference in goal models do not help user in any readability aspect [Liaskos 2010, Jureta 2010]. In can be argued that surcharge of information increases the pressure on the designer or stakeholders. The way the data is represented and made available to the design engineer can be rendered more readable.

Currently, there are a variety of tools available for the GORE modeling such as Objectiver based on KAOS [Objetiver 2013], other recently introduced RE-TOOLS based upon recently introduced light weight RMLs like VLML [Glinz 2010]. Still there are numerous issues to be solved by a RML and lots of lessons learned during all there years of RE needs to be brought together to a standard GORE language. A standardized GORE notation based on KAOS seems to be most appropriate for this unifications of lessons learned and hence we propose in this work few modifications into the KAOS modeling notations to the benefit of RE community.

Table 2.3: Comparing popular Gore RMLs

Approaches	Keypoints	Advantages	Disadvantages
i* [Yu 1997]	It focuses on strategic dependency and intentionality of the various stakeholders. Internal rationales of the stakeholders are taken into account. Agent oriented framework, two components: Strategic dependency model and Strategic rationale model. Reasons for the need of the system under study. The goals are refined till they are satisfied by a task and resource.	Allows to represent the strategic relationships between the stakeholders explicitly. It allows to model critical situations by using soft-goals and their contributions associated with Strategic rationale model	Preference can not be taken into account, scalability issues with goal models, difficulty to represent domain assumptions, constraints, and core-optional features. Non functional requirements are treated as softgoal, but no means to arrive at them.
Tropos [Castro 2002]	Based on i* syntaxtual framework, allows to enrich i* graph with formal specification language: Formal tropos	Model checking can be carried out, other benefits similar to i*	Similar to i*
Knowledge Acquisition in autOmated Specification (KAOS) [Van Lamsweerde 2001]	The goal model starts with the decomposition of goals into sub-goals until arriving at requirements, software agents take care of requirements and requirements of human agent as interfaces. Its a top-down model, beginning with most strategic goal to subgoals. It is composed of four types of models: goal model, object model, operational model, responsibility model. Provides mixture of semi-formal, formal, and qualitative reasoning.	Tree based layout models allow better comprehension. Ease of understanding to technical and non-technical stakeholders, obstacle modeling can be carried out for risk analysis, it can model roles as responsibilities, NFRs are treated as obstacle or constraint and thus allows more precision for implementation.	Four types of models are used which could be confusing. Preference can not be taken into account, scalability issues, constraints, and core-optional features. Strategic dependency between stakeholders cannot be taken into account explicitly.

Table 2.3 provides a comparison of popular RMLs and their benefits and limitation.

2.2.1.1 Early Phase Requirements Modeling Problems

Early phase requirements engineering starts, once the requirement elicitation process starts following to the interviews, questionnaires, ethnography and other elicitation techniques mentioned in literature. Through all the elicitation techniques the information gathered is converted to textual documents, often known as user-stories. These user-stories form the foundation of the requirements modeling (RM) processes. We have identified a few of the problems which seek attention and proper resolution during RM.

Ease of Scalability: recent empirical studies using i* GORE methodology have shown that scalability can turn out to be big problem when modeling requirements for a sufficiently large projects either with different viewpoints or integrated modeling [Easterbrook 2005].

Traceability of goals: often, during RM goals are elicited through stakeholders and as the project evolves, the complexity of the models may increase to an extent where it becomes tedious task to answer why a particular goal exists in the model and which particular stakeholder needs it. Also, it can be equally cumbersome to link a goal to a particular user story previously elicited, owing to syntactic differences [Easterbrook 2005]. There is need of dedicated mechanism to link goals to the user-stories previously documented during elicitation.

Preference of multiple stakeholders over goals: in a systems engineering project, it is of great importance that most of stakeholders are satisfied with the various decisions taken during the product development and with the final resulting end product. A higher satisfaction among the stakeholders can be guaranteed if the various stakeholders preferences are taken into account in a transparent and holistic manner during the goal modeling.

Multiple view-point requirement modeling: during the requirement modeling multi-view point modeling is often instrumental in understanding the system under study. Multi-view requirements modeling allows separation of concerns and can help to elaborate particular aspects of the system under study. But it becomes tedious task to combine these multiple views and present one single coherent and comprehensive models. Often, the resultant combined model is inconsistent and hard to understand, which demands significant amount of resources.

Modeling of core and optional features: modeling of core features and optional features from the very beginning of project can allow the engineers to have better understanding of the systems under study and lots of effort and resources can be saved if they can be modeled in early phase of RM. There are some dedicated feature modeling languages in the literature but this often leads to redundant efforts.

Representation of domain assumptions: domain assumptions or beliefs are often implicit during the requirement modeling but often lead to goals and requirements. They are usually held by the stakeholders and sometimes designers.

Their implicit nature during the RM may cause potential traceability errors and may cause worries for the quality control of the product.

2.2.2 Proposed Formulations on Requirements

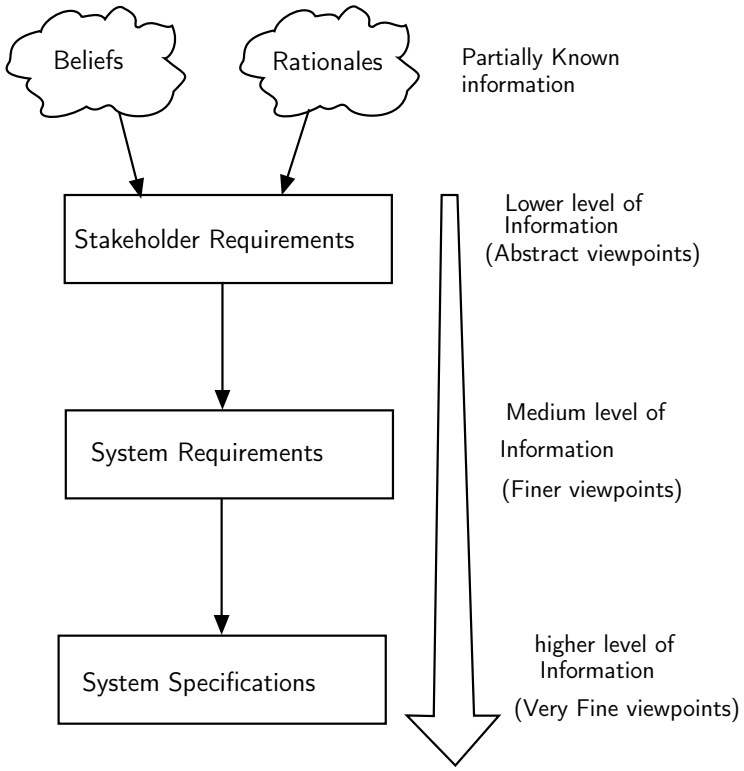


Figure 2.3: Relating Requirements, Rationales and Viewpoints

Previously, Question 1.1,*what are the relationships between the various requirement artifacts, customer requirements and system requirements ? what is their significance to clients and other stakeholders ?* was raised in Chapter 1. In order to respond to this question a theory of requirements is developed, which tries to analyze requirements from a different angle. It states that requirements are projection of stakeholders rationales in product with his beliefs. In other words, together with his set of beliefs and his rationales a stakeholder arrives at a need or a stakeholder requirement. A rationale in turn is composed of multiple viewpoints which are of interest to stakeholder, but this fact is often hidden from himself. Figure 2.3 shows the proposed formulation. It shows that as we move from stakeholder requirements toward the system specifications, the degree of clarity in information increases gradually from very abstract viewpoints to very specific fine viewpoints.

Stakeholder Requirements as Projection of Rationales

Stakeholder requirements can be thought of an abstraction of stakeholder rationales, which may be known or unknown to him. An aware stakeholder knows these rationales and can easily identify his requirements more precisely. This is the reason why lots of requirements in an organization come from a small set of participant often known as 20-80 rule of Pareto. It is observed that rationales are govern by the law's of nature: laws of physics, laws of economics, laws of chemistry, etc., and hence relationship between them are logical. They can either be directly proportional, inversely proportional and a few times independent. There is still something which is more important to dig out—the beliefs. Beliefs are hard to elicit from the stakeholders, often they remain implicit in the user stories and costly iterations are sometimes needed to really reach them. This is often the case in software based systems engineering.

The relationship between belief, rationale and stakeholder requirements can be assumed in mathematical equation as below:

$$f_B(b_1, b_2, \cdots, b_n) \textcircled{?} f_R(r_1, r_2, \cdots, r_m) \Longrightarrow f_{Sre}(re_1, re_2, \cdots re_l) \tag{2.1}$$

In Eq.(2.1) f_B is a set representing the stakeholder beliefs and f_R is the set representing the rationales of stakeholders. Unknown operation $\textcircled{?}$ leads to f_{Cre}, representing the resultant set of stakeholder requirements. Eq.(2.1), shows the complex process of the how the stakeholder beliefs and rationales together play a role in generating an array of customer requirements. f_B, f_R are often hidden and the only known thing is f_{Cre}, which comes as a result of elicitation process.

Rationales as projection of Viewpoints

As the requirements are projection of rationales together with the beliefs, similarly rationales are projections of viewpoints. Rationales can be decomposed into a set of viewpoints, together with a importance vector, which provides the strength of their preference held by a particular stakeholder. This vector of importance is implicit to the stakeholder and difficult enough to elicit completely.

System Requirements and System Specifications

System requirements and design specifications can also be expressed in terms of viewpoints and rationales. System requirements can be thought of composed of coarser granularity of viewpoints. System specifications are the very fine granularity level projections of viewpoints. The whole process of requirements engineering can be thought of discovering of these viewpoints, rationales and the weight vector held by the various stakeholders.

2.2.3 Proposed Comprehensive Requirements Modeling Language

Proposed approach is implemented as Comprehensive Requirement Modeling Language (CReML), which is based on Goal-Oriented Requirements Engineering

(GORE) methods. CReML can be used complementarily with the Unified Modeling Language (UML) or System Modeling Language(SysML) and helps to provide better design specifications and insights to the system under study. It provides way to model the requirements representing the core and optional features of the system from early phases of RE and the stakeholders' preferences associated to them. It equally provides ways to visual modeling, which are more scalable and comprehensible to the engineers and the various stakeholders using layers of abstraction, visibility and invisibility. Classical GORE method is augmented using view-points as a tool to take in account the non-functional requirements which can be carried out from the very beginning and be carried out separately and integrated later. The non-functional requirements are made contractual by associating test cases which represent their functional implementations (as all non-functional requirements are transformed into appropriate functionality [Pinheiro 2004]). Figure 2.4, shows the goal meta-model of the proposed CReML, Figure 2.5 shows the responsibility meta-model. Figure 2.6, shows the CReML strategy meta-model. Table 2.2 explains the syntax and semantics of CReML.

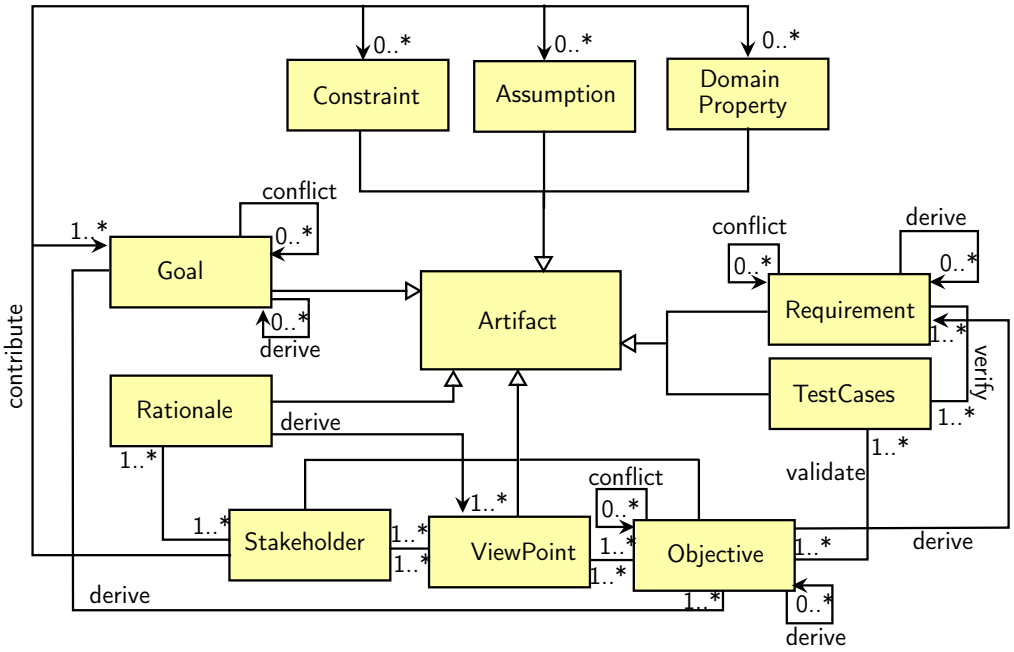


Figure 2.4: CReML Goal Meta-model

A goal model is used to understand the problems and its formulation as goals, and objectives, which lead to formulation of systems requirements. The strategic model allows to represent the strategic relationships existing between the stakeholders, i.e., their dependency upon each other for their success or failure. Strategic model allows to understand the strategic relationships existing in the environment of its customers and how together they can succeed or fail in carrying out their missions. The responsibility model comes later when the modeling starts. The responsibility model

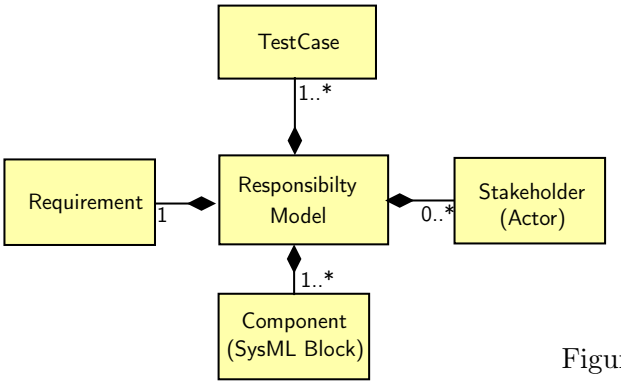


Figure 2.5: CReML Responsibility Meta-model

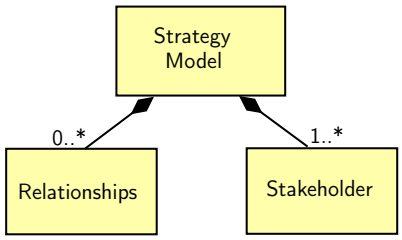


Figure 2.6: CReML Strategy Meta-model

allows to understand the allocation of requirements to the design components, their associated test cases and corresponding actors (if necessary). *CReML* necessitates *SysML/UML* block diagram to represent the components, other *SysML* diagram can also be used together with *CReML* to understand the system behavior (Out of scope of this thesis).

Table 2.4: Requirement Artifacts definition in CReML

Graphical Diagrams	**Semantic meaning**
 Goal	*Goals* represents the fundamental state, that the stakeholder would like to achieve by using the system under study. A system can have one primary goal and many other secondary goals. Goals may not be strictly measurable or tangible but stakeholders agree upon certain conditions for determining the acceptability of goal by system under study.
 Rationale	*Rationales* are the fundamental reasons, why the goals needs to be achieved by the system under study. Rationales are extracted from stakeholders. A single stakeholder may have multiple rationales. These rationales are actually linked to various responsibilities and roles played by a given stakeholder.
 Viewpoint	*Viewpoint* is a certain perspective of the system which is of interest to a particular stakeholder. Viewpoints allow to visualize system from a particular aspect while hiding unnecessary details.
	Continued on the next page

Table 2.4 – Requirement Artifacts definition in CReML continued

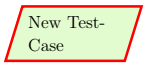
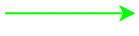
Graphical Diagrams	**Semantic meaning**
 Objective	*Objectives* are the measurable set of tasks and conditions, which the system needs to meet in order to accomplish tasks. Goals projected with a particular viewpoint lead to a objective in a direction of particular viewpoint to achieve the goal.
 Constraint	*Constraints* are the limitations imposed on the system by the non-development stakeholders or they may represent some challenges to overcome by the system.
 Domain-Property	A domain-property can defined as a knowledge or information about the domain of the system under study which is uniformly acceptable by all stakeholders: technical or non-technical and which can be verified and validated using scientific methods.
 Assumption	Assumptions are hypothesis or non-verifiable information or conditions which are considered valid for the system under study (but they can be false). They are close to domain-property but domain-properties can be verified and be easily validated and hence are always correct.
 Requirement	*Requirements* specify, from the stakeholders' viewpoint, what characteristics it is to possess and with what magnitude in order to achieve the stakeholders' objectives. Requirements are derived from a particular or a set of objectives, constraints, assumption and domain properties.
 Test Case	*Test Case* provide the means to Verify and Validate a requirement partially or fully, they define the environment of the test, input conditions, output conditions, pass conditions and failure condition.
Component	*Component* is the physical representation of the design concepts as a result of system modeling process. It is responsible to show the end product of the previously developed requirements. They can be the structural or behavioral UML/SysML diagrams.
Conflict	*Conflict* relationship is used to represent the conflict occurring between two objectives, or two requirements and hence between two source stakeholders. Conflict means that the implementation of the two requirement artifact cannot be achieved by the system under study at the same time.
	Continued on the next page

Table 2.4 – Requirement Artifacts definition in CReML continued

Graphical Diagrams	**Semantic meaning**
 Derive	*Derive* relationships represent the parent-child relationships existing between the various requirements artifacts. A derive relationship exists between the goal and rationales, rationales and viewpoint, viewpoint and objective, objective and requirements, constraints and requirements, domain-property and requirements, assumption and requirements, and between requirements and requirements.
Contribute	*Contribute* relationship is used to represent the direct contribution of information about requirement artifacts from an stakeholder for system under study, such as constraint, domain property, assumption or goal.
 Stakeholder	*Stakeholders* of the project are the entities which have genuine interest in the project. They are of two types: stakeholders from the client side or end-users and stakeholders responsible for the development of the project. In our terminology, we use Agents also as an stakeholder, as they interact with the system.

Proposed approach aims to use and devise techniques previously used and matured in domain of software engineering for the benefit of systems engineering community. Many advances in the RE community come from software industries. These advances provide new opportunities to systems engineer to make their process more lean and mean. This approach is designed for early phase requirements engineering. It is not here to replace use-cases or requirements block used in UML/SysML, it is their precursor and complementary technique to them in RE activities.

This approach identifies nine types of early phase RE artifact: *Stakeholders, Goals, Rationale, Viewpoint, Objectives, Constraints, Domain Property, Assumptions, Requirements and Test cases*, three types of early phase relationships : *Contribute, Derive, Conflict* and other contributing artifacts *Components*. Requirement artifact diagrams and their semantic meanings is shown in Table 2.4. They are used to address the problems previously raised in Subsection 2.2.1.1. There are three types of models introduced in CReML: *Goal models*, *Responsibility model* and *Strategy model*. User stories, interview statements and statements are necessary to start using the *CReML*, this step of requirement elicitation activities is out of scope if this thesis. *CReML* models are result of the analysis of these user-stories and other statements by the requirements engineer. *Goal model* and *Strategy models* are developed simultaneously. Strategy models can be started once the goal modeling begins. *Strategy model* and *goal model* provide inputs to each other over different iterations to better understand the system goals and environment. *Responsibility model* is development can be carried out once the *goal-model* and the *strategy model* are available.

In **first stage** Goal models and strategy models are developed simultaneously. For *Goal modeling* the stakeholders are identified and their goals are extracted out of their corresponding available user stories. Since the stakeholders contribute the goals the relationship between the stakeholders and goal is called *contribute*. These stakeholders are then analyzed and their responsibilities are analyzed taken in account both in presence and absence of the system under study. This analysis leads to various rationales of the corresponding stakeholders role. These rationales of the stakeholders provide the basis of strategy modeling. Rationales provide the inputs regarding how the different stakeholders can be satisfied. Strategy models provide the statements (in form of rationales), which bind together the stakeholders and the statements for their potential conflict. At this stage stakeholders' preference about the various goals are gathered and core and optional goals are identified. This preference gathered over the goals from different stakeholders provides the inputs for the strategy formulations for system development. Stakeholders' preference about traceability of various rationales are also gathered, they are later used to formulate the traceability policies according to the needs of the stakeholders.

In the **second stage**, the various viewpoints are which are of concern to each stakeholders are gathered; a viewpoint is a particular aspect of the system under study which is of interest to stakeholder: client or developer. The analysis of viewpoints lead to the formulation of objectives corresponding to a particular stakeholder. This allows us to understand very clearly the capabilities stakeholder wants to acquire with respect to each viewpoint. At this very stage, potential conflicts among the objectives can be observed, a conflict relationship is marked for further resolution and negotiation for the magnitude of accomplishment of particular objectives.

In the **third stage**, the *responsibility model* is created by separately mapping the stakeholders, viewpoint, objectives, constraints and assumptions. Mapping of objectives with the actors (roles of stakeholders) allows to model the responsibility and point of interactions between the agents (roles of stakeholders) and system. This mapping allows to determine the constraints and assumptions held by the stakeholders and their interrelationship.

In the **final stage**, the *Goal model* is enriched with the assumptions, and constraints previously extracted from stakeholders. Developers held domain properties are included at this stage to transform them together with objectives into achievable requirements. Each objective may lead to one or more requirement.

Responsibility modeling can be carried out once the Goal modeling starts. Responsibility model is basically the network of requirements, their test cases, their logical and physical design component representations and necessary actors. Responsibility modeling provides higher level traceability of the requirements artifact which can be used later different purposes. Goal models are responsible for providing pre-requirement traceability, whereas responsibility model is accountable for providing the post-requirement traceability.

2.3 Writing Natural Language Requirements

Writing good requirements is one of the most important task. Writing requirements is a technical task so it is different from other types of writing. Previously, we mentioned and framed **Question 1.3:** *how to write requirements in a uniguous manner, which are also easy to understand?* In this Section, we are concerned with the Natural language for writing requirements, i.e. natural language requirements (NLR). Although, it is an imperfect way of expressing requirements, it remains the only universal means of expression that covers the huge variety of concepts needed [INCOSE 2012]. Previously, it had been well debated in literature whether natural language should be used for writing requirements or not [Kovitz 2002, Berry 2005], well it is hard to correctly answer this question but natural language remains the de facto industry standard for writing requirement documents, as it allows to communicate with great expressiveness between stakeholders and developers. Ben Kovitz [Kovitz 2002] has previously quoted that "*requirements communication, to be effective, must work human-to-human. Formal notations, with their total lack of ambiguity, are the end point of software development, not the start.*" What he pointed out was that the context is more important than the description itself and its only the natural language which fully enables you to capture and communicate the context and the description together in the two way direction of customer and developer. Whereas there are numerous issues yet to be solved as previously highlighted in Chapter 1, and this provides the motivation to work in this direction. Writing of requirements documents has multiple aspects to be carefully balanced such as readability and usability [INCOSE 2012, Hooks 1993, Hull 2011]. A requirement document which is hard to understand is as good as no requirement document. Usually, the model of requirements may depend on the policies and practices used by the organizations. A few general *requirements templates* or *boiler plates* have been previously mentioned in the literature like VOLERE [Robertson 2006].

2.3.1 State of Art of NLRs Writing Techniques

In software engineering literature, requirement ambiguity and requirement precision are inversely proportional to each other, the more a requirement is ambiguous the lesser is its precision. The more a document is precise, the better the quality of a requirement is. The earlier works have laid emphasis on resolving ambiguity of requirement sentences and lesser work has been produced regarding the precision of requirement. The earlier works of Ryan [Ryan 1993] have showed that the actual role of natural language processing for requirement engineering is limited, as there could be no machine intelligent enough to understand every context. The writing styles have evolved for writing unambiguously [Berry 2003, Berry 2005, Berry 2004, Sommerville 1997, Kovitz 1998, Dupré 1998]. Manual ambiguity detection techniques like inspection [Kamsties 2001, Kamsties 2000], together with various tools have been developed to write requirements in natural language [Ambriola 1997, Huyck 2000]. The various techniques other than natu-

ral language have been introduced like constrained natural language, semi-formal languages and formal languages for writing requirements, however the ambiguity and understandability problem remains intact. The earlier works have mentioned the reverse requirements with no test case associated for non-functional requirements [Berry 2003], just for avoiding ambiguity. Previously, none of work has reported the concept of understandability associated with negative requirements. Supakkul *et al.* [Supakkul 2010] proposed usage of requirement patterns for reuse in elicitation of non-functional requirements using knowledge base. Cysneiros *et al.* [Cysneiros 2004] advocated the usage of Language Extended Lexicon (LEL), which is based on vocabulary capture approach for writing non-functional requirements [do Prado Leite 1993].

Earlier works [Svahnberg 2008] have been produced linking requirements as artifacts consumed by the decision making process outside of the requirement engineering process.

In general, there are three types of approaches which have previously addressed the ambiguity problem of requirements: approaches that define linguistic rules and analytical keywords [Fabbrini 2001a, Fabbrini 2000, Wilson 1997], approaches that define guideline rules [Juristo 2000, Tjong 2008] and approaches that define specific language patterns to be used in writing the natural language requirements (NLRs) [Denger 2003, Ohnishi 1994, Rolland 1992].

In the first kind of approach [Wilson 1997], software requirements and specification documents' quality aspects are defined and such as completeness, correctness, traceability, uniguity, etc., and other indicators for lack of quality such as imperatives, continuances, directives, options and weak phrases. Fabbrini *et al.* [Fabbrini 2000] further refined the ambiguity aspects of requirements itself called requirements sentence quality (RSQ) and requirements document quality (RDQ) in global. RSQ related indicators include implicit subject, multiple sentence, optional sentences, underspecified sentences, vague sentences and weak sentences. RDQ related indicators include comment frequency, reliability index, under referenced sentences and unexplained sentences. Their finding about RSQ and RDQ were implemented in an automatic tool called QuARS (Quality Analyser of Requirements Specifications).

In the second type of approach, Juristo *et al.* [Juristo 2000], classified requirements into two types static and dynamic. They defined Static Language Utility (SLU) and Dynamic Language Utility (DLU) structures, which is specified by a formal grammar and a set of natural language structures transformable into predicate logic. This transformation allows to achieve a set of unambiguous requirements. Another guidelines based approach [Tjong 2008] provides set of of guideline rules and a inspection checklist for writing less ambiguous requirements the guideline rules and checklist were implemented in an experimental lexical analyser tool called SREE. Upon inspection of the NLRs, its notifies the user about the potential ambiguity in the document, leaving space for the user to act upon and disambiguate a truly ambiguous statement.

In the language patterns approach for writing NLRs, Ohnishi [Ohnishi 1994]

developed a software specification method with help of a visual requirement definition language (VRDL) and a text base language (X-JRDL). VRDL allowed to define shape and semantics of an any icon to specify requirements and X-JRDL language based on case grammar allowed to define case structures of any new verbs to specify requirements. These languages were used under a concept called requirement frame model. It allowed to distinguish three kinds of frame models: Noun frame, Case frame and Function frame. [Rolland 1992] *et al.* working with database development domain, defined patterns and cases based on the Fillmore's case system: Agentive, Instrumental, Dative, Factitive, Locative and Objective for writing requirements. They defined several classes of verbs and clauses and distinguish the main linguistic patterns: elementary patterns which allow associating clauses to syntactic units of clauses and sentence patterns that allow associating cases to clauses of a sentence. Denger *et al.* [Denger 2003] developed an approach for reducing the ambiguity in NLRs with the use of natural language patterns , authoring rules and document templates. They outlined distinct language patterns such as functional requirement sentence patterns, event patterns, reaction patterns, computational patterns, relationship patterns, exception patterns, patterns for special aspects and non-functional requirement sentence pattern. A few other work also exist, notably on Controlled English [Fuchs 1996], which is a subset of natural language with restricted syntax and semantics. Experiments with Controlled English were carried out on Attempto [Fuchs 1996], which translated the specifications written in Controlled English into discourse representation structures and logic programming language (Prolog). Their findings indicated that Controlled English could be used for specifications, but there is need for establishing the small number of easy to remember functions to formulate sentences in Controlled English.

Table 2.5, summarizes the key-points of the three approaches and their corresponding advantages and disadvantages. It is difficult to say with whom we can compare our actual work, as we have addressed the precision problem of non-functional requirements, which is a type of ambiguity problem. Our work is actually very different from the previously existing approaches, however it can be located into placed into a category with natural language patterns for NLRs with some exceptions. Earlier, it has been advised to use only affirmative phrases for requirement description, in contrast to our proposed work which advocates that it can be useful to use negative phrases to explore quality requirements and can be used as a measure to prevent ambiguity and understandability problem.

2.3.1.1 Issues Identified with Writing of NLRs

Ambiguity of NLR We confront ambiguity in everyday life in different forms: conversations, signs, pictures, newspapers, documents etc. Sometimes their implications are trivial and sometimes significant to us. In the context of system engineering the role of ambiguity is significant and is a determining factor for success or fail-

Table 2.5: Comparing Different Approaches for Ambiguity Problem in NLRs

Approaches	Keypoints	Advantages	Disadvantages
Linguistic rules and analytical keywords [Fabbrini 2001a], [Wilson 1997]	Model of language attributes and indicators used in evaluating the quality of existing NLRs. Frequently used keywords, phrases and sentence structure are grouped and counted by computer program	Effective in detecting defects and ambiguous NLRs	Depends a lot upon different contexts, human intervention is required
Guidelines rules [Tjong 2008], [Juristo 2000], [Berry 2003]	Guiding rules that can detect defects, ambiguities and weak phrases .	Avoids incorrect construction of NLRs, rules functions as a checklist.	Restricts freedom of writing NLRs
Language patterns [Denger 2003], [Rolland 1992], [Ohnishi 1994]	Linguistic patterns used as reference, set of authoring rules and document templates, limited vocabulary language.	Allows to provide transformation rules	Requires considerable efforts for devising patterns

ure of project [Auriol 2008]. Ambiguity infers poor or opaque understandability of documents. The usability of a requirements specification document depends on the quality with which the document is written; consciously or subconsciously the development teams expect document to be unambiguous and easy to understand. The understandability of a requirement is usually forgotten by requirement engineers as an attribute of document, but they expect it subconsciously [Svahnberg 2008].

Understandability of NLR In literature the definition of understandability of requirement documents is usually taken from English dictionaries: *able to be understood*. [Fabbrini 2001b] defines understandability with respect to requirement specification, as the capability of the Requirements Specification Document to be fully understood when read by the user. It can also termed to be readability or comprehensibility of a document [Kamsties 2003].

In our approach we categorize the understandability of a requirement document as good, if it enables its readers to operate upon them with ease, enables transparent decision making and provides a pathway for the test cases while remaining in conformance with the user.

2.3.1.2 Defining Ambiguity

In literature the definition and scope of ambiguity is considered different in different disciplines like philosophy, linguistics, law and, of course, systems engineering. Here

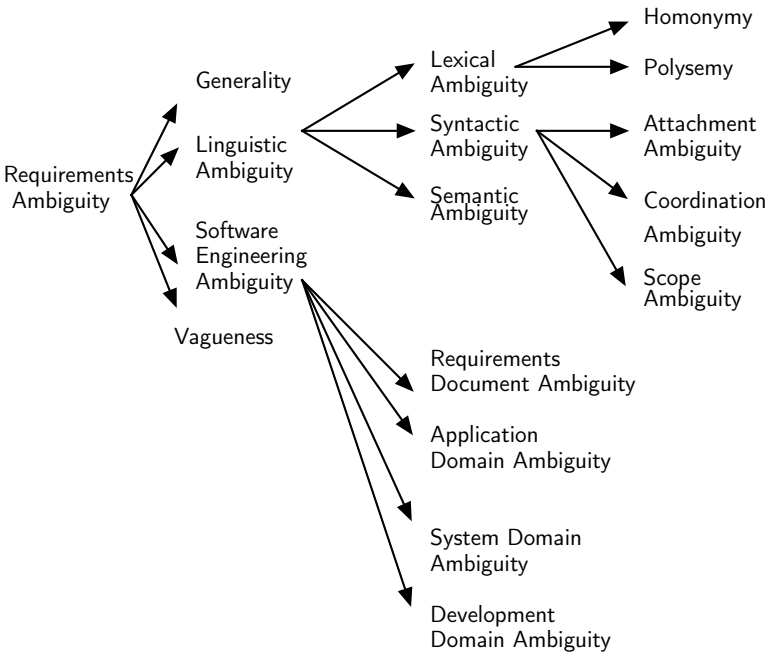


Figure 2.7: Taxonomy of Ambiguity Types [Berry 2003]

we have tried to highlight few of the existing definitions of ambiguity in the literature of software engineering.

The IEEE recommended practice for software Requirements Specification defines an unambiguous requirement: a SRS (Software Requirement Specification) is unambiguous, if and only if, every requirement stated therein has only one interpretation [IEEE 1998]. Implying that, a requirement is ambiguous if it is not unambiguous.

According to [Kovitz 2002], ambiguity is a relation between a description and reality in which the distinctions in the description fails to guide you when you meet the reality. [Huyck 2000] defines, a sentence is ambiguous if it has more than one complete interpretation. [Leffingwell 2003] defines ambiguity as: a requirement is ambiguous if there is a probability of being misunderstood. [Kamsties 2001, Kamsties 2000, Berry 2003] extend further the concept of ambiguity to context knowledge and they used three complementary definitions to curve out RE-ambiguity, the first definition defines a requirement as ambiguous if it has multiple interpretations despite the reader's knowledge of the RE (Requirement Engineering) context. The second definition states that the required RE context necessary to disambiguate a requirement consists of four parts; the requirement document, the application domain, the system domain, and the development domain. The third definition [Kamsties 2000, Berry 2003], implies that a requirement allows multiple interpretations if it contains linguistic or RE-specific ambiguity. Linguistic ambiguity arises independently from any context and comprises vagueness and genuine ambiguity in the form of lexical, requirements-document, application-domain,

system-domain and development-domain ambiguity.

Figure 2.7, shows the taxonomy of ambiguity that can be encountered in a natural language requirement specification [Leite 2004]. It is clear from Figure 2.7 that the requirement ambiguity can be traced to system/software-engineering ambiguity, lexical, generality, vagueness or all of them [Leite 2004]. [Leite 2004] describes all of these types of ambiguities with examples.

2.3.1.3 Understandability and Ambiguity Problem in RE

In the context of RE, the poor understandability of requirements document implies wastage of resources [Sommerville 1997]: it will demands more efforts from the reader and hence will cost time and money.

In RE context, an ambiguous or misunderstood requirement may lead to faulty development of product. The product deviates from the actual needs of the stakeholders and may upset both the client and the developer. Therefore early detection and resolution of ambiguity in requirements documents is recommended. As natural language allows many structural possibilities of sentences, one sentence that seems ambiguous to one, may be interpreted uniguously by another. The faulty interpretation may arise because of cultural differences, linguistic differences, methodological differences, or as trivial as geographical or climatic differences.

A perfectly unambiguous requirement would be when everybody has the single interpretation of it. Currently, the projects are getting more and more complex leading to highly complex requirements, and this goal of unambiguous requirement seems very difficult to achieve with natural languages and current writing styles.

With formal languages there is a big communication barrier to overcome when dealing directly with clients. A poor knowledge of the particular formal language by clients can not help in establishing correct set of requirements. Also it is not possible for all the stakeholders to express themselves in a particular formal language [Altenhofen 2010].

The main problem of semi-formal notations is their lack of precise semantics for modeling notations, which may lead to incorrect interpretations and misunderstandings for corresponding models. Another problem linked to semi-formal or constrained natural language is that, there is difficulty with expressiveness and convenience of usage. Similar to usage of formal languages, it is not possible for all the stakeholders to use the new set of grammatical rules to write sentences, or to learn a new set of notations to express fine details, in the case they do so, there is always a fear of something being left out.

In a complex system, at times it becomes very difficult to transform a system needs into a precise verifiable requirement statement which can be easily contracted with the client. Usually, it turns out to be qualitative or non-functional requirements with which the requirement engineers and clients/stakeholders face this problem [Cysneiros 2004]. It remains a challenge to elicit such requirements from the stakeholders, as they are not clear to stakeholders as are functional requirements.

Problem of Negotiating Test Cases

The quality requirements derived from soft goals are often hard to quantify, i.e., to know exactly when the system fulfills them exactly. Such requirements are often bone of contention among the developers and other stakeholders. There are few methods in the literature to quantify such quality requirements.

2.3.2 Proposed Approach for Writing Requirement

The negation technique we propose considers the ambiguity definition given by Kamsties [Kamsties 2000] and understandability definition as given in Section 2.3.1.1. We suppose that the stakeholders and requirement engineer share a common system glossary for referencing and there is no ambiguity in glossary definitions of terminologies. The idea is that if you want to convey 'circle' do not convey it as 'diagram', otherwise it may be misunderstood as 'rectangle'.

The quality requirements test cases are difficult to be agreed upon by the stakeholders and requirement engineers. The traditional way of writing requirements specification is using the affirmative phrases but it gets difficult to link a test case with which the qualitative requirement can be verified. It is not the inverse requirement [Berry 2003, Berry 2004] which cannot be verified with a test case, but instead they are the characteristics or symptoms which could be tested and verified.

Such requirements can be identified easily, as they are non-functional or quality requirements, they seem to be opaque for both client and supplier. Another symptom for such requirement is that, it seems difficult to establish a test case for verification of such requirement.

Usually, it seems to be difficult for the document writer to explain in detail such requirements with traditional affirmative sentences alone. Such requirements written in affirmative sentences are usually written following the rules and writing styles imposed, making them less interactive and useful [Kovitz 1998]. We have found that while using the negative phrases in combination with affirmative phrases the effort is greatly reduced and test cases can be easily established to verify and validate the requirement. Use of negation for such requirement seems very natural and is easily accepted by the reader.

For example: The Need of a *comfortable chair* could be expressed as:

(a) the user should feel relaxed even after prolonged usage or as,

(b) the user should not feel any pain or cramp in his body, even after prolonged usage.

The second phrase seems more natural and turns out to be verifiable. We relate this fact later with our interviews and experiments in Section 2.3.3.

We also propose that a complex requirement statement should be written with complementary requirement statements, that restricts the possible deviated interpretations of the original requirement. We advocate the usage of negative sentences for restricting the mind of readers from interpreting the sentences in the wrong direction and guiding them towards the exact interpretation. Figure 2.8, shows the idea

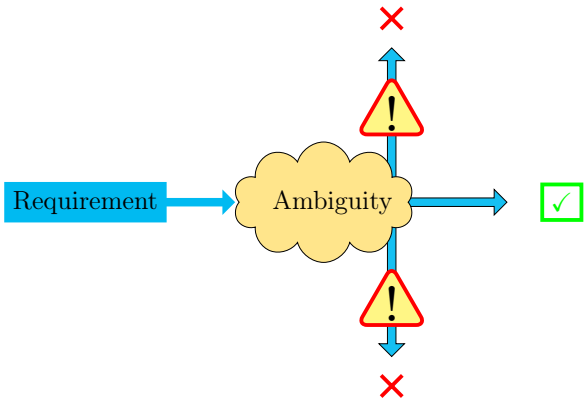


Figure 2.8: Restricted Interpretation

of restricted interpretation of the requirement statement. We propose that as soon as a document writer with suitable domain knowledge writes down a requirement in natural language, he should add the relevant negation statements which could otherwise be interpreted as the original requirement with the same context. Each requirement specification written with its complementary statements should be validated with the corresponding stakeholder. It should be verified that the stakeholder has also understood the complementary statements.

For example, if an auto-mobile company wants to develop a golf-car, it receives some of the stakeholder needs such as :

(a) *The car shall be comfortable.*

(b) *The car shall be reliable.*

(a) *The car shall be ecological.*

Table 2.6 shows how a requirement engineer will write the user needs in to requirement statements, using the two techniques; first with only affirmative sentences and second with affirmative and negative sentences. The table shows that the negative statements have increased the understandability of the requirement sentences and it helps greatly to generate the corresponding test cases.

For instance, to validate if the car is comfortable, a test case based on the simple affirmative requirement sentence would be difficult to conceive, whereas to validate whether the usage of car doesn't cramp your limbs would be easier. In this case, 'X' number of people could be made to sit on the car seats for a determined duration and later asked if they had any sensation of strain, cramp or pain in their limbs.

Similarly, it is easy to verify the pollution index of the car with a negative requirement statement, as it links directly to an associated test case. It is evident that the negative requirement statement helps the requirement engineer to have a better insight of product. Also, it demands less effort to conceptualize and also lessens the amount of effort for corresponding test cases to be generated for validation and verification of the product.

Table 2.6: Table of Requirements Using and Without Using Negation

Affirmative	**Affirmative + Negative**
a)The user shall be relaxed. b)The user shall remain at ease.	a)The usage of car shall not cause strain in the user's body. b) The user shall not get cramped while using car. c)The usage shall not induce postural deformity.
a) The car shall be safe. b) The car shall turn or move as per user's actions. c)The car shall provide protection to the user in case of accident.	a) The car shall not fail its missions. b) The car engine, breaks, and gears shall never fail before a particular duration. c) The car shall not get crushed in half, in case of a head on collision.
a)The car shall be pollution free. b) The car shall be 95% recyclable.	a) The car shall not release green house gases or toxic waste in to the environment. b) The car shall not produce noise more than 55 decibels.

2.3.3 Experiment and Empirical Findings

We proposed an experiment and a survey to know, whether negation really helps in explaining an idea or whether it really increases the understandability of ideas. We tried to verify this fact in two steps :

1. Questionnaire
2. Interviews

The language of interview and questionnaire were optional : English or French. Subjects involved in the questionnaire and interviews were of different nationalities and scientific background. Subjects had good command over English and had abilities to analyze and interpret a notion with scientific perspective. Total number of subjects, who participated in these questionnaires and interviews was 31. The survey was carried out in two steps. In the first step, first half of them were interviewed and other half put through the questionnaires. After an approximate gap of three months the second step of survey was carried out, which involved the same process but with the two groups switched with questionnaires and interviews. Till the end of second step, the subjects were not provided with the explanations of what the survey is upto. This was done to avoid any prejudice on affirmation or negation of interview or questionnaires. On an average the duration of an interview was of 8 minutes. The questionnaire contained four qualitative attributes, with 1-2 questions for every attribute. In the second part of the survey which was carried out after

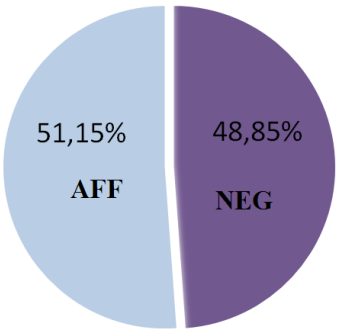


Figure 2.9: Gross Percentage of Affirmative Phrases vs Negative Phrases

some gap of days (approximately 3 months) the two previous groups were again surveyed but with swapping of questionnaires and interviews, keeping the duration of interview and type of questionnaires similar as before.

Questionnaire

The questionnaire we carried out contained phrases describing the quality of a product, for example comfortable, ecological, etc. In the first section, all the qualitative properties were written in affirmative phrases and in the second section, the same properties were written in a set of affirmative and negative phrases reinstating the same property. The reader was asked which one of the two types of explanation seems clearer and offers better understandability and less ambiguity. The readers were asked to rate the understandability of the two types of presentation on a scale of 1-10, with 10 meaning excellent and 1 referring to poor understandability.

Results

The results obtained showed that 60% of readers found it more understandable, when a qualitative property is written down together with negative phrases. 20% of readers said that both are equally understandable, rest of 20 % said it depends on the prior knowledge of a concept.

Interviews

We interviewed the other half of selected peoples for interview sessions. We asked them to explain a complex concept of an object. We asked them how they define a particular characteristic quality of a system. We tried to put cases from real life scenarios like reliability, comfort, or usability of a product. We counted the number of phrases utilized to explain the concept and counted the total number of negative phrases one used during the conversation for explaining the idea. Then they were asked, whether negative phrases were coming more easily in mind then affirmative phrases to explain the same concept.

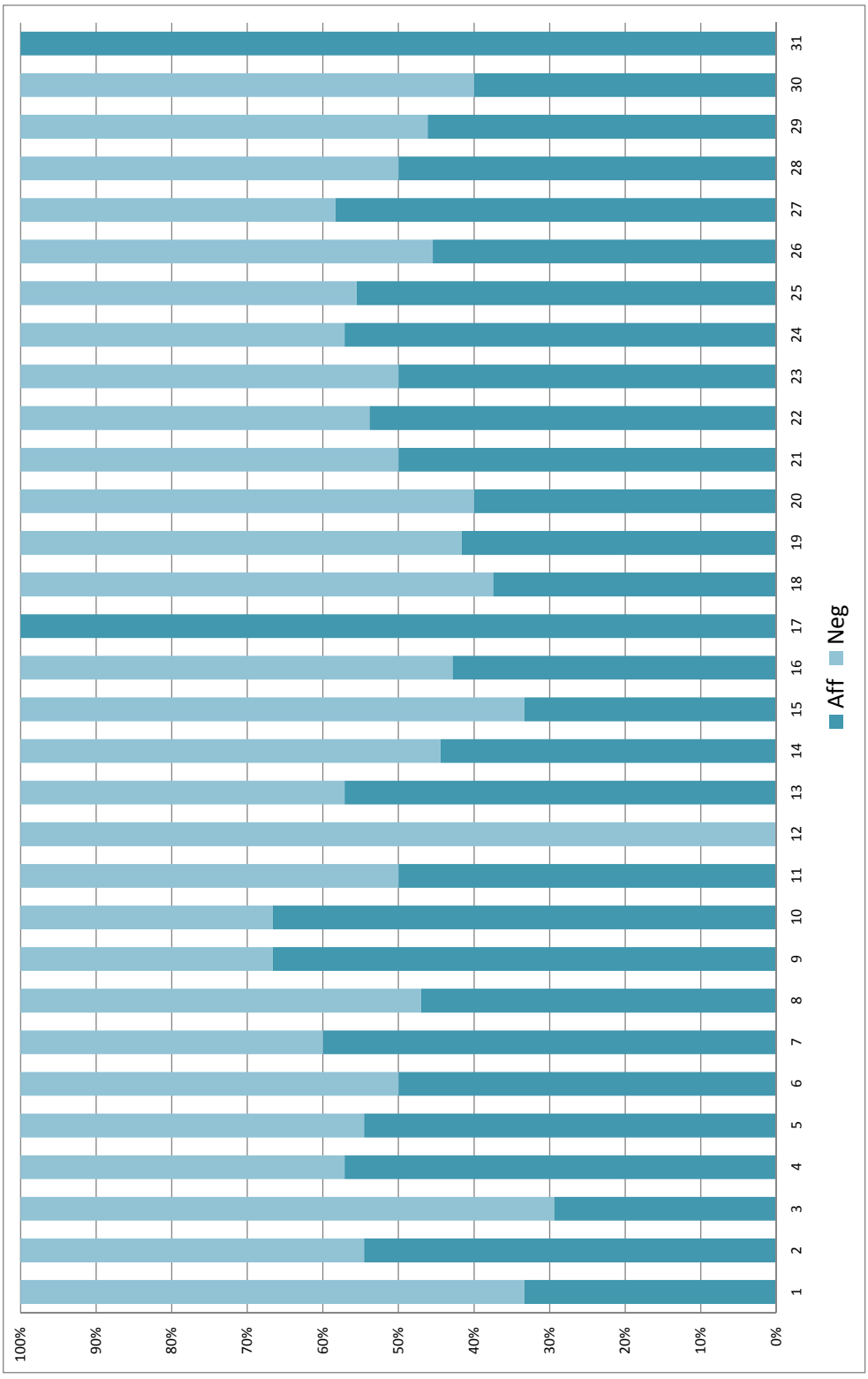


Figure 2.10: Individual Percentage of Affirmative Phrases vs Negative Phrases Production

A few examples of questions that were used with the majority of individuals are as: *What do you mean by comfortable chair? How would you define reliability in context of a car?* etc.

Results

The total number of phrases produced in the interview sessions was 348 phrases by 31 individuals, out of which 178 were in affirmation, and 170 were in negation. Figure 2.10, shows the individual distribution of negative and affirmative phrases. Figure 2.9, shows that roughly 48,85% of total phrases utilized to express a non-functional requirement were in negation, whereas 51.15% were in affirmation. On average, of the various interviewed peoples utilized, 89% of people used a blend of negative and affirmative phrases to elaborate a concept. After the interviews we pointed out to the individuals their pattern of sentence production and asked them why they tend to use negative sentences. 80% of them said that for explaining some notion, negative phrases seem to be more intuitive and natural.

2.3.4 Using Negation to Negotiate the Requirements

As empirical study carried out by us shows that, clients use negation as part of their conversation in natural language. It is very unnatural and artificial to provide them requirements documents written only in affirmative phrases. Although, it depends a lot on many factors: cultural, geographical, etc., in many cultures negation is completely avoided. Some discussions with experts in RE domain provided the reasons for using only affirmative sentences. The reason they provided could be summarized as: "*requirements written in negative phrases can be tormenting to developers*". But, given to the benefits the negation allows in framing the user needs, it can be used as tool to negotiate the requirements. These aspects of negation can be very useful in designing and negotiating the verification and validation test cases with the customers. With our empirical study, we would like to recommend to appreciate the use of negation while negotiating the requirements and test cases.

2.4 Discussion

We have provided some theoretical formulations of requirements. They may seem to be too abstract and hard to understand. But these formulations present a theory to understand requirements in depth. We have provided reasons for their existence in the system. The link between the stakeholders, their beliefs, rationales, requirements and viewpoints is new and allows to argue and understand a given requirement.

Although it is hard task to go through each and every requirements and rationale and stakeholder and differentiate them to find their roots. But some hard to understand and ambiguous requirements can be put through this process, which makes them explicitly clear and contractual. Our theory of requirements explains the vagueness of the requirements and need of the stakeholders, as it provides the

reasons for their ambiguity and intangibility. We provided some means and tools to make this requirements elicitation process more semi formal and try to bridge the gap of natural and semi-formal modeling.

Still, there is little to say how to help the design team in detecting all the viewpoints explicitly which are of interest to stakeholder and more importantly the stress vector of each stakeholder that he holds for the particular rationales. Even if one brings to light the various viewpoints held by a particular stakeholder, it is hard to claim that there are non other left. But some techniques can be devised to narrow down the list of view-points that can be held by a particular stakeholder and approximate stress vector.

Like any GORE based language CReML may suffer from similar issues, but the remedies of all the GORE based language lies in the transforming it into tool. The tool developed to implement the CReML has overcome a few of the challenges and makes it more scalable and allows to design highly complex diagrams while still remaining understandable by using layers of abstractions between the various artifacts. Artifacts which require a particular users attention are presented to him rather than presenting the whole complex system at a time.

Using negation as a tool for avoiding ambiguity also poses some challenges, it could be hard to say which requirement is appropriate for using negation. There are some cultural challenges too which are associated with using negation, in some cultures negation is reputed to have evil meaning, whereas in some cultures negation is used commonly and has nothing to do with evil sense. During our study, we tried to find the reasons that why it is recommended to use affirmative sentences for writing requirements. We could not find the precise reference which gives the reasons for secluding negation. During some discussions, a few of researchers suggested that some persons find negation to be *cruel*, hence they try to avoid it.

2.5 Conclusion

This chapter started from state of art of requirements engineering techniques for modeling and writing natural language requirements. Various definitions in the literature were reviewed for understanding the notions related to requirements. Issues with the current state of art of requirements modeling and writing natural language requirements were raised. A theory of requirements was presented with the aim of better understanding the requirements and the other notions related to it.

In order to take the benefit from the theoretical formulations, a requirements modeling language *CReML* is introduced, which tries to overcome the inherent challenges linked to the scalability issues of complex requirement diagrams. We integrated *CReML* with SysML to take benefit from the existing system modeling infrastructure. The proposed a graphical modeling language which is capable of functionalities typical to popular GORE techniques like i* and KAOS and other functionalities which are of concern to systems engineers and other stakeholders. Proposed language and supporting tool allows to represent the preferences of the

various stakeholders on the various goals and objectives. It allows to model both the core and optional features of the system under study. The goals can be traced back to the user stories which are linked to the goal modeling diagram. The responsibility and interaction among the agents is separately modeled and can be integrated if the developer wishes. The other interesting capability our tool provides is to model the rationales using view-points. The stakeholder rationales are projected and divided in to various viewpoints from the very early stage, which allows to better understand the user requirements. The end-product of goal modeling leads to system requirements which can be allocated to the UML/SysML diagrams. Our tool supports a few of the diagrams of the UML/SysML notably Use-case diagram and Block definition diagram. This is to provide direct traceability throughout the V-cycle.

We have handled other issues related to natural language requirements such as ambiguity and understandability. Devised some methods to avoid ambiguity using negation and used them for negotiating test cases. A few of the surveys were carried out to reinforce belief in the theoretical formulations and results were encouraging. Restricted negation technique offers interesting aspects as a tool against the ambiguity and understandability problem in context of RE. It is argued that for quality notions it is comparably easier to elaborate using negative sentences, than with purely affirmative ones. Findings with surveys have reaffirmed that for quality concepts negation is more natural than affirmation. In the context of requirement engineering, it can be used successfully as it seems more or equally natural and intuitive as affirmative ones. Our examples have manifested that, a good blend of affirmative and negative sentences can provide in depth view implications of a quality requirement. Interviews confirm that the user may prefer negative phrases and they may come intuitively. Fundamental argument is that, if some phrases seem to be natural, they are easily understood and accepted. Negation are natural in a natural language, so they should be banished from requirements rather than use them as a tool.

Chapter 3

Requirements Traceability

Contents

3.1 Introduction 61
3.2 Requirements Traceability 62
3.2.1 Requirements Traceability Processes and Problems 63
3.2.2 Traceability Recovery Challenges 68
3.3 State of Art of Requirement Traceability 70
3.3.1 Information Retrieval Based Techniques 70
3.3.2 Structurally Rule Based Techniques 74
3.3.3 Linguistically Rule Based 75
3.3.4 Transformation Rule Based 75
3.3.5 Other Miscellaneous Based 76
3.3.6 Works on Traceability Maintenance 82
3.3.7 Traceability For Systems Engineering 83
3.4 Proposed Solution for Traceability Problems 84
3.4.1 Semantics of Relationships for Requirement Traceability . . . 85
3.4.2 Planning and Managing Traceability Strategy 87
3.4.3 Trace Creation Process 89
3.4.4 Trace Maintenance Process 93
3.4.5 Trace Usage 96
3.4.6 Using Traceability Information for SE Activities 96
3.4.7 Comprehensive Traceability During Project Development . . 98
3.5 Discussion 101
3.6 Conclusion 102

3.1 Introduction

In systems engineering activities, requirements management occupies a considerable amount of space, both in technical processes and project processes. Requirements management processes consists of providing the platform for identification, management and implementation of client needs or requirements during a product life cycle. Requirement traceability is one of the activities of requirements management process which governs the life of requirements from their inception to their

implementation and thereafter. Requirement traceability deals with identifying or creating the trace links between the artifacts and using them further for conflict resolution, consistency checking, change impact analysis, testing, verification and validation. These activities depend on the quality of trace links identified earlier. The trace link identification process is also called traceability recovery. Traceability recovery techniques can be broadly classified in to three categories: manual, semi-automatic and automatic. Requirement traceability recovery techniques are of great interests in systems engineering as they are used in product life cycle management process. Success of a SE project depends on various factors, and requirements management activities are one of them. Adequately carried out RM activities ensure that the system is correctly defined, verified, validated and deployable.

In SE process, requirement traceability is one of the anticipated attribute for quality control. INCOSE handbook [Haskins 2011] for SE, mentions and demands special attention for traceability, but provides no insight about the methods or techniques for it. The most of the research work on requirement traceability is recent and it has gradually gain importance as industries look for better products and development processes. Still, industries have to gain lot from traceability.

In this chapter, Section 3.2 presents the requirements traceability, its processes and problems with the current traceability schemes. It also identifies the challenges faced by the systems engineers while selecting a particular recovery technique. Section 3.3 aims to bring together all the contemporary requirement traceability techniques, as part of an extensive literature survey, to identify the various existing requirement traceability techniques, their various benefits, limitations or inconveniences linked with the usage. Section 3.4 presents the proposed approach for the various problems raised. Next, the approach is discussed with its benefits and limitations in section 3.5. Finally, conclusions are presented in Section 3.6.

3.2 Requirements Traceability

Definition 3.1. In words of Gotel and Felkinstein [Gotel 1994], *"requirements traceability refers to the ability to describe and follow the life of a requirement, in both a forwards and backwards direction (i.e., from its origins, through its development and specification, to its subsequent deployment and use, and through periods of on-going refinement and iteration in any of these phases)."*

The requirement traceability principle states that [Gotel 2011] "*there is no ability to trace without a track and there is no ability to lay a track without making signs*". In accordance with the above mentioned principle, requirement traceability process consists of four activities: trace creation, trace recovery, trace maintenance and trace usage. Trace creation is the process of lying down the signs across the length and breadth of artifacts. Trace recovery is the practice of extracting links using the signs laid before to find the information flow and the relationships between the artifacts. Trace maintenance can be seen as the process of maintaining and updating the signs between the artifacts, which reflect the instantaneous and past states of the

system under development. Trace usage involves the set of activities which use this information held by the links for other SE activities like: change impact analysis, configuration management, verification & validation, installation, etc.

Every organization implements its own suitable guiding principles for requirement traceability which are known as 'traceability policies'. Traceability policies define which information dependencies between requirements should be maintained and how this information should be used and managed. The existing literature shows that information like *who, why, what, where, when,* etc., and other relationships are maintained for efficient requirement traceability [Spanoudakis 2005, Shukla 2012b]. There are already many methods for traceability maintenance like described in literature [Cleland-Huang 2003, Drivalos-Matragkas 2010, Nguyen 2005, Shukla 2011], without efficient maintenance, a traceability scheme becomes useless and is ignored by the users. Another important concern with traceability is of trace generation. There are numerous techniques in literature describing various methods to recuperate trace from artifacts. The majority of existing literature in requirement traceability is in context to software engineering. Although, we cover the traceability recovery schemes for both software engineering and SE, we try to present the traceability schemes in a broader picture of SE.

Requirements traceability can be broadly classified into various types: *pre-requirement traceability* and *post-requirement traceability* [Gotel 1994], *upward traceability* and *downward traceability* [Pinheiro 2004].

The *pre-requirement* traceability is accountable for all the mechanism before the stakeholder requirement(needs) are conceptualized. The *pre-requirement* traceability reveals the complex network of links between the stakeholders, their various roles, their rationales & beliefs, viewpoints and their various requirements.

The *post-requirement* traceability tracks this projection of requirements at various stages of the project life cycle. Various other notations may describe the traceability schemes like: upward-downward traceability, *functional traceability*, *non-functional traceability*, *intra requirement traceability*, *extra requirement traceability.*

3.2.1 Requirements Traceability Processes and Problems

The problem of requirements traceability in software and systems engineering exists because we lack the precise track on which we can trace the artifacts. As previously quoted by Gotel *et al.* [Gotel 2011], *"there is no ability to trace without a track and there is no ability to lay a track without making a sign".*

We have understood that, there cannot be a single technical solution which can address all of the challenges of traceability, in fact each of the challenge is an own class of problem. If a technique strives to achieve them all, it is deemed to fail. Each of them needs to be addressed individually systematically and holistically, and then only later integrated to form a whole solution to traceability problem. They cannot be solved in isolation, rather they can only be achieved when they are solved in a concurrent manner while one solution communicating with other. Figure 3.1, shows the current model of traceability process in industry. The requirements traceability

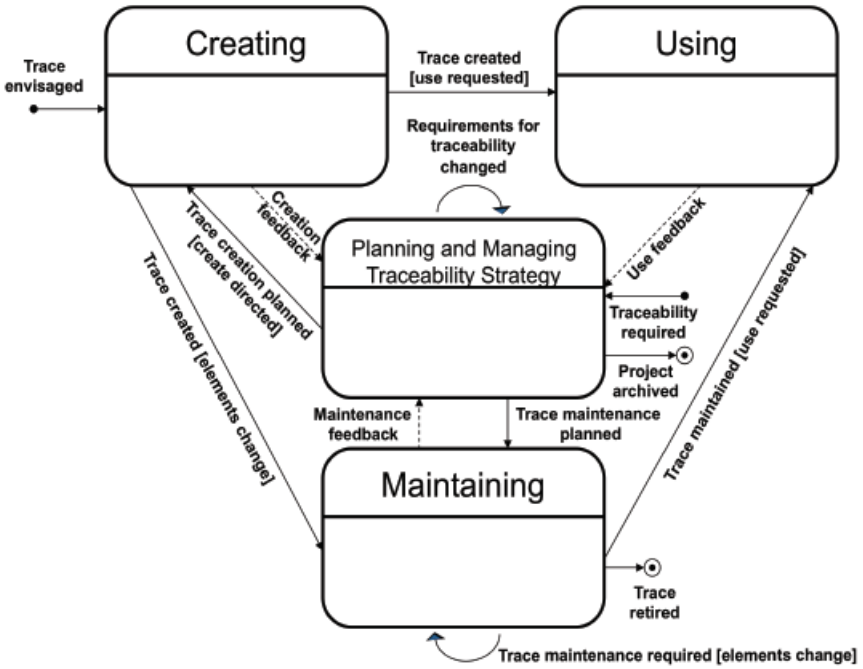


Figure 3.1: Current Traceability Process [Gotel 2012b]

problems have other set of organizational problems, for instance, there is no consensus over the fundamental vocabulary of requirements traceability, although recent works have started in this direction [Gotel 2012b]. There is no meta-model of trace which is generally accepted by all. The trace community has yet to achieve consensus on multiple fundamentals of traceability. The semantics of various existing relationships are yet to be standardized. The actual scenario is far more alarming, the systems engineering community demands extensively the traceability, but the tools on which they depend are not equipped to provide it. Even a few of them provide it remains confined to one tool, or if it claims to provides for another set of tools , then linking them remains manual. Traceability across the heterogeneous tools remains the bottleneck of requirements traceability in systems engineering projects. Fundamentally, the ideal solution would be to have a completely automated traceability scheme, but with the current technical limitations, it is not possible to fully automate the *pre-requirement* traceability. It is hard, to extract rationales and beliefs from human, as well as non-human stakeholders automatically. The pre-requirement traceability can only be tracked by involving the stakeholders in discussions and taken into account the *rationales* and *beliefs* of the stakeholders. The stakeholder identification process should also be traced, i.e., the techniques used by the teams to properly discover the stakeholders. The post-requirement traceability problem can be divided into three parts: *creation*, *generation*, and *maintenance*. All three activities are carried out in a context boundary called *traceability strategy* which includes planning and managing organizations' traceability schemes.

3.2.1.1 Traceability strategy

Traceability strategy is all about deciding what to trace and how much to trace at what cost. Traceability strategy planning is critical task, it has to take in account organization's budget and resources, clients traceability needs, or a particulars standard's traceability compliance needs. Upon the budget and considerations regarding the available human and technical resources it is decided how much to automate and what should be automated ? How much fine granular traceability should be achieved ? The choice of approach: manual, semi-automatic, automatic; techniques; tools; other traceability infrastructure is made during strategy formulation. Deciding upon whose traceability requirements are to be fulfilled and to which extent are part of strategy formulations. Figure 3.1 shows that strategy formulation and planning as the central task to establish requisite requirement traceability.

3.2.1.2 Trace Creation

The process of linking various requirements to artifacts at various levels of life-cycle is called trace generation. The trace creation problem is about how to create traces in such a manner that at a given instant all the artifacts are accountable, i.e., we know about the parameters like *'who'*, *'why'*, *'what'*, *'where'*, *'when'* and the trace relationships existing between the particular source artifact and the target artifact. The trace extraction process can be assumed as an on-demand trace creation process, i.e., when the system analyst demands the traceability between the two artifacts at same or different level in system. Determining the relationships between the two artifacts is a complicated process. It requires comprehensive understanding of the interaction between the various artifacts/components with their immediate environment and their interactions globally. The interactions can be at any level or view. The interactions from the multiple aspects of the system can ease the trace determining process [Shukla 2012a].

Among the various available traceability recovery techniques, the information retrieval based techniques are being used commercially by various industries for automatic link creation. This class of traceability recovery techniques are measured using two parameters called *precision* and *recall* [Antoniol 2000]. *Precision* is the fraction of the retrieved documents that are relevant to the search, as given by Eq.(3.1). *Recall* is the fraction of the documents that are relevant to the query that are successfully retrieved, as given by Eq.(3.2).

$$precision = \frac{|\{relevant\ document\} \cap \{retrieved\ documents\}|}{|\{retrieved\ document\}|} \tag{3.1}$$

$$recall = \frac{|\{relevant\ document\} \cap \{retrieved\ documents\}|}{|\{relevant\ document\}|} \tag{3.2}$$

The information retrieval based techniques are yet far from being perfect and have issues with the problem of poor precision and low recall. A poor precision leaves space for considerable human involvement despite of automatic retrieval, rendering the traceability recovery to be semi-automatic.

Similarly, in the case of poor recall, a missed relationship can be disastrous and may lead to considerable loss of resources. Therefore, human intervention is necessary to be sure that the recall is sufficient. But this two tier approach for inspection can have scalability issues, or in worst case some traces may be ignored deliberately or unknowingly. This leads to an erroneous traceability where the user or analyst has low confidence in the system.

3.2.1.3 Trace Generation

The process of extracting the links which indicate the relationships between the artifacts at different phases is called trace generation or recovery. The various traceability recovery techniques can be broadly classified into various classes: information retrieval based, structural rule based, linguistic rule based, transformation rule based and many other miscellaneous techniques.

A major difficulty of the requirements traceability recovery problem is about providing the fine grained traceability, i.e., ability of traceability scheme to provide very precise location of the relevant artifact in the provided documentation. Fine-grained traceability provides the links between the various customer requirements and the artifacts generated during the development activity and later during other project's activities. such as, customers requirements to the design components, installation plans, user manuals, configuration plan, etc. In the case of software engineering products, it can be compared to provide the exact location of a function in a source code, and in case of another system engineering product (hardware intensive system) the precise location of the implemented system functionality.

In the case of software engineering project, providing fine grained traceability can be much easier than the systems engineering projects. The part of source code in a software system can be identified with the very precise line numbers of the implemented functions or methods. Contrarily, in a hardware intensive system this is fairly complicated. For example, the machine drawings are usually drawn with various types of projections and with heterogeneous tool sets. Drawing of systems and sub-systems are drawn from different levels and perspectives. It can be cumbersome task to locate very precisely functionalities drawn into the such set of available drawings.

Another problem of traceability recovery process is linked with the heterogeneous set of artifacts. An acceptable traceability recovery technique should be able to provide the traceability of the artifacts across the various types of artifacts, i.e., a technique should be independent of types of input artifacts. The traceability recovery process should also address problems linked to the maintenance of the artifacts or the evolution of the requirements and their proposed solutions.

The requirements traceability recovery scheme should enhance the confidence of its users. There are many metrics proposed in literature to measure this, but the one more popular are *precision* and *recall*. False positives or false negatives are biggest challenge to a traceability recovery process. A false positive will lead to unnecessary work, whereas, a false negative will necessarily lead to huge amount of rework. Both

of these activities are highly undesired in an enterprise.

3.2.1.4 Trace Maintenance

Trace maintenance poses another serious challenge to traceability, without proper trace maintenance, everything previously achieved and maintained is supposed to be doomed. A complex traceability maintenance scheme is not appreciated by the analysts, whereas, a traceability scheme which provides proactive maintenance with minimum amount of interactions is desired by the analysts. The requirements traceability is a continuous activity, involving people of various levels to participate continuously and maintaining a perfect communication channel among them for avoiding any information lapse. A good communication channel can help to figure out inconsistencies in the interpretation of requirements among various stakeholders which is very necessary for requirements engineering activities. Besides the communication there are various issues in traceability maintenance. Maintenance is the activity of updating and modifying already existing traceability relationships [Schwarz 2010]. The traceability maintenance can be further divided into four major problems: cost of maintenance, dangling traces, information loss and low value of trace for low-end users.

Cost of maintenance: As the requirements are continuously evolving through the life of a project, requirements are added, removed or modified. The links between these evolving requirements need to be maintained. In a sufficiently complex system, the number of requirements can vary up-to a few thousand requirements depending upon the granularity. Maintaining these requirements can be a tedious task involving lot of computational and human resources.

Dangling Trace: A dangling trace is one which points nowhere or it lacks either a source or a target. Such situation may arise due to human or system error during the course of a continuous evolution of a fairly complex system. They may also arise due to changes in the system model rendering some part of old system out of the boundaries of new system and hence it becomes difficult to trace them with respect to new requirements.

Information Loss: Whenever a new requirement is added to the system it needs to be linked to other requirements and available artifacts. The corresponding owners of the linked artifacts should be informed and advised to bring up the necessary changes. Similarly whenever an artifact is removed or altered or its dependency changes all the information should be communicated to the various stakeholders. This task usually involves maintaining these fine grained relationships and continuous update of such information usually leads to loss of data and hence information. We claim this information to be important as they are result of earlier high level discussions and decisions which involved certain cost. If any such information is deleted permanently then in case of a future discussion there is chance that development team may reach a similar decision which was earlier found to be inutile. This may happen due to a probable change in the team or may be just of a simple absence of a member, which is quite possible as project development may take sufficiently long

time.

Increasing value of Trace for low end users: For the low end users traceability seems to be a monotonous task and they are reluctant to involve themselves in traceability process. They do not find it very useful for their objectives and hence traceability does not offer them sufficient valorization for their work.

The questions raised in Section 1.4.2 are based on this section and are addressed in this chapter.

Question 1.7: how to engineer requirements such that requisite amount of trace requirements are implemented throughout the SE of life cycle of product?

Question 1.8: how to do requirements traceability in SE projects?

Question 1.9: how to maintain requirements traceability in SE project ?

Question 1.10: how to provide the cost effective requirements traceability?

Question 1.11: how to provide purposeful requirements traceability to stakeholders?

Question 1.12: can we estimate the cost of requirements traceability? If yes how? if no why?

3.2.2 Traceability Recovery Challenges

The traceability recovery is only one step of the complete requirement traceability process. The choice of traceability recovery technique is very critical to the success of a project. Research works in this field have addressed many problems of requirement traceability but still there are some issues ahead. We discuss some of the challenges faced by the traceability schemes or more precisely recovery schemes and tools. *Automatic generation and verification of links:* It is still among one of the big issues of requirement traceability problem. There are traceability tools which claim to be fully automatic and fully capable of generating the traceability relationships among the artifacts; this generation process is followed by verification of traceability links, because still now the various information retrieval based traceability recovery techniques do not have 100 percent recall or 100 percent precision; the computing systems are not able to verify these semantic aspects flawlessly or better than human, therefore, manual verification of the relationships is the usual procedure. This manual verification is time consuming, and costly; this can also have scalability and reliability issues. A semi-automatic process but with credible verifiability is better choice than a fully automatic technique which is partially dependable. Research community should try to focus on this problem, as mere generation does not solve the problem. Therefore a process which can both generate and verify the traceability links with proven credibility will be solution to this problem.

Inconsistency: During the evolution of project, dependencies among the artifact changes and the relationships among them may remain intact or get changed. This is one of the biggest challenges faced by requirements engineer to track the traceability relationships among the artifacts. Maletic *et al.* [Maletic 2003] proposed a conformance analysis based on a formal hypertext model. They proposed using latent semantic indexing (LSI) signatures to detect changes in the documents and

track them using conformance graph. The challenge ahead is how to equip the traceability techniques to maintain consistency? How to highlight inconsistency using traceability techniques? How to make traceability a more active entity in the SE or software engineering?

False positives or negative: The another challenge with semi-automatic and automatic traceability techniques is of false positive, with the IR techniques available today the analyst has to decide thresholds with trial and error methods. For different projects this can be cumbersome, and time taking to find the exact value, as the number of false positives grows up too rapidly when similarity of artifacts pair decreases. One challenge is to develop confidence among analyst regarding tools and techniques. If an analyst encounters large number of false positive or false negative notifications during process, analyst stops using the tool and prefers using his experience and expertise to draw the links among the artifacts, and hence the traceability problem remains unsolved, costing significant amount of resources.

Scalability: Some traceability recovery techniques use indexing of terms, for retrieving the correct link between the artifacts, but this technique may have scalability issues, with respect to a sufficiently large project. The timely indexing of terms of complete set of documents can be very costly and time taking. It may not be possible also index all the documents available at different geographical locations, which is quite possible in today's scenario owing to distributed locations of sub-contractors or teams.

Automatic detection of relationship type: It is necessary to know about the type of relationship existing between the artifacts. The various types of relationships need to be identified between artifacts, to make them useful to the analyst. The current traceability recovery schemes do not provide a mean to automatically detect the types of relationships between the artifacts. Or if they provide, there is always a lack of confidence, which can only be verified with a human intervention.

Traceability across heterogeneous artifacts: Traceability recovery techniques are not well equipped to handle fine grained traceability relationships across heterogeneous artifacts. As the various documents used for developing the system are in different file formats, it is still cumbersome to link them properly. These heterogeneous artifacts are hard to be linked in a fine-grained manner. The vocabulary used in different documents may mean same or different things; hence, semantic aspects of different documents available in different format cause this problem. Semantic correctness can only be verified with human intervention.

Forward or reverse traceability policy: Traceability recovery should be a parallel activity to the development, but many recovery techniques are either implemented for forward traceability or backward traceability, which creates a gap between the current state of project progress and traceability state. It is also very difficult to select which type of traceability policy should be implemented or used in a project. As not all projects are similar, they have different needs for recovery policies: depending upon the type of project; software intensive or non-software intensive project; long duration project or short duration project; small size project or large size project.

3.3 State of Art of Requirement Traceability

Current literature on traceability contains ample work on need, and generation of traceability. Traceability literature is vast, as mention Gotel *et al.* [Gotel 2012a] the first paper on traceability arrived fifty years ago. The recent works [Gotel 2012b, Gotel 2012a] of CoEST[1] have formulated and elaborated the major traceability challenges and provided the new insight in to the actual traceability problem. There are few previous works [Cleland-Huang 2012], which have highlighted the issues of strategy formulation for traceability with purpose, but to best of our knowledge there is no comprehensive policy which provides the insights of traceability needs of each stakeholder. There is no traceability policy cost comparison and measurement technique in the literature to best of our knowledge, for both software and systems engineering projects. There are seminal works on value based requirements traceability in literature such as Egyed *et al.* [Egyed 2005b], and by Cleland-Huang *et al.* [Cleland-Huang 2012] in which they discuss the return on investment using heterogeneous techniques for maximum benefit with optimal efforts.

The earlier analysis of traceability problem [Gotel 1994] presented the difference between the *pre-requirement* and *post-requirement* traceability. Requirement traceability techniques can be broadly classified as: manual, semi-automatic or fully automatic. Manual techniques are those, in which direct human intervention is necessary for trace creation, quality of traceability fully depends on his expertise. In semiautomatic traceability scheme potential traces are proposed to the requirement engineer, and following to an analysis of proposed traces, the requirement engineer establishes the links between the artifacts. In fully automated traceability scheme human intervention is not necessary and traces are created and deployed, based on code synthesis using certain algorithms.

Otherwise, traceability recovery schemes can also be classified as follow:

1. Information retrieval based (IR)
2. Structurally rule based
3. Linguistically rule based
4. Transformation rule based
5. Other miscellaneous techniques

Table 3.1 provides a comparative analysis of the various traceability techniques, with their suitability for application domain of Software engineering or SE.

3.3.1 Information Retrieval Based Techniques

Information retrieval strategies calculate the similarity between a query and a document. Grossman and Frieder [Grossman 2004] define a as: a **retrieval strategy** *is an algorithm that takes a query Q and a set of documents $\{D_1, D_2, \cdots, D_n\}$ and identifies the similarity coefficient $SC(Q, D_i)$ for each of the document $1 \leq i \leq n$.*

[1] http://www.coest.org/

The group of documents over which the retrieval is performed is called *collection* or *corpus*. The quality of results in IR is measured using two metrics: precision and recall. Precision and recall are given by Eq.(3.1) and Eq.(3.2).

Vector space model (VSM) was introduced by Salton, Wong, and Yang [Salton 1975] as a technique for automatic indexing. It defines a document space consisting of documents D_i, each identified by one or more index terms T_j; the terms may be weighted or unweighted according to their importance between 0 and 1. Each document is represented by a t-dimensional vector: $D_i = (d_{i,1}, d_{i,2}, \cdots, d_{i,t})$, where d_{ij} represents the weight of the jth term.

Given the index vectors for two documents, it is possible to compute a similarity coefficient between them, $S(D_i, D_j)$, which reflects the degree of similarity in the corresponding terms and term weights.

Antoniol *et al.* [Antoniol 2002] introduced the usage of information retrieval technique based on VSM and *latent semantic indexing* (LSI) schemes for traceability recovery. They used $if - idf$ metric for retrieval effectiveness [Salton 1988]. According to this metric, the j^{th} element $d_{i,j}$ is derived from the term frequency $tf_{i,j}$ of the j^{th} term in the document D_i and the inverse document frequency idf_j of the term over the entire set of documents. The term frequency tfi, j is the ratio between the number of occurrences of word j^{th} over the total number of words contained in the document D_i. Antoniol *et al.* [Antoniol 2002] compared these results with *probabilistic retrieval models*. They carried out experiments with two case studies and suggested that both IR models are suitable for the problem of recovering traceability links. They show that the probabilistic model achieves higher values of recall with smaller cut values and makes little progress towards 100 percent of recall. On the other hand the VSM starts with lower recall values and makes regular progress with higher cut values towards 100 percent of recall.

Hayes *et al.* [Hayes 2003] considered the traceability problem as IR problem and implemented the *Vanilla VSM* and its two variants: with key-phrases and using thesaurus retrieval. In the first modifications they augmented the traditional vector model, by associating a list of technical terms or key- phrases with the document repository. When the model building software detected a technical term, it was added to the vocabulary and was treated as an ordinary term. This allowed augmenting the relevance of matches related to technical terms, and excluding some coincidental match. In the second modification based on thesaurus retrieval, they used a simple thesaurus. Each entry of thesaurus is a triplet $(k_i, k_j, a_i j)$, where k_i and k_j are vocabulary terms and $\alpha_{i,j} \varepsilon [0, 1]$ is perceived similarity coefficient of k_i and k_j. During the model building stage, thesaurus entries are recognized and added to the vocabulary as new word.

The main change in the behavior of this method with respect to classical Vanilla VSM and Vanilla VSM with key-phrases comes during the query processing stage. When computing the similarity between the query requirement and a document set, the standard cosine computation receives an add-on that is generated by matches found via the thesaurus.

The *probabilistic models* computes the similarity coefficient (SC) between a

query, and a document as the probability that the document will be relevant to the query. Probability theory is used to compute a measure of relevance between a query and a document. Probabilistic retrieval system computes the weight of a term with respect to its probability of occurrence in a document. There are many probabilistic models like simple term weight model [Robertson 1977], which is based on probability ranking principle, non-binary independence model [Yu 1989], Poisson model [Robertson 1994], component-based models [Kwok 1990], language models [Mori 1997]. Antonial *et al.*[Antoniol 2002] used a *probabilistic model* based on stochastic language model [Mori 1997] for information retrieval.

Deerwester *et al.* [Deerwester 1990] introduced an indexing scheme for information retrieval based on the semantic meaning of the words called *Latent Semantic Indexing* (LSI). LSI is based on vector space model. The earlier available IR techniques (VSM...) were using the keywords to index the terms, which had more chances of error as many people may not use the same term to represent same concept, this causes errors on the systems which fully depend on syntax and not on the semantics. They used singular value decomposition matrix (SVD) to filter out noise between two documents, such that two documents that have semantically similar terms are located close to each other in a multidimensional space. Marcus and Maletic [Marcus 2003] used first time latent semantic indexing methods for traceability recovery. Earlier they had used LSI for measuring similarities between source code elements. They carry out experiments and showed that LSI performs at least as well as Antoniol's [Antoniol 2002] methods of probabilistic and VSM IR techniques. Like others they also measured the quality of results in form of two IR metrics: recall and precision. Lormans and Deursen [Lormans 2006] explored whether LSI can help to trace requirement traceability in design and test, they equally carried out research to look if LSI based techniques can be plausible solution for traceability coverage [Lormans 2005]. Their findings say that, if the documents to be investigated are set up according to a well-designed traceability structure then LSI can reconstruct it. They say that LSI based techniques are not yet capable to fully replace a systematic requirement management process. Gross *et al.* [Gross 2007] used LSI for component procurement and conducted three case studies, they supported the usage of LSI for establishing the implicit relationships between different components and requirements explicitly. Mahmoud *et al.* [Mahmoud 2011] explored the aspects of source code indexing for automated tracing. LSI has been used for semantic clustering by Kuhn *et al.* [Kuhn 2007] for reverse engineering of systems, in other words reverse traceability.

The *Jensen-Shannon (JS) similarity model* is an IR technique proposed by Abadi *et al.* [Abadi 2008], it is driven by a probabilistic approach and hypothesis testing techniques [Gutman 1989]. It represents each document through a probability distribution i.e. each artifact is represented by a random variable, where the probability of its states is given by empirical distribution of terms occurring in the artifact. They conducted various experiments on real world datasets and compared their technique with LSI, VSM (implemented as lucene package), PLSI (probabilistic LSI) and SDR (Sufficient Dimensionality Reduction) techniques with various dimensions.

Their findings indicated that VSM and JS have almost equal performance; results showed SDR bit behind VSM and JS whereas LSI and PLSI gave worst performance. PLSI was significantly behind all the other techniques tested and compared. SDR is also an information retrieval technique but its applications on traceability are not explored elsewhere except by them, so we do not discuss SDR further.

Inference networks use evidential reasoning to estimate the probability that a document will be relevant to a query. They model the probabilistic retrieval models and enhance them to include additional evidence that a document may be relevant to a query. The essence of an inference network is to take known relationships and use them to "infer" other relationships. The two types of techniques which have been discussed in literature are **Bayesian** and **linear inference**.

Omoronyia *et al.* [Omoronyia 2011] describe a traceability generation approach based on *Bayesian inferencing technique* to model relevance of artifacts associated with traceability links. It depends on the interaction events trails left behind by collaborating developers while working within a development environment. The Bayesian inference technique explores the use of single and multiple variables as evidence nodes to determine the conditional probability of relevant use cases, developers, and code artifact entity instances.

Omoronyia *et al.* [Omoronyia 2011] also used a *linear inferencing technique*. Similar to the Bayesian technique it is also dependent on the interaction trail left behind by the developers. The linear inference technique accumulatively determines the relevance of use cases, system developers, and code artifacts to the requirement traceability link. Unlike Bayesian model, linear inference techniques capture context size dimension for the entities involved in each interaction event. It use work contexts graphs that characterize the situation of entities in work environment. It also uses a sphere of influence (SOI), the SOI ratio is used to represent the relative influence an entity exacts on the collaboration space. SOI ratio of an entity is defined as the ratio of the total number of unique entity instances directly associated with an entity (size of work context) compared to the total number of unique entity instances in the environment.

Capobianco [Capobianco 2009] introduced a technique based on numerical analysis for recovering traceability links between code and software documentation called *B-Spline Method*. They claim it to be better than vector space models and latent semantic indexing. They say that it can be comparable or sometimes better than probabilistic model i.e. Jensen-Shannon method. The proposed approach models the information contained in a software artifact by particular interpolation curves of plots mapping terms and frequency on the artifact. Like other IR techniques this traceability recovery technique is applied in two steps: artifact indexing and artifact classification. During the artifact indexing process, they indexed only nouns contained in the artifact. The extracted information is stored in a matrix called term-by-document matrix, with m being the number of all terms within the artifact, and n being the number of artifact in repository. During the artifact classification process, the B-spline approach is used: the artifacts are represented as set of points in the Cartesian plane. Once obtained the B-spline curve for each artifact a_i and a_j,

the similarity between two artifacts can be defined calculating the distance between two B-spline curves i.e. $Bspline_i(t)$ and $Bspline_j(t)$.

3.3.2 Structurally Rule Based Techniques

Structurally rule based techniques for traceability can be of many types: some involve program analysis, some are based on formal description of specifications.

Egyed [Egyed 2001, Egyed 2003, Egyed 2004a], and Egyed *et al.* [Egyed 2002, Egyed 2005a, Egyed 2007] introduced *scenario driven approach* for traceability, their work was similar to work of Lange and Nakamura [Lange 1997] in some aspects like: visualization of trace information by execution of program. *scenario driven approach* generates traceability information based upon observation of execution of test scenarios. This is a reverse engineering approach which traces artifacts after a product is available. This technique is strongly iterative by running software systems and observing dependencies. Scenario driven approach [Egyed 2004a, Egyed 2004b, Egyed 2005a] requires an observable and executable software system, a corresponding software model and a scenario describing test cases or usage scenarios of the software systems or it components and a monitoring tool (for monitoring which lines of code were executed).

A forward engineering approach given by Richardson and Green [Richardson 2004] for traceability generation is based on *program synthesis*. They claim it to be fully automatic as program code is derived based on the specifications of the behavior. Program can be changed completely by changing the specifications. The work is based on the automated software engineering group's work on domain-specific program synthesis systems (AUTOBAYES, AMPHION [Whittle 2001] and AUTOFILTER) at NASA Ames Research Center. They describe two types of traceability techniques: deep traceability and surface traceability. Deep traceability scheme is a heavyweight approach for traceability; it involves augmenting the program synthesis systems so that calculations carried out by the synthesis program are annotated with information on what the calculations were and why they were made. They claim that this approach works well for a purely deductive synthesis. In surface traceability scheme, the relationships between source and target are mapped automatically in two steps: in the first step, the synthesis system (or compiler) is applied to the source to generate target. The original source is called nominal source and the corresponding generated target is called nominal target. In the next step, small changes (perturbations) are made to the source (perturbed source) and the corresponding target programs are synthesized from it (perturbed target). As long as the synthesis process is deterministic, differences between the nominal and perturbed target program can only be caused by the differences between the nominal and perturbed source. They tested their technique on two examples: AUTOFILTER program synthesis systems and GNU GCC compiler.

3.3.3 Linguistically Rule Based

Zisman *et al.* [Zisman 2003] proposed an automatic traceability recovery technique based on natural language processing techniques used during requirements engineering. They used a requirement object model expressed in UML; a model which specifies the main parts, and their relationships of the product, it plays a role in the interactions with the users and delivers the common functionality. In proposed technique, they limited the requirement documentation to only three parts: commercial requirements specifications (CRS), functional requirements specifications (FRS) and requirement object model (ROM). They defined and used three types of relationships: overlaps, realizes, and requires relationship among the artifacts and documentation. The generation of traceability relations is based on two types of traceability rules: requirement to object model rules (RTOM) and inter-requirement rules (IREQ).

RTOM rules give means to trace relationship between requirement statements, and use case and analysis object models. These rules are used to match syntactically related terms in CRS and FRS with semantically related terms in ROM. IREQ rules are used to generate traceability between CRS and FRS. The natural language requirements are processed with a tool called CLAWS [CLAWS 2013] to convert them in to .xml format files with respective 'part-of-speech' tags representing different part of sentence. The two types of rules allows mapping of relationships using the corresponding CRS.XML, FRS.XML and ROM.XML files. They have implemented a tool and carried out experiments on requirement documentation of TV products. The results show interesting aspects with variation of recall between 0.46-0.81, and variation of precision between 0.52-0.94.

Spanaudakis *et al.* [Spanoudakis 2004] further carried out work based on the RTOM and IREQ rules given by Zisman [Zisman 2003]. They considered requirement documentation to be composed of: requirement statement documents use case documents and analysis object models. They described four types of relationships between artifacts: overlap relation, $requires\_execution\_of\_relations$, $requires\_features\_in$ relations and $can\_partially\_realise$ relations. They improved the RTOM and IREQ rules to accommodate newer modifications. They allowed customizing their approach by adding new type of relations and RTOM or IREQ rules or by modifying or deleting them. They developed a machine learning algorithm to create rules to trace hard to generate relationships.

3.3.4 Transformation Rule Based

The transformation rule based techniques are usually the one employed during model driven engineering, when one model is transformed in to another. The rules which define this transformation are responsible for this traceability.

The domain specific *deductive program synthesis* also allows such type of automatic traceability features. The Amphion/NAIF project and Amphion/NAV project at NASA Ames research center used deductive synthesis for space systems program

synthesis [Whittle 2001]. Amphion/NAIF system generated source code based on the high-level graphical specification describing input/output functions. In parallel to code generation, the NAIF system generates in parallel to proof that this implementation is correct. As NAIF project was limited to a domain of space science, NAV project tried to design state estimation software.

The Amphion projects are based on a domain theory. The domain theory specifies the types and operation signatures in the domain, and axioms describing the implementation of the abstract operations in terms of concrete operations. The domain theory also explains each axiom providing documentation about their meaning. The process of deductive synthesis submits the specifications and the axioms of the domain theory to the synthesis engine. The engine proves that the specification is a consequence of the domain theory, and returns a proof and witness terms for the output variables. The code generator records a trace of the application of the transformations.

Similar to the program synthesis approach, the OMG's model driven engineering [Soley 2000] allows the creation of systems based on a meta-model and also allows transformation from one meta-model to another meta-model. The rules between the meta-model and model contain the traceability information, equally between two meta-models the transformation trace information is stocked as rules written in a transformation language like, ATL [INRIA], QVT [OMG 2013], etc. of type of system.

The transformation rules between platform specific models are the trace information. A model driven traceability approach described by Anquetil *et al.* [Anquetil 2010] employs special independent modules called trace extractors for recovering information from artifact repository. The other existing traceability approaches in model driven engineering [Galvao 2007] consider model transformation as mechanism to generate trace links, apart from model transformation, the other technique used for traceability recovery is through merger languages. The merger languages use two separate models: primary model and trace model. They are merged together to produce an annotated model for inspection on demand.

3.3.5 Other Miscellaneous Based

Apart from above discussed techniques there are other possible ways to recover traceability information, we discuss below some of major works based on other techniques.

Xiaofan *et al.* [Chen 2011] tried to enhance the performance of traceability link recovery using IR by combining it with three other techniques: regular expression, key phrases, and clustering. They tried to put together these three techniques to overcome the limitation of each other. Their technique comprises of VSM as base technique which is responsible for link recovery. The regular expression technique is used to find all of occurrences of class names in documents, to augment the number of retrieved links at high cut points. Key phrases are used to extract key words from comments of code to provide a brief summary of each class's description comment

and use these to augment VSM technique's link recovery. Clustering is a division of set of objects in to groups of similar objects. Preliminary results were encouraging for both recall and precision.

Sherba *et al.* [Sherba 2003] have proposed an approach based on *hypermedia and information integration*. It allows the generation of new traceability relations based on relationship chaining. This approach uses special information integrators, and open hypermedia services. Integrators can discover and create traceability relations between software artifacts and other previously defined relations. Open hypermedia systems enable the creation and viewing of relationships in heterogeneous applications as well as the traversal of those relationships within and between applications. The new identified relations can be generated based on indirect and transitivity dependencies, complex dependencies containing more than one source or destination elements being related (anchors), intersection of anchors, or matching of pre-defined conditions between artifacts or relations.

Antoniol *et al.* [Antoniol 2000] introduced the concept of *modeling programmer behavior* for traceability recovery. In their technique they assumed that programmers tend to process applicationŰdomain knowledge in a uniform way, applying a set of unknown rules when writing code, more precisely when choosing identifiers. Thus program item names of different code chunks, related to the same high level documents or concepts are likely to the same nature. The technique says that, the rules adopted by the programmers choosing identifiers, those rules are implicitly represented by the joint probability distributions estimated on training sets.

Aponte *et al.* [Aponte 2011] advocated the use of *software artifact summarization* for improving the traceability link recovery using information retrieval techniques. They proposed the text summarization to create summaries for text based software artifacts. For other types of artifacts they suggested usage of hybrid summarization that combines textual and structural information. They argued for the generation of normative summaries which are capable of representing the original artifacts. They found that a wide variety of documents need to be summarized because IR based tools use heterogeneous software artifacts. To generate summaries at various stages they used various techniques like: text retrieval; structured based vocabulary; natural language processing; structural information etc.

To improve requirement traceability, Cleland-Huang *et al.* [Cleland-Huang 2005] introduced the concept of using supporting evidences for *dynamic requirements traceability*. Based on a basic probabilistic network model [Wong 1991, Wong 1995], she introduced three enhancement techniques: using hierarchical modeling, using artifact clustering, and using semi-automated graph pruning. The probabilistic network retrieval model consists of a graphical and a quantitative component. The graphical component is a directed acyclic graph (DAG), where the nodes are random variables, and the arcs are link pairs of nodes and represent association between variables. The DAG encodes the model's conditional independence assumptions among the variables.The results of the three different approaches gave interesting results; they showed varying abilities to enhance the retrieval of requirements trace.

Reflexion technique introduced by Murphy *et al.* [Murphy 1995] offers interesting aspects for requirements traceability. Although, it is rather a technique for artifact summarization but can be employed for traceability recovery and used equally for the traceability activities like: change impact analysis, program comprehension, and design conformance. *Reflexion technique* is primarily employed for using the drift between the design, and implementation of software systems. The analyst defines a high level model of interest, extracts a source model from the source code, and defines a mapping between the two models. A software reflexion model is then computed to determine where the engineer's high level model does and does not agree with the source model.A reflexion model summarizes a source model of a software system from the viewpoint of a particular high-level model.

Dagenais *et al.* [Dagenais 2007] proposed a technique for tracing concern implementation in a software project using *structural patterns*. Their technique tracks the implementation of concerns throughout the multiple versions of a software system by leveraging existing structural patterns that may exist among the elements in a mapping. The concerns can be mapped to the source code in many ways, including manual entry to sophisticated automated entries. Two popular techniques used for concern mapping are: extensionally, intensionally. In extension mode of concern implementation, the corresponding elements of the source code are recorded, and mapped to the concern, through the signatures of the code. This technique is poor in case of software code evolution. The other technique in which concerns are listed through patterns is called intensions; they describe their characteristics. This type of mapping requires more effort from the programmer.

Recent research work on using *ontologies* for traceability recovery has shown interesting aspects. Zhang *et al.* [Zhang 2006] introduced a technique using text mining(TM) system based on ontologies for traceability recovery. They used text mining for linking the artifacts produced during the life cycle of a project. Their technique provides formal ontological representation of both source code and document artifacts. The ontologies capture structural and semantic information conveyed in artifacts and allow recovering traceability links between software implementation and documentation at semantic level. The traceability links between the ontological representations of source code and documentation are linked through existing ontological alignment techniques. Alignment techniques try to align ontological information from different sources on conceptual or instance levels. The TM system can also directly take input from the results of the source code analysis when detecting named entities. Manual definition of new concept or relationships is also allowed to establish links that cannot be detected by automated alignment.

Table 3.1: Comparative Analysis of Various Traceability Techniques

Name/*Type of Technique*	**Advantages**	**Disadvantages**	**Tool**	**Application Domain**
Vector Space Model/*IRB*	Ease of usage & integration in to the SE & Software engineering projects. Requires least amount of manual intervention for setting up the traceability process in the existing project.	Maintenance is an issue, Recall & precision can be varying, linking does not takes into account the semantic aspects of artifacts, trace creation depends strongly upon the mnemonics used	Reqtify ArchTrace	S/W& SE projects
Latent Semantic Indexing/*IRB*	Similar to VSM, allows to capture the similarity of underlying concepts, better recall and precision than VSM	recall & precision can be varying, inefficient with huge dataset or large number of documents	RETRO	S/W& SE projects
Probabilistic Model/*IRB*	Similar to VSM,	Information loss is possible, issues with recall & precision, scalability issues with huge data set, comparatively more effort is required for query and document representations		S/W& SE projects
Jensen Shannon/*IRB*	Better precision for large dataset, robust over different parameters than any other IR based technique	Issues similar to VSM, LSI	Tool exist(Name unknown)	S/W& SE projects
Bayesian inference/*IRB*	Allows to capture relationships with semantical links, trace links with purpose can be obtained	Recall & precision can be varying, organization policies may cause hinder to use this approach	Eclipse	S/W projects

Table 3.1 – continued from previous page

Name/*Type of Technique*	**Advantages**	**Disadvantages**	**Tool**	**Application Domain**
Linear inference/*IRB*	Allows to capture relationships with semantical links, specific contextual information can be provided, relevance ranking of artifacts can be obtained	Does not provides comprehensive requirements traceability, rather a tool for traceability	Eclipse	S/W projects
B-Spline Method/*IRB*	Robust, semantic relationships partially are taken into account, lesser false positive, better than VSM, LSI			S/W& SE projects
Scenario Driven approach/*SRB*	Ease of usage, could be used for validation of traces also, can be used on legacy systems too	Needs specially an observable version of software, can only be applied once software is ready to use, difficulty in determining type of dependencies	Trace-Analyzer, STRADA	S/W projects
Program Synthesis/*SRB*	very Simple, very reactive	Accuracy and coverage can be varying		S/W projects
Linguistically Rule Based Techniques	Automatic generation of traceability, rapid, different types of traceability information, better recall & precision	only syntactic relations are taken into account, creation of traceability rules demands extra work and are very subjective(vary from one project to another)	Tool exist (Name unknown)	S/W projects
Model Based Traceability/*TRB*	Good for the SE, special tasks like simulation of subsystems		MBSE Editors	S/W& SE projects
IR Combination Approaches	Similar to VSM, better recall and precision,	depends upon the mnemonics used, depends upon the inputs provided at different steps	Tool exist(Name unknown)	S/W & SE projects

Table 3.1 – continued from previous page

Name/*Type of Technique*	**Advantages**	**Disadvantages**	**Tool**	**Application Domain**
Hypermedia and Information Integration	Quality traceability can be achieved, even complex relationships	More studies needs to be carried out	TraceM	S/W projects
Modeling Programmer Behavior	Can be fully automated, other benefits similar to IR based techniques	Issues with estimations of probability, depends upon the mnemonics used, requires training for S/W developers	Tool exist(Name unknown)	S/W projects
Software Artifact summarization	Quality traceability, light weight, decreases amount of effort during trace examinations	Requires considerable efforts from the developers	Tool exist(Name unknown)	S/W projects
Dynamic Requirement traceability	High recall & better precision	Requires effort for initializing inputs, datasets may behave differently	Poirot	S/W& SE projects
Reflexion Technique	Lightweight, approximate, Scalable	Multiple iterations are required to generate reflexion models, strongly depends upon the human resources	Reflexion Model Tool	S/W projects
Structural Patterns for traceability	Easily detects modifications in different versions,high level traceability can be achieved in much easy way, resilient to evolutionary changes	Requires more effort from programmer, fine grained traceability cannot be achieved	ISIS4J	S/W projects
Using Ontologies	Correct traceability can be achieved, semantically linked traceability	Requires considerable efforts from the developer throughout the development process, time taking	SOUND	S/W& SE projects

3.3.6 Works on Traceability Maintenance

Compare to traceability generation fewer work have addressed the traceability maintenance [Cleland-Huang 2003, Nguyen 2005, Mader 2009, Drivalos-Matragkas 2010, Seibel 2010]. Cleland-Huang *et al.* [Cleland-Huang 2003] proposes publish-subscribe mechanism, a relationship between artifacts is registered to a central server. The evolution is represented by the series of change event. When a requirement is changed, the subscribers are notified about the change and they may bring the potential changes to their artifacts. It allows complete removal of requirement artifacts.

Another event based scheme [Mader 2009] uses a tool called Ttracemaintainer but it uses only UML structural models. Another similar tool to Ttracemaintainer is ArchTrace [Murta 2006], it addresses the consistency and evolution of trace links between software architecture models and their associated code. Another approach for evolving traceability for heterogeneous artifacts [Hong 2010] gives interesting insights about which information should be traced for corresponding artifacts so that fine-grained differencing can be used to identify evolution.

The graph-based traceability schemes exist in literature like [Pinheiro 1996, Lucia 2007, Schwarz 2010]. Schwarz *et al.* [Schwarz 2010] recommends the complete deletion of traceability links hence in this respect it is like our maintenance model, but it insists the trace maintenance using the technique based on [Cleland-Huang 2003], but essentially they are based on transformation models.

Some earlier works have recommend versioning schemes for traceability maintenance of artifacts [Nguyen 2005], but with the versioning schemes it becomes hard to see the evolution at an instant. Other significant approaches is state-based [Drivalos-Matragkas 2010].The state-based techniques employ syntactic differences between different versions of model. Some use text differencing to identify change. The other techniques for managing traceability based on evolution, use policy based support [Seibel 2010]. An important aspect of various traceability models is of the traceability recovery scheme. To reduce the cost of traceability, use of semi-automatic and automatic mechanism for traceability recovery is advocated. This is an important aspect, as for a fairly large sized project creating traces manually can be tardy.

ADAMS [Lucia 2007, De Lucia 2010] uses a latent semantic indexing scheme for traceability recovery from the checked in artifacts. There are many schemes based on IR (information retrieval) and vector space model techniques. The majority of traceability tools equipped with semi-automatic or automatic recovery techniques are plagued with 'false positive' problem [Lucia 2007]. The tool ADAMS uses an event notification scheme and claims automatic traceability recovery scheme and other modules for project management. It also uses a versioning scheme for traces, but still some information loss is still possible owing to complete removal of artifacts before the version release. There are many traceability models, but most of the systems are overly complex and do not address the chronological evolution and information loss problem in particular.

3.3.7 Traceability For Systems Engineering

Systems engineering standards specifically mention the creation of requirements traceability between the various artifacts created and used during design activities [ISO 2008, EIA632 2005]. The most of the traceability recovery techniques that are found in literature of requirements engineering have their origins in software products or software engineering. In the literature of SE, it is recommended to follow requirement traceability, but there is hardly any prescribed description of traceability techniques, which should be applied while carrying out SE. The traceability creation and recovery is usually manual in early systems development phase. Apparently, techniques like model transformation, structural rule-based, or information retrieval have limited applications for traceability recovery in non-software intensive systems. One of the popular tools used for handling requirement management activities in industry is DOOR [Eng 2002], which is also used for requirement traceability, but usually creation of links between the requirements is done manually.

Industries have much to gain from automatic traceability recovery techniques. Information retrieval (IR) techniques used for traceability in software consider the source code along with documents to provide fine grained traceability. The same procedure cannot be applied as it is to hardware intensive systems, for the reasons of accuracy or precision; valid information can be missed and outcomes can be disastrous for a project. Until now, no information retrieval technique has claimed to achieve 100% recall even with a lower degree of precision.

The scenario based traceability techniques can have scalability and reachability issues if they are applied on the systems, as in a fairly large system (complex) it is not possible to monitor the whole system at a time, and derive the dependency relationships between the subsystems; for another reasons, it is not feasible to lead a project successfully without having the traceability or dependency information a priori. For small projects also, unlike software systems, we cannot monitor every relationship in a non-software intensive system. The transformational rule based techniques have limited application for traceability recovery; they can be utilized up until systems modeling and design. For implementing fine grained traceability in SE suitable metrics should be carefully designed which are capable of answering the various needs of system analyst or stakeholders. The automatic traceability recovery techniques are not equipped enough, to reply to the needs of non-software intensive systems currently.

The industrial challenge of requirement traceability has not been popularly addressed. A report available at website of AFIS [AFIS 2013] shows that tools like DOORS and Reqtify are used together complementarily by many enterprises, to overcome the limitation of one another. DOORS employs parsing techniques to capture requirements, and Reqtify uses Perl regular expression extraction mechanism for creating links between existing documents, glossary, indexes, etc.

Some tools also use knowledge-based techniques like ontologies for creating traceability links between the artifacts. Currently the techniques above mentioned are still not 100 percent reliable as they may cause false negation or false positive er-

rors. Industries do not consider Reqtify a requirement-engineering tool, but only a complementary tool to DOORS for traceability, also DOORS do not correspond to the needs of requirement engineers; maintenance issues are among the biggest challenging issues with it. For doing SE, the traceability recovery and maintenance techniques need to be more active entities in the SE process. The SE processes should try to holistically integrate the traceability process within it, making it integral part of itself, considering its implications to wide range of other tasks or process such as: decision making, conflict resolution, change impact analysis, simulations, tests, verification, validation, end user evaluations, etc.

Traceability relationships should be reasonable and verifiable; they should be able to answer the "raison de être" of every entity or artifact in the system. It should not only capture the rationale behind every need but also provide process and means to show how they are implemented at next levels of life-cycle. Rule-based techniques [Zisman 2003, Spanoudakis 2004] like IR techniques also show interesting aspects for traceability recovery, with varying precision and recall degrees depending upon artifacts. But still the problem remains unsolved as they too do not report consistent high level of recall and precision.

There are also issues regarding, how much traceability information should be maintained, and up to which extent they should be traced at different level. Egyed *et al.* [Egyed 2005b] tried to determine the cost quality trade-off for software traceability. In this respect, software and systems share similar problems. There is no theory which can correctly answer these needs of modern Software and SE. The traceability policies for different types of projects should be adapted adequately to address the special needs of projects. The traceability policies which are used for a very large size project should not be implemented on a small or medium size project or vice-versa, because in the first case it would lead to over-expenditure and in second case it may lead to inadequate traceability.

3.4 Proposed Solution for Traceability Problems

To address the challenges and questions raised upon traceability problems an approach is devised. The approach lies in establishing a network of *traceability infrastructure* across the life-cycle, as shown in Figure 3.2. A dedicated trace repository is established, which is connected bidirectionally with the heterogeneous tool sets and communicating with a standard trace file format across the product development cycle. The product development cycle consists of different phases as shown in Figure 3.2, and can be assumed of five layers. The traceability recovery process can be solved by dividing it into subtasks.

The *pre-requirement* traceability and the *post-requirement* traceability are addressed separately and then integrated to form a comprehensive solutions to the traceability problems. The stakeholder identification process is also covered in the traceability recovery process. This traceability recovery process aims at uniquely identifying each entity or artifact. The effectiveness of traceability process lies in

the manner the data are managed, to avoid lot of data that serves no purpose except leading to unnecessary burden. Relationships defined in Chapter 2 Section 2.2.3 are sufficient for providing the pre-requirement traceability, however they are valid for post-requirements traceability too. *Derive*, *conflict* and *contribute* relationships are reused here, with their semantics unchanged. As in literature, semantics of traceability relationships have differences originating from different authors. In order to use traceability relationships precisely, the semantics of traceability relationships need to be clarified precisely. Semantics to various relationships existing in post requirement traceability process, which can help to automate the entire process are presented in next section.

3.4.1 Semantics of Relationships for Requirement Traceability

A *trace* is a *relationship* existing between a source artifact and a target artifact. Previously, various terminologies were defined and used in Chapter 2, they are used here too. Semantics needed for defining the relationships used in post-requirement traceability are considered here. A *design element* is defined as an artifact that models the structural or behavior of a component. Various requirements artifact were introduced in Chapter 2, they are used in proposed approach.

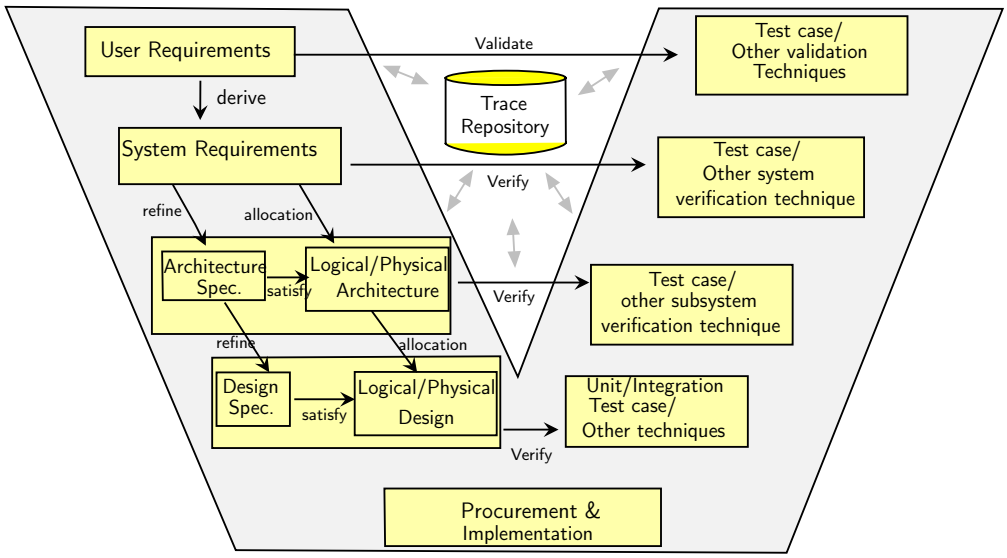


Figure 3.2: Traceability Relationships and Infrastructure

Figure 3.2, shows the locates the occurrence of traceability relationships in the Vee-model of life cycle together with the necessary infrastructure. Figure 3.3 shows the trace information model used in our approach. A trace is a type of relationship, existing between a source artifact and a target artifact. A trace itself may be composed of other traces. An artifact in the system can be alive or dead depending upon the conception stage. An artifact owns a few of the data about it, *'who', 'why', 'what', 'when', 'where', 'how', 'verify'* and the *'time stamp'* are iden-

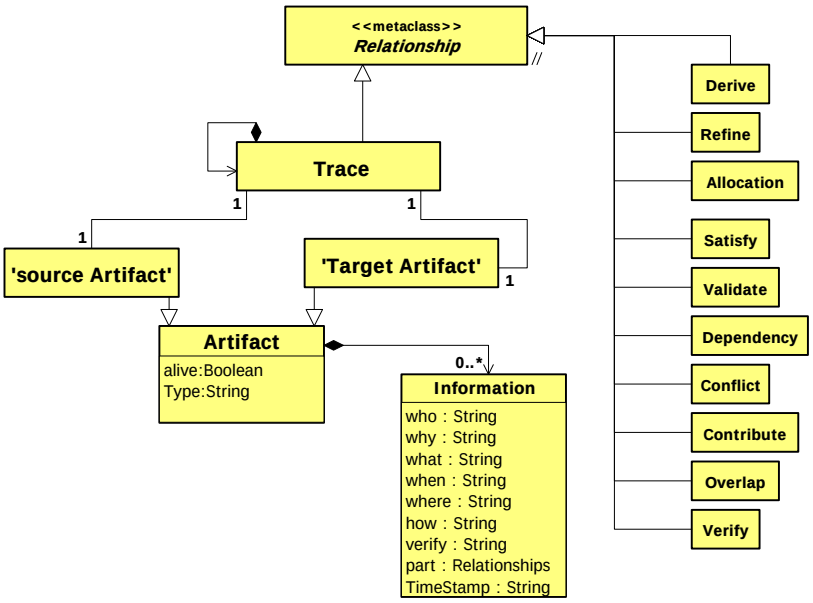


Figure 3.3: Trace Information Model

tified as information fields it may have. A few artifact may exist without all of the information fields, depending upon the type of artifact. *Who* block provides the information about the primary origin of the stakeholders of the requirement. It can have single or multiple values. *Why* block provides the information about the rationales for existence of the corresponding requirement. *'Where'* block provides the location of the implementation of the requirements in the system and hence provides fine grain traceability. *What* block provides the information about the role of the requirement, the function with which it is associated. *When* block, provides the temporal information about the implementation of the requirement in the system. *How* block provides the information about the mode of implementation of the requirement. It reveals the information about the realization of the requirement in system. *Verify* block provides the information about, how it can be proven that the associated requirement is successfully implemented in the corresponding system. *Time stamp* provides the information about the introduction of artifact into the conception process.

In our approach semantics of trace *relationships* are defined as follows:

- *Derive* is the relationship existing between a user need (Stakeholder requirement) and a system requirement, such that the first implies the later.
- *Refine* is the relationship existing between a system requirement and an architecture specification, or between an architecture specification and a design specification such that the first implies the later.
- *Allocation* is the relationship existing between a system requirement and proposed system architecture, or between a system architecture and a design

component such that the first implies the later.

- *Satisfy* is the relationship existing between an architecture and an architecture specification or between a *design element* (component) and a design specification, such that the later is characteristic of first.
- *Validate* is the relationship existing between a stakeholder requirement(need) and its corresponding test cases.
- *Verify* is the relationship existing between a system-requirement and verification test cases or between a specification and test cases or between a design component and its corresponding unit/integration test cases.
- *Conflict* is the relationship existing between two requirements such that their co-existence cannot be achieved as such without negotiation; or between two architectures or between two design components such that their co-existence cannot be achieved as such without negotiation.
- *Dependency* is the relationship existing between two requirements such that they are dependent to each other to exist; or between two design elements such that they are dependent to each other to exist.
- *Overlap* is the relationship existing between two requirements or needs such that they lead to common set of specifications.
- *Contribute* relationship is used to represent the direct contribution of information in context of a particular requirement artifact from a stakeholder for system under study.

Relationships *derive* and *refine* represent the two different levels of abstraction in the life-cycle. Similarly *allocation* and *satisfy*, represent the two different levels of abstraction in the life-cycle. Figure 3.4 shows the proposed traceability process in form of an activity diagram. Whenever a project starts, traceability is required. First activity with traceability involves planning and managing traceability strategy. RE activities may add, modify or remove artifacts which should be followed by creation of traces. This is followed by maintenance of traceability information. Usage of traceability information provides feedback, which is used for traceability planing and strategy. Various activities mentioned in Figure 3.4 are detailed in next sections.

3.4.2 Planning and Managing Traceability Strategy

Although it is difficult to extract exhaustively all the *rationales* from the various stakeholders, the first step towards the pre-requirement traceability is to take into account all the rationales and valid beliefs held by various stakeholders. Rationale extraction is usually manual, involving fair amount of interaction. Here we do not go into details about how to do it. For managing the traceability strategy, we propose a *rationale* map.

A *rationale map* consists of three matrices: Stakeholder-Rationale (St_R), Stakeholder-Preference (St_r_P) and Rationale-Rationale (R_R). The St_R is

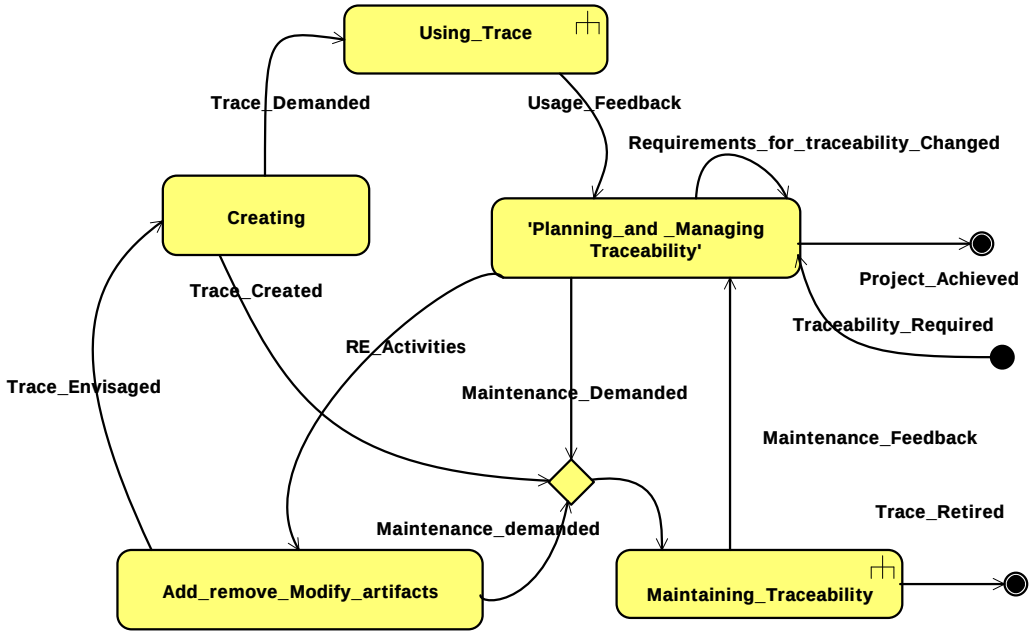


Figure 3.4: Proposed Traceability Process

a matrix of size $m \times n$, where m is the number of stakeholders of the project and n is the number of rationales elicited, given by Eq.(3.3). The St_R matrix allows to trace the origins of a rationales to stakeholders. The St_r_P matrix is also of size $m \times n$, it allows to represent the traceability preferences of various stakeholders towards the elicited rationales in the life cycle, given by Eq.(3.4). The R_R matrix is a $n \times n$ matrix, where n is number of rationales, given by Eq.(3.5). It is a symmetric matrix with diagonal composed of either of all zeros or all ones. It allows to take in account the potential relationships between the various rationales and hence requirements.

- For each entry $a_{ij} \in (St\_R)$, specifies whether stakeholder i is origin of rationale j or not.

$$a_{ij} = \begin{cases} 1 & \text{If stakeholder } i \text{ is origin of rationale } j \\ 0 & \text{If stakeholder } i \text{ is not origin of rationale } j \end{cases} \tag{3.3}$$

- For each entry $b_{ij} \in (St_r\_P)$, specifies stakeholder i's traceability preference of rationale j in the product life cycle.

$$b_{ij} = \begin{cases} 2 & \text{If stakeholder } i \text{ strongly needs traceability of rationale } j \\ 1 & \text{If stakeholder } i \text{ weakly needs traceability of rationale } j \\ 0 & \text{If stakeholder } i \text{ is indifferent to traceability of rationale } j \\ -1 & \text{If stakeholder } i \text{ weakly dislikes traceability of rationale } j \\ -2 & \text{If stakeholder } i \text{ strongly dislikes traceability of rationale } j \end{cases} \tag{3.4}$$

- For each entry $c_{i,j} \in (R\_R)$, specifies rationale i's relationship with rationale j.

$$c_{ij} = \begin{cases} 1 & \text{If rationale } i \text{ is directly proportional to rationale } j \\ 0 & \text{If rationale } i \text{ is independent to rationale } j \\ -1 & \text{If rationale } i \text{ is inversely proportional to rationale } j \end{cases} \tag{3.5}$$

The graphical model of rationale map is converted to matrices and enriched manually for eliciting the preferences and dependencies existing between the rationales.

The *rationale* map allows to understand the relationship existing between stakeholders and rationales, and relationships between the various rationales. CReML goal-modeling allows to take in account the various rationales by creating rationale map from user stories and interviews as mentioned in Chapter 2.

3.4.2.1 Towards *Purposed* Tracing

The St_r_P matrix allows the systems engineer to invest suitably towards the stakeholders' sensitivities vis-à-vis traceability of rationales and requirement in life cycle. This allows to associate the various requirements with their end-users who are most demanding for their corresponding traceability, before starting the project. Sharing of this St_r_P matrix with other organizations or projects can also be helpful and can be helpful to provide patterns of purposeful tracing.

3.4.2.2 Towards *Cost-effective* Tracing

As in any organization all the stakeholders are ranked and weighted before kick-starting the project, the product of the stakeholders' weights and St_r_P, St_R matrices provides insights into the amount of relative benefit the particular requirements traceability will provide. This information can be very useful when planning the budget of traceability.

3.4.3 Trace Creation Process

3.4.3.1 Making Traces

Trace creation process assumes, trace engineer invests suitable amount of effort to properly create them. A *Trace* can provide relationships using the data like 'who', 'why', 'what', 'where', 'how', 'when' and 'verify' of an artifact. In order to avoid confusion at different layers of development, we use two other block: *How-fine* and *Validate*. Their semantics is similar to 'how' and 'verify' but they are used at different layers.

- *How-fine* is the 'how' block for details design phase, it provides the measurable precise information about the realization of the customer requirement in system.

- *Validate* is the 'verify' block that provides the information about, how it can be proven that the stakeholder requirement is successfully implemented in the corresponding system. The information is in form of links to associated test cases and validation techniques (acceptance test, crash tests, etc.,).

The systematic trace creation is carried out throughout the life-cycle of project, hence at every phase of life cycle we create some part of trace. AS the project life cycle can be assumed to be divided in five layers from top to bottom. Phases in one layer are planned together, phase in left followed by one in right. The creation with respect of every phase of project life cycle is described below:

- In the customer needs elicitation phase, we assume that we already have the St_R matrix, it can also be called as *who-why* matrix, as it maps the 'who' & 'why'. Following to this, map *Stakeholders* and the *customer needs* in a matrix progressively during customer needs elicitation phase, this provides the *who-what* matrix. As after the customer needs elicitation phase, the system validation plan is already started, for each customer requirement the validation plan is mapped providing the *what-verify* matrix. In *who-why* matrix st_i represents the i'th stakeholder and r_j represents the j'th rationale. In *who-what* (who-wht) matrix st_i represents the i'th stakeholder and cr_j represents the j'th customer requirement. In what-validate(wht-vl) matrix vl_i represents the i'th validation plan.

$$\begin{array}{c} \textit{who-why} \\ \begin{matrix} st_1 \\ st_2 \\ \vdots \\ st_m \end{matrix} \end{array} \begin{array}{c} \begin{matrix} r_1 & \cdots & r_n \end{matrix} \\ \begin{bmatrix} 1 & \cdots & 1 \\ 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 0 \end{bmatrix} \end{array}, \quad \begin{array}{c} \textit{who-wht} \\ \begin{matrix} st_1 \\ st_2 \\ \vdots \\ st_m \end{matrix} \end{array} \begin{array}{c} \begin{matrix} cr_1 & \cdots & cr_l \end{matrix} \\ \begin{bmatrix} 1 & \cdots & 1 \\ 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 0 \end{bmatrix} \end{array} \tag{3.6}$$

$$\begin{array}{c} \textit{wht-why} \\ \begin{matrix} cr_1 \\ cr_2 \\ \vdots \\ cr_l \end{matrix} \end{array} \begin{array}{c} \begin{matrix} r_1 & \cdots & r_n \end{matrix} \\ \begin{bmatrix} 1 & \cdots & 1 \\ 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 0 \end{bmatrix} \end{array}, \quad \begin{array}{c} \textit{wht-vl} \\ \begin{matrix} cr_1 \\ cr_2 \\ \vdots \\ cr_m \end{matrix} \end{array} \begin{array}{c} \begin{matrix} vl_1 & \cdots & vl_n \end{matrix} \\ \begin{bmatrix} 1 & \cdots & 1 \\ 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 0 \end{bmatrix} \end{array} \tag{3.7}$$

- In the system requirements definition phase, system requirements are derived from customer requirements. And following to that system verification plan (set of test cases) is also planned. In *what-how*(wht-how) matrix sr_i represents the i'th system requirement and cr_j represents the j'th customer requirement. In *how-verify*(vf) matrix vf_i represents the i'th system requirement verification plan.

$$\begin{array}{c} \textit{wht-how} \\ \begin{matrix} cr_1 \\ cr_2 \\ \vdots \\ cr_m \end{matrix} \end{array} \begin{array}{c} \begin{matrix} sr_1 & \cdots & sr_n \end{matrix} \\ \begin{bmatrix} 0 & \cdots & 1 \\ 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 0 \end{bmatrix} \end{array}, \quad \begin{array}{c} \textit{how-vf} \\ \begin{matrix} sr_1 \\ sr_2 \\ \vdots \\ sr_m \end{matrix} \end{array} \begin{array}{c} \begin{matrix} vf_1 & \cdots & vf_n \end{matrix} \\ \begin{bmatrix} 1 & \cdots & 1 \\ 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 0 \end{bmatrix} \end{array} \tag{3.8}$$

- In the architecture and analysis phase system requirements are allocated to a proposed system architecture, architecture specifications are derived from proposed system architecture. And following to that, architecture verification plan is also planned using the architecture specifications. In *how-where* matrix sr_i represents i'th system requirement and sa_j represents the j'th system architecture component. In *where-howfine*(whr-hf) matrix as_i represents i 'th system architecture specification, In *how-vf* matrix vf_j represents the j 'th architecture specification verification plan(set of test cases).

$$\begin{array}{c} \textit{how-whr} \\ \begin{matrix} sr_1 \\ sr_2 \\ \vdots \\ sr_m \end{matrix} \end{array} \begin{array}{c} \begin{matrix} sa_1 & \cdots & sa_n \end{matrix} \\ \begin{bmatrix} 1 & \cdots & 1 \\ 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 0 \end{bmatrix} \end{array}, \quad \begin{array}{c} \textit{whr-hf} \\ \begin{matrix} sa_1 \\ sa_2 \\ \vdots \\ sa_m \end{matrix} \end{array} \begin{array}{c} \begin{matrix} as_1 & \cdots & as_n \end{matrix} \\ \begin{bmatrix} 1 & \cdots & 1 \\ 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 0 \end{bmatrix} \end{array} \tag{3.9}$$

$$\begin{array}{c} \textit{how-hf} \\ \begin{matrix} sr_1 \\ sr_2 \\ \vdots \\ sr_m \end{matrix} \end{array} \begin{array}{c} \begin{matrix} as_1 & \cdots & as_n \end{matrix} \\ \begin{bmatrix} 1 & \cdots & 1 \\ 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 0 \end{bmatrix} \end{array}, \quad \begin{array}{c} \textit{how-vf} \\ \begin{matrix} as_1 \\ as_2 \\ \vdots \\ as_m \end{matrix} \end{array} \begin{array}{c} \begin{matrix} vf_1 & \cdots & vf_n \end{matrix} \\ \begin{bmatrix} 1 & \cdots & 1 \\ 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 0 \end{bmatrix} \end{array} \tag{3.10}$$

- In the detail design phase, design components are allocated to architecture and refined set of design specifications are derived from selected design components. The unit/integration test plan is also planned for the design components. In *where-howfine*(whr-hf) matrix as_i represents i'th architecture specification and ds_j represents the j'th design specification. In *where-finewhere*(whr-fwh) matrix ds_i represents i'th design specification and dc_j represents the j'th design component. In how-vf matrix vf_j represents the j'th design specification test case.

$$\begin{array}{c} \textit{how-whr} \\ \begin{matrix} sa_1 \\ sa_2 \\ \vdots \\ sa_m \end{matrix} \end{array} \begin{array}{c} \begin{matrix} dc_1 & \cdots & dc_n \end{matrix} \\ \begin{bmatrix} 1 & \cdots & 1 \\ 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 0 \end{bmatrix} \end{array}, \quad \begin{array}{c} \textit{whr-hf} \\ \begin{matrix} dc_1 \\ dc_2 \\ \vdots \\ dc_m \end{matrix} \end{array} \begin{array}{c} \begin{matrix} ds_1 & \cdots & ds_n \end{matrix} \\ \begin{bmatrix} 1 & \cdots & 1 \\ 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 0 \end{bmatrix} \end{array} \tag{3.11}$$

$$\begin{array}{c} \textit{how-hf} \\ \begin{matrix} sa_1 \\ sa_2 \\ \vdots \\ sa_m \end{matrix} \end{array} \begin{array}{c} \begin{matrix} ds_1 & \cdots & ds_n \end{matrix} \\ \begin{bmatrix} 1 & \cdots & 1 \\ 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 0 \end{bmatrix} \end{array}, \quad \begin{array}{c} \textit{how-vf} \\ \begin{matrix} ds_1 \\ ds_2 \\ \vdots \\ ds_m \end{matrix} \end{array} \begin{array}{c} \begin{matrix} u/i_1 & \cdots & u/i_n \end{matrix} \\ \begin{bmatrix} 1 & \cdots & 1 \\ 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 0 \end{bmatrix} \end{array} \tag{3.12}$$

- In the implementation plan is mapped to component plan using the fine where-when matrix as shown in below in Eq.(3.13). In where-when matrix dc_i represents i'th design component and im_j represents the j'th implementation plan.

$$\begin{array}{c} \textit{where-when} \\ dc_1 \\ dc_2 \\ \vdots \\ dc_m \end{array} \begin{array}{c} \begin{array}{ccc} im_1 & \cdots & im_n \end{array} \\ \begin{bmatrix} 1 & \cdots & 1 \\ 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 0 \end{bmatrix} \end{array} \tag{3.13}$$

In total with above mentioned different types of traceability matrix are needed for a life-cycle of nine phases for a very fine grained traceability. To achieve very coarse granularity lesser matrices would be required. These matrices provide a comprehensive fine grained, traceability throughout the life cycle and later. The matrices shown by Eq. (3.6) can only be created manually. The matrices shown in Eq. (3.7), (3.8), (3.9), (3.10), (3.11),(3.12) and (3.13) can be created assisted with software module in *SysEngLab* tool.

3.4.3.2 Trace Generation

Trace Generation is basically to find the relationships between the source artifact and the target artifact. Once the trace creation process is fully accomplished. The various trace matrices are stored in an trace repository, which can be shared with all the participants. The trace matrices allow bidirectional traceability at any moment, if all the required matrices exist.

Finding traceability relations between the entities

The previously defined matrices in section 3.4.3.1 can be used to determine the relationships existing between the various requirements artifact.

Derive: Derive relations existing between the customer requirement and system requirement are traced using $what - how$ matrix, every entry in column of matrix means the requirement is derived from the customer requirement lying in the same row.

Refine: Refine relations existing between the system requirements and architecture specifications is calculated using $how - howfine$ matrices.

Allocation: Allocation relationship is calculated using the $how - where$ matrix. Each entry with value '1' represents an allocation.

Satisfy: Satisfy relationship is calculated using the $where - HowFine$ (whr-hf) matrix. Each entry with value '1' represents a satisfy relationship.

Validate: Validation relations exists at top layer of life cycle between the customer requirements and the system validation phase. They are traced using the $wht - vl$ matrix.

Verify: Verify relations exists at three layers of life cycle, between system requirements definition phase and system verification phase, architecture & analysis phase and subsystem verification phase, and detail design and unit/integrated testing. They are traced using the $how - vf$ matrices of their respective phases.

Conflict: Conflict relations represent the potential conflicts arising between the various requirement. The R_R matrix is looked upon and, as the entries with the values equal to '-1' represents inverse proportionality, they are potential for a conflicting set customer requirements. Identified Customer requirements are traced forward to system requirements specification. At any moment of life cycle any artifact can be looked up for conflicts.

Dependency: Dependency relationship between customer requirements can be determined using $wht - how$ matrix, at the end of system requirements definition phase. Between system requirements using $how - where$ at the end of architecture analysis phase. Between architecture specifications using $how - where$ et the end of detail design phase.

Overlap: An overlap relation between two customer requirements or system requirements arises when they share same set of rationales, it can be calculated using the $what - why$ matrix.

Contribute: Contribute relation between customer and requirements is given the *who-what* matrix.

A trace from customer requirement to design component is hence the resultant set of relationships of type derive, refine, allocation, and satisfy. And a comprehensive trace would also include validation and three types of verification relationships.

3.4.4 Trace Maintenance Process

Previously in Figure 3.4, *'maintaining traceability'* was identified as part of traceability process. Figure 3.5 presents the details of *'maintaining traceability'* activity or trace maintenance process as an activity diagram. The trace maintenance process employs techniques to keep the relations between the various artifacts updated and valid. Traceability mechanism is based on the graphical traceability techniques in which artifacts are represented as nodes and traces are links between the two or more artifacts. The root nodes of a traceability tree can be stakeholders or artifacts like: rationales, requirements, etc. Previously Figure 3.3, showed that an artifact can be dead or alive. This concept of dead or alive is used in trace maintenance. The traceability mechanism, has three actions defined: addition, modification, and rejuvenation; they can be applied both on traces and artifacts; there is no deletion operation but instead another sub-operation of modification called suspension. Suspension is envisaged to provide similar functionality like deletion, which permits to keep the track of trace evolution. Each node/artifact maintains two additional lists, one for the dependencies or links, which are pointing to a alive artifact, and one which maintains the names of dead child artifacts.

The proposed concept of dead and alive artifacts, is shown in Figure 3.6 and 3.7. Alive artifact is one which is coherent till date and represents the current state of

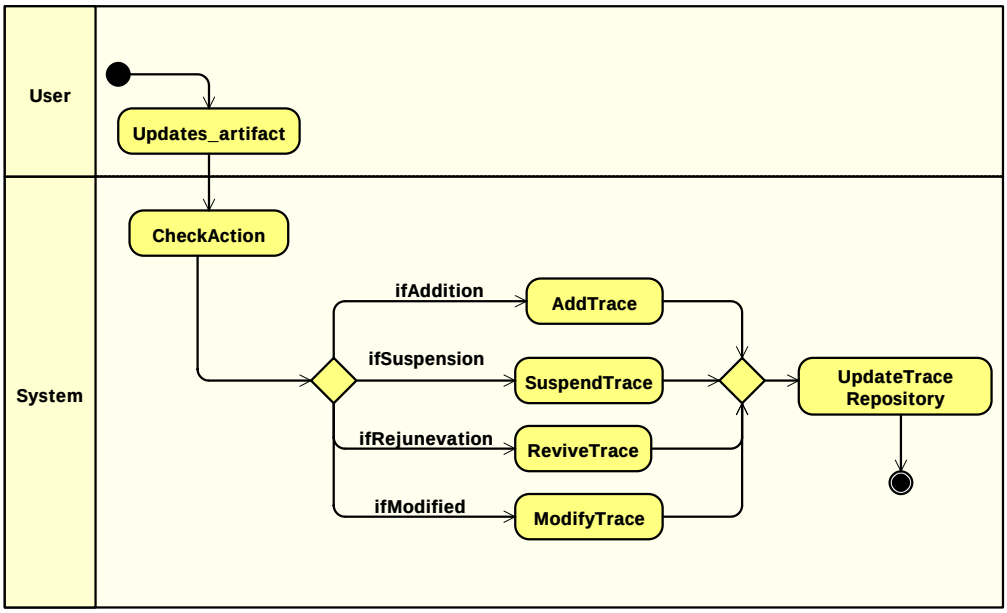


Figure 3.5: Proposed Traceability Maintenance Trace

system development, whereas dead artifact is one which is obsolete with respect to current state of system evolution but still holds information which shows the chronological evolution of system. In our traceability maintenance system, a validated artifact is never deleted completely rather it is simply marked to be dead and is stored as dead artifact linked to its precursor artifact in life cycle.

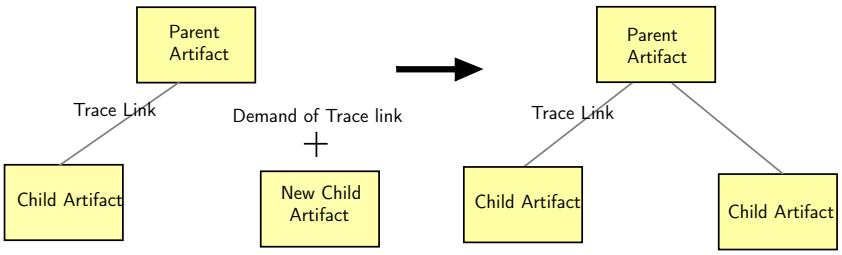


Figure 3.6: Trace Addition [Shukla 2011]

1. *Addition operation.* Figure 3.6 shows the addition operation, when an artifact is created a trace is created pointing from the parent node to the recently created node. All the necessary data are filled and the node is initialized.

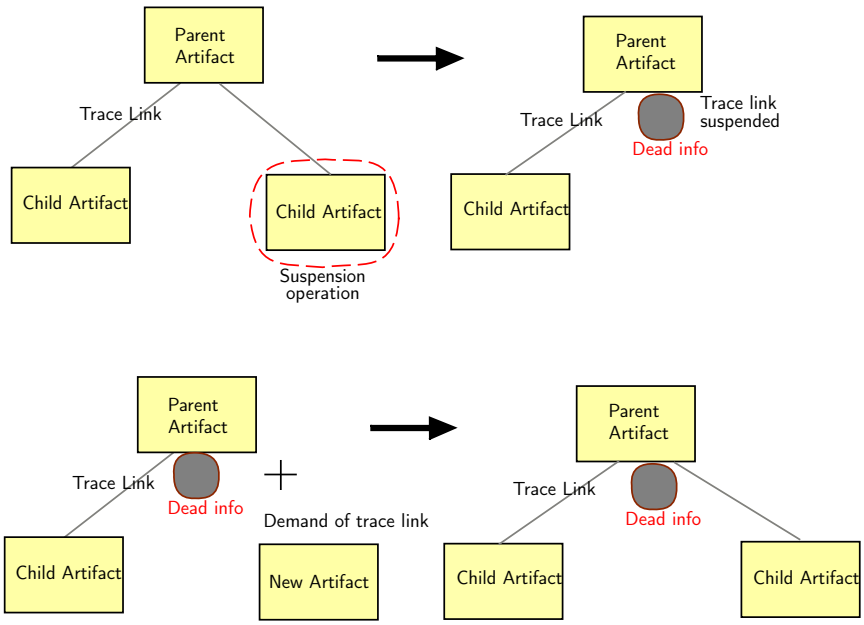


Figure 3.7: Trace Suspension [Shukla 2011]

2. *Modification operation.* Modification operations are of two types change and suspension.

   - *Change.* In case of modification change operation whenever data are updated the earlier existing data are marked dead and the newer ones take their place and are marked alive.
   - *Suspension operation.* Modification-suspension operation is one when an artifact is no longer coherent with the current system state, and user actually wants to remove it, in this case the artifact is marked dead and is suspended and instead of complete deletion from tree it is moved one level up and is added to the list of dead artifacts of corresponding node. Figure 3.7 shows the modification-suspension operation. The other consequence to modification-suspension operation is that all the links from the various other artifacts which were pointing to dead artifact are added to the list of dead pointers.

3. *Rejuvenation operation.* A rejuvenation operation permits to change the status of a trace from dead to alive. This operation can only be applied when all the pre-artifacts to current artifact are alive or controlled i.e., all the earlier artifacts which were the existential reason for the current artifacts should have been taken in account suitably.

3.4.5 Trace Usage

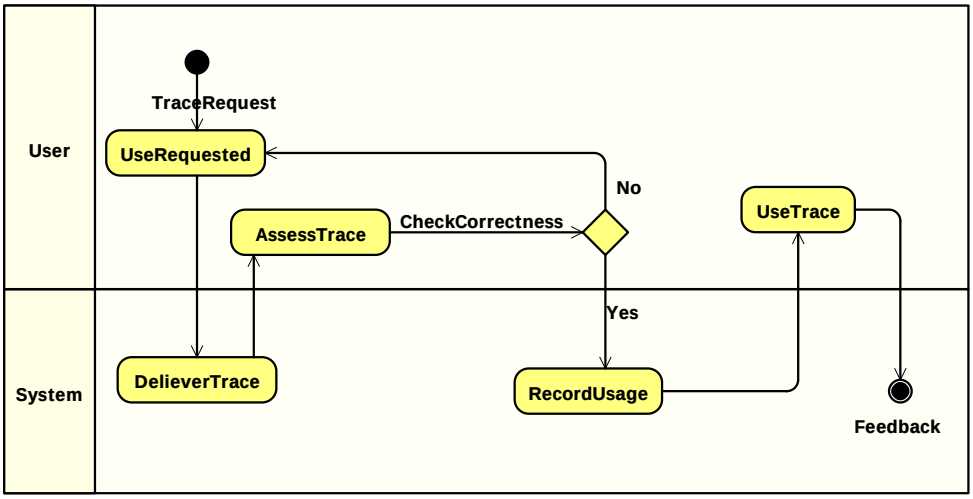


Figure 3.8: Proposed Traceability Usage

Previously in Figure 3.4, *'Using'* activity was identified as part of traceability process. Figure 3.8 presents in detail the the *'Using'* activity. It shows how the traces should be rendered and used for the various activities. Once the trace request is received the trace information is generated according to the trace user and the activity concerned. Then it is checked for conformity, if the required granularity is matched with the users requirements. If the trace information thus generated is suitable for the activity demanded, it is used and its usage record is stored and updated. Otherwise, if the trace information generated is unsuitable a request for trace information generation is send again with requisite granularity for the activity.

3.4.6 Using Traceability Information for SE Activities

The roles of various trace blocks can be defined to solve many other activities during the project life cycle or later during deployment: change impact analysis, configuration management, product upgrades or updates, maintenance, component reuse, etc. Usually, there is vast amount of information, which needs to be properly analyzed before carrying out these activities. A classification of such information helps to carry out these tasks rapidly.

Change impact analysis (CIA) is an activity carried out to measure the ripple effect on the system, brought by some changes in the existing components of the system or because of some system upgrades. The trace table elements can be used to calculate this change impact analysis very efficiently. Primarily, only the *'relationship'*, *'who'*, *'what'*, *'why'*, and *'how'* blocks of a component need to be analyzed to calculate the CIA brought, instead of searching through the length and width of the documents. Upon occurrence of an event, depending on during which phase it does occurs, the corresponding impact of event can be analyzed

using the matrices previously developed in each layer. $What - Why$ matrices allow to detect which requirements can be potentially effected and $What - How$, $How - Where$, $Where - HowFine$ and $Where - FineWhere$ allows to locate precisely the components impacted.

System upgrade or update involves the activities which enhance the functional or non-functional capacity/capability of the system. This is usually carried out after the delivery of the product. Usually, involved with the large size products, for which enhancing capacity of the system is much cheaper than buying a new system with desired capability, for example aircrafts, ships, or buildings. System upgrading is very close to CIA, so trace block involving CIA are needed and additionally *'where'* block.

Product reuse of a component involves studying only the *'why'*, *'how'*, and *'what'* block of a component in the traceability information. The information gathered provides the functionality of the system and its procedure and the various reasons for its usage or recycling. This can also be very helpful in finding the reusable components in the system.

Maintenance process involves the activities, which intend to avoid the system degradation or activities which restore the degraded system state to the ideal or near ideal state, so that all the functionalities expected from the systems can be delivered to user. Maintenance process of a component needs minimally the *'relationship'*, *'what'*, and *'how'* blocks.

Configuration management (CM) is needed for large products, when the different customers need different functionalities out of the system. Their different functional or non-functional product needs are addressed individually. The various functionalities are implemented as various components, which can be integrated to the same system. To carry out these CM activities all the blocks *'relationship'*, *'who'*, *'what'*, 'why', 'where', *'verify'*, *'when'* and *'how'* are needed. As, it is a highly complex task to maintain the different configurations of the same system , the traceability information in form of blocks can provide valuable input to the various CM tools.

Understanding system is necessary for end user from various perspectives to evaluate the system and to quickly assess the system. The data blocks *'relationships'*, *'what'*, *'why'*, *'where'*, and *'how'* are needed to understand the system. Before operationally deploying a complex system, the operator or the user needs to understand completely the system. Minimum two matrices $What - Why$ and $How - Where$ are needed to understand the system.

Prioritizing requirements can also be done using the *'why'*, *'what'*, and *'who'* blocks. These blocks provide sufficient information to give the priority to the requirements using the functionality and rationales.

Measuring project progress can be done using the *'why'*, *'what'*, *'who'*, *'where'* and *'when'* blocks. The progress in project implementation and accountability is achieved using the above mentioned blocks. This can save valuable analyst time in predicting the system completion or failure or delays in system delivery.

Manual generation or **system documentation** is an important activity to

define the various mode of utilization of system, doing this activity necessitates proper understanding of sub-systems, their constraints, etc. The relationships between the various components and their interaction should be known. The *'who'*, *'what'*, *'why'*, *'where'*, and *'relationship'* blocks are needed to create the proper manuals. These blocks can also help to rapidly analyze and measure the adequateness of the manuals or system documentation.

3.4.7 Comprehensive Traceability During Project Development

Considering the Vee-model as shown in Figure 1.3 for the project life cycle. Previously in Chapter 2, it is mentioned that the requirements can be globally categorized into functional or non-functional requirements. Functional requirements lead to the functionalities and non-functional requirements are converted to the functional feature and often account for the magnitude and strength of the functionalities. In Section 3.4.1, comprehensive traceability entities were mentioned as different fields: *'why'*, *'what'*, *'when'*, *'how'*,*'how-fine'*, *'where'*, *'verify'*,*'validate'*, *'who'*, *'relationship'*.

During ***User Requirements elicitation***, the *'what'* block should have data about the functionality that is expected from the system. Eventually, as the non-functional requirements are also implemented as some functionality in the system they should be treated as functional requirements. In the end of 'user needs' level, the user needs are recorded and *'what'* block is filled with the key functionality associated with the corresponding user need. The value of the *'what'* block at this level is singular entity. The *'who'* block value is multiple and contains the name of corresponding stakeholders responsible for the requirement. The *'why'* block gathers the rationale behind the requirements, it can have multiple values. The *'why'* block values are retained throughout the Vee-model at various levels. Hence, this quality can be later used as for linking trace blocks throughout the life-cycle [Shukla 2012b]. The *'where'* and *'how'* blocks at this stage occupy vague values, which the analyst assumes to locate in future. The values can be left unfilled also. The values are filled later at other stages when the features are designed and decisions are made about the components in system. *'validate'* block contains the information about the planned validation test for the user requirement. *'when'* block contains the abstract temporal information about the planned requirement implementation, which is in proportion with the duration of project development. As, at this stage the exact functional requirements are not very precise, the *'relationship'* blocks for them empty, while the qualitative or non-functional requirements of system can have relationships a prior, for example, in case of an auto-mobile 'speed' and 'body weight' can have a prior relationships of conflict and dependency.

At the ***System requirements*** derivation level all the user requirements functional or non-functional are converted in to functional system requirements. The 'What' block of each requirement should mention the name of operation that is expected from it. The 'Who' block contains the name of the user requirement which needs the functionality. The *'why'* block gathers the rationale behind the system

requirement, which is similar to the higher level user requirement, it can have multiple values. The *'how'* block at this stage still occupies the vague values or null, they can only be filled at later stage. The *'where'* blocks still occupies vague values. The 'When' blocks at the end of systems requirements phase contain the temporal information about the implementation of the operations. The *'verify'* block contains the information about the planned validation tests for the system requirements. The 'Relationship' at this level start to be organized, the refinement relations are analyzed for various user requirements and system requirement, 'Who' block is used to determine the various relationships.

As, at the ***Architecture and analysis level***, more fine work about requirements is carried out, various solutions and concepts are proposed for various implementations of requirements. At this stage every artifact introduced or produced during the development process is identified with corresponding requirements, therefore the *'Who'* block in this level can have multiple values. The 'Who' block contains the information of the various system level requirements which lead to the artifact. Similarly, the *'Why'* block at this level can have various rationales linked to the artifact with various system requirements. 'What' block at this level contains the exact functionalities expected from the component. As, components at this stage start falling in to shape, an approximate structure of the system is known by the end of this level. Design engineers start having presumptions about the location implementations of the system. The *'Where'* block, contains the location of the component in the system. The *'Verify'* block at the end of this stage points to the various the sub-system tests. The 'Relationship' block contains the various relationships between the sub-systems at the end of this stage. These relationships are carried up to the higher levels of system requirements and user requirements, and the vague values of the relationships at these levels are replaced by the more suitable values determined by the analysis at architecture and analysis level. At the end of this level the concept selection for various sub-systems component is made and decision-making information is stocked as *'Why'* in various system component levels.

Detail design step starts with detailed inputs from the architecture and analysis stage, the 'Who' block at this stage should contain the information about the refined requirement from the systems requirement stage and the information about the selected concepts at analysis and architecture stage. At this stage the selected concepts are explored and developed. *'why'* block at this stage contains the various rationales with which each component is linked, at this level we know the exact interactions of the components. The *'what'* block contains the exact functionality of each component similar to the *'what'* block at superior level. *'how-fine'* block at this level contain the precise information about the components and their working principles. The *'verify'* block contains the information about the unit and integration tests of the corresponding components which are planned together with the detailed design. The 'Relationship' blocks for the components at this stage evolve as the interactions between the various components are known at this stage; they are filled with respect to various other interacting components (requirements) and are

also propagated to the higher-level verify blocks of analysis and architectural stage, system requirements and user requirements stage. The 'Where' block at the end of this stage contains, the very precise location of the implementations of the requirements in the system. The *'when'* block at this stage contains the precise temporal information about the implementation of the component, it may also contain the temporal dependencies with the other components (requirements). At the end of detailed design the systems engineer has all the necessary data and valid simulation results to go ahead finally with the selected concept.

The ***implementation phase*** is carried based on the results obtained from detailed design. This phase finally takes input from the detail design and other previous phases and executes the project implementation plan. The data in various blocks should remain same as in detailed design stage in *'who'*, *'what'*, *'why'*, *'verify'* and *'where'* blocks. The values in *'when'*, *'how'*, and *'Relationship'* block depend on the project execution plan, the various resource allocation strategies and the temporal relationships between them before execution of the project. Once, the project is executed the, all the desired information is collected and stocked: the mistakes, violation of temporal constraints of the implementation and their reasons, etc.

At the end of the implementation phase, the product is available in form of various sub-systems, following to this, unit and integrated testing is carried out of the product. At this stage we have a system in which, each system requirement is allocated to one or more sub-system. So, the *'who'*, *'what'*, *'why'*, *'verify'*, *'where'*, *'when'*, *'how'*, and *'Relationship'* blocks at this level have practically the same information as of detailed design phase.

Similarly, the ***Sub-systems verification*** has all the trace blocks in accordance with the analysis and architectural plan of the system. The three blocks *'where'*, *'When'* and *'Relationship'* may not contain any value at this stage. They should identify the exact component with the very precise system requirements and the verification process as per the verification strategy used by the team. The *'Who'* block should give the data about the system requirements and the verification strategy, *'What'* block specifies the verification process and *'Why'* provides the rationale behind the system requirement, and *'How'* contains the data about the procedure for verification, *'Verify'* gives the information about verification model proving that verification model is valid.

For the ***System verification*** plan, the trace block contains the similar information as previous sub-system verification plan but at a more abstract level. The trace blocks at system validation plan validate that each user requirement is implemented in to the system.

The *'who'* block at the ***System validation*** plan point to various user needs (functional and non-functional). *'What'* block specifies the validation process and 'Why' block provides the rationale behind the user requirement. *'How'* contains the information about the procedure of the validation plan of the user needs in system. *'Where'* localizes every user requirement in the system. There can be *'relationships'* between the validation plans of different requirements. *'Validate'* block shows the information about the validity of the validation process about the truthfulness and

end user acceptance of validation plan. *'When'* trace block may have void value, or the temporal order of execution of the system validation plan.

3.5 Discussion

The main reason for the failure of many traceability recovery approaches in providing desired quality of requirement traceability can be contributed to their being passive in nature. Passive nature of traceability activity means that they are not being used repeatedly in the system engineering activities. Whereas, regular use of traceability information can benefit a lot in the systems engineering processes.

Another reason for their poor acceptance by the systems engineers in the enterprises is that they are always implemented as a side activity of another major system engineering processes. Currently, the traceability activities in industries begin only after the completion of one phase of the life cycle, whereas our approach does not waits to begin the traceability activity. The traceability activity of a system engineering phase starts with the beginning of the corresponding phase.

Earlier, there were issues with the acceptance of traceability activity by the enterprise personnel, but with the recent arrival of automated and semi-automated tools, there is enthusiasm about it. However, the automated and semi-automated traceability recovery techniques are plagued with other issues, such as: *precision* and *recall*. Unlike the information retrieval based techniques for trace generation, our approach does not have issues with precision and recall of the trace. On the contrary with a suitably integrated software module in design tools, with feeble manual efforts, we can assure very high quality comprehensive traceability even for very large number of requirements.

The manual efforts invested in our approach are never wasted. They are always asset for the development team and for the knowledge base. The knowledge acquired through our approach can also be used later as resource for better decision making for the ongoing projects and the projects to come. The proposed approach augments the status of requirement traceability from merely being the instrument of verification and validation to a more active entity of decision-making and knowledge base synthesis. Our approach is different from other existing traceability recovery approaches. Our proposed technique usage recommends some policies to be incorporated into the system engineering process, which begin as soon as stakeholder identification process.

Although it is very difficult to capitalize upon the social issues linked with the requirements traceability, the presented approach attempts to reduce the burden of traceability in a more organized and systematic manner. The proposed approach is designed to be integrated into the main systems engineering processes by becoming part of it. Up to a certain extent it can provide valuable help, but to expect to have 100% traceability throughout the life-cycle would be naive. More than tooling of traceability other measures have to be designed, which can valorize the effort of analyst and developers during various phases of life-cycle. Certainly, motivation and

environment of work have lot to do with improving the traceability. In principle, the majority of graph-based traceability tools are more or less similar, plagued with similar deficiencies. We would recommend a semi-automatic traceability recovery technique. As, in a fairly large system a fully automatic mechanism can lead to false-positive notifications, which can be errant for requirement engineers. The current traceability mechanisms based on information retrieval (vector space models, latent semantic indexing, and probabilistic model etc.), structural rule-based, linguistically rule based, transformation rule based or other hybrid techniques are still error prone and needs to be improved.

Proposed traceability maintenance technique can be coupled with any traceability recovery technique, and used efficiently. A part of this chapter addresses vital issue of information loss; for example, in a fairly large project which has duration of several years, it is possible that one artifact which was previously decided not to be included in the product owing to a certain constraint, is reintroduced. If the analyst had removed this artifact from system, the information regarding its exclusion was lost which was valuable to the project, and hence it costs again time and money, only to be discovered later regarding its deficiency. We claim that this 'artifact evolution information' is useful and should not be lost whether the decision regarding the artifact is finally affirmative or negative.

The major limitation of event based traceability approach is of scalability: as the number of messages generated passes a certain limit, it becomes difficult to handle so many notifications manually [De Lucia 2010]; even reduced subscription cannot answer this problem. This maintenance problem is addressed by proposed approach, as the cost of maintenance using proposed approach is fairly less as compared to other techniques, as for every artifact updated the information which is obsolete becomes part of the parent node in the form of dead info and the pointing trace is also removed and stocked as dead info with parent node, this eases the work of requirement engineer. In a large project with an event based notification procedure, with our proposed technique, the deletion operation on any artifact would be executed without the overhead of notification and then trace deletion request from lower level artifact owner to a higher level artifact owner.

3.6 Conclusion

This chapter provides a small step towards approaching the traceability challenges. This chapter presented a SE methodology for a few of the traceability problems existing in the literature. The traceability problem is divided into various subproblems: creation, generation, maintenance, cost estimation, etc. Systematic solutions are designed within the framework of pre-requirement traceability and post requirement traceability; necessary and minimum relationships are identified for providing traceability for *pre-requirements* and *post-requirements* traceability. Suitable metrics for modeling the purposed traceability policy and return on investment are provided, which can later be used for decision making process.

The basic idea of our methodology is to make the requirement traceability process more and more reactive and formal with precise semantics of relationships. We provided a trace creation process which can be easily implemented semi automatically. The proposed traceability approach well for both *pre-requirement* and *post-requirement* traceability, with the desired granularity as demanded by the stakeholder or user. An improved approach for traceability maintenance is designed. The proposed traceability model emphasizes on maintenance with efficient maintenance schemes. Maintenance technique provides interesting solution to the dangling trace problem, which can immensely help to reduce the tediousness of tracing process. The Solution offers a plausible solution to the information loss problem as the information ever generated in the development process remains in system to provide the exact trace of evolution of the system.

With the ease in trace maintenance process the cost of maintenance can be reduced noticeably as the dangling pointer problem is solved the effort in maintenance is reduced and hence less time and less human resources are engaged to do the same task. The software tool on which it is implemented is a composite platform for RE&RM, system modeling and decision-making called *SysEngLab*. RE&RM module implements the CReML and the requirement traceability technique.

Presented traceability techniques are currently in use on a SE research project called IBIS at LAAS-CNRS, presented as a case study in Chapter 5. Our approach can be used as a tool for organizing complexity in the composite system design by providing ready to use traceability from previous step. Still, overcoming the social issues of traceability problem remains a challenging task and more research work needs to be done to address the traceability challenges.

Chapter 4

Decision-Making in SE

Contents

4.1 Introduction . . . 105
4.2 Decision Analysis in SE . . . 106
4.2.1 Issues with Decision Making . . . 109
4.2.2 Criteria Weighting Problem in Systems Engineering . . . 110
4.3 State of Art of Decision Making . . . 112
4.3.1 Multi Criteria Decision Making Methods . . . 112
4.3.2 Game Theory Based Conflict Resolution and Negotiation . . . 117
4.3.3 State of art of Criteria Weighting Techniques . . . 118
4.4 Proposed Methodology . . . 123
4.4.1 Roles in Decision making for SE . . . 124
4.4.2 Prerequisite to technique . . . 125
4.4.3 Methodology . . . 125
4.4.4 Optimality Check . . . 131
4.5 Simple Example . . . 131
4.6 Analysis and Comparison With Other Technique . . . 137
4.6.1 Optimality Check . . . 137
4.6.2 With Entropy . . . 137
4.6.3 With Rank Order Centroid . . . 138
4.6.4 With Eigen-vector from AHP . . . 138
4.7 Discussion . . . 139
4.8 Conclusion . . . 140

4.1 Introduction

System design activity begins after the requirements are fixed. Once the requirements are fixed various system behavior models are proposed. The follow-up process of these behavior models is physical allocations to subsystems and components. At this step various problems arise: how to select the best set of subsystems and components to implement the behavior and services demanded by requirements. As there are numerous set of functional or physical solutions proposed, it becomes tedious task to analyze the various available possible configurations for system design.

Often, this is the case of multi criteria decision making (MCDM). In a multi criteria decision making, this step of criteria weighting is very critical for the selection of the right product. Often these criteria are traced-back to a set of multi-disciplinary stakeholders participating in the evaluation process. Usually, these stakeholders differ upon their weighting of this particular set of criteria with other stakeholders. This chapter provides answers to the various question previously raised in Chapter 1, Section 1.4.3.4, which are as revisited below.

Question 1.13: *How to generate weights for a set of criteria for group decision making ?*

Question1.14: *How to remain transparent to all the stakeholders or decision makers in decision process ?*

Question1.15: *How to do group decision making and negotiate ?*

Question1.16: *How to resolve a conflict ?*

Question1.17: *How to provide a requisite level of transparency to the various decision makers ?*

Section 4.2, presents the general decision making process and some issues linked to it. Section 4.2.2 presents the criteria weighting problem. Section 4.3.3 presents the state of art of criteria weighting techniques. Section 4.4 presents proposed approach. Section 4.5 presents an example of our approach. Section 4.7 discusses about the advantages and disadvantages of our approach, and the industrial readiness assessment. Finally, Section 4.8 derives the conclusion .

4.2 Decision Analysis in SE

Howard [Howard 1980] defined decision making in very simple terms as: "*decision making is what you do when you don't know what to do.*" Reformulating words of Howard, Parnell *et al.* [Parnell 2011] defined decision making in context of SE as: "*a decision is an irrevocable allocation of resources.*" Decision analysis is a formalism, a logical procedure for decision making [Howard 1980]. In other words, decision analysis is a structured and formal viewpoint that relates how a course of action would lead to a result[Sage 2000].

In literature there are two popular schools of philosophies in decision making: *alternative focused thinking* (AFT) and *value focused thinking* (VFT). AFT is considered as reactive way of decision making and VFT is proactive way of decision making. Keeney [Keeney 2009] defines value as: "*values are what we care about. They are the principles used for evaluation.*" Value focused thinking stresses upon understanding the values of the stakeholders first and then generating the set of alternatives. This set of alternative is evaluated later upon the criteria representing the values. Decisions can be broadly of two types: *decision problems* and *decision opportunity*. Decision problems are said to be one which arise as result of an external event, whereas decision opportunities are one arising as a result of an internal event necessitating an action. Table 4.1 presents the comparison of AFT and VFT in case of decision opportunities and decision problems.

Table 4.1: Comparing Alternative Focused and Value focused thinking [Keeney 2009]

Alternative Focused thinking	**Value focused thinking**		
For decision problems		*For decision opportunities*	
		Before specifying strategic objectives	*After specifying strategic objectives*
1. Recognize a decision problem	1. Recognize a decision problem	1. Identify a decision opportunity	1. Specify values
2. Identify alternatives	2. Specify values	2. Identify a decision opportunity	2. Create a decision opportunity
3. Specify values	3. Create alternatives	3. Create alternatives	3. Create alternatives
4. Evaluate Alternatives	4. Evaluate Alternatives	4. Evaluate Alternatives	4. Evaluate Alternatives
5. Select an alternative	5. Select an alternative	5. Identify a decision opportunity	5. Select an alternative

Table 4.2: Decision Contexts

Under certainty	**Under uncertainty**			**Conflict-Cooperation**
	Probabilistic uncertainty	*Probabilistic imprecision*	*Information Gap*	
Action results in only one outcome	Several possible outcomes with known probability	Several possible outcomes with unknown probability	Several possible outcomes with unknown probability	Several possible outcomes
Correct decision structural model	Correct decision structural model	Correct decision structural model	Approximate decision structural model	Approximate & unstructured decision model
No parametric Uncertainties, all stakeholder utilities known	Known parametric Uncertainties, all stakeholder utilities known	Unknown parametric Uncertainties, all stakeholder utilities known	Unknown parametric Uncertainties, few stakeholder utilities known	Uncertainties, Assumed stakeholder Utilities, stakeholders objectives may differ

First and second column show how the decision are made when facing a decision problem using the AFT and VFT approaches respectively. Third and Fourth column of Table 4.1 show how the decision opportunities are handled in case when strategic objectives are not defined and in case when strategic objectives are defined.

Mitchell *et al.* [Mitchell 1997] define a DM as a stakeholder which possesses an urgency to find a solution to the dilemma facing the system, the power to select and implement a solution, and a recognized legitimacy by all stakeholders to make this selection. In other words a DM can be defined as a stakeholder who shares a genuine responsibility in a decision opportunity or decision problem.

Essentially, SE itself is value focused, as product developed is for the values of the client or customers. But there are situations where AFT and VFT may find usage simultaneously. In practice, whenever a conflict arises DMs are rushed in, and they might already find alternatives in place to resolve the conflict and take a decision. In such cases AFT and VFT are quite essentially same or have very subtle difference in practice. In a SE project, decisions some time need to be made as a reaction to a situation and most often as proactive way to create value in system.

Decision context in SE can be broadly classified in three types depending upon the environment in which decision is made: decision under certainty, decision under uncertainty, and decision under cooperation or conflict. Table 4.2 summarize the class of decision context upon which decisions are taken. They are compared over the results of action, the decision model, the outcome uncertainties and utilities of the stakeholders. The decision under uncertainties can be sub-classified into three types: under probabilistic uncertainty, under probabilistic imprecision, and under information gap. In SE often decisions are not made in isolation, but in sequence. In some cases, we can wait for the result of a decision to take another decision and in other cases, it might not be possible. This leads decision making in SE to the class of decision under uncertainty. In some cases stakeholders involved in a particular decision may differ upon the objectives to achieve. This leads to forming of groups of stakeholders in conflict.

The decisions under conflict & cooperation are based upon game theory, where the conflict or cooperation occurs according to the difference and agreements upon the objectives of stakeholders [Fang 1993, Sage 2009]. Strategic decision occurring at the higher levels of management can derive benefits by using conflict & cooperation based decision models of game theory [Sage 2009]. The decisions at higher management level are less structured then the decision made for physical design solutions. Aspects related to the strategy of organization is out of scope of this thesis.

Decision making in the SE is a type of multi participant MCDM, in other term multi criteria group decision making. This chapter presents the decision making in the light of evaluation of requirements, alternatives of design component, etc., however technique presented in Section 4.4 may be formulated to use for strategic decisions of organizations.

4.2.1 Issues with Decision Making

Previously, AFT and VFT were presented, it is clear that SE involves both types of challenges of AFT and VFT, but essentially its a VFT problem. Considering SE project as decision opportunity, the issue faced are:

1. Who should provide the value of system ?
2. Is it the design engineer, one who can determine the value of the system who tries his best to design a new system and is the father of this system or is it the end-customer/client who will use this system and will be the future owners of the system ?

There are arguments from both sides designer and developers themselves will hardly use these systems themselves and are little influenced economically or socially by the system they design. On the other hand end-users are the one who will be actually benefited or harmed by the systems they use, they are the one actually influenced economically, socially or politically sometimes through these systems. These arguments mostly fall in favor of end-users. Given the arguments which indicate that its the end-user who has necessity of the system demanded, so the actual system value should be decided by the end-users, as developers are only the means to acquire the system. But than, there are many end-users who will use the same system for different purpose, can they provide the whole value of system ? The system ultimately serves to a more abstract level users like: companies, defense forces, transport authorities, health authorities, countries, or whole world itself. This question might be very hard to answer, but needs to be answered for a successful product to be designed.

Although, decision making is long researched topic and holds high importance in SE during various activities, particularly in selection of design components. There are still multiple issues to tackle such as: how to select the stakeholders for a particular decision ? Once a stakeholder is selected how to measure his stake or importance in decision? Can we assume all stakeholders to have equal stake ? How to handle this fact that some values might be totally irrelevant to one stakeholder and some for other? These questions need to be answered, because a lot depends upon their inclusion or exclusion and on their stake in decision. Even if we ignore these question and look at criteria weighting problem, there are questions like: how to accommodate the various stakeholders preference into one set of criteria ? There are also problems like how to elicit the preference itself ? How much precision should be demanded from stakeholders ? And if the precision demanded is correct for a particular stakeholder or if he can really provide this precision with accuracy and if this can be validated that his preference are actually correct ? Many of these question have roots in the psychology and human behavior and cognitive sciences.

Colin and Eden previously devised a technique for categorizing the stakeholders: *power vs. Interest grid* [Eden 2013]. *power vs. Interest grid* is shown in Figure 4.1. Power and interest grid has been used extensively in various projects for

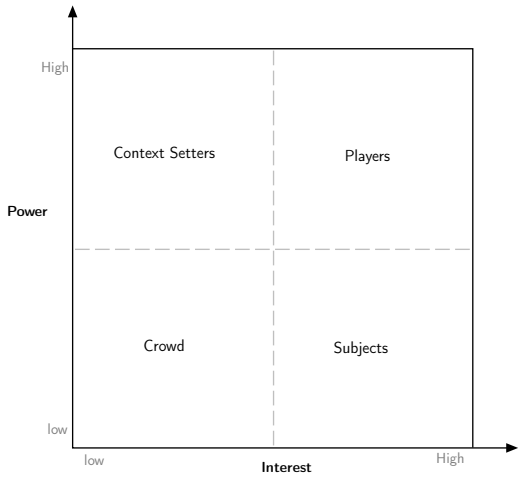


Figure 4.1: Interest Vs. Power Grid [Eden 2013]

stakeholder categorization [Bryson 2011, Eden 2013]. For interested reader various techniques for stakeholder analysis for strategic planning are discussed in [Bryson 2011, Eden 2013, Mahoney 2012].

There are other issues regarding the elicitation of preferences of the stakeholders upon the value previously elicited. There are issues regarding the roles played by the systems stakeholders, who should be responsible to what? There are issues of the transparency of the decision making process how to make the decision process more and more transparent when deciding upon the components. Some of questions raised are of course out of the scope of this thesis but majority of them are researched and some pertinent solutions are designed.

Some questions were previously introduced in the Chapter 1, some are added in this section. The research questions previously were formulated in light of these issues.

The list of additional questions raised is as follow:

Question 4.1. How to select the stakeholders for a particular decision ?

Question 4.2. How to weight the stake of stakeholders in a decision ?

Question 4.3. How much precision should be demanded from stakeholders ?

Question 4.4. Who should provide system value ?

Question 4.5. Who should evaluate alternatives upon these values ?

4.2.2 Criteria Weighting Problem in Systems Engineering

In a systems engineering project, it is of great importance that most of stakeholders are satisfied with the various decisions taken during the product development and with the final resulting end product. A higher satisfaction among the stakeholders

can be guaranteed if the various stakeholders criteria weights are taken into account in a transparent and holistic manner.

The *criteria weighting* is critical part of the MCDM problem during analysis of alternatives in a systems engineering project. Different stakeholders own different views towards the different criteria, their perception of weights differ from each other which makes it very tedious and difficult to come up with an agreement on a particular set of criteria weights. This often causes the conflicts among the stakeholders, which may halt the progress of the project, if it remains unresolved.

The *criteria weighting* problem refers to the problem of generating a single array of criteria weights from multiple arrays of criteria weights emerging from different stakeholders. Often, such problems start with demanding the DMs' weights for all the involved criteria and then provide a mechanism to find the mean weight. This approach assumes that the DMs' know their particular weights, which is often not supported by any evidence. The *criteria weighting* problem faces four critical challenges:

— Evidence of validity for criteria weights.

— Transparency about DMs' participation.

— Scalability of criteria weighting.

— Low Cognitive load on DMs.

Evidence of validity for criteria weights refers to the means which can prove that the assumed criteria weight is correct for a DM or to be able to reason why it is correct for a particular DM. Transparency of DM' participation in criteria weighting is necessary to at least make sure that the criteria weights are not illicitly decided in a manner to favor a particular DM' preferences, or in other words to make sure that every participant DM is satisfied with his bit of contribution in the process. The scalability of the techniques for a sufficiently large number of criteria which may arise in a systems engineering project.

In a systems engineering project the number of broad criteria hardly rise more than ten. These broad criteria later can be divided in to multiple criteria in next level. The *criteria weighting* technique should be able to address this hierarchy of criteria. The fourth most important challenge is about the amount of cognitive load that a technique poses on the DM. If majority of DMs find it difficult to use the technique for weighting the criteria, then the technique has less chances to be used in the process. Whereas, if a technique which poses less cognitive load on the DMs, can easily win over others and find acceptability in the approach, even if it provides less accurate results.

Another aspect of the criteria weighting problem is about the uniform satisfaction among the stakeholders, pointed out by the well known *Arrow's impossibility theorem* [Arrow 1963], which states that DM can find no procedure that can combine individual's rankings of alternatives to obtain single unified rankings. Which is based on some assumptions and states that there is no way to satisfy all the stakeholders completely.

$$\begin{matrix} ST \\ \begin{bmatrix} St_1 \\ St_2 \\ \vdots \\ St_m \end{bmatrix} \end{matrix} \textcircled{?} \begin{matrix} \begin{matrix} c_1 & c_2 & \cdots & c_n \end{matrix} \\ \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m1} & w_{m2} & \cdots & w_{mn} \end{bmatrix} \end{matrix} \Longrightarrow \begin{matrix} \begin{matrix} & & w & \end{matrix} \\ \begin{bmatrix} w_1 & w_2 & \cdots & w_n \end{bmatrix} \end{matrix} \quad (4.1)$$

Eq.(4.1) represents the criteria weighting problem mathematically. In Eq (4.1) on the left hand side St_i refers to i'th stakeholders weight, c_j refers to the j'th criterion, and w_{ij} refers to the weight of the i'th stakeholder for the j'th criterion. On the right hand side w_i refers to the final derived weight of i'th criteria.

4.3 State of Art of Decision Making

4.3.1 Multi Criteria Decision Making Methods

Multi criteria decision making is one of young and very well known branch of decision making. Multi criteria decision making (MCDM) can be divided into two branches multi-objective decision making (MODM) and multi attribute decision making (MADM) [Zimmermann 2001]. However MADM and MCDM are currently used as synonyms and refer to the same class of problems [Triantaphyllou 2000]. MODM focuses on problems with continuous decision space. Whereas MADM concentrates on the problems with discrete decision space. SE projects involve multi criteria decision making (MCDM), and MCDM involves evaluation of various alternative solutions upon a set of criteria. The result of multi criteria decision making is the best alternative which secures the highest score with the predefined criteria. These criteria are often conflicting but demanded by the different stakeholders. Usually, these criteria are weighted in an order to represent their stake in the final selection. Often stakeholders have to make trade-offs their one criterion for another or one value for another. All MCDM methods can be thought of containing following three steps:

- Calculate the weight of the criteria for evaluation.
- Evaluate each alternative solution separately on each criteria.
- Calculate the overall score of each alternative by aggregating the evaluations on each criteria.

Some of the popular MCDM methods are multi attribute value theory (MAVT), multi attribute utility theory (MAUT) [Keeney 1993], utility theory [Fishburn 1970], analytic hierarchy process (AHP) [Saaty 1980, Saaty 1990], analytic network process (ANP) method [Saaty 1996], minimum weighted squares method, weighted sum method (WSM), weighted product method (WPM) [Triantaphyllou 2000]. WSM method is one of the easiest and commonly used approach, based on additive utility assumption, particularly for single dimensional analysis. WPM method like WSM

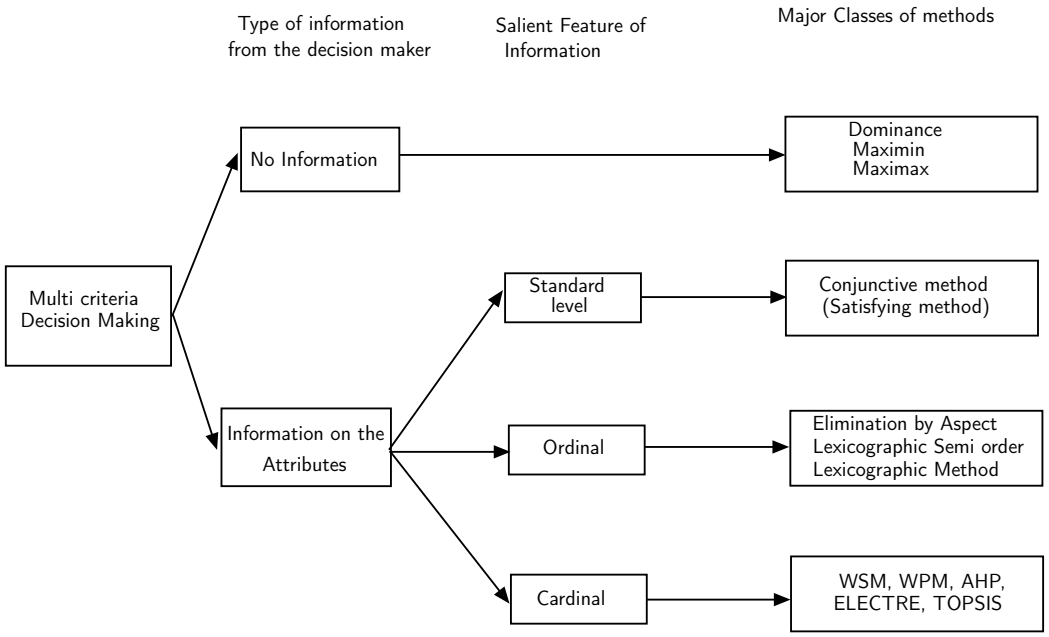


Figure 4.2: Taxonomy of MCDM Methods [Chen 1992]

is simple and easy to use and allows dimensionless analysis. AHP and its various variants have gain much popularity. It uses eigenvector to derive weights for criteria and carry outs pairwise comparisons using a cardinally deployable verbal scale. Inspite of a few of the issues AHP gained popularly for SE, INCOSE handbook uses it one MCDM methods for SE [Haskins 2011].

Reference point method for vector optimization [Wierzbicki 1998], takes into account the goals and targets to be acgived and models the the ideal and anti-ideal reference points to find the solution concepts, this method is used for satisficing problem. Reference point method could provide valuable help in determining the set of possible set of alternatives in AFT approach of SE.

The technique for order preferences by similarity to an ideal solution (TOPSIS) [Hwang 1981] method is relatively close to the principle of reference points method. It is based on a simple principal that the best alternative should have the shortest distance from the positive ideal solution and largest distance from the anti-ideal solution in some geometrical sense. This method can also be used for criteria weighting.

PROMETHEE family of methods [Brans 2005] are another class of outranking methods which are equally popular to for MCDM problems. PROMETHEE is well supported by the decision support system Decision Lab, with a well documented procedure to carry out group decision. PROMETHEE method is adapted to problems where limited numbers of alternatives are evaluated over several criteria.

Measuring attractiveness by a categorical cased evaluation technique (MACBETH) [Bana e Costa 1994, Bana e Costa. 2005] uses pairwise comparison to express strength of preference on semantic scale, for value increments in moving from

performance levels. MACBETH creates scales or indices of value for each criteria involved in the evaluation process, using their range from various alternatives in competition. These indices are dimensionless and with each level they too have a value associated. Evaluation of an alternative requires summing up all the scores an alternative gets, which is product of the weight of criterion and value of index level of alternative on that criterion. This process often leads to a narrow set of alternatives, which with further iterations may eventually lead to best solution.

Potentially all pairwise rankings of all possible alternatives (PAPRIKA) method is another MCDM method [Hansen 2008]. Similar to MACBETH, criteria have value levels. Decision maker is presented with the all possible undominated pairs(a pair of alternatives where one is characterized by higher level of one criteria and another is characterized by higher level of another criteria) defined on just two criteria. Decision maker is asked to provide his preference for each pair which can be strict preference or indifference. Corollaries are generated from a few of these available preferences and ranks of the alternatives are generated. Criteria level values are derived from these rankings using linear programming. PAPRIKA method is essentially a tool for screening problem. Its application to SE could be in selection of set of alternatives for evaluation. Also group decision problem could be challenging with PAPRIKA.

ELECTRE family of Methods [Roy 1968, Roy 1978, Roy 1973] (elimination and choice expressing reality) are also popular in SE applications. ELECTRE I, has improved over time to ELECTRE II, III, IV, Tri. ELECTRE methods use outranking relationships and create the final ranking of alternatives. ELECTRE methods basically analyze the dominance relationships existing between the alternatives. They are particularly good when evaluating large number of alternatives, but can have some issues with contrasting the results. Like PAPRIKA it could be used as a tool for screening alternatives in SE.

The VlseKreterijumska Optimizacija I Kompromisno Renseje (VIKOR) method was developed for multi criteria optimization in complex systems [Opricovic 2002, Tzeng 2002]. It is similar to TOPSIS and ELECTRE method in many ways, as its chooses the solution which has shortest distance to ideal solution and farthest from negative ideal solution in a geometrical space. It determines the compromise ranking list, the compromise solution, and the weight stability intervals for preference stability of the compromise solution obtained from the initial weights [Opricovic 1998]. Unlike vector normalization used in TOPSIS, linear normalization is used in VIKOR.

A few descriptive models of MCDM based on Fuzzy set theory [Zadeh 1965], have been proposed. Fuzzy set theory tries to model the ambivalent perceptions and preferences of the decision makers. In literature multiple hybrid approaches have been devised using Fuzzy set theory such as Fuzzy WSM, Fuzzy WPM, Fuzzy AHP, Fuzzy revised AHP, Fuzzy ANP, Fuzzy Topsis, etc [Triantaphyllou 2000]. Even fuzzy approaches have been used more or less successfully in academia they suffers from some practical issues in their elicitation [Belton 2002]. It could be very hard to actually exploit them in real projects with various stakeholders in SE.

Surrogate worth-trade off (SWT) method [Haimes 1974] helps to choose a preferred solution from a set of non-dominated set of solutions. Previously it has find usage in environmental engineering problems. It is four step process with first step involving non-dominated solutions, second step obtains relevant trade-off information. Third step solicits decision-makers assessment of each objectives improvement and degradation. The best solution is one which remains stable given to all the chances of improvements and degradations. This approach is based on AFT. The way popularly known MCDM techniques are used, it gives an impression that they are used in AFT.

A few of Comparative study of popular MCDM methods are found in literature [Triantaphyllou 2000, Opricovic 2007]. Triantaphyllou devised few metrics for comparison of MCDM techniques. In his simulation based studies MCDM techniques are compared upon two evaluation criteria: ranking inversions, ranking indiscrimination [Triantaphyllou 2000]. His studies assumed that decision makers are rational. Through his studies he tried to answer the question :"*which decision making method should be used to choose the best decision making method ?*" Although, with no concrete conclusion could be derived, his work mentions that decision maker should be vigilant before accepting the results from the MCDM methods, only if they are resilient enough in terms of ranking inversions and ranking indiscrimination. For certain problems one may never know what the best decision is, even if the perfect knowledge of the decision problem and input data are assumed. These studies indicate the choice of MCDM method is strongly depends upon the problem context, and still there is lack of sound theory and method to select the right method for right problem.

More studies need to be carried out based on classes of methods with appropriate problem frames [Opricovic 2007]. Still there is shortage of metrics and validation models to correctly assess a particular MCDM technique. Surprisingly, most of the methods mentioned above are very difficult to use operationally in a real industrial scenario. Except for some which are used for problems such screening patients for surgery, or candidates for immigration in developed countries.

The discussed MCDM methods often consider only one decision maker, which makes it difficult to actually apply the technique in real life scenarios. They mostly assume that the decision maker is rationale. MCDM techniques which derive weights of the criteria for selection in the later stages of decision making such as: PAPRIKA, MACBETH are applicable to situations where the AFT is used.

A few of the discussed methods have been extended to support group decision making, but still there is lack of research which takes into account the transparency and cognitive efforts required for the group decision making using MCDM techniques. Distinction of roles played by different types of decision makers needs to be clarified to know a priori the responsibilities of different decision makers.

Table 4.3, presents a comparison of the various researched MCDM techniques with their strengths and weaknesses.

Table 4.3: MCDM Techniques

MCDM Technique	**Strength**	**Weakness**
WSM	Ease of usage and understanding	Violates additive utility assumption (applicable only to single dimensional cases)
WPM	Dimensionless analysis, ease of usage and understanding	No upper bound in scores cause difficulty in interpreting the scores
AHP	Pair-wise comparison allows precision and accuracy, provides rationale strategy, deterministic	High cognitive load, could be difficult to judge two alternatives by semantic scales, issues with rank reversal, needs consistence information
ANP	Provides rationale strategy, benefits similar to AHP, more structured problem modeling	High cognitive load, Very complex task to make the network, highly cognitive demanding, issues with transparency to stakeholders
MAVT	Provides rationale strategy, allows to take in account risk	High cognitive load, determining value functions could be very difficult
MAUT	Provides rationale strategy, allows to take in account risk	High cognitive load, determining utility functions could be very difficult
TOPSIS	Ease of defining the ideal and anti ideal solution, ease of determining rankings of the solutions	May fail to determine the best alternative, the solution selected may not be the best one as relative distances from ideal and anti ideal are not considered
ELECTRE	Good for analyzing large number of alternatives upon few criteria	Time consuming, may fail to determine the best alternative
MACBETH	Takes into account the qualitative preferences, helps in strategy formulation, inconsistency checks help to avoid errors	Cognitive load, time taking, order of preference elicitation may lead to different results
PAPRIKA	Good for screening problems, generates inferences from the basic preferences elicited, less cognitive load	Higher degree of levels can increase the amount of effort and time required for analysis. Criteria Weights are derived later part of process.
PROMETHEE	Good for evaluating limited number of alternatives,	time taking, selecting the right type of utility function can be subject to errors and inconsistencies

Continued on the next page

Table 4.3 –MCDM Techniques

MCDM Technique	Strength	Weakness
VIKOR	Uses linear preference function, easier in its class of methods, similar as PROMETHEE and ELECTRE methods	Time Consuming process
Fuzzy methods	Allows to represent the qualitative preferences(uncertainty), can handle inconsistent information, allows better precision	Difficulty to elicit the preferences comprehensively,
SWT	Correctness, traceability of preferences	High cognitive load, difficulty for determining low ranked alternatives

4.3.2 Game Theory Based Conflict Resolution and Negotiation

Game theory based techniques evolved to tackle the conflict resolution problem. Normal form of the game model can be used to model a conflict in which each DM chooses a course of action without knowing the choice and course of action of the adversaries [Von Neumann 1937]. A tree structure of the game is called extensive form, with its nodes representing the choices available to the decision maker to move forward or backward. Although it can provide information available in an organized manner, it has some scalability issues.

Graph model for conflict resolution (GMCR) methodology is a further extension of game theory based conflict modeling and resolution techniques. It emerged as a popular technique for strategic conflict resolution [Fang 1993]. GMCR greatly improved the theory and practice of conflict analysis. In the GMCR methodology, each DM has a directed graph that records the unilateral moves available to DM to move the conflict from one state to another. Several contributions were brought forward to handle the various elements of the conflict such as the status quo analysis [Li 2004a], strength of preferences [Hamouda 2004], unknown preferences [Li 2004b], emotions [Obeidi 2005], attitudes [Walker 2009] and fuzzy preferences [Bashar 2012].

Game theory based conflict resolution techniques are suitable to handle conflict at strategic levels. Applying game theory based techniques for conflict arising during product design faces several challenges. Most important of them is transparency of the decisions and decision process available to the stakeholders. Transparency to the stakeholders can not be compromised, when resolving a conflict arising between stakeholders in system design. System design process demands clarity, justifications, trust and strong willingness to collaborate and cooperate. Absence of these attributes only leads to game theory based decision process such as strategic conflicts. This factors limits the application of game theory based techniques in resolving conflicts occurring at system design levels. However, GMCR and other

game theory based techniques have found application in strategic conflicts arising at international levels such as trade disputes, boundary disputes, environmental problems [Fang 1993].

4.3.3 State of art of Criteria Weighting Techniques

In the literature of multi-criteria decision making, often the problem of *criteria weighting* comes along the multi-criteria decision analysis. The vast literature on criteria weighting techniques found mentions of large number of different techniques. Criteria weighting methods can be classified primarily into two types subjective and objective.

Popularly known objective methods are equal weights method [Dawes 1974], rank sum method, rank reciprocal method, rank exponents method [Stillwell 1981], entropy method [Shannon 1949, Shannon 2001], regression method [Albright 2009], variance method, least mean square, standard deviation method [Diakoulaki 1995], criteria importance through inter-criteria correlation(CRITIC), and other methods based on multi-objective optimization techniques such as LINMAP method.

Rank sum, rank reciprocal and rank exponents methods were proposed by Stillwell *et al.* [Stillwell 1981]. Equal weights method requires no additional information except the number of criteria and assigns equal weights to each criteria [Dawes 1974]. Rank sum method allocates weight to the criteria which correspond top their ranks, normalized by the sum of ranks. Rank reciprocal method gives weights based upon the reciprocal of the ranks. The non-normalized weights are given as $1, 1/2, \cdots, 1/n$. Normalized weight are achieved by dividing each term by the sum of these weights. Rank exponent method requires specific knowledge of the exact weight of the most important criteria. Decision maker measures the weight of the most important criteria on 0-1 interval and use it for determining the other weights using the equation Eq.(4.2).

$$w_i = \frac{(N - r_i + 1)^z}{\sum_{j=1}^{N}(N - r_j + 1)^z} \tag{4.2}$$

where, r_i is the rank of the ith criteria. N is the total number of criteria, z is the undefined measure of dispersion in weights called rank exponent. If the $z = 0$ the equal weights are allocated. If $z = 1$, it leads to rank sum weights.

Entropy, in decision theory is a criterion for the amount of uncertainty, represented by a discreet probability distribution, in which there is an agreement that a broad distribution represents more uncertainty than does a sharp one [Shannon 1949, Shannon 2001]. The Entropy method can be used to determine the weights of criteria by the Eqs.(4.3),(4.4).

$$E_j = -\frac{\sum_{i=1}^{m} p_{ij} \ln(p_{ij})}{\ln(m)}, \qquad i \in [1, m], j \in [1, n] \tag{4.3}$$

$$w_j = \frac{1 - E_j}{\sum_{k=1}^{n}(1 - E_k)}, \qquad j \in [1, n] \tag{4.4}$$

The Entropy method is be used to determine the provide an index which represents the variation between the performance index of criteria, it provides a higher score to a criteria, whose values vary most.

Standard deviation method [Diakoulaki 1995] like entropy method allocates smaller weight to the criteria if it has similar importance across various instances of criteria weight. It determines the criteria weight in terms if their standard deviation. This approach is compared to entropy method is less accurate.

CRITIC method is based on standard deviation method [Diakoulaki 1995]. The methods starts with a priori evaluation matrix of alternatives upon a set of criteria with references as ideal values and anti-ideal values. The evaluation is transformed into a matrix of relative scores. Standard deviation is calculated for the vectors in the matrix, which calculates quantified intensity of contrast of each criterion. Linear correlation coefficients matrix is calculated between the previously available evaluation score vectors. This provides the measure of conflict that each criteria is having with rest of others. The two available information are used to generate weights for each criteria by multiplicative aggregation. This is followed by normalization to obtain normal weight summing up to 1. A few case studies are available which use CRITIC method in alternative selection.

Least mean square method like SD and CRITIC also gives higher weight to the criteria which have stronger contrasts in the a priori available evaluation matrix of alternatives. A criteria which receives similar values on various alternatives is awarded lesser weight.

Variance and regression techniques [Wang 1970] can be used to for criteria weighting like other statistical techniques. Not many studies mention the usage of variance and regression methods for criteria weighting. Fundamentally, Standard deviation, CRITIC, and least mean square method give higher weights to criteria which have bigger diversity in the evaluation grid.

Subjective methods take a deeper insight into the problem and often involve deep analysis of the problem. Easiest approaches for subjective weighting in literature mention direct rating and point allocation.

In direct rating and point allocation, decision makers rates directly the weighting criteria upon his analysis of criteria. Direct rating has been used since prehistorical times. Modern studies have focused on direct rating and point allocation from an analytical angle. Point allocation methods involves allocation of points or distribution of upto 100$ to the criteria set. Comparison of two approaches was studied by Bottomley *et al.* [Bottomley 2000, Doyle 1997]. Their studies showed that direct rating and point allocation are different approaches although they look same or minor variant of each other. They lead to different results. Their studies claim that direct rating should be preferred over point allocation.

Simple multi attribute rating technique (SMART) [Edwards 1977] finds mentions in literature as a whole process of rating alternatives, which tries to determine weights in more organized way. It asks decision maker to choose the least important criteria, allocates 10 points to it. The rest of the criteria are allocated points with reference to the least important one. There is no upper limit to the points allocated.

In SWING method [Von Winterfeldt 1986, Edwards 1977] decision makers is asked to allocate the weights in a range to criteria. In the next step the decision maker is asked to select the criterion that he would like to change most from its worst to best value(Swing). This criterion is given the highest points, 100. This process is repeated to all other criteria, hence providing relative weights to all other criteria. The final weights are the normalized scores.

Barron developed the rank order centroid (ROC) method [Hutton Barron 1992] for generating weights directly from the ranking of the criteria set. Roc method was successful in generating reasonable weights without full information about the attributes gathered [Barron 1996a].

Edwards and Barron combined SMART and Swing and called it SMARTS (SMART using Swing). SMARTS was introduced as an improvement over SMART which corrected one previous error of SMART [Edwards 1994] regarding the range of the attribute weighted. They also introduced SMART exploiting rank (SMARTER) method [Edwards 1994, Barron 1996b] based upon the idea of using centroid method. SMARTER method was improvement over SMARTS with lesser participation demanded from the decision maker using surrogate weights from centroid.

Eigenvector method in AHP [Saaty 1980, Saaty 1990] is used for evaluating criteria weights. Verbal preferences over the criteria are elicited and transformed to cardinal values in form of ratios. A pairwise comparison matrix is prepared for the criteria set, where each criteria is compared with others on ratio scales. Based upon this matrix criteria weight can be generated using various techniques such as: arithmetic mean method, least square method, etc. As the individual perception is hardly coherent, the degree of consistency is measured by consistency ratio to check if the comparison is good enough.

Trade off method [Keeney 1993], described by Keeney and Raifa judges criteria in pair. They judge two hypothetical competing alternatives upon two criteria. First alternative has the best consequence level of criteria one and worst consequence level of criteria two and second alternative has worst consequence level of criteria one and best consequence level of criteria two. Next, they measure how much the decision maker is willing to shed the most important criterion in order to change the other one to best level. This approach is very subjective and assumes that the decision makers are rationale.

A recently introduced prescriptive method called CROC [Riabacke 2012] has tried to overcome many of previous issues of criteria weighting problem. First step involves elicitation of the rankings of criteria based upon their perceived importance. The most important criteria is given 100 points. The decision maker expresses the importance of least important criteria in relation to the most important one. In the next step, an analogue visual scale is created and criteria are set on this scale. This visual scale allows to show the relative distances between the criteria on cardinal scale. Then ROC method is used to provide weights to the ordinal weights, augmented with cardinal weights.

Interval methods [Walley 1991] has been devised to handle preferential information in a range. Interval methods [Jiménez 2003, Jiménez 2006] are another class

of methods for criteria weighting in a lesser precise way. A range of possible values is shown as an interval. They are claimed to be best suited for group decision making. As they allow big enough imprecisions between the different individual perceptions [Jiménez 2006].

There are some other subjective techniques such as 5Ws & H technique, which use subset of SWING and SMART techniques for criteria weight generation [Čančer 2012]. They devised a couple of questions for both SWING and SMART methods. First question for SWING is "*which criterion change from the worst to the best level is considered most important ?*" and second question is "*with respect to this change importance, how many points less and how many points are given to other criteria change ?*", first question for SMART is "*which criterion change from the worst to the best level is considered least important ?*" and second question is "*with respect to this change importance, how many points more and how many points are given to other criteria changes ?*" A few other techniques based on the visualization aids for selecting the right stakeholder weight such as: cognitive mapping, fishbone diagrams, mind mapping, etc., have evolved and implemented in multiple software decision support system.

A few other works have tried to address criteria weighting problem from a very different angle Simos *card playing techniques* [Simos 1990, Figueira 2002] associates a playing card with criteria, the participants are asked to rank these cards from the least important to most important. The rank order of playing cards determines the perceived importance. The first criteria in the ranking is least important and last card in the order is most important. If the two criteria are equally important these are given same ranking. To allow associate strong preference between the cards, another set of white cards is used. The participants are asked to put white cards between two successive colored cards. The number of white cards represents the degree of difference between two criteria. The rank positions are divided by the total sum of positions of the considered criteria set. Thus leading to the criteria weight set as normalized score summing up to 1.

There are some other hybrid approaches of many different techniques, but in almost all of them they started with assumption about the particular criteria weight. The combination weighting technique is one of the popular technique which uses principles of multiplicative and additive synthesis to generate the combined weight. Multiplicative synthesis is shown by Eq.(4.5). Additive synthesis is given by Eq.(4.6).

$$w_j = \frac{w_{1j} W_{2j}}{\sum_{j=1}^{n} w_{1j} W_{2j}} \tag{4.5}$$

where w_{1j}, w_{2j}, w_j are subjective weight, objective weight and combination weight of the jth criteria respectively.

$$w_j = k w_{ij} + (1 - k) w_{2j} \tag{4.6}$$

where k is the linear combination coefficient and $k \geq 0$.

The previously mentioned techniques depend completely on human judgment about the preference weighting to get the weight, but seldom human can provide reasoning about the preference weight. To best of our knowledge, in the literature of decision making, there is no comprehensive way to reason the weight of the decision maker, to allocate systematically the weights at the various levels of the preferences. In this respect our work is very different from the previous work. We provide a mechanism to weight the human perception and link it with the mathematical formulations to derive the criteria weight. To have more robust criteria weighting, we have used multiple algorithms to find the weight.

Table 4.4: Subjective Criteria Weighting Techniques

Weighting Technique	**Strength**	**Weakness**
SWING	Structured technique for weighting, normalized cardinal weight, rapid	Cognitively demanding, no measure to validate the correctness, no guidance available for group weighting
ROC	Derives weight with ordinal rankings, without any cognitive load	Weights are mechanical, lower ranking criteria are severely punished, weights generated may not reflect the real preferences, no Guidance available for group weighting
SMART	Structured technique for weighting, cardinal weight	Cognitively demanding, rating inconsistencies, rating depends upon range of criteria value, other issues similar to ROC Based, no Guidance available for group weighting
SMARTER	Structured technique for weighting, cardinal weight	Cognitively demanding, other issues similar to ROC Based, no guidance available for group weighting
SMARTS	Structured technique for weighting, cardinal weight	Cognitively demanding, other issues similar to ROC Based, no guidance available for group weighting
Eigenvector	Ratio weight, measures of consistency known, different participants can provide different matrices, consistency check is available for group weighting	Highly demanding cognitively, hard to validate the preferences, time taking, semantic interpretation of preference phrases can be misleading
CROC	Ordinal ranks, cardinal weights taken into account (through graphical scale) for the difference between the cardinal difference between the ranks	Direct ordinal ranking process is bound to create errors or some inconsistencies, (perhaps impact of different graphical scales needs to be studied, issues with GDN needs to be studied)
Interval	Interval weights, closeness to correctness, less cognitively demanding	Precision could be difficult to obtain,no guidance available for group weighting

Continued on the next page

Table 4.4 –Subjective Criteria weighting Techniques

Weighting Technique	Strength	Weakness
Trade-off	Pairwise comparison, cardinal, Correctness	Highly demanding cognitively, time taking, explicit difficulty for weighting less important criteria, no guidance available for group weighting
Point Allocation	Cardinal, rapid, less cognitively demanding	Possibility of errors,no guidance available for group weighting
Simos card playing	Ordinal, less cognitively demanding	possibility of errors, no guidance available for group weighting
Direct rating	Cardinal, rapid, less cognitively demanding	Could be very challenging cognitively, possibility of errors,no guidance available for group weighting
5W&H	Same advantages as SWING/SMART, more structured,	Highly demanding cognitively, similar challenges like SWING/SMART,no guidance available for group weighting
Rank Sum	Rapid, less cognitively demanding	Possibility of errors, does not takes into account the cardinal differences
Rank Exponent	Rapid	Fairly demanding cognitively, does not takes into account the cardinal difference, prone to errors,

Table 4.4 shows the comparison of subjective criteria weighting techniques available in the literature, discussed previously. Most of the techniques discussed previously are more apt for single person decision making, with an assumption of decision maker as rational. Some of these techniques have tool implementation, like CROC. This chapter does not discusses any of the technique which use weight elicitation directly from uses visual tools, which do not take any account of rational justification.

It is clear that the objective weighting techniques need various vectors or sets of criteria weights to operate upon and determine the final set of criteria weights. The choice of using an objective criteria weight generating technique can hugely affect the final result. Objective weighting techniques have strong mathematical base, that allows to avoid the participation from the decision makers. The problem with the subjective criteria weighting technique is about how to assimilate the various weights obtained from the different decision-makers.

4.4 Proposed Methodology

Different roles in the decision process and prerequisite to the techniques are provided before the technique and an optimality check procedure is defined for the decision process. The proposed approach is a six step process. We assume that all the necessary stakeholders who are decision makers for the particular criteria set are already identified for the system under study.

4.4.1 Roles in Decision making for SE

In first step towards categorizing the stakeholders involve in the SE project. The stakeholders of a SE project could be classified in two classes: client/end-users and developers. Both client/end-user and developer stakeholders play different roles in decision making. Client/end-user are the system value providers or criteria weight determiners and developer stakeholders are evaluators.

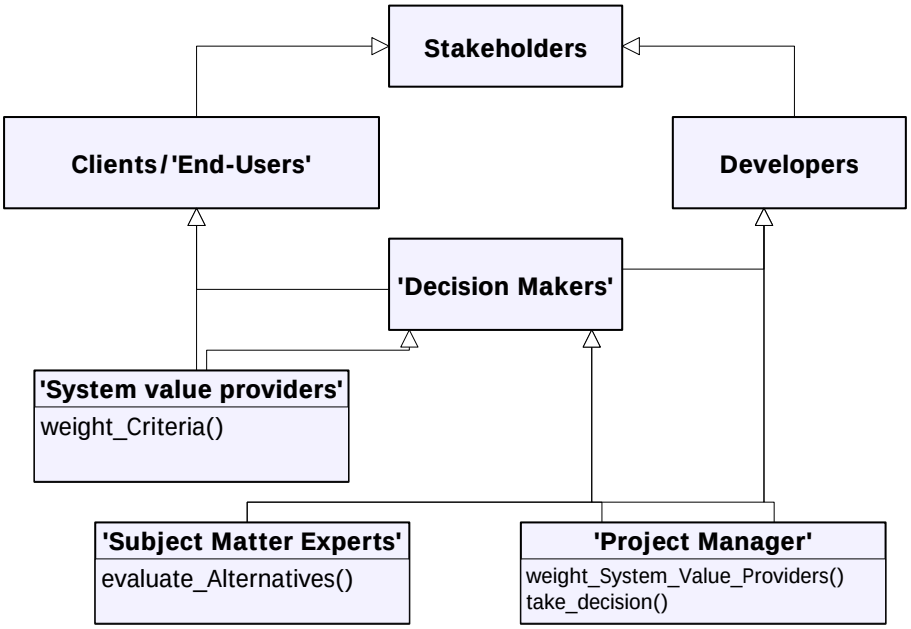


Figure 4.3: Decision Makers Classification

Figure 4.3 shows the proposed classification of SE stakeholders. Criteria weight determiners are the decision makers who would like to provide a measurement of a particular quality in the system upon which evaluation should be carried out. The criteria weight determiners are the general stakeholders which have no interest in the technical specifications rather than quality of product only. Whereas evaluators are the decision makers who have the competence to evaluate the alternative on various parameters and provide a score relevant to their characteristics. These are often the domain experts often known as subject matter expert in SE. Project manager is a type of developer stakeholder who should be responsible for weighting system value provider and take the final decision upon the analysis and evaluation done by subject matter expert. A multi participant MCDM problem in SE can be conceived as two step problem, in first step criteria weights are determined using the preferences of the general decision makers and in second step evaluation by the subject matter experts of the alternatives upon the previously determined criteria set.

4.4.2 Prerequisite to technique

Preference modeling can be done using the classical preferences mentioned in various Utility theories [Fishburn 1970, Roy 1991, Roy 1984, Tsoukias 1992], and in MAVT [Keeney 1993]. Similar preference modeling is also used in game theory based approaches like GMCR techniques [Fang 1993, Hamouda 2004].

- $D = \{d_i\}$, $i\,\varepsilon\,[0, N]$, is the set of decision makers (DMs), d_i involved in the concerned conflict..
- 'C' is the set of distinguishable criteria, satisfying $2 \leq |C| < Z$. For each stakeholder set 'C' consists of three distinct subsets H, M, L, such that: $\{H\} \cup \{M\} \cup \{L\} = \{C\}$, and $\{H\} \cap \{M\} = \{M\} \cap \{L\} = \{H\} \cap \{L\} = \{\phi\}$. Where cardinality of each set can be different.
- For each $d_i \in D$, there is set of preference relationships $\{\gg_{d_i}, >_{d_i}, \sim_{d_i}\}$ defined over 'C'.
  - For each $d_i \in D$, a complete binary relation $\gg_{d_i}$ on C, specifies DM d_i's strong preference over C. If $s, t \in C$, then $s \gg_{d_i} t$ means the DM d_i strongly prefers s to t.
  - For each $d_i \in D$, a complete binary relation $>_i$ on C, specifies DM d_i's weak preference over C. If $s, t \in C$, then $s >_{d_i} t$ means the DM i weakly prefers s to t.
  - For each $d_i \in D$, a complete binary relation $\sim_{d_i}$ on C, specifies DM d_i's indifference over C. If $s, t \in C$, then $s \sim_{d_i} t$ means that s to t are indifferent to DM d_i.
  - The relation $\gg$ and $>$ are asymmetric, $\sim$ is symmetric and reflexive and the triple $\{\gg_{d_i}, >_{d_i}, \sim_{d_i}\}$ is complete.

The proposed approach can be divided into six steps: stakeholder categorization, stakeholder weighting, criteria categorization, preference modeling, score generation and evaluation. They are described in following section.

4.4.3 Methodology

Step 0: Determine Relevant Stakeholders

Decision makers which can provide the system values are analyzed and determined. This first step may differ in different organizations depending upon their policies. Method for determining the stakeholder could not be generalized. This methodology assumes that system value providers or relevant stakeholders are already determined and available.

Step 1: Stakeholder Categorization

In order to carry out multi-participant MCDM, it is essential to determine the importance of the stakeholders who are decision makers for the decision prob-

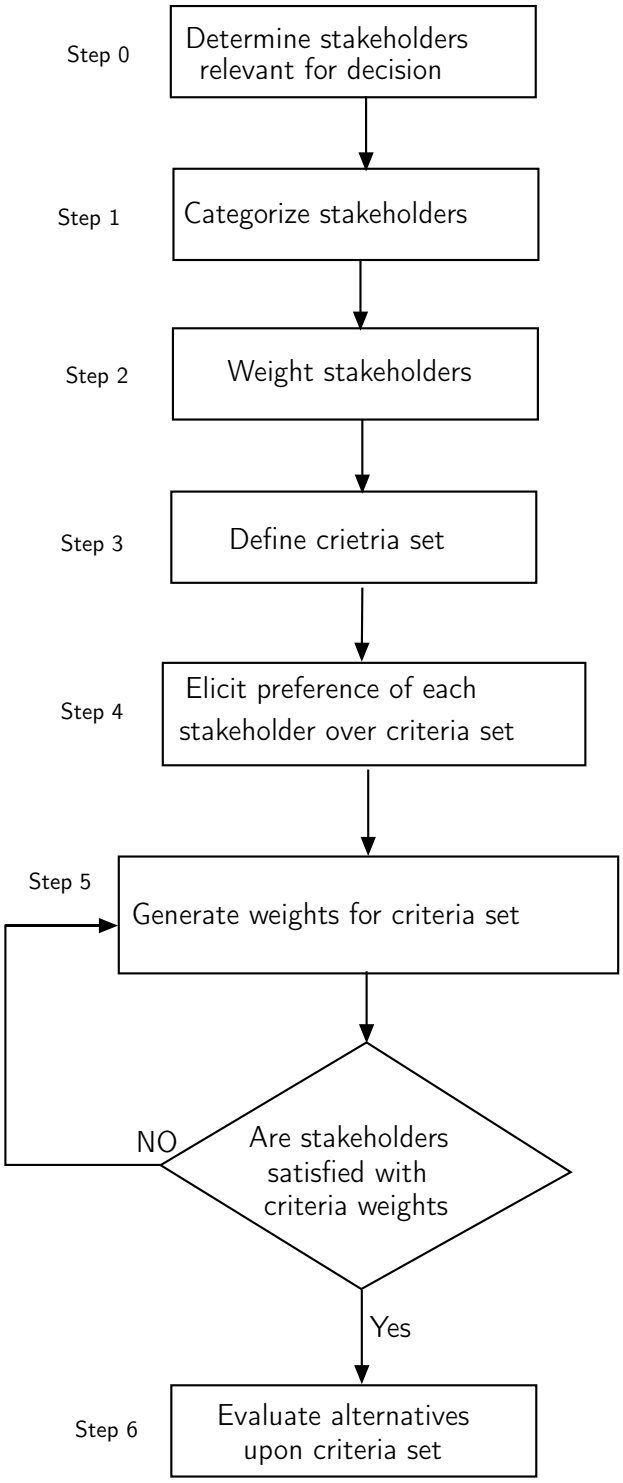


Figure 4.4: Proposed Decision Methodology

lem. Their particular importance in the weighting process should be taken into account. For this purpose a *influence vs. interest grid* is used as shown in Figure 4.5. The *influence vs. interest grid* derives its motivation from the one mentioned in [Bryson 2011, Eden 2013] but is more apt for stakeholder categorization in SE. *Influence vs. interest grid* allows to map stakeholders stake in the decision process. The grid can be divided into five zones which represent the strength of their stake in a particular decision: stakeholder with very strong stake, with strong stake, with medium stake in decision, with weak stake in decision and with very weak stake in decision.

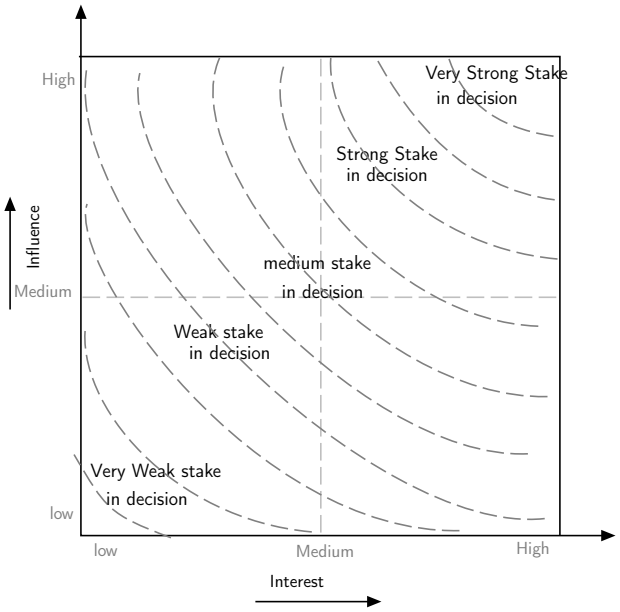


Figure 4.5: Interest vs. Influence Grid in Decisions

Step 2: Stakeholder Weighting

Once the stakeholders are categorized using the *Influence vs. interest grid*, an order of their importance or ranking is available to the project manager or chief decision maker. This ranking order can be converted to cardinal weights using the surrogate weight generation technique based on *decreasing utility functions* as shown in Figure 4.6. For each given ranking a *utility function* is can provide a cardinal weight. The choice of function depends upon the decision maker and may choose to use one or more functions with different degrees of risk averseness, or risk proneness.

Step 3: Criteria Categorization

All the decision makers involved in the decision process should categorize the agreed set of criteria in to three sub sets high preference (H), medium preference (M), low preference (L), according to their perception of utility of the criteria. Hence, the

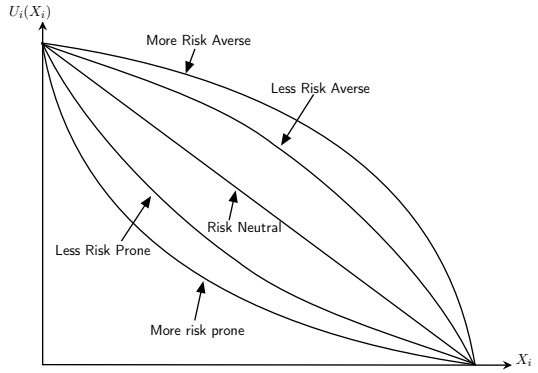


Figure 4.6: Decreasing Utility Function [Keeney 1993]

sorting of criteria is such that the perceived utility is in the order $p(H) > p(M) > p(L)$.

Step 4: Preference modeling over criteria

Each DM $i \in D$, creates the preference matrix P_i, over the criteria $j \in C$, given by Eq.(4.7). The previously created three subsets H, M, L are used as reference for creating the preference matrix in the next step.

$$P_i = \begin{array}{c} \\ c_1 \\ c_2 \\ \vdots \\ c_j \end{array} \begin{array}{c} \begin{array}{cccc} c_1 & c_2 & \cdots & c_j \end{array} \\ \begin{bmatrix} 0 & p_{12} & \cdots & p_{1j} \\ p_{21} & 0 & \cdots & p_{2j} \\ \vdots & \vdots & \ddots & \vdots \\ p_{j1} & p_{j2} & \cdots & 0 \end{bmatrix} \end{array} \tag{4.7}$$

where the value p_{ab} of a DM $i \in D$ is given by Eq(4.8) below:

$$p_{ab} = \begin{cases} 2 & \text{If } a \text{ is strongly preferred criterion than } b \\ 1 & \text{If } a \text{ is weakly preferred criterion than } b \\ 0 & \text{If } a \text{ is indifferent to criterion } b \\ -1 & \text{If } a \text{ is weakly disliked criterion than } n \\ -2 & \text{If } a \text{ is strongly disliked criterion than } b \end{cases} \tag{4.8}$$

As the matrix P_i is skew-symmetric, it is sufficient to express the value of p_{ij} to get p_{ij} and vice versa. Once P_i matrix is available, the criteria are ranked in order with maximum number of $2, 1, 0, -1\ and - 2$ respectively. It is possible for a DM to have two or more criteria securing exactly same ordinal ranking, depending upon the entries in matrix.

Step 5: Generating scores

The scores are generated using a simple marking process consisting of a maximum of j number of moves, where $j = |C|$, in which every DM $d_i \in D$ starts marking the

criteria set C. The marking algorithm is shown as Algorithm 1. In this marking process every DM, d_i starts marking with the most preferred criterion towards the least preferred criterion till all criteria are marked. The result of the marking process is that all criteria are marked in a $j \times j$ matrix. The net score of each criteria by each algorithm is the sum of the total score obtained by each marking at different steps of process. The scores can be generated using the utility algorithms, depending upon

Data: D, C, N
Result: Criteria marked according to their ranks
D=1, N=1;
while $\forall\ i \in C$ *not marked* **do**
  if *Rank of* C_i *is unique* **then**
    mark C_i ;
    N++;
  else
    mark $\forall\ i \in C$ where rank $C_i == C_j$;
    N:=N+2;
  end
end

Algorithm 1: Marking Process Algorithm

the risk averseness, risk neutrality or risk proneness as shown Figure 4.6. There are two approaches to calculate weights from the available ranking from criteria categorization, preference and final ranking:

1. Taking into account only the rankings of the available criteria set and using the different utility functions.
2. Taking into account the degree of preference difference between the various categories of criteria and use different utility functions for generating criteria weights.

The net score of each criteria by each algorithm is sum of the total score obtained at each marking. The scores are calculated using the utility functions. Slight variations in Eq.(4.9), (4.10) and (4.11) with parameters j, d and c_i can generate different risk averse, risk prone and risk neutral algorithms for scoring.

$$score(c_i) = k - j_i \tag{4.9}$$

where k is some constant used as index term, j is some variable dependent on i.

$$score(c_n) = score(c_1) - (n-1).d \tag{4.10}$$

where $score(c_1)$ is maximum score used as index term, d is some constant.

$$score(c_n) = score(c_1) - j_n \tag{4.11}$$

where $score(c_1)$ is maximum score used as index term, j is some variable dependent on n.

Once the scores are calculated they are normalized to obtain the score as shown in Eq. (4.12).

$$Normalised\ score(c_i) = \frac{score(c_i)}{\sum_i score(c_i)} \tag{4.12}$$

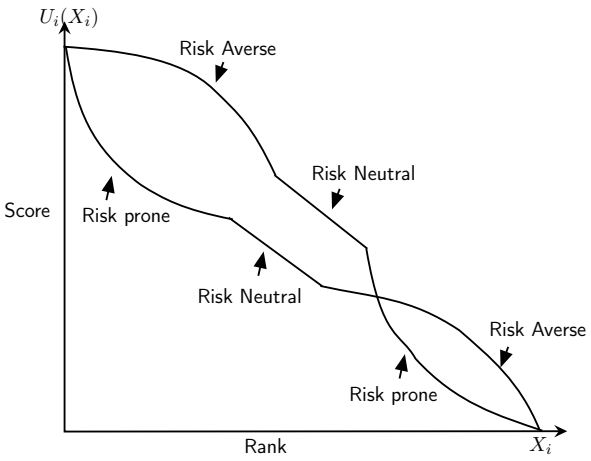


Figure 4.7: Taking in Account Differences in Utilities of Categories

Figure 4.7, shows the two different cases of using different utilities for three categories, in the first case higher criteria are weighted using risk prone, medium using risk neutral and lower using risk prone. In second case higher criteria are weighted using risk averse, medium using risk neutral and lower using risk prone.

Step 6: Evaluating Alternatives on various criteria

Once the criteria weights are available the domain expert are then involved to evaluate the alternatives. As in industrial practice, domain experts use relative scoring techniques based on symbols which often represent a cardinal value such as: $++$, $+$,$+-$, $-$, and $--$, each representing cardinal scores of 100, 80, 60, 40 and 20 respectively [Ulrich 2008]. Domain experts may choose to put precise evaluation scores if they want to provide the precision between two weights of same sign. The alternative with the highest net score is the selected as the best one. Table 4.5 shows the evaluation grid, Alt_i shows the i'th alternative and E_{mn} shows the evaluation score of m'th alternative for n'th criteria.

Table 4.5: Multi Criteria Alternative Evaluation

Criteria	Alt_i	Alt_j	$\cdots$	Alt_m
c_1	E_{i1}	E_{j1}	$\cdots$	E_{m1}
c_2	E_{i2}	E_{j2}	$\cdots$	E_{m2}
$\vdots$	$\cdots$	$\cdots$	$\ddots$	$\cdots$
c_n	E_{in}	E_{j3}	$\cdots$	E_{mn}
Net Score	$\sum E_i$	$\sum E_j$	$\cdots$	$\sum E_m$

4.4.4 Optimality Check

Optimality check can be carried out on the different types of scoring functions. In order to carry out the optimality check, it is necessary to assume that all the stakeholders have equal stake in decision process or have equal weight in decision. We recommend to use the mean distance of criteria rankings as a metric to determine the limits of optimality for a particular utility function. The mean distance is given by the formula:

$$\overline{C} = \frac{\sum_i \Delta c_i}{|C|} \tag{4.13}$$

Where, Δc_i is the difference between the ranking obtained as a result of the scoring function and the actual ranking of a criteria c_i, and $|C|$ is the total number of criteria under consideration. For a given function the limits of of mean distance signify the acceptable ranking distance for each stakeholder, i.e., the mean difference of ranking of DM and resultant ranking obtained, when the DMs' weight is not considered in decision process. A scoring function is said to be $[a, b]$ optimally acceptable, if its minimum acceptable distance $\overline{a}$, is such that $\overline{a} \geq a$ and its maximum acceptable distance $\overline{b}$ is such that $\overline{b} \leq b$.

It should be the responsibility of project manager or a unbiased party to devise the optimality limits for the different types of scoring function during the decision making process. Optimality check technique allows to have an estimate of the degree of satisfaction of the stakeholders with the final scores.

4.5 Simple Example

To show the ease of usage of our technique, we present here an example of Criteria weighting problem. We take an example of a hybrid car. The design team in a automotive industry wants to design a hybrid car and they need to weight design criteria to proceed with the selection of alternatives.

Step 0

The project manager determines five stakeholders which are decision makers for the early phase design concepts: (a) design team, (b) marketing team, (c) end-users representative, (d) safety engineers and (e) maintenance or after sales service team. The set of involved DMs is $D = \{a, b, c, d, e\}$, the criteria set be $C = \{c_1, c_2, c_3, c_4, c_5, c_6\}$. The interpretation of various criteria is explained in Table 4.6. Four design alternatives for the hybrid propulsion system, under evaluation are: petrol-diesel (PD), diesel-gas (DG), gas-electric (GE), and diesel-electric (DE). Their specifications and other details are mentioned in Table in 4.7.

Table 4.6: Design Criteria of a Hybrid Car

Criteria	**Description**
c_1	Flexibility of usage
c_2	Maintainability
c_3	Robustness
c_4	Aesthetic value
c_5	Environment Friendly
c_6	Ease of manufacture

Step 1

The decision makers (DMs), $D = \{a, b, c, d, e\}$ are categorized using the *Influence vs. Interest grid*. The grid is shown in Figure 4.8, which provides DM's stakes in decision. Clearly, the order of importance of their stake is as follow: $c > a > d > b > e$.

Step 2

To convert the rankings into the weights, we use the surrogate weight generation technique based on decreasing utility functions. A risk averse function based on Eq.(4.11) is used to generate weights with $score(c_1) = 80$, $j = 3.2^n$. The resultant weights are shown in the Eq.(4.14) below.

$$\begin{matrix} a & b & c & d & e \end{matrix}$$
$$\begin{bmatrix} 0.238 & 0.18 & 0.258 & 0.22 & 0.103 \end{bmatrix} \tag{4.14}$$

Table 4.7: Hybrid Car

Design-Concepts	**Possible Specifications**
Petrol-Diesel Engine	*Stroke* - 2, 4, 6, Wankel *Volume* - 800cc, 1000cc, or 1200cc *Engine Cooling* - Air , Water , Oil
Diesel-Gas Engine	*Stroke* - 2, 4, 6, Wankel *Volume* - 800cc, 1000cc, or 1200cc *Engine Cooling* - Air , Water , Oil
Gas-Electric Engine	*Stroke* - 2, 4, 6, Wankel *Volume* - 800cc, 1000cc, 1200cc *Engine Cooling* - Air , Water , Oil *Electric Power* - 55hp, 60hp, 65hp, 70hp *Type* - Series wounding DC Motor, Permanent Magnet DC Motor, 3-Phase Induction Motor
Diesel-Electric Motor	*Stroke* - 2, 4, 6, Wankel *Volume* - 800cc, 1000cc or 1200cc *Engine Cooling*- Air , Water , Oil *Electric Power* - 55hp, 60hp, 65hp, 70hp *Type* - Series wounding DC Motor, Permanent Magnet DC Motor, 3-Phase Induction Motor

DM	H	M	L
a	c_1, c_5	c_2, c_4	c_6, c_3
b	c_4, c_6	c_1, c_2	c_3, c_5
c	c_3, c_4	c_2, c_1	c_5, c_6
d	c_6, c_3	c_2, c_1	c_4, c_5
e	c_2, c_3	c_1, c_6	c_5, c_4

Table 4.8: Design Criteria categorization

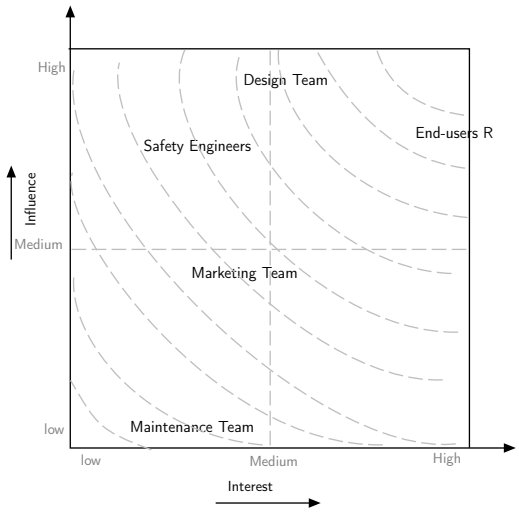


Figure 4.8: Categorization Using Interest vs. Influence Grid

Step 3

The stakeholders (DMs) which are system value provider categorize criteria set according to their perception of criteria as shown below in Table 4.8.

Step 4

The DMs create their preference matrices according to their categorization from step 1 . The preference matrix of stakeholder a, b, c and d are shown in Eq.(4.15), (4.16) and (4.17) respectively.

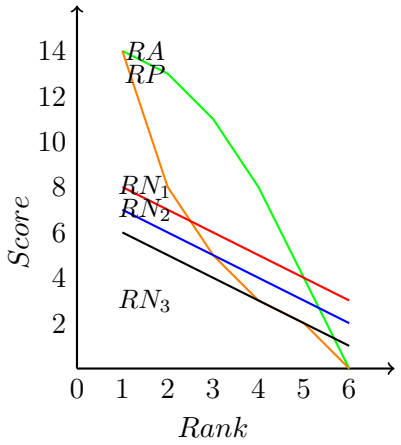
$$P_a = \begin{array}{c} \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \end{array} \begin{array}{c} \begin{array}{cccccc} c_1 & c_2 & c_3 & c_4 & c_5 & c_6 \end{array} \\ \begin{bmatrix} 0 & 1 & 2 & 1 & 1 & 2 \\ -1 & 0 & 2 & 1 & -2 & 2 \\ -2 & -2 & 0 & -1 & -2 & 0 \\ -1 & -1 & 1 & 0 & -1 & 1 \\ -1 & 2 & 2 & 1 & 0 & 1 \\ -2 & -2 & 0 & -1 & -1 & 0 \end{bmatrix} \end{array}, P_b = \begin{array}{c} \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \end{array} \begin{array}{c} \begin{array}{cccccc} c_1 & c_2 & c_3 & c_4 & c_5 & c_6 \end{array} \\ \begin{bmatrix} 0 & 1 & 2 & -2 & 1 & -2 \\ -1 & 0 & 1 & -2 & 2 & -1 \\ -2 & -1 & 0 & -2 & 1 & -2 \\ 2 & 2 & 2 & 0 & 2 & 1 \\ -1 & -2 & -1 & -2 & 0 & 2 \\ 2 & 1 & 2 & -1 & 2 & 0 \end{bmatrix} \end{array} \tag{4.15}$$

$$P_c = \begin{array}{c} \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \end{array} \begin{array}{c} \begin{array}{cccccc} c_1 & c_2 & c_3 & c_4 & c_5 & c_6 \end{array} \\ \begin{bmatrix} 0 & 1 & -1 & -1 & 1 & 1 \\ -1 & 0 & -1 & -1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 2 & 2 \\ 1 & 1 & -1 & 0 & 2 & 2 \\ -1 & -1 & -2 & -2 & 0 & 1 \\ -1 & -1 & -2 & -2 & -1 & 0 \end{bmatrix} \end{array}, P_d = \begin{array}{c} \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \end{array} \begin{array}{c} \begin{array}{cccccc} c_1 & c_2 & c_3 & c_4 & c_5 & c_6 \end{array} \\ \begin{bmatrix} 0 & -1 & -1 & 1 & 1 & -2 \\ 1 & 0 & -1 & 1 & 1 & -2 \\ 1 & 1 & 0 & 2 & 2 & -1 \\ -1 & -1 & -2 & 0 & 1 & -2 \\ -1 & -1 & -2 & -1 & 0 & -2 \\ 1 & 1 & 1 & 2 & 2 & 0 \end{bmatrix} \end{array} \tag{4.16}$$

$$P_e = \begin{array}{c} \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \end{array} \begin{array}{c} \begin{array}{cccccc} c_1 & c_2 & c_3 & c_4 & c_5 & c_6 \end{array} \\ \begin{bmatrix} 0 & -1 & -1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 2 & 2 & 1 \\ 1 & -1 & 0 & 2 & 2 & 1 \\ -1 & -2 & -2 & 0 & -1 & -2 \\ -1 & -2 & -2 & 1 & 0 & -1 \\ -1 & -1 & -1 & 1 & 1 & 0 \end{bmatrix} \end{array} \tag{4.17}$$

Step 5

The DMs play a simple marking process according to their preferences. In this this example we have used five algorithms for calculating the score, one risk neutral and one risk averse and three risk neutral algorithms with slight difference in degree of neutrality.



$$Markings = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} c_1 \quad c_2 \quad c_3 \quad c_4 \quad c_5 \quad c_6 \\ \begin{bmatrix} (a) & (e) & (c) & (b) & 0 & (d) \\ 0 & 0 & (d,e) & (c) & (a) & (b) \\ (b,e) & (c,a,d) & 0 & 0 & 0 & 0 \\ (c,d) & (b) & 0 & (a) & 0 & (e) \\ 0 & 0 & (b) & (d) & (c,e) & (a) \\ 0 & 0 & (a) & (e) & (b,d) & (c) \end{bmatrix} \end{array} \tag{4.18}$$

Figure 4.9: Score vs. Rank

After the marking is over, the DMs agree on a set of score generating algorithm. By using the first score generating algorithm as mentioned in Eq.(4.11), the scores can be obtained. The weights obtained are shown in Table 4.9. Next step is to normalize the score of each criteria using the Eq.(4.12).

Table 4.9: Design Criteria Scores

	Algorithms				
Criteria	Risk-N1	Risk-N2	Risk-N3	Risk-P	Risk-A
c_1	5.99	4.99	4.306	6.18	10.269
c_2	6.02	5.02	4.02	5.56	10.758
c_3	5.76	4.76	3.76	6.556	8.531
c_4	5.62	4.62	3.62	5.738	8.658
c_5	4.31	3.31	2.31	2.626	4.538
c_6	5.26	3.86	3.26	5.305	7.19

After using the five algorithms and normalization we have a set of criteria weights as shown in matrix below by Table 5.14.

Table 4.10: Normalized Design Criteria Weight

	Algorithms				
Criteria	Risk-N1	Risk-N2	Risk-N3	Risk-P	Risk-A
c_1	0.181	0.187	0.202	0.193	0.205
c_2	0.182	0.189	0.188	0.173	0.215
c_3	0.174	0.179	0.176	0.205	0.170
c_4	0.17	0.173	0.17	0.179	0.173
c_5	0.13	0.124	0.103	0.081	0.090
c_6	0.16	0.145	0.153	0.166	0.144

Once the criteria weight matrix is available, the final criteria can be obtained either by the mean of weights obtained for each criteria or by accepting any particular criteria array. In this example we took the mean of the four array of criteria weights and the resultant criteria weight array is represented by Eq.(4.19).

$$\begin{matrix} c_1 & c_2 & c_3 & c_4 & c_5 & c_6 \end{matrix}$$
$$\begin{bmatrix} 0.194 & 0.189 & 0.18 & 0.173 & 0.106 & 0.154 \end{bmatrix} \tag{4.19}$$

Step 6

Once the criteria weight are available the subject matter experts of propulsion system can evaluate and provide the scores of the various alternatives under study. As previously mentioned the weighting notation can vary and each notation may correspond to an cardinal number.

Table 4.11: Multi Criteria Alternative Evaluation

Criteria	$Alt_{(PD)}$	Alt_{DG}	Alt_{GE}	Alt_{DE}
$c_1 = 0.194$	$++(100)$	$+-(60)$	$++(100)$	$-(40)$
$c_2 = 0.189$	$+-(60)$	$--(20)$	$++(100)$	$+(80)$
$c_3 = 0.18$	$++(100)$	$--(20)$	$--(20)$	$+-(60)$
$c_4 = 0.173$	$-(40)$	$+(80)$	$++(100)$	$+(80)$
$c_5 = 0.106$	$--(20)$	$+-(60)$	$++(100)$	$+(80)$
$c_6 = 0.154$	$++(100)$	$+-(60)$	$+-(60)$	$--(20)$
Net Score $\sum w$	73.18	48.46	79.04	59.08

Upon evaluating the various alternatives on the given criteria set, the most suitable alternative for the propulsion system is third alternative Gas-Electric propulsion system, which secured the highest scores during the multi participant MCDM. Table 4.11 shows the evaluation grill withe the various scores obtained by the different alternatives during analysis.

4.6 Analysis and Comparison With Other Technique

4.6.1 Optimality Check

In example mentioned in Section 4.5, if the weight of decision-makers are not taken into account then the ranking orders achieved would be different. Using the risk averse function in Fig 4.9, the ranking obtained while not considering stakeholder weights is $c_2 > c_1 > c_3 > c_4 = c_6 > c_5$, different to actual rankings $c_2 > c_1 > c_4 > c_3 > c_6 > c_5$. In this case the mean distance is calculated for each stakeholder for risk averse function using Eq.(4.13) is: $\overline{C}_a = 1.33$, $\overline{C}_b = 1.667$, $\overline{C}_c = 1.667$, $\overline{C}_d = 1.33$, and $\overline{C}_e = 0.667$. In the resultant limits are found to be [0.67-1.67]. If it lies within the acceptable optimal limits the function is acceptable. For example [0.67-1.67] lies within acceptable limits of [0.5-2.0], hence it is acceptable.

4.6.2 With Entropy

Entropy method of information theory [Shannon 1949, Shannon 2001] is a popular objective method for criteria weights generation, which finds mention in literature. Entropy, in decision theory is a criterion for the amount of uncertainty, represented by a discreet probability distribution, in which there is an agreement that a broad distribution represents more uncertainty than does a sharp one [Shannon 1949, Shannon 2001]. The entropy method can be used to determine the weights of criteria by the Eqs.(4.3),(4.4).

The entropy method is used to determine an index which represents the variation between the performance index of criteria, it provides a higher score to a criteria, whose values vary most. The entropy method is given by the Eqs. (4.3), (4.4). The comparison of results obtained using our approach and the results obtained after applying entropy method can provide more insight about suitability of our approach.

Upon using the Eqs. (4.3), (4.4), the entropy of criteria weights are estimated to be as shown below by Eq.(4.20), which provides a measure of degree of randomness in the criteria weights, and provide a criteria weight according to simple principle of higher entropy leads to higher criteria weight. In our case, the entropy method provides criteria weights which cannot be accepted in a systems engineering project, because it gives unreasonably very high weight to criteria c_5, which was initially least important to all DMs. This examples shows that entropy method based approaches are not suitable for criteria weighting in systems engineering projects.

$$w \begin{matrix} c_1 & c_2 & c_3 & c_4 & c_5 & c_6 \end{matrix} \begin{bmatrix} 0,069 & 0,1072 & 0,033 & 0,148 & 0,516 & 0,126 \end{bmatrix} \tag{4.20}$$

4.6.3 With Rank Order Centroid

Rank Order Centroid(ROC) is a popularly emerged technique for surrogate weight generation technique allows to generate weights for any given number of criteria rapidly. But its disadvantage lie in the disproportionate distribution of criteria. ROC technique is often severe with criterion associated to lower rankings and thus making it unsuitable for a fair trial of alternatives. For example, four given criteria are bound to get weight distribution as 0.52, 0.27, 0.15 and 0.06, thus making first criteria overweighted and leading to unfair distribution.

4.6.4 With Eigen-vector from AHP

With eigenvector based techniques [Saaty 1980, Saaty 1990], it is assumed that the decision-makers are rational and capable of determining perfect ratio of preference between two criteria. Although, theoretically it provides very good results based on the assumptions but often for the stakeholders, providing precise comparison ratio is difficult and mostly a guess. There is no tool to help the stakeholders to determine this ratio. For example it would be very difficult to measure if the ratio of preference between two criteria is 1/6 or 1/7.

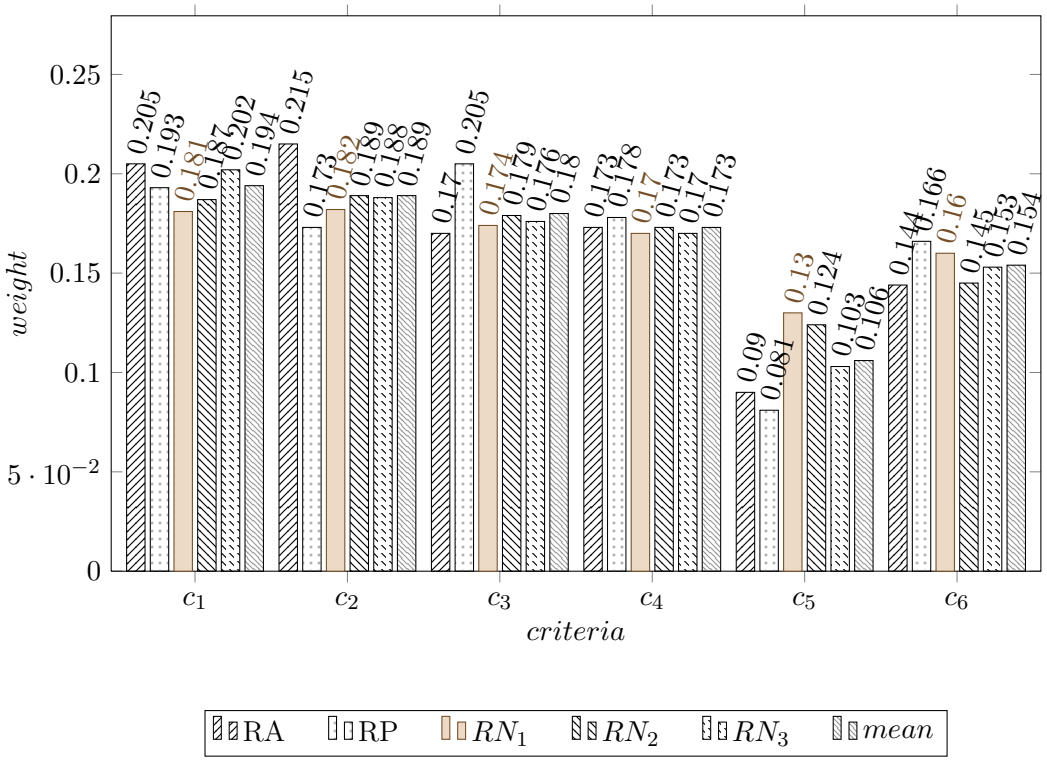


Figure 4.10: Criteria vs. Weight

4.7 Discussion

Figure 4.9, shows the score generating algorithm used in the example and Figure 4.10 shows compares the resulting normalized weights obtained by the algorithms used.

Presented approach tried to provide a methodological solution to one of the ignored problems of MCDM. Most of MCDM techniques mentioned in the literature use these criteria weights which often lack in representing the true nature of preferences of system value providers. Some techniques derive weight in later stages, which may give rise to conflicts between the stakeholders. Often, this is not the case in SE component selection. The proposed technique provides means to allow any number of DMs to achieve an appropriate array of criteria weights, while remaining in synergy with other DMs criteria weights. The trace for validation and transparency is obtained in form of pairwise comparison matrix of criteria. One of the benefit of proposed approach is about the scalability of the technique, it can easily take in account a large number of DMs with different perception of criteria weights and even large number of criteria if needed. But with respect to systems engineering, it would not be fruitful to employ large number of criteria as recommended by the SE good practice guides. The INCOSE SE handbook [Haskins 2011] recommends to have 7-9 criteria in a project. If the number of criteria are too many it would be advised to create a hierarchy of criteria and then use this technique hierarchically.

In addition to ease of application, technique provides other multiple benefits. It helps a DM to understand his perception of criteria better and of the other DMs. It also helps to avoid conflicts among the DMs, which often arrive during the decision making process. The transparency of the approach allows easy negotiation of the criteria weights and hence maximum number of DMs are satisfied with their contribution. The low cognitive load that our technique demands can be important factor for acceptability of the technique. Criteria weighting technique can be coupled with a variety of MCDM approaches, such as PROMETHEE, TOPSIS, WSM, WPM, ELECTRE, AHP, ANP, etc.

The decision makers are free to propose their own utility functions or scoring algorithms based on the ordinal ranking achieved. This allows a more conducive environment for criteria weight negotiation, as the whole process is transparent, with no black-box process involved.

It is possible that the different stakeholders in the projects have different importance. In proposed approach, the project managers can use their preference matrix to generate the DMs' weight, as is the case in a real SE project. Proposed approach offers possibility to weight the different DMs involved in a project while reasoning his priority using a structured framework. For stakeholders weight generation only one preference matrix is required, i.e., only of responsible Project manager. While, for weighting DMs it would be advised to use a set of risk averse, prone and neutral utility functions according to the categorization obtained by them. If there is huge difference between the cardinal preference, risk prone type of algorithm should be used. If they are rather perceived to be closed cardinally risk averse is advised to be

used. Similarly, if they are perceived to be equally separated linearly, risk neutral should be used.

There can be arguments that the utility score generation algorithms can not be accepted as weighting mechanism. But the literature shows that, every individual DM has a utility function, which he uses consciously or unconsciously while providing scores directly as it usually happens. Here we attempted to provide a formalism to use this conscious/unconscious utility function, in order to help other DMs to understand the perceptions of each other. The benefit that our approach provides over other is that, it involves the DMs to methodologically provide the ordinal ranking by first demanding them to attempt to categorize them in three categories. This categorization provides input to the next step; depending on the preference of the various DMs, a range of scoring algorithm can be applied and weights can be obtained.

In the beginning our approach demands slightly more participation from the DMs, as compared to the other approaches, but once the weights are methodologically obtained they are certainly more reliable then the other contemporary approaches with least amount of conflict. Better decisions allow to design the right systems, with more stakeholders getting more confident about their product.

Decision making in SE has to deal with very complex problems, created by the organization and stakeholders itself. A very robust process is required to avoid the unnecessary influence from a particular stakeholder. This is the problem of strategy formulation of decisions in SE. The proposed process is designed to be transparent and keep trace to validate the preferences and take in account at the same time the ordinal and cardinal difference. In literature only CROC allows to do same. While CROC provides no guidelines to take in account the preferences from multiple decision makers (who are system value providers), the proposed approach allows to do so.

4.8 Conclusion

This chapter presented the situation faced by systems engineers when they have to make a decision about choosing a solution for a design element from a set of alternatives. In this chapter, we provide a technique which allows to assimilate the various criteria weights from the different stakeholders to provide a single array of criteria weights which can be uniformly accepted by all the different decision makers or stakeholders. The major contributions are as follows:

— It provides a holistic way to integrate the different criteria weights from different stakeholders to provide a single array of criteria weights using the classical preference modeling.

— It shows that all stakeholders are uniformly satisfied with the proposed technique.

In this current work, we have provided a systems theory of how the criteria weights can be obtained using the classical theory of preference modeling. This approach

provides multiple benefits compared to other existing approaches. Usually in systems engineering project, the engineers rely upon their intuition to provide weights, and later use other technique to combine the different DMs' preferences. Our approach provides a formalism to this systems engineer intuition and hence provides the reasoning for the various weights achieved. Our approach is very easy to understand and use and demands very low cognitive load from the engineers. It allows to formally provide the scores using the DM' drawn utility functions: risk prone, risk averse, or risk neutral; it provides a mechanism to combine them together to come up with a uniformly acceptable solution. In future, we look forward to link the simulation of the DMs' preferences with the design library, in order to shorten the decision time. Our approach can easily be applied to the class of methods which require information on the attributes to carry out a decision analysis.

We provided a methodology to use this *UROW* criteria weighting technique holistically to make a decision based on the value of the system provided by the customer stakeholder (system value provider) executed by the subject matter experts. In order to provide maximum satisfaction from the characteristics and services they required with the quality and constraints they imposed upon the developers. We classified the various stakeholders according to the various roles they play in the decision making of a systems engineering project.

Chapter 5

Integrating Requirements Engineering and Decision Making

Contents

5.1 Introduction . . . 143
5.2 Comprehensive Methodology: Integrating Concepts . . . 144
5.2.1 Methodology . . . 144
5.2.2 Tool Support: SysEngLab . . . 147
5.3 Case Study: Iron Bird Integrated Simulator . . . 152
5.3.1 Assumptions . . . 154
5.3.2 IBIS Stakeholders Needs Elicitation . . . 154
5.3.3 IBIS System Requirements Definition . . . 165
5.3.4 IBIS Architecture Design and Analysis . . . 171
5.3.5 Landing Gear Detail Design . . . 173
5.3.6 Deciding Specifications using our Technique . . . 175
5.3.7 Deciding Design Components . . . 177
5.4 Requirements Traceability . . . 182
5.4.1 Purposed Traceability . . . 183
5.4.2 Cost-effective Traceability . . . 183
5.4.3 Pre-requirement traceability . . . 183
5.4.4 Post-requirement traceability . . . 186
5.5 Limitations and Conclusions . . . 188

5.1 Introduction

Previously in Chapter 2, 3 and 4, we have presented requirements engineering & management and decision making methods respectively. These methods needs to be integrated in order to make them really useful for SE. This chapter provides an approach to combine previously brought contributions together for a comprehensive methodology for requirements engineering and decision making. It also presents the tool and modules developed to support the proposed methodology. In order to demonstrate the feasibility and readiness level of the applicability of our comprehensive approach developed for SE, this chapter presents a case study applied

on a real project called IBIS. The case study allows to have first hand evaluations of the previous contributions and the comprehensive methodology itself.

Section 5.2, presents the comprehensive methodology by integrating the previously developed concepts of RE&M and decision making. Section 5.3, introduces the case study called IBIS and illustrates how the comprehensive methodology can be used for a SE project. Section 5.4 summarize the results on post-requirements and pre-requirements traceability achieved. Finally, limitations and Conclusions of our study are presented.

5.2 Comprehensive Methodology: Integrating Concepts

Figure 5.1, shows the proposed comprehensive methodology in form of Gantt chart. Previously developed concepts are used and integrated in a chronological order of their usage during the development life cycle. The proposed methodology can be divided into eight sections principally. The first six sections involve the RE, design and decision making methods and are of core of our comprehensive methodology mentioned in this chapter. Once the project gets green signal from necessary authority, the project kick-starts.

5.2.1 Methodology

First phase: In the first step primary stakeholder of the system is identified. User stories, interviews and other statements from him are gathered. In the next step the other secondary stakeholders are identified and their statements, user stories, etc., are gathered similarly. Following to the analysis of the stakeholder statements, the context modeling of the system under design is started. Next step involves the weighting of the various stakeholders which are the potential customers of the system. The weighting at this level is carried out to calculate their stake in any decision at the system level, such as decisions determining the missions and services of the product at system level. Traceability between the stakeholders and user stories and statements is maintained as simple table at this stage.

Second phase: First step of the second phase takes as input the previously collected user-stories and other various statements from the stakeholders and identifies the goals (state they want to achieve) of the various stakeholders. In the next step, their rationales are identified and mapped as a graph. Following to this, the stakeholders' values are identified, i.e., system characteristics upon which system goals can be measured, once values are identified their importance is measured using the input from the previous step. Following to this, stakeholders are asked to weight the various system goals identified by the requirements engineers. In the next step the traceability needs of the stakeholders are identified in order to formulate the traceability policies required by the stakeholders and institutions involved with the project. The traceability policies for the project are designed and configured according to the needs of the stakeholders.

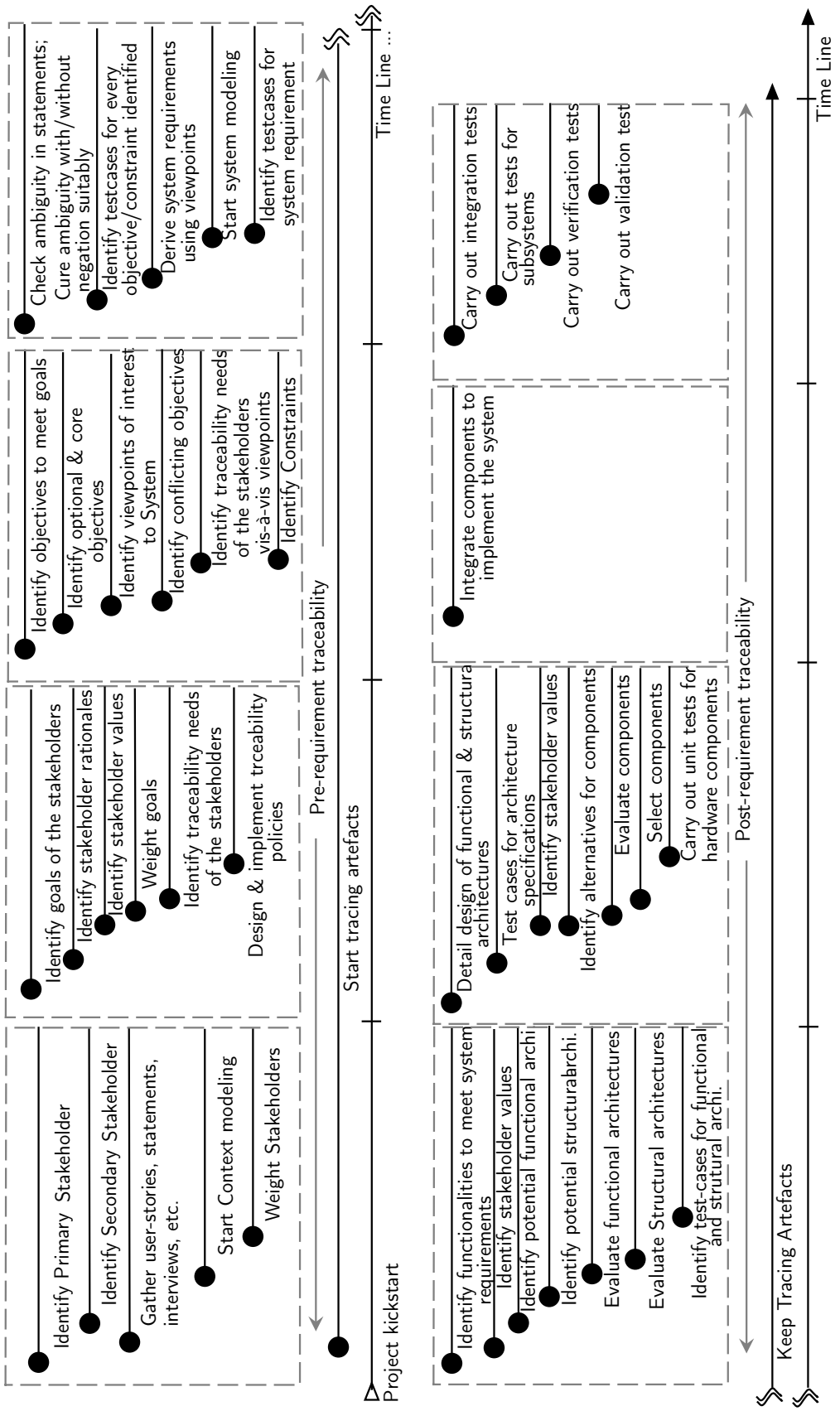


Figure 5.1: Gantt Chart of Proposed Comprehensive Methodology

The traceability process starts for the project according to the policies desired by stakeholders. Requirements artifacts produced till now are traced to the stakeholders and their user stories, statements, etc.

Third phase: Previously identified goals of the stakeholders are taken as input and a set of objectives are identified, which when accomplished permit to achieve the goals. In the next step, the identified objectives are marked as optional or core according to their importance perceived by the requirements engineers and validated later by the stakeholders vis-à-vis constraints, emerging from their side. Next step involves identification of the viewpoints which are of interest to the system, requirement engineers and the stakeholders. Analysis of previously identified objective in the light of identified viewpoints leads to the determination of the potentially conflicting objectives. In the next step traceability needs of the stakeholders vis-à-vis the identified viewpoints is elicited and traceability policies are fine-tuned for each stakeholder. The final step of third phase focuses on identification of the constraints imposed on the system by the stakeholder. Already identified objectives and viewpoints ease this task and previously gathered user stories and statements form the input to this task.

Fourth phase: Fourth phase begins with checks for the ambiguity in the previously produced requirements artifacts and statements. The ambiguous statements are treated and rendered unambiguous by using suitable techniques, such as negation or other techniques mentioned in literature. Design of test-cases can be carried out once the objectives and constraints are treated for ambiguity. An ambiguous objective would be hard to be associated with test-cases. Once the test-cases for the objectives and constraints have been designed, the process of transforming the objectives into system requirements starts. System requirements are derived using the various viewpoints, which allow them to be quantifiable, precise and measurable. System modeling can be started once a few of the system requirements are available, taking inputs from the previously carried out context modeling of the system environment. Test-cases for the system requirements are designed at this stage, once a set of system requirements are available.

Fifth phase: This phase starts with identification of the functionalities demanded by the system requirements previously identified. This is followed by the design of different functional architectures, which can provide the accommodate the desired functionalities. Once the functional architectures are available to the requirement engineers, their evaluation is carried out and a selection is made for implementation. Next, Various potential structural architectures are identified which can support the proposed functional architectures. This is followed by the evaluation of the structural architectures and a selection for final implementation. Finally, test-cases for the functional and structural architectures are identified.

Sixth phase: Detail design of the functional and structural architectures is carried out and precise design specifications are established. Test cases for the design specifications of functions and structures are decided. This is followed by the identification of alternatives for the physical implementation of the components. Once the set of alternatives is available to the designer and subject matter experts

(SME), evaluation of the alternatives/components can be carried out, and final selection are made for the implementation. Prior to physical implementation unit-tests are carried out on the hardware components.

Seventh Phase: All the necessary components are acquired or manufactured and the implementation of the system is carried out during this phase. All the subsystems are integrated to realize the whole system. All the software components are installed and system is readied for the various tests.

Eight Phase: Predefined integration testing is carried out on the system, followed by the subsystem verification, system verification and finally the system validation tests are carried out. Upon successfully passing all the testing procedures the system is ready for delivery.

5.2.2 Tool Support: SysEngLab

A software platform is designed and implemented which supports CReML for RE, traceability technique for RM and the decision making technique for various purposes, the platform is called **SysEngLab**: it is composed of two major components: RE&M module, and decision-engineering module [Shukla 2013].

5.2.2.1 Engineering requirements

The GORE-based notation, previously developed in Chapter 2 is implemented as RE module. This module is also capable of tracing the requirements artifacts produced during the development process. RE module allows to engineer requirements, categorize them based on their types such as: functional, performance, quality, etc. Ambiguity checks for the requirements which are marked for quality can be carried out using various techniques based on previous research of natural language requirements or manual analysis and proposes them to write with negation and requests user for their quantification using metrics of measurements. Eventually, it demands for one or more test case for validation of user requirements with precise condition of acceptance by the end-user or client.The user requirements are hence prepared for a validation plan with at least one test case against each identified user requirement. System requirements are derived following to the user requirements elicitation and similarly to each user requirements, for each derived system requirement atleast one test case is designed for system verification, with a agreed upon environmental context and input data and preconditions and postconditions.

Tagging User Stories with Goals: During the stakeholders requirement elicitation process various techniques are used to elicit the stakeholder requirements. Often, empirical studies have shown that during the first encounter with the clients the needs, desires, and wants are first hand written down in natural languages. The implemented tool allows to create tags for the various user stories based on the goals determined allowing to keep trace of exact origin of a goal in a user story. Often, these requirements represented by various stakeholders also represents the various roles attached to them, which are sometime hidden in the first iteration of

the project. The stakeholder identification process first should be carried out to determine all the potential stakeholders with their all potential roles. With each of their roles there are some potential rationales attached. Requirements are actually projection of these rationales in stakeholders' statements often known as user needs or stakeholder requirements. This information which provides links to the stakeholder requirements and various roles is critical for providing the traceability in the later stages.

Integrated Traceability: Traceability approach as mentioned in Chapter 3 is implemented in the RE module. As previously mentioned, the problem of traceability lies actually the way it is done. Usually, requirement traceability is carried out when it is demanded by the quality control departments, i.e., when it is solicited. Whereas, our proposed tool tracks the links from the very beginning of the goal modeling, every requirement artifact is linked to its parent and child artifact and the root is linked to the user stories. In fact, the responsibility of traceability is shared by the requirement engineer and the tool itself. Tool establishes the links in the order as it is used by the requirement engineer, by using the markings established by him when linking different requirements artifacts with viewpoints and rationales. RE module in our tool allows to model the traceability preference of the system, from the very early stage the system tracks which stakeholder has more affinity to which goal and in which viewpoint. As the goal models can be bridged to some of UML/SysML diagrams (Use-case and Block definition diagram) the traceability is continued to the next stage of system design. Upon demand of a particular stakeholder, he is provided with the traceability of the requirements artifacts which are of his interest or what he demands according to his preference matrix.

Modeling preference In a systems engineering project, it is of great importance that most of stakeholders are satisfied with the various decisions taken during the product development and with the final resulting end product. A higher satisfaction among the stakeholders can be guaranteed if the various stakeholders criteria weights are taken into account in a transparent and holistic manner. Preference modeling of the goals and prioritization of requirements is essential activity, our tools allows to elicit and model both of them over the goal and requirements notations. Unlike goal preference modeling in [Liaskos 2010], our approach takes in account the preferences of group of stakeholders and calculates the net preference of each goal using the integrated decision module, and shows it above the goal notation. Similarly, the core and optional features of the system under study can also be marked to keep track of requirements of a product line.

Including Boiler Plates: The requirements diagram used in goal-modeling are equipped with boiler plates mentioned in [Hull 2011, (RWP) 2012] to help user to write the requirements. The boiler templates aids user to put their capability, capacity, constraint, performance requirements and other type of requirements.

Generation of reports: Our tool supports automatic generation of reports supporting various types of formats. The requirements specification document can be generated in pdf format, user stories can be exported using excel, goal-model, strategy-model and responsibility models can be generated in image forms, trace-

ability information can be generated in form of matrix with demanded parameters.

5.2.2.2 Decision module: Evaluation, Conflict resolution and Negotiation

Decision module of *SysEngLab* is responsible for carrying out the decision process as required at different steps of system design. It implements the decision methodology mentioned in Chapter 4. It is also used in resolving the conflicts arising among different stakeholders, essentially surfacing from different objectives they sought to achieve. A conflict may break out about an action to be taken or not to be taken. *SysEngLab* allows to gather the preferences of the various stakeholders over the different types of requirements artifacts generated during RE activities. Conflicts may arise between two requirements arising from different objectives to achieve. To resolve such conflict the requirements need to be measured and quantified. These requirements are evaluated over a set of criteria such as: cost, benefit, etc., to resolve the conflict. This allows to readily resolve the conflicts and propose choices, to negotiate the design options available to the stakeholders. The stakeholders' preferences regarding the traceability are also provided according to their particular interests and demands and decisions made are stored for future references. But still to validate the preferences elicited from the stakeholders a simulation can be carried out, to confirm their choices for the technologies or macro family of technologies that their preferences lead to.

Simulating preferences for solution: The criteria weighting evaluation is always a difficult step for high level stakeholders who have simply limited knowledge about the technical aspects of system they want. For this purpose a simulation of solutions is proposed by decision module of *SysEngLab*, which could be retained if criteria weights are accepted by the stakeholders. This high level simulation is individually processed for each stakeholder. A stakeholder could display the effect of his/her choice of criteria weights. This simulation is based on data bound to some system components. At the beginning of a new project, Subject matter experts (SME) could draw a very simple organic (or functional) system breakdown. During this step, they can add on each high level component approximative values for criteria. This approach is suitable because SME have skills on available technologies. They don't need to precisely know the features of each components of the real system breakdown. At this step, SME can only categorize the parts of solutions into sets of criteria values, for example they can chose between low, medium or high. The aim of this simulation is to provide stakeholders an approximative solution if they confirm their criteria weighting. Let's illustrate this approach with a very simple example. This one deals with the design of a new powered transport system proposed to city inhabitants for their short distance trips. Let the stakeholders be:

— end users' representative,

— technical engineer of the product development company,

— business engineer of the product development company,

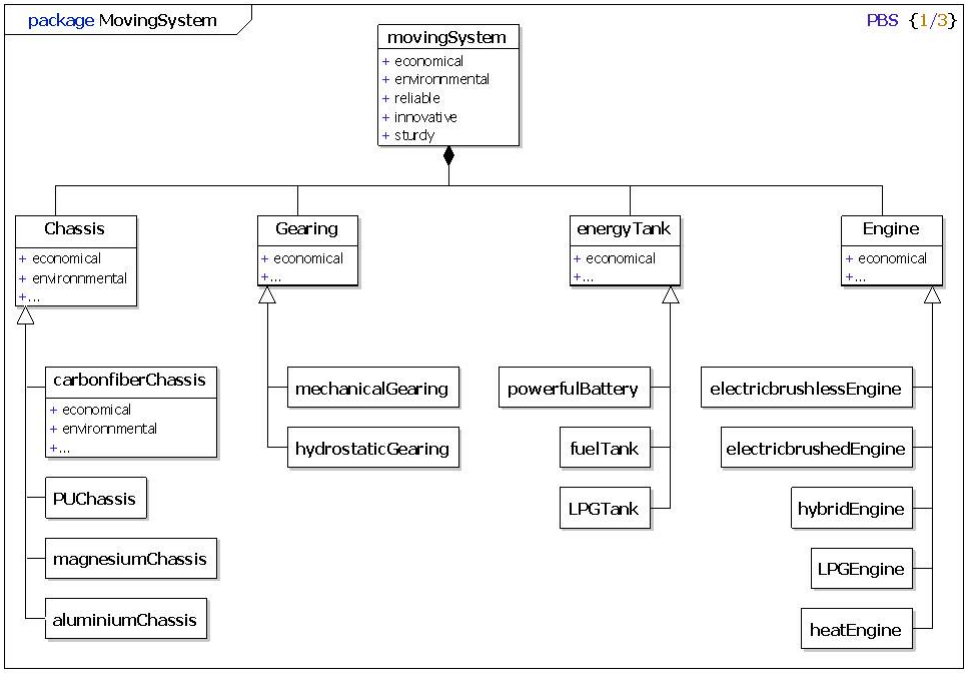


Figure 5.2: Example of Solution Components

— city representative in-charge of city traffic.

In a very simplified way, let the design criteria be:

— economical,

— environmental,

— reliable,

— innovative,

— repairable.

Without knowing the details of solution, design experts have some elements of solution as far as the breakdown is concerned, for example:

— chassis,

— engine,

— energy tank,

— gearing.

For each element of this breakdown, they are able to give some categories of solutions. Fig.5.2 gives an example of partial breakdown.

For each categories of solution, SME could add approximative criteria values. The chassis could be in carbon, magnesium, aluminum or in polyethylene. For each

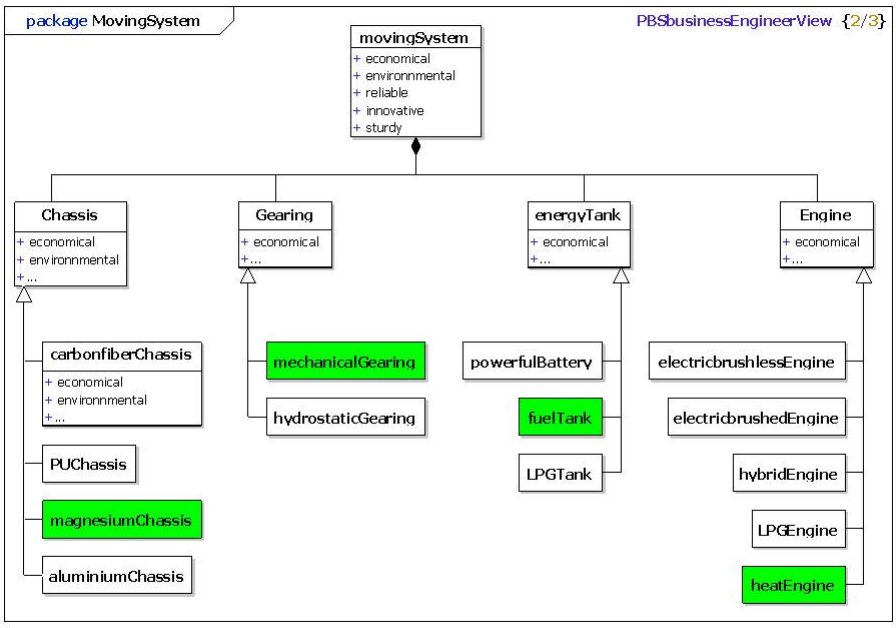


Figure 5.3: Solution According to Business Engineer's Center of Interest

technology, subject matter experts are able to approximatively weight each element of this structure.

For each technology, the SME are able to make a "categorization" for alternative solutions based on the proposed criteria set, without the details of specifications, it is not accurate sizing, but SME know about the advantages and disadvantages of basic technologies involved. This simulation does not attempt to validate a solution but tries to establish a more precise relationship between a criterion and a family of solution. Considering the choice of the two following stakeholders. The sales engineer wants a conveyance which is economical, reliable and robust. He will therefore fill his preference matrix accordingly. He may not have realized that his criteria preference matrix guides the choice solution to polyethylene chassis and using an old noisy and polluting engine.

The ecological criterion may not be a priority for him, but the solution proposed would not be commercially very attractive. It would be difficult to place the product in the market demand. Similarly, considering the mayor of the city, responsible for public transport system. He wants an ecological and innovative solution, that represents modernity and dynamism of the city. He does not realizes that his preference matrix may lead towards the solution of the latest technologies that are likely to be less reliable and expensive to maintain for a product designed for intensive use. In these two examples, we see that the perception of a stakeholder to the criteria is necessarily biased. The head of the city was not opposed to having a robust and

reliable, but he can interpret these criteria with his knowledge, which are a certainly limited on technologies. His perception represents the criteria "robust" and "reliable" with respect to its repository of life for the project in question: a modern car, a scooter, ... Now consider the difference of preferences of the business engineer, they lead to a solution with magnesium chassis, mechanical gearing, with heat engine and a fuel tank as shown in Figure 5.3. The three stakeholders differ a lot completely and sometimes due to partial understanding of the whole cycle of business and product. In such cases, this simulation can aide the stakeholders to really understand each other preferences and validate or refute their preferences. To summarize, this simulation is a mean, proposed to help each stakeholder when choosing values for the weight of importance of the design criteria. Simulation offers the ability to view a draft solution based on criteria elements informed by SME.

5.3 Case Study: Iron Bird Integrated Simulator

Our developed approach is applied on a project called **Iron Bird Integrated Simulator (IBIS)**. IBIS project aims to develop a fully functional platform capable of demonstrating the life cycle of simulation used in a SE project. Engineering schools of Institut Nationale des Sciences Appliquées de Toulouse and Université Paul Sabatier plan to use the IBIS platform later in their curriculum for teaching & research purpose. A few other industrial collaborators had also showed interest in the platform but plan to participate later, once the platform is deployed.

Application of our methodology during case study consists of:

- Application of proposed RE methods
  - User requirements elicitation,
  - Preference modeling and requirements prioritizing,
  - Writing better requirements,
  - Requirements modeling.
- Application of proposed RM methods
  - Creating value based requirements traceability,
  - Traceability creation, recovery, maintenance & usage,
  - Estimation of effort & cost of traceability,
  - Purposed tracing of requirements,
  - Automatic detection of relationships between artifacts.
- Application of decision making and conflict resolution techniques
  - Stakeholders weighting,
  - Criteria weighting,
  - Evaluation of alternatives,
  - Conflict resolution wherever applicable.

To carry out all these mentioned activities, *SysEngLab* is used which implements the proposed methods introduced in previous chapters. *SysEngLab* is explained with all its modules and capabilities in Annexe A. Section 5.3.5 develops the landing gear detail design and component selection for the landing gear, the criteria for selection are weighed and evaluation of alternatives is carried out. In order to understand the IBIS platform, it is necessary to have prior understanding of simulation. A brief introduction to simulation and its life-cycle is presented in the follow up of this section. The US DoD [USDoD 1993] defines simulation as a method for executing a model over time. A simulation is also a technique for testing, analysis or training in which real systems or models representing these real systems are used.

Growing complexity of the systems poses hurdles to know the behavior of the system in its entirety. These complexity problem of systems have led to development of multiple techniques for Verification and validation (V&V) of the systems. One of the popular approach is simulation, which avoids tedious task of carrying out numerous manually applied set of stimuli over the system, while observing its behavior. Simulation consists of building a computer executable model of system and its environment, and exercising test cases upon it, which could be generated both automatically and manually. Simulation helps to reduce development cost and time.

Simulation life-cycle can be divided into four phases [Foures 2013]. The transition from one simulation platform to another requires more or less effort, depending on the type of systems developed, and tests which are required to complete the phase. Figure 5.4 illustrates the chronology of platforms. It is difficult to establish a precise location for each method. Depending on the activities and policies of the development team and the type of system to be deployed, these phases will be forced to slide, and thus appear more or less early in the development cycle.

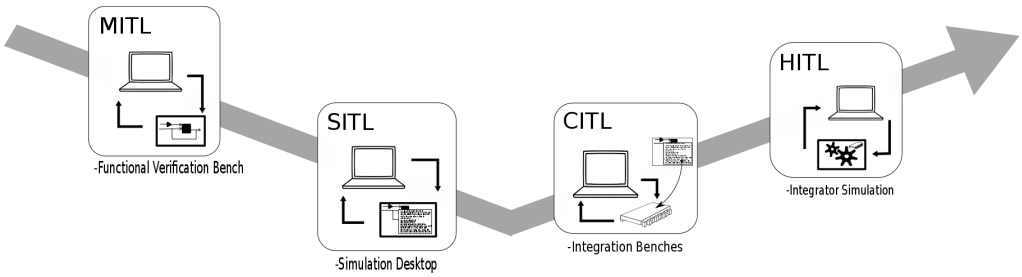


Figure 5.4: Life Cycle of Simulation in SE [Foures 2013]

- MiTL: "Model in the loop" consists of recovering a model describing the behavior of the system to be validated by simulation.
- SiTL: "Software in the loop" consists of recovering a program that implements the model in the target language. This implementation can lead to bias due to model implementation constraints. A new phase of validation and/or verification is required to ensure the semantic equivalence.
- CiTL: "Controller in the loop" consists of validating the behavioural equiva-

lence of the program after controller integration. All control systems is still simulated.

- HiTL: "Hardware in the loop" consists of replacing successively simulated systems by physical systems. The environment remains simulated.

Figure 5.4, shows the life-cycle of simulation from Model in the loop(MITL), to Software in the loop (SITL), to Controller in the loop (CITL) and final stage of simulation Hardware in the loop (HITL). IBIS is conceived to provide a platform to support the life-cycle of simulation based SE for a drone aircraft. IBIS system is supposed to be part of many other activities of research and teaching carried out for the M&S and SE courses at Université Paul Sabatier and Institut National des Sciences Appliquées de Toulouse. In this Chapter, general context of IBIS system is presented. Rest of chapter presents the landing gear system of IBIS in detail and its MiTL and SiTL.

5.3.1 Assumptions

During the study a set of assumptions were made, some of which which come from the clients and were adopted by consensus with other stakeholders. Assumptions used in study are as follows:

1. Given the size of the flightless aircraft, it could require special resources from university and laboratory to host it, which is out of scope of funding constraints.
2. The flightless aircraft simulator is preferred over the flight capable aircraft by the laboratory and other surrounding agencies (assumptions made by the researchers).
3. The visitors shall be interested in interacting with the flightless aircraft and try to understand its role during the development cycle.
4. 'Students' include both the university masters students and the doctoral students (which are also researchers).
5. A platform to carry out the experiments on M&S can contribute significantly to develop the theoretical as well as practical knowledge base of the university.
6. A realistic platform would attract researchers from around the world to test the theoretical formulations of M&S and hence strategic outreach to the research community could be achieved (assumptions made by the researchers).

5.3.2 IBIS Stakeholders Needs Elicitation

In this section, IBIS stakeholders requirements elicitation process is detailed. In the first step stakeholders potentially associated with the IBIS project are identified and their user stories are collected. User stories form the basis of the customers needs. Customers needs are later improved using negation to avoid ambiguity. Validation

tests are designed for the customer needs thus collected. Stakeholders preferences upon the various needs are thus collected and recorded.

5.3.2.1 Stakeholder Identification

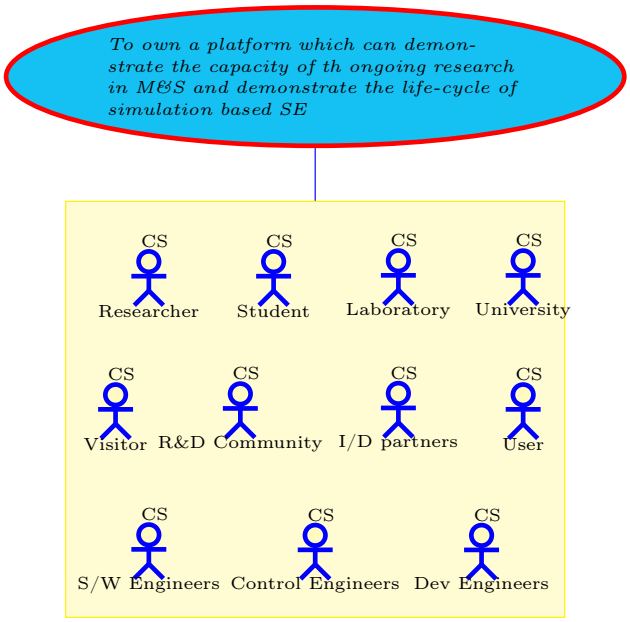


Figure 5.5: IBIS Context Diagram

IBIS projects is a result of vision of the researchers and Ph.D. students working with the complex system simulation, with aspects related to the validation of simulation. To implement and test their theories for validating a simulation they needed a real platform and hence IBIS project was conceived. Figure 5.5, presents the stakeholders initially identified which could be of interest to IBIS. The primary goal identified for the IBIS project is "*To own a platform which can demonstrate the capacity of the ongoing research in M&S and demonstrate the life-cycle of simulation based SE*". Four client stakeholders are identified which are directly link to the IBIS project for either hosting it or as end-user: "*Researcher*", "*Students*", "*University*" and "*Laboratory*". Four other potential stakeholders are identified which may have interest in IBIS: "*Research Community in whole*", "*Industrial partners*", "*Visitors*", and "*Others miscellaneous*". In the first step towards the acquisition of platform in IBIS project, started with gathering of the "User-Stories" as shown in Table 5.1.

Table 5.1: User Stories

Stakeholder Name	**User Stories**
Researchers	Researchers would like to acquire a platform which is capable to demonstrate the life cycle of a simulation, in context of systems engineering. The desired platform should enable him to validate or refute models of the systems. The platform should have all necessary simulation modules for an aircraft system. Researchers want to establish a research axis in the university and associated laboratory based on Simulation based Systems Engineering and use the "Acquired Platform" as the experimental system and case-study during the project work of students. The platform should enable researchers to use it in university for teaching with other professors of engineering domains like, control engineering, architecture design, safety engineering, etc. The platform should provide students the actual know-how of simulation in complex systems engineering. Also the system should be attractive enough for general public and other visitors who visit laboratory during *"Journée porte ouverte"* or *"Fête de la science"*. The researcher wants to use this platform in order to attract the research community's attention and wants other researchers to use their theories and metrics used in systems engineering M&S on the platform. The platform should be able to validate and refute system models put through the simulation. The platform would increase the collaboration between the academic-industrial collaboration, and academic-academic collaboration which surely accounts for quality research publications and also the improvement of quality of education imparted in University. Collaboration with research Community will lead to development of new theories and models of simulation, which can lead to overall large scientific contribution. Also the platform would be attractive for the youngsters and other visitors who are interested in science or wants to pursue a scientific career. ***Roles***: End-user, Maintainer, Configurator
Student	Students would like to work on a real simulation platform and implement the theoretical knowledge imparted to him in university on a real system. He would like to acquire an experience which is useful for his future industrial career. ***Roles***: End-users

Continued on the next page

Table 5.1 —-Continued from the previous page

Stakeholder Name	User Stories
University	University would like to diverse its portfolio of courses on Simulation. A Simulation platform would help to increase the quality of education and education facility in the University. A platform could enhance endogenously the quality of research in domain and overall image of University. This will keep University competitive in general. The platform should be able to be stationed in the system design laboratory with surface area equal to that of shelfs available. ***Roles***: Host-organization
Laboratory	Laboratory would like to host the platform, as it would certainly increases the quality and quantity of research in Simulation domain, which one of the most sought by industries. Platform would enable laboratory to provide better consulting to the industries which are its customer in multiple projects and hence will improve the overall reputation of the laboratory. This platform would enable to bridge the gap between the industrial needs and academic research. The platform should be able to be stationed and remain functional in a closed room. ***Roles***:Host-organization
Industrial partners	An Industrial partner would seek to use a platform which can provide reliable results about the simulation of a model. A platform which could be instrumental in V&V of the models developed for customer application. The platform should be able to guide the user about the modeling choices. ***Roles***: Collaborator
Research Community	Research community would like to participate in developing theories with the platform. Platform could be great help to develop collaboration with the researcher and hence with the University and Laboratory. It could be instrumental in developing new theories in M&S, with real test case. ***Roles***: End-user, Collaborator.
Others	A platform should allow it to attach with reality and demonstrate it purpose in a very intuitive manner. It should be able to attract new young mids for scientific study and research. A visitor may also be interested business opportunity linked with this platform. ***Roles***: Visitor, End-user, Collaborator

5.3.2.2 IBIS Stakeholder Requirements

After the stakeholders are identified, stakeholder requirements are elicited. A few stakeholder requirements are provided explicitly by the stakeholders and find men-

tion in their user stories explicitly or implicitly, such as the technical constraints imposed. The rest of them need to be engineered. First step towards eliciting the customers' needs involves determining their 'rationales'. Rationales can be extracted from the previously collected user stories and interviews from the stakeholder. Figure 5.6 shows the rationale map of the various stakeholders. Rationales are traced to the user stories in Table 5.1. Rationale map aides to create the strategic model of the system in order to guide the system designer for the potential conflict arising owing to the various rationales held by the stakeholders.

Following to the mapping of rationales, IBIS goal model is created. The primary goal is divided in to sub-goals. each subgoal is traced to its source stakeholder as shown in Figure 5.7. Rationales are further contemplated upon till they can provide hint of the viewpoint held by the respective stakeholder. The viewpoints are hence derived from the rationales, which are actually held by the stakeholders. A minimal representation of relationship between viewpoints and rationales relevant to IBIS are shown in the Figure 5.8. It shows three "*Viewpoints*" derived from the previously elicited rationale. Figure 5.9, shows the constraints imposed by the stakeholders on the IBIS project.

The sub-goals are divided into various objectives, which represent the customer needs. The derived objectives for the IBIS system are shown in Figure 5.11. The objectives can be marked as *optional* or *core* as viewed by the corresponding stakeholder. By default, objectives are core until they are marked with an 'O' in top right corner. Objective "*platform shall allows to assess the students*" is marked optional in Figure 5.11, rest of them are core objectives by default. Objectives are typically the customer needs. A set of validation test case is designed for each objective, test-cases provide the contractual base between the developers and clients.

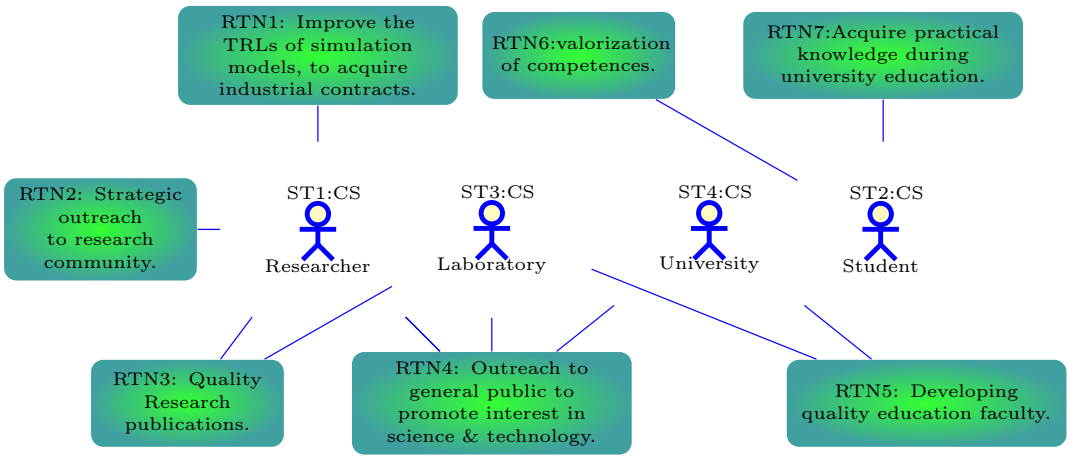


Figure 5.6: IBIS Rationale Map

Constraints imposed by the customer are also elicited at this stage and are taken in account during next steps of system design. Although, constraints are also type of customer need, they are treated specifically and account for system

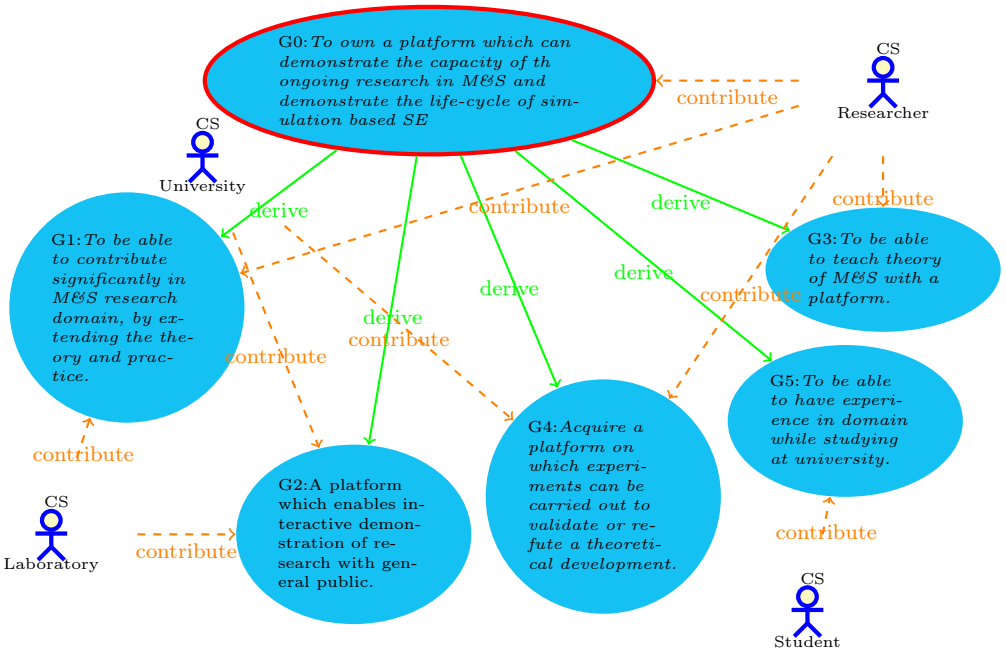


Figure 5.7: IBIS Goal Diagram

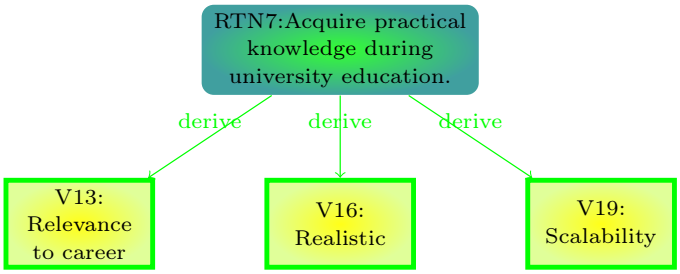


Figure 5.8: IBIS Rationale Viewpoint

requirements derived in next phase. IBIS project has three constraints regarding: space, energy or propulsion system and a funding constraint. Space constraints forces designer to design the system such that it does not requires more than $9m^2$ of space in laboratory/university premises. Energy constraint implies to use non ignitable source of energy in the system. Funding constraint limits the expenses on software and hardware components of the system within € 10,000. Space constraint comes from the corresponding host university/laboratory authorities. Energy and funding constraints come from the relevant laboratory authorities.

Figure 5.7, shows the goal-model of the IBIS system and the involved client stakeholders described in goal-model of our tool *SysEngLab*. A total of five sub-goals are derived from the primary goal. Figure 5.10, shows the primary stakeholders and the viewpoints held by them for the project development.

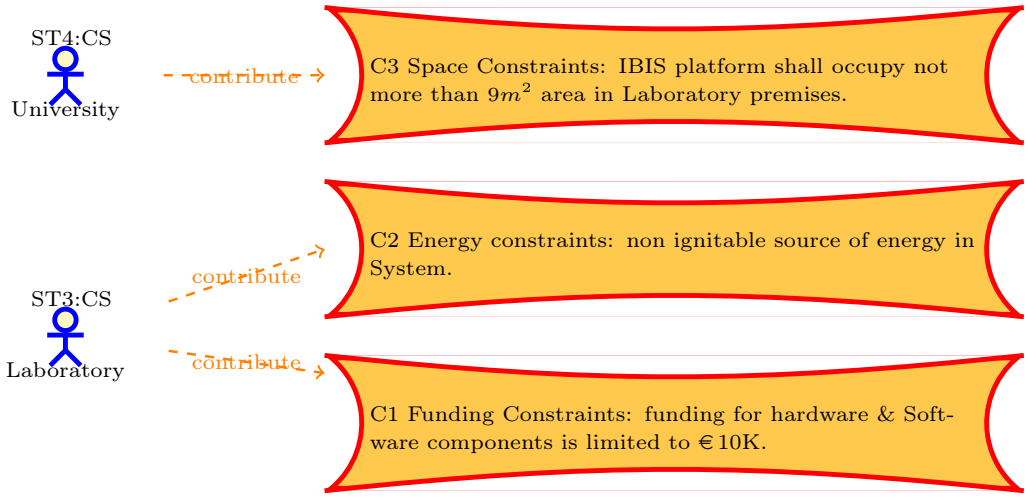


Figure 5.9: IBIS Constraints

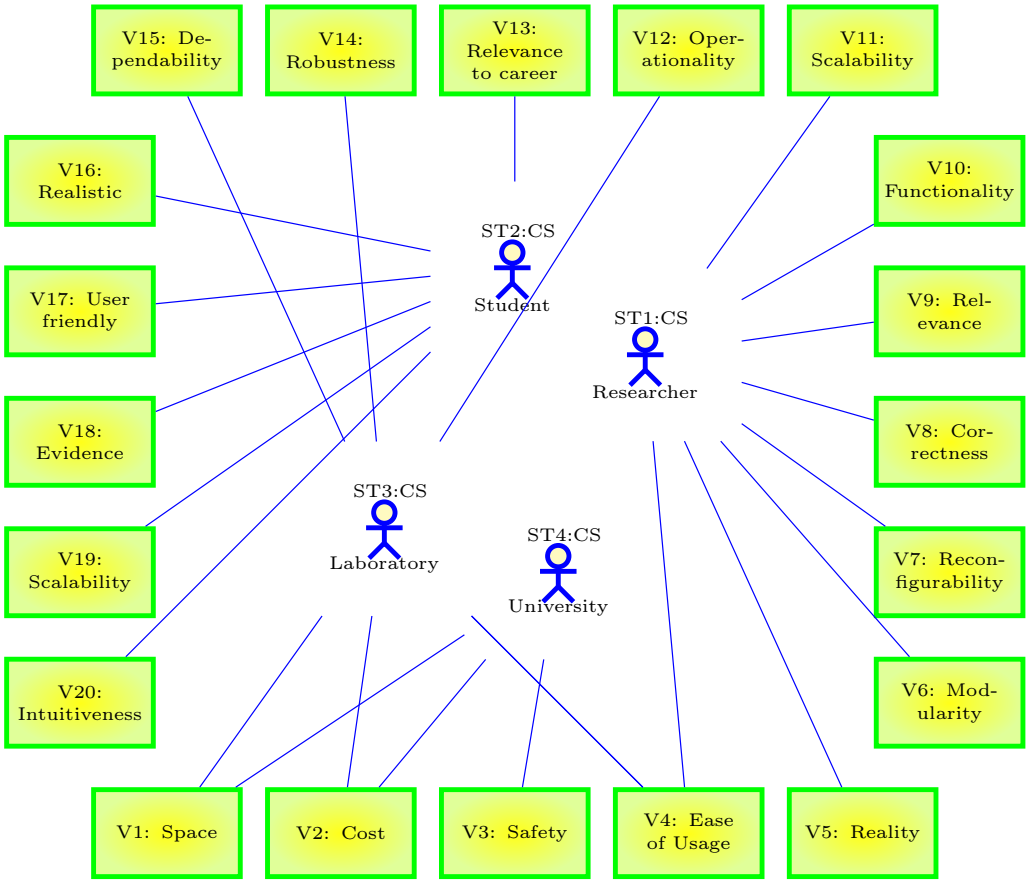


Figure 5.10: Primary Stakeholders and Viewpoint

Figure 5.11: IBIS Objectives

5.3.2.3 Prioritizing Goals Using Our Approach

Prioritizing subgoals is necessary before further development is carried out. Prioritization of subgoals forms the basis of decisions and inputs to purposed requirements traceability to come in future. Project manager asks for the preferences of the stakeholders: researcher, university representative, laboratory representative and students representative. Project manager creates his own preference matrix of stakeholders, representing his perceived importance of the stakeholders. The preference matrix leads to ranking and a risk averse function is used to convert those rankings into weights shown in Figure 5.12. Eq.(5.1) shows the preference matrix

and the resultant weights for the stakeholders.

$$P_{PM} = \begin{array}{c} \\ r \\ u \\ l \\ s \end{array} \begin{array}{c} \begin{array}{cccc} r & u & l & s \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 2 \\ -1 & 0 & 0 & 1 \\ -1 & 0 & 0 & 1 \\ -2 & -1 & -1 & 0 \end{bmatrix} \end{array}, \quad \begin{array}{c} \begin{array}{cccc} r & u & l & s \end{array} \\ \begin{bmatrix} 0.32 & 0.26 & 0.26 & 0.16 \end{bmatrix} \end{array} \tag{5.1}$$

Table 5.2: Prioritizing Goals

Origin	**Sub-goals description**
Researchers, University	*G1 :To be able to contribute significantly in M&S research domain, by extending the theory and practice.*
University, Laboratory, Researchers	*G2:A platform which enables interactive demonstration of research with general public.*
Researchers, University, Laboratory	*G3:Acquire a platform on which experiments can be carried out to validate or refute a theoretical development.*
Researchers, University	*G4:To be able to teach theory of M&S with a platform.*
Student	*G5:To be able to have experience in domain while studying at university.*

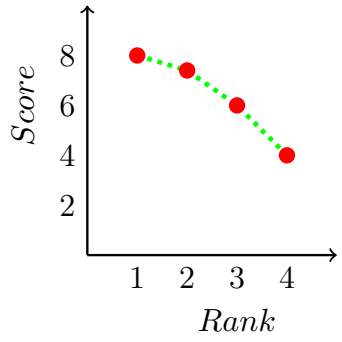


Figure 5.12: Stakeholder Weighting

Table 5.3: Goal Categorization

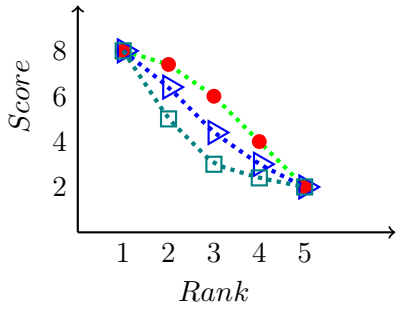
DM	*High*	*Medium*	*Low*
Researchers	G_1, G_3	G_4	G_5, G_2
University	G_1, G_4	G_2, G_5	G_3
Laboratory	G_3, G_1	G_2	G_4, G_5
Student	G_5, G_4	G_3, G_1	G_2

Table 5.2 shows the various sub-goals and their origins as traced from user stories. Each stakeholder is asked to arrange the goals in three categories: high (H), medium(M), and low(L), depicting their corresponding perceived level of priority. Table 5.3 shows the goals categorization by the corresponding stakeholders. Then their preferences are elicited using the preference matrices shown by Eq.(5.2) and Eq.(5.3). Once the Preference matrices of stakeholders are available they have their ranking available for the different subgoals. The simple marking process consisting of putting their rankings one after another till all the subgoals are marked is shown by Eq.(5.4). In the following step weight for each subgoal is calculated using the

three types of discrete utility function as shown in Figure 5.13 with the help of Eq.(4.11). The calculated weights are shown in Table 5.4, where $R-A$, $R-N$, $R-P$ and nm refer to risk averse, risk neutral, risk prone and normalized score respectively. Weights are based on the utility function with discrete values with risk averseness, risk neutrality and risk proneness. The Final weight is taken using the mean of the three weight arrays.

$$P_r = \begin{array}{c} \\ G_1 \\ G_2 \\ G_3 \\ G_4 \\ G_5 \end{array} \begin{array}{c} \begin{array}{ccccc} G_1 & G_2 & G_3 & G_4 & G_5 \end{array} \\ \begin{bmatrix} 0 & 2 & 1 & 1 & 2 \\ -2 & 0 & -2 & -1 & 0 \\ -1 & 2 & 0 & 1 & 2 \\ -1 & 1 & -1 & 0 & 1 \\ -2 & 0 & -2 & -1 & 0 \end{bmatrix} \end{array}, P_u = \begin{array}{c} \\ G_1 \\ G_2 \\ G_3 \\ G_4 \\ G_5 \end{array} \begin{array}{c} \begin{array}{ccccc} G_1 & G_2 & G_3 & G_4 & G_5 \end{array} \\ \begin{bmatrix} 0 & 1 & 2 & 1 & 1 \\ -1 & 0 & 1 & -1 & 1 \\ -2 & -1 & 0 & -1 & -1 \\ -1 & 1 & 2 & 0 & 1 \\ -1 & -1 & 1 & -1 & 0 \end{bmatrix} \end{array} \tag{5.2}$$

$$P_l = \begin{array}{c} \\ G_1 \\ G_2 \\ G_3 \\ G_4 \\ G_5 \end{array} \begin{array}{c} \begin{array}{ccccc} G_1 & G_2 & G_3 & G_4 & G_5 \end{array} \\ \begin{bmatrix} 0 & 1 & -1 & 2 & 2 \\ -1 & 0 & -1 & 1 & 1 \\ 1 & 1 & 0 & 2 & 2 \\ -2 & -1 & -2 & 0 & 1 \\ -2 & -1 & -2 & -1 & 0 \end{bmatrix} \end{array}, P_s = \begin{array}{c} \\ G_1 \\ G_2 \\ G_3 \\ G_4 \\ G_5 \end{array} \begin{array}{c} \begin{array}{ccccc} G_1 & G_2 & G_3 & G_4 & G_5 \end{array} \\ \begin{bmatrix} 0 & 1 & -1 & -1 & -1 \\ -1 & 0 & -1 & -2 & -2 \\ 1 & 1 & 0 & -1 & -1 \\ 1 & 2 & 1 & 0 & -1 \\ 1 & 2 & 1 & 1 & 0 \end{bmatrix} \end{array} \tag{5.3}$$



$$Game = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} G_1 & G_2 & G_3 & G_4 & G_5 \end{array} \\ \begin{bmatrix} \text{r,u} & 0 & \text{l} & 0 & \text{s} \\ \text{l} & 0 & \text{r} & \text{s,u} & 0 \\ 0 & \text{u,l} & \text{s} & \text{r} & 0 \\ \text{s} & \text{r} & 0 & \text{l} & \text{r,u} \\ 0 & \text{r,s} & \text{u} & 0 & \text{r,l} \end{bmatrix} \end{array} \tag{5.4}$$

Figure 5.13: Criteria Weighting

Goals	Algorithms						
	$R-A$		$R-N$		$R-P$		*mean*
	score	*nm*	*score*	*nm*	*score*	*nm*	
G_1	6.97	0.242	6.75	0.256	6.299	0.30	0.266
G_2	5.346	0.185	4.19	0.159	2.949	0.14	0.161
G_3	5.807	0.201	5.306	0.201	4.655	0.222	0.208
G_4	5.998	0.208	5.77	0.218	3.641	0.173	0.199
G_5	4.668	0.162	4.353	0.165	3.43	0.163	0.163

Table 5.4: Goal Weights

5.3.2.4 Writing Stakeholder Requirements using negation

In Figure 5.11, high level customers requirements are modeled as objectives. Upon their analysis a few of the needs particularly which represent the qualitative aspects of the system are perceived to have some ambiguity and seem hard to be contracted. We use the negation method as mentioned early in the Chapter 2, to avoid the ambiguity and make it more clear.

Table 5.5: Writing Requirements with Negation

Affirmative	**Affirmative + negative**
Platform shall be safe to use.	*Using platform shall not cause any health hazards.*
Platform shall be modular.	*Platform components shall not be corrupted by the immediate components.*
Platform shall be reconfigurable.	*Platform shall not stop user to change the system modules on his command.*
Platform shall be easy to use.	*Platform shall not take more than five steps to start using the platform.*
Platform shall be easy to deploy.	*Platform shall not take more than 10 minutes to go critical.*
The platform should be able to remain functional in a closed hall.	*Platform shall use non ignitable source of energy in system.*

Constraint imposed by the laboratory "*The platform should be able to be stationed and remain functional in a closed room,*" can be interpreted easily upon analysis that, laboratory can not host a platform with gaseous emissions or with considerably noise polluting components. This constraint can be simplified and written as above.

5.3.2.5 Negotiating Stakeholders Requirements

From the very beginning of the elicitation process the stakeholders requirements or needs are negotiated and marked as either: core feature or optional feature. This is done by the objective blocks property of being marked as optional or core as shown in Figure 5.11, objectives "*O15: platform shall allow to assess the students*" and " *O18: platform shall be able to provide suggestions to students for M&S*" are marked with an '**O**' in north east corner. If designing test case for a given stakeholder need seems to be difficult to understand, it means that the need is still not very clear. Different stakeholders may not have the same impression about the priority of requirements. Only a few of the requirements pose this problem and need prompt solution. Prioritizing needs can be done using the same process as used for prioritizing goals in Subsection 5.3.2.3. For each customer need elicited a validation test case is proposed, which forms the basis of the contract.

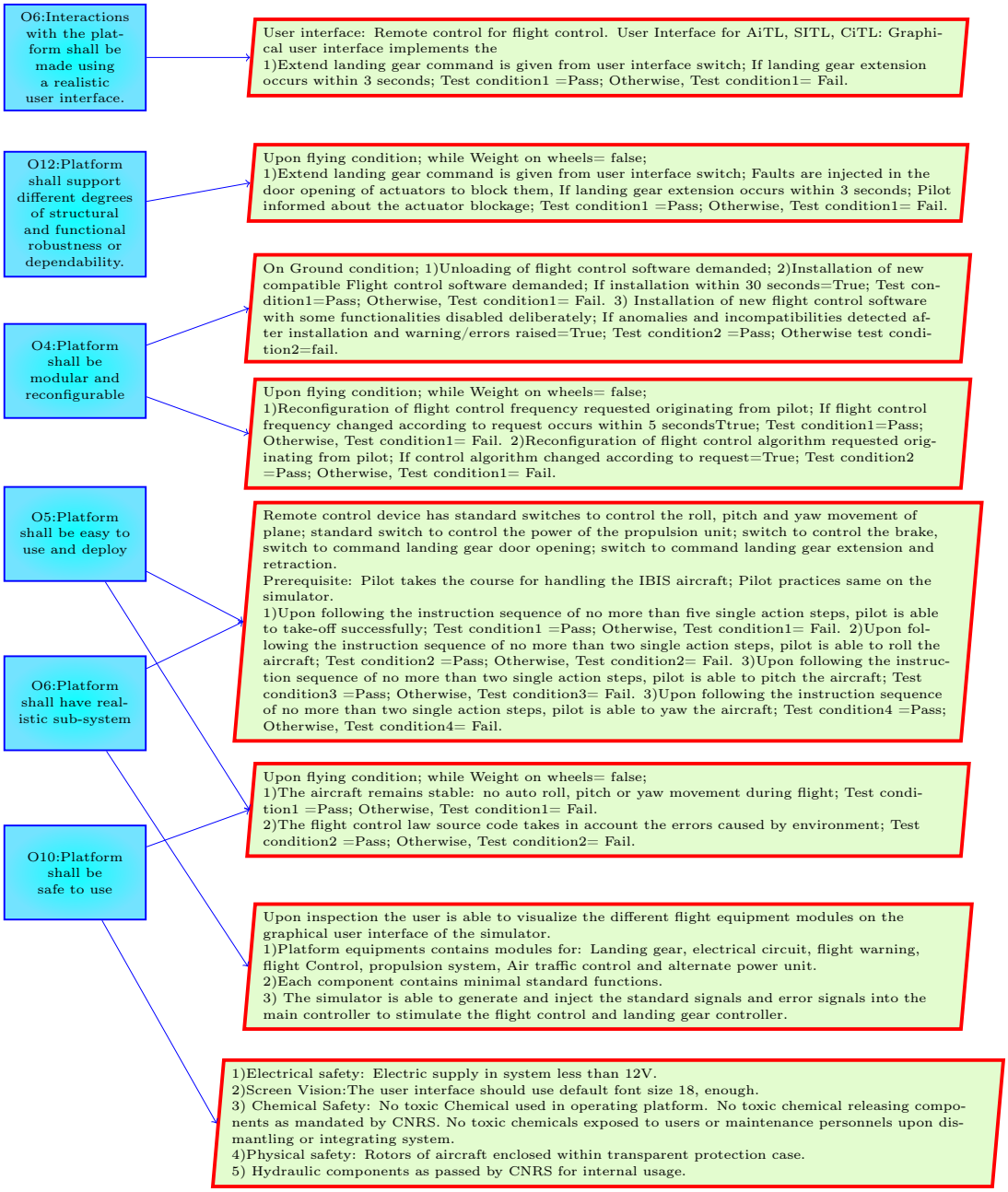


Figure 5.14: Test Case for Customer Requirements

5.3.3 IBIS System Requirements Definition

Figure 5.15, presents the architecture of the IBIS platform in the form of the SysML block definition diagram. System requirements modeling is carried out in parallel with system modeling, both activities provide inputs to each other which lead to better formulation of the system.

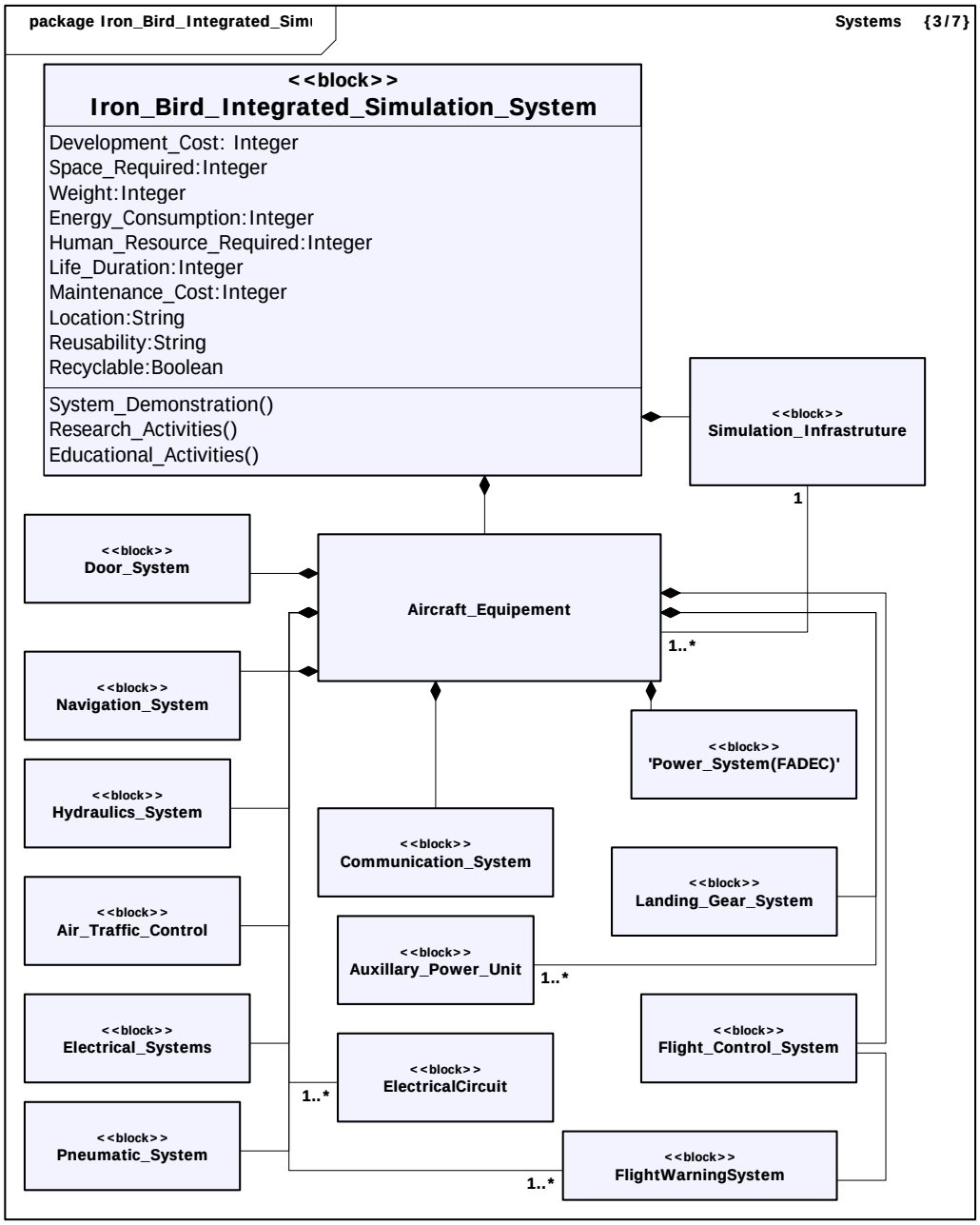


Figure 5.15: Architecture of IBIS Platform

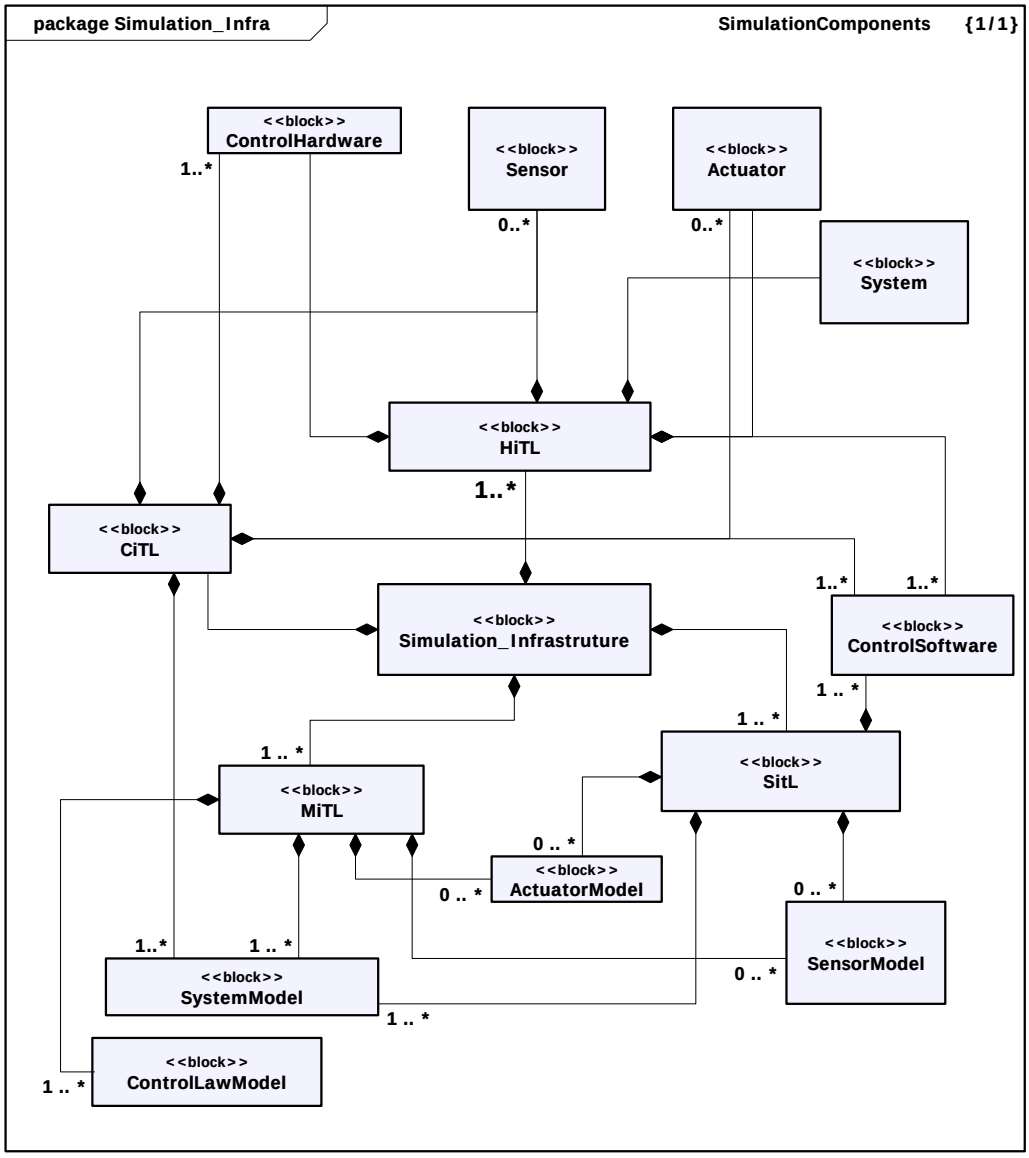


Figure 5.16: IBIS Simulation Infrastructure Platform

5.3.3.1 Landing Gear Requirements

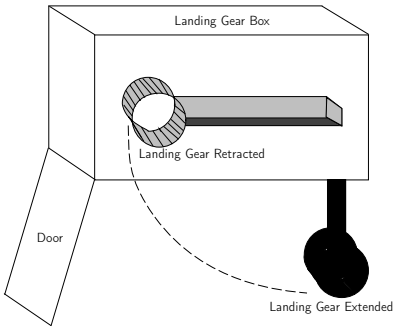


Figure 5.17: IBIS Landing Gear

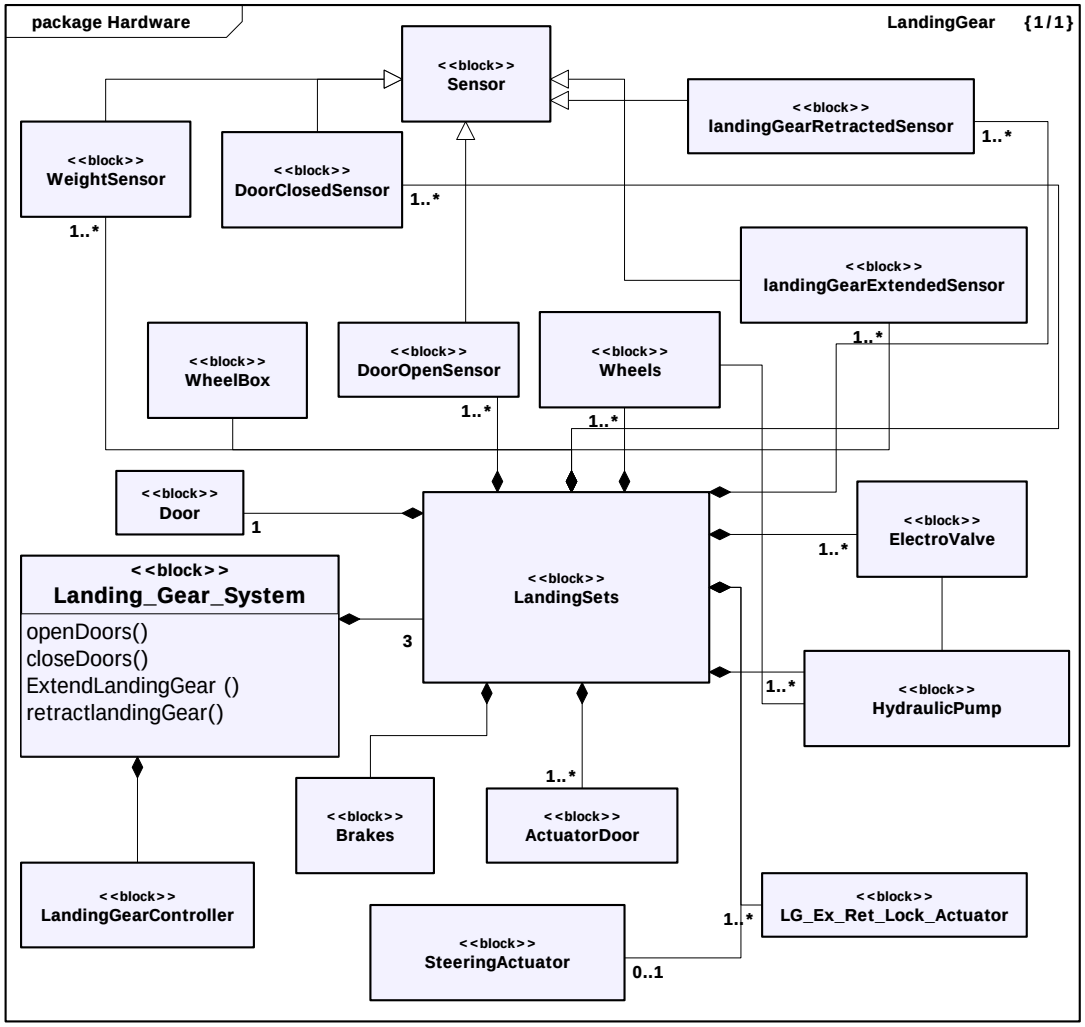


Figure 5.18: IBIS Landing Gear System

Figure 5.17 shows an abstract sketch of landing gear. Figure 5.18 presents the modeled landing gear in SysML block definition diagram. Landing gear simulation requirements are derived from the previously modeled objectives. Figure 5.19, 5.20, show some of the derived landing gear system requirements.

Figure 5.19: IBIS Landing Gear System Requirements

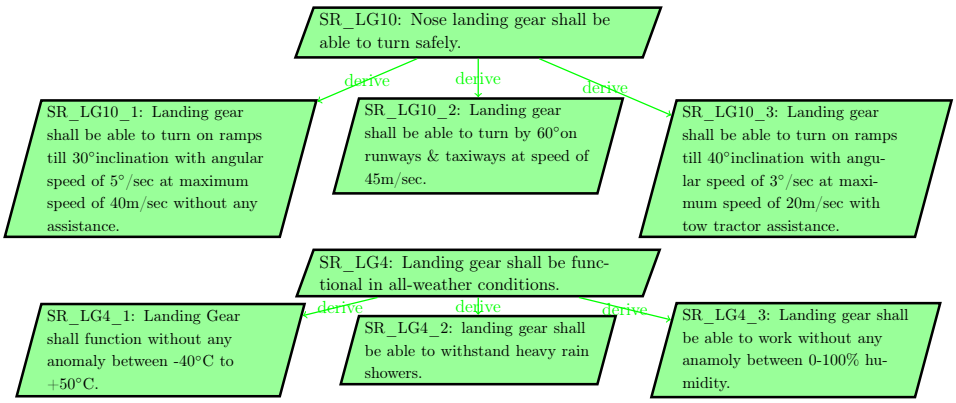


Figure 5.20: Landing Gear Derived System Requirements

Requirements which appear to be general are derived till they become more measurable or quantified. For example Figure 5.20, shows one example of derived requirements. The three system requirements derived from the original system requirement allows to quantify the aspect it wanted to express.

5.3.3.2 Landing Gear MiTL Requirements

Figure 5.21: IBIS Landing Gear MiTL Requirements

Figure 5.16, presents the IBIS simulator infrastructure. Figure 5.21 shows the MiTL requirements derived from the objective "Platform shall support Landing Gear MiTL." These requirements may lead to other derived requirements.

5.3.3.3 Landing Gear User Interface Requirements

Figure 5.23, presents the landing gear user interface requirements as modeled in *CReML* using *SysEngLab*. User interface requirements are derived from the previously modeled objective: "landing gear user interface." Sixteen user interface requirements are listed in the *CReML* diagram.

5.3.3.4 Writing System Requirements using negation

The non-functional requirements related to the quality of system such as: reconfigurability, modularity, robustness can be explored further and written in an non-ambiguous form using suitable usage of negation. A test case is designed to verify the requirement and forms the basis of contract between the client stakeholders and the developer stakeholders.

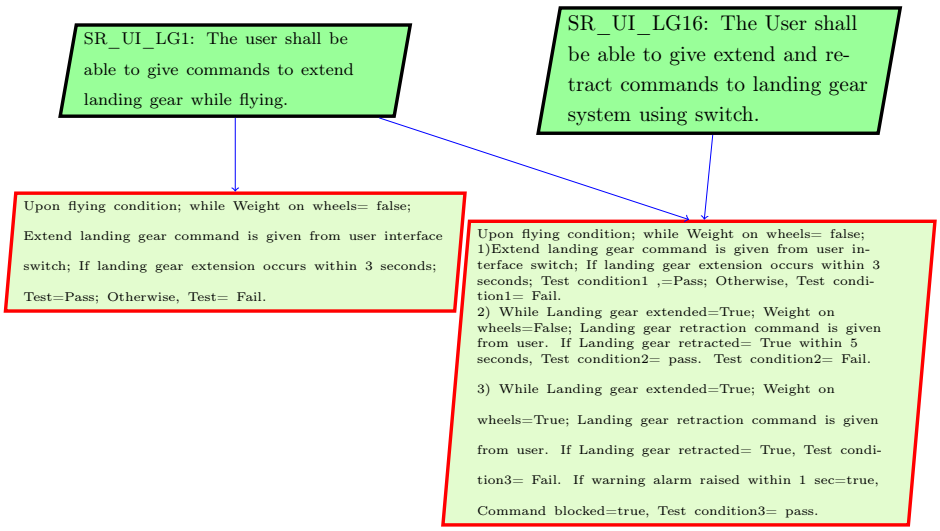


Figure 5.22: Validation Test Cases for Landing Gear System Requirements

5.3.4 IBIS Architecture Design and Analysis

As, architecture design and analysis are quasi-parallel activities, as soon as the user needs are elicited, system requirements definition process starts together with architecture design. Figure 5.15, presents the model of IBIS platform. IBIS Platform is composed of the two components: *aircraft equipment* and *simulation infrastructure*. *Aircraft equipment* typically groups the subsystems which constitute the aircraft and *simulation infrastructure* constitutes the systems which are responsible for carrying out the simulation of the aircraft equipment.

Figure 5.16, shows the simulation infrastructure required by the IBIS platform. IBIS simulation infrastructure is composed of the four different levels of simulators: MiTL, SiTL, CiTL and HiTL. MiTL is composed of system model, control laws model and actuator models (algorithm coded in different language then target). SiTL block is composed of sensor models, actuator models, control software and system model (algorithms coded in target language). CiTL is in-turn composed of control hardware, control software, sensors, actuators and system model. HiTL is composed of system itself, its control hardware and software, sensors and actuators of the system.

Among the aircraft equipments, only the *Landing gear* is detailed in this chapter. Figure 5.18, presents the proposed architecture of the landing gear system. Landing gear is composed of landing sets and landing gear controller. Each landing set is composed of wheels, wheel box, door, hydraulic actuators, valves, and a set of sensors for monitoring landing gear. Weight sensor monitors if the weight of the aircraft is on the wheels or not. Door close and open sensors provide the status of the door if its open, or close. Landing gear's extension and retraction is monitored by two other dedicated sensors.

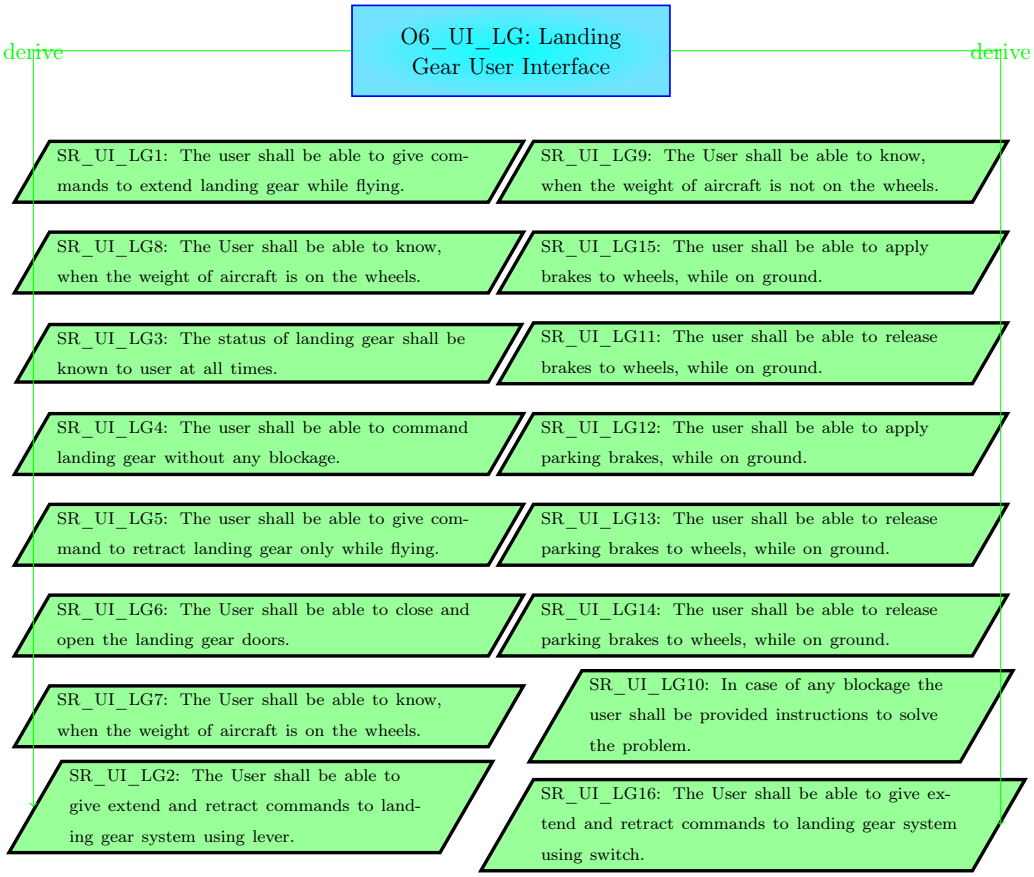


Figure 5.23: IBIS Landing Gear User Interface Requirements

5.3.5 Landing Gear Detail Design

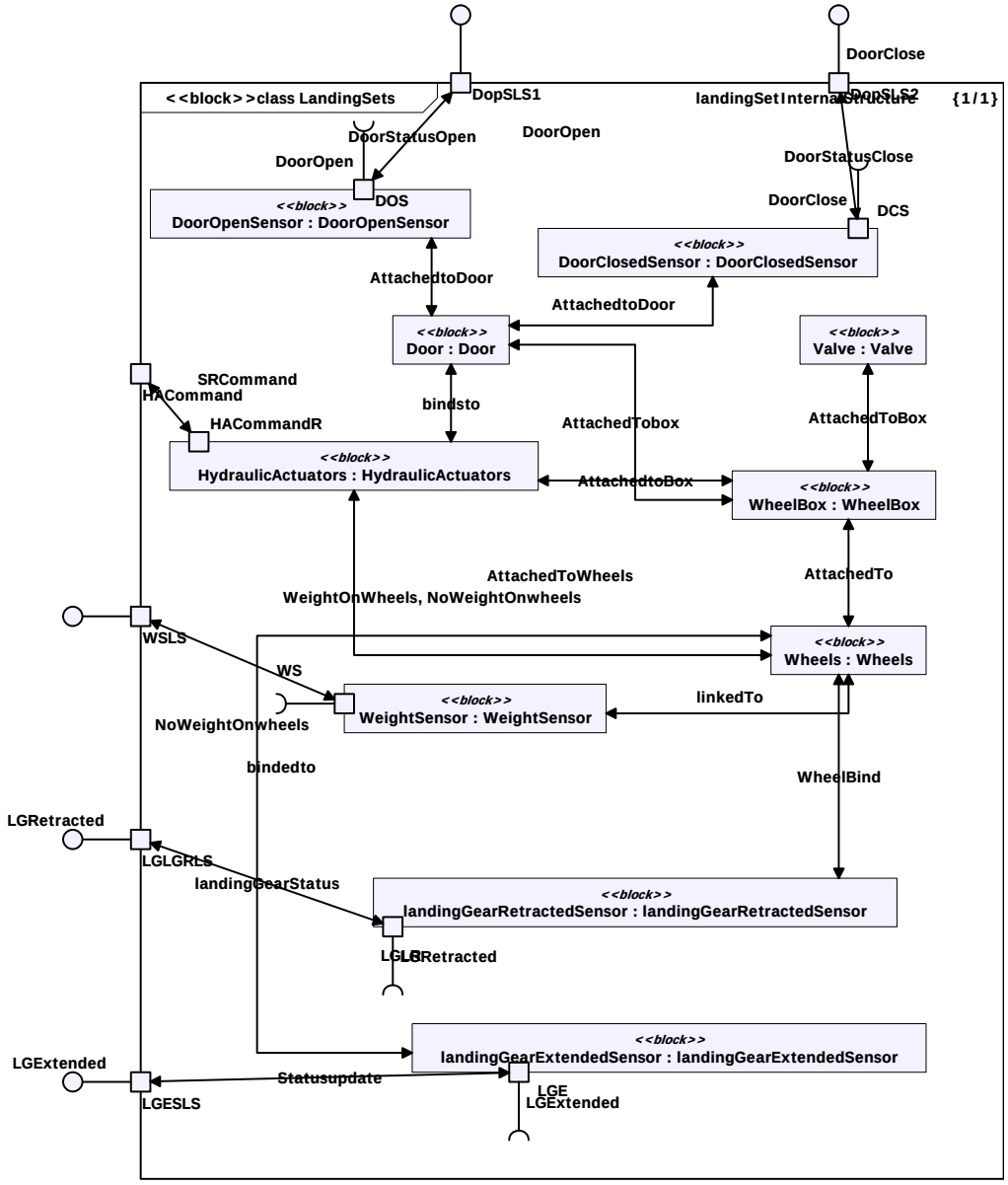


Figure 5.24: IBIS Landing Set Detail Design

Detail design is carried out to provide in-depth composition of the landing gear. Figure 5.24, shows the detail design of landing set. Signals and interfaces are designed which allow to communicate between the sub-systems. Detail design forms the basis of the system specifications and selection of components.

5.3.5.1 Design Specification for Landing Gear

As the system requirements are allocated to the architecture designs, the system specifications are refined set of system characteristics. System specifications for the landing gear and the MiTl infrastructure are mentioned below.

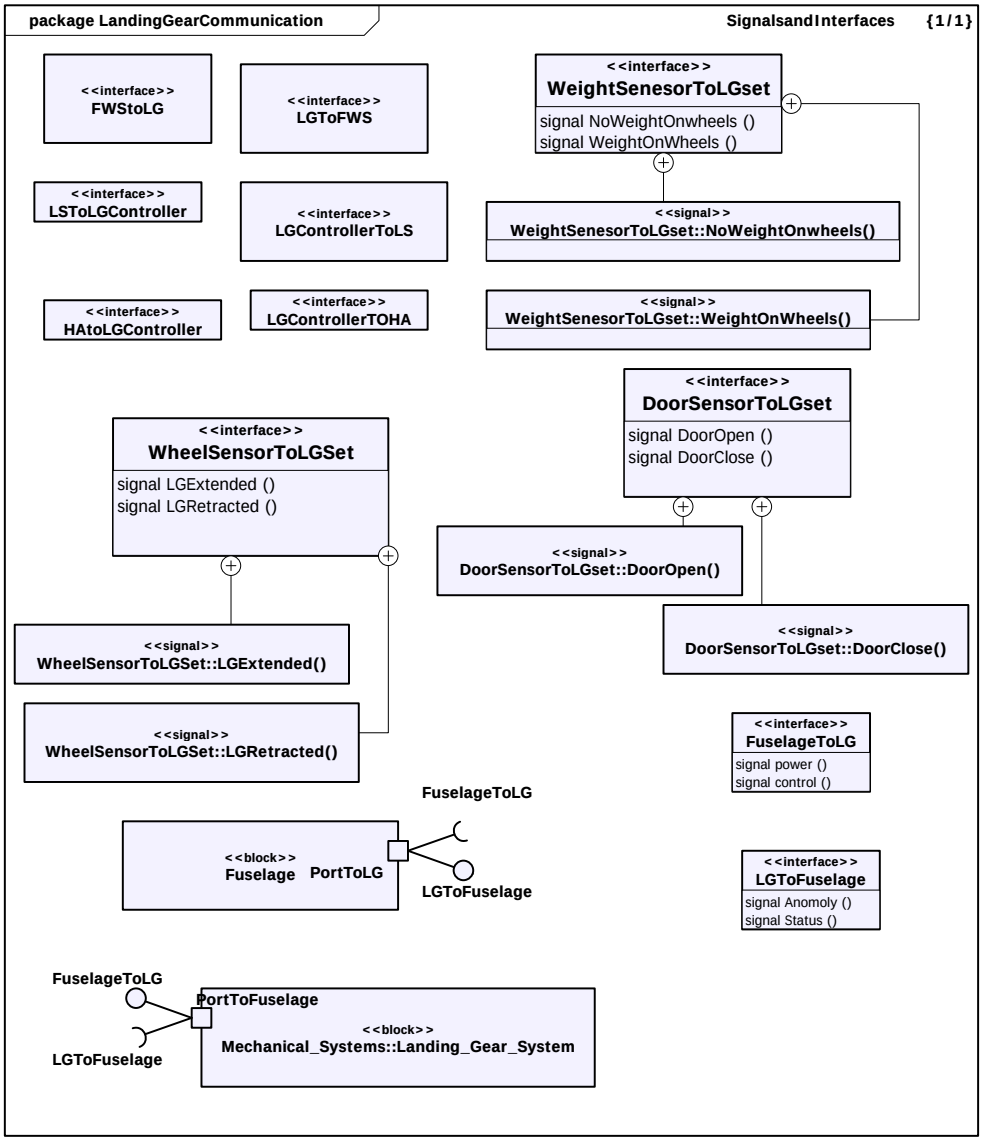


Figure 5.25: Landing Gear Signals and Interfaces Specifications

Specifications for modularity: IBIS platform should implement bus-modular architecture for plugging in various aircraft equipment modules. *Communication bus between hardware and controller.* options available: AFDX, ARINC 429, CAN, TTP. *Weight Sensors*, door open and door close sensors, wheel retracted and extended sensors, Valves and hydraulic actuators.

5.3.5.2 Design Specification for MiTL of Landing Gear

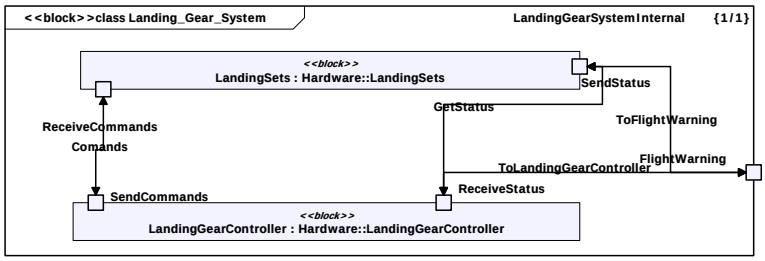


Figure 5.26: Landing Gear Signals and Interfaces Specifications

5.3.6 Deciding Specifications using our Technique

Degree of redundancy needs to be determined, in order to satisfy few of the requirements mentioned previously.

Step 1: Categorizing Stakeholders Categorizing stakeholders according to their influence and interest in the design.

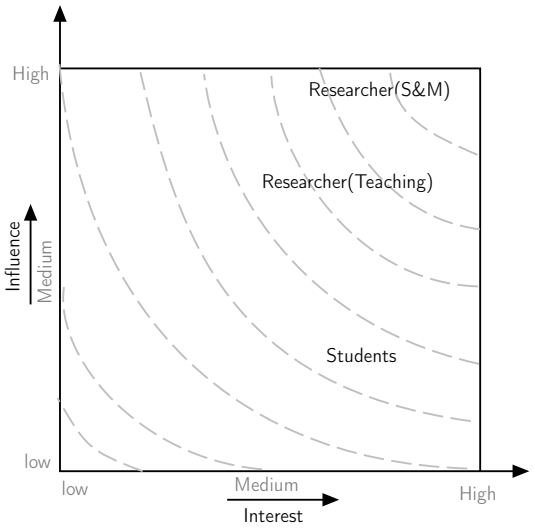


Figure 5.27: Categorizing Stakeholder

Criteria	Description
c_1	Development time
c_2	Cost
c_3	Benefit

Table 5.6: Design Criteria for Specification Selection

Step 2: Weighting Stakeholders Project manager uses a risk averse function to weight the stakeholders associated with the robustness requirements. Previously derived rankings provide the base for weighting. The weight for each ranking is shown in Figure 5.28.

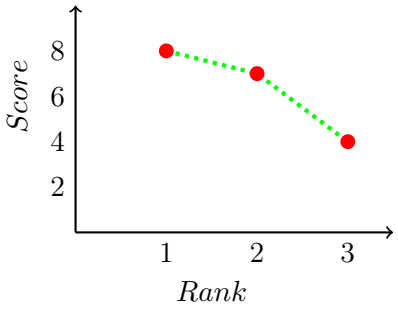


Figure 5.28: Categorizing Stakeholder

Stakeholder	weight
$(a) Researcher(M\&S)$	0.421
$(b) Researcher(Teaching)$	0.368
$(c) Students$	0.21

Table 5.7: Stakeholder Weight

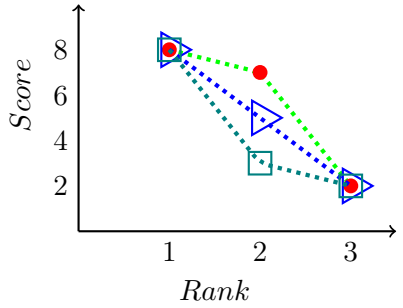
The weight for each stakeholder is shown in Table 5.7.

Step 3: Criteria Categorization Given to the number of criteria under consideration, this step can be ignored.

Step 4: Stakeholders' Preference Modeling

$$P_a = \begin{array}{c} \\ c_1 \\ c_2 \\ c_3 \end{array} \begin{array}{c} \begin{array}{ccc} c_1 & c_2 & c_3 \end{array} \\ \begin{bmatrix} 0 & 1 & -1 \\ -1 & 0 & -2 \\ 1 & 2 & 0 \end{bmatrix} \end{array}, P_b = \begin{array}{c} \\ c_1 \\ c_2 \\ c_3 \end{array} \begin{array}{c} \begin{array}{ccc} c_1 & c_2 & c_3 \end{array} \\ \begin{bmatrix} 0 & 2 & 0 \\ -2 & 0 & -2 \\ 0 & 2 & 0 \end{bmatrix} \end{array}, P_c = \begin{array}{c} \\ c_1 \\ c_2 \\ c_3 \end{array} \begin{array}{c} \begin{array}{ccc} c_1 & c_2 & c_3 \end{array} \\ \begin{bmatrix} 0 & 1 & -2 \\ -1 & 0 & -2 \\ 2 & 2 & 0 \end{bmatrix} \end{array} \tag{5.5}$$

Step 5: Criteria Weight Generation The DMs carry out the marking process according to their preferences. In this example we have used three algorithms for calculating the score, one risk neutral and one risk averse and one risk neutral algorithms with slight difference in degree of neutrality.



$$Game = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} c_1 & c_2 & c_3 \end{array} \\ \begin{bmatrix} \text{b} & 0 & \text{a,b,c} \\ \text{a,b,c} & 0 & \text{b} \\ 0 & \text{a,b,c} & 0 \end{bmatrix} \end{array} \tag{5.6}$$

Figure 5.29: Generating Criteria Weights

	Algorithms		
Criteria	$R-A$	$R-N$	$R-P$
c_1	9.944	6.255	5.941
c_2	2	2	2
c_3	10.576	7.939	8.045

Table 5.8: Criteria Weights

	Algorithms		
Criteria	$R-A$	$R-N$	$R-P$
c_1	0.44	0.386	0.371
c_2	0.09	0.123	0.125
c_3	0.47	0.49	0.503

Table 5.9: Normalized Criteria Weights

$$\begin{array}{ccc} c_1 & c_2 & c_3 \end{array} \\ \begin{bmatrix} 0.399 & 0.111 & 0.49 \end{bmatrix} \tag{5.7}$$

Step 6: Evaluating Alternatives Table 5.13, shows the evaluation of degree of redundancy required by the different stakeholders. Legends $0 - redundt$, $2S - redundt$, $3S - redundt$, and $3M - redundt$ mean zero redundancy, double same type redundancy, triple same type redundancy, and triple mixed type redundancy respectively.

Table 5.10: Evaluation Degree of Redundancy

		Alternatives			
Criteria	weight	0-redundt	2S-redundt	3S-redundt	3M-redundt
c_1	0.399	++	++(95)	++(90)	+
c_2	0.111	++	++(95)	+	+(75)
c_3	0.49	−−	+−	++(95)	++
Net Score	$\sum w$	60.8	77.85	91.34	89.245

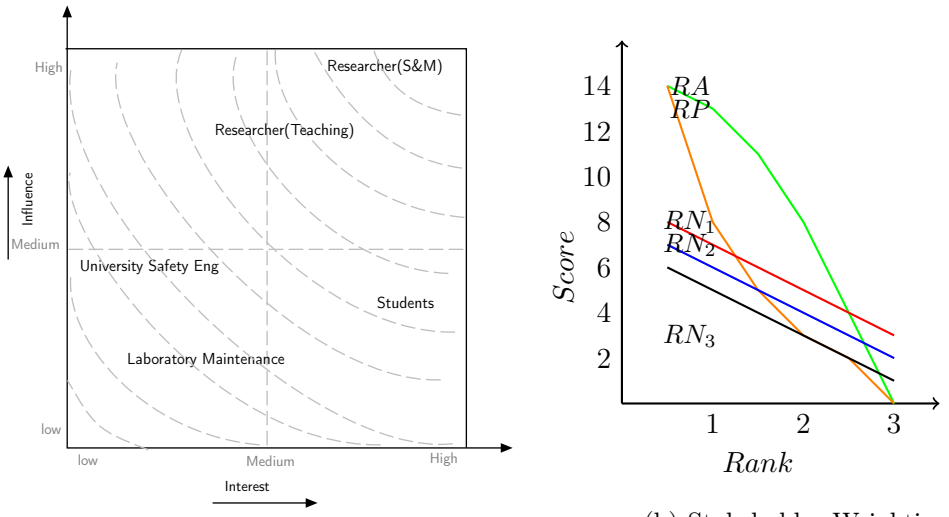
As the third option for 3S-redundancy secures the highest score on the evaluation matrix. The stakeholders agree upon the 3 pair of same type of redundant actuators.

5.3.7 Deciding Design Components

Table 5.11: Design Criteria for Landing Gear Component Selection

Criteria	**Description**	**Comopsed of Sub-Criteria**
c_1	Functionality	*Endurance, speed of operation, weight*
c_2	Initial cost	*Initial expenditure for acquisition , time for delivery*
c_3	Durability	*Tolerance to harsh conditions, water, electric charges, tolerance to shocks, duration of life while regular usage*
c_4	Dependability	*Rate of failure, tolerance to faults, potential hazards*
c_5	Maintainability	*Time required to maintain, cost of maintenance*
c_6	Ease of integration	*S/W, H/W, & other interface requirements*

The project manager of IBIS determines five stakeholders which are decision makers and system value providers makers for the design components selection: (a)



(a) Categorizing Stakeholders

(b) Stakeholder Weighting

Figure 5.30: Stakeholder Categorizing and Weighting

Researcher(M&S), (b)Researcher(Teacher), (c) Student representative, (d) Maintenance team and (e) University-Laboratory Safety Personnel,

Step 1: Categorizing Stakeholders The decision makers which are system value providers are DMs $D = \{a, b, c, d, e\}$ are categorized using the *Influence vs. Interest grid*. The grid is shown in Figure 5.30a, which provides DM's stakes in decision. Clearly, the order of importance of their stake is as follow: $a > b > c > e > d$.

Step 2: Weighting Stakeholders To convert the rankings into the weights, we use the surrogate weight generation technique based on decreasing utility functions. A risk averse function based on Eq.(5.8) is used to generate weights with $k = 80$, $j = 3.2^i$.

$$score(c_i) = k - i.j \quad , k = Maxscore,\ j \in \mathbb{Z} \tag{5.8}$$

The resultant weights are shown in the Eq.(5.9) below.

$$\begin{matrix} a & b & c & d & e \end{matrix}$$
$$\begin{bmatrix} 0.238 & 0.18 & 0.258 & 0.22 & 0.103 \end{bmatrix} \tag{5.9}$$

Step 3: Stakeholders' Preference Modeling of criteria The DMs categorize criteria set according to their perception of criteria as shown below in Table 5.12.

Table 5.12: Design Criteria Categorization

DM	*High*	*Medium*	*Low*
a	c_1, c_5	c_2, c_4	c_6, c_3
b	c_4, c_6	c_1, c_2	c_3, c_5
c	c_3, c_4	c_2, c_1	c_5, c_6
d	c_6, c_3	c_2, c_1	c_4, c_5
e	c_2, c_3	c_1, c_6	c_5, c_4

Step 4:Stakeholders' Preference matrix The DMs create their preference matrices according to their categorization from previous step. The preference matrix of stakeholder a, b, c, d and e are shown in Eq.(5.10), (5.11),and (5.12) respectively.

$$P_a = \begin{array}{c} \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \end{array} \begin{array}{c} \begin{array}{cccccc} c_1 & c_2 & c_3 & c_4 & c_5 & c_6 \end{array} \\ \begin{bmatrix} 0 & 1 & 2 & 1 & 1 & 2 \\ -1 & 0 & 2 & 1 & -2 & 2 \\ -2 & -2 & 0 & -1 & -2 & 0 \\ -1 & -1 & 1 & 0 & -1 & 1 \\ -1 & 2 & 2 & 1 & 0 & 1 \\ -2 & -2 & 0 & -1 & -1 & 0 \end{bmatrix} \end{array} \tag{5.10}$$

$$P_b = \begin{array}{c} \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \end{array} \begin{array}{c} \begin{array}{cccccc} c_1 & c_2 & c_3 & c_4 & c_5 & c_6 \end{array} \\ \begin{bmatrix} 0 & 1 & 2 & -2 & 1 & -2 \\ -1 & 0 & 1 & -2 & 2 & -1 \\ -2 & -1 & 0 & -2 & 1 & -2 \\ 2 & 2 & 2 & 0 & 2 & 1 \\ -1 & -2 & -1 & -2 & 0 & -2 \\ 2 & 1 & 2 & -1 & 2 & 0 \end{bmatrix} \end{array}, P_c = \begin{array}{c} \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \end{array} \begin{array}{c} \begin{array}{cccccc} c_1 & c_2 & c_3 & c_4 & c_5 & c_6 \end{array} \\ \begin{bmatrix} 0 & -1 & -1 & -1 & 1 & 1 \\ 1 & 0 & -1 & -1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 2 & 2 \\ 1 & 1 & -1 & 0 & 2 & 2 \\ -1 & -1 & -2 & -2 & 0 & 1 \\ -1 & -1 & -2 & -2 & -1 & 0 \end{bmatrix} \end{array} \tag{5.11}$$

$$P_d = \begin{array}{c} \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \end{array} \begin{array}{c} \begin{array}{cccccc} c_1 & c_2 & c_3 & c_4 & c_5 & c_6 \end{array} \\ \begin{bmatrix} 0 & -1 & -1 & 1 & 1 & -2 \\ 1 & 0 & -1 & 1 & 1 & -2 \\ 1 & 1 & 0 & 2 & 2 & -1 \\ -1 & -1 & -2 & 0 & 1 & -2 \\ -1 & -1 & -2 & -1 & 0 & -2 \\ 1 & 1 & 1 & 2 & 2 & 0 \end{bmatrix} \end{array}, P_e = \begin{array}{c} \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \end{array} \begin{array}{c} \begin{array}{cccccc} c_1 & c_2 & c_3 & c_4 & c_5 & c_6 \end{array} \\ \begin{bmatrix} 0 & -1 & -1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 2 & 2 & 1 \\ 1 & -1 & 0 & 2 & 2 & 1 \\ -1 & -2 & -2 & 0 & -1 & -2 \\ -1 & -2 & -2 & 1 & 0 & -1 \\ -1 & -1 & -1 & 1 & 1 & 0 \end{bmatrix} \end{array} \tag{5.12}$$

Step 5: Criteria Weight generation The DMs carry out the marking process according to their preferences as shown in Eq.(5.13). In this example we have used five algorithms for calculating the score, one risk prone (Risk-P) and one risk averse (Risk-A) and three risk neutral(Risk-N1, Risk-N2, Risk-N3) algorithms with slight

difference in degree of neutrality. The resultant criteria weight set is shown in Eq.(5.14).

$$Game = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{cccccc} c_1 & c_2 & c_3 & c_4 & c_5 & c_6 \end{array} \\ \begin{bmatrix} \text{a} & \text{e} & \text{c} & \text{b} & 0 & \text{d} \\ 0 & 0 & \text{d,e} & \text{c} & \text{a} & \text{b} \\ \text{b,e} & \text{c,a,d} & 0 & 0 & 0 & 0 \\ \text{c,d} & \text{b} & 0 & \text{a} & 0 & \text{e} \\ 0 & 0 & \text{b} & \text{d} & \text{c,e} & \text{a} \\ 0 & 0 & \text{a} & \text{e} & \text{b,d} & \text{c} \end{bmatrix} \end{array} \tag{5.13}$$

Table 5.13: Design Criteria Scores

	Algorithms				
Criteria	Risk-N1	Risk-N2	Risk-N3	Risk-P	Risk-A
c_1	5.82	4.85	4.6	6.7	9.92
c_2	5.78	4.81	3.84	5.07	10.37
c_3	5.86	4.61	3.75	6.42	8.19
c_4	5.52	4.55	3.58	5.61	8.67
c_5	3.052	3.27	2.18	2.66	4.57
c_6	5.07	4.1	3.13	5.18	6.78

Table 5.14: Normalized Design Criteria Weight

	Algorithms				
Criteria	Risk-N1	Risk-N2	Risk-N3	Risk-P	Risk-A
c_1	0.187	0.1851	0.2182	0.1927	0.204
c_2	0.186	0.184	0.182	0.160	0.214
c_3	0.1884	0.176	0.1778	0.2029	0.166
c_4	0.17	0.17	0.17	0.17	0.18
c_5	0.098	0.125	0.1034	0.098	0.094
c_6	0.16	0.15	0.15	0.163	0.139

$$\begin{array}{cccccc} c_1 & c_2 & c_3 & c_4 & c_5 & c_6 \end{array} \\ \begin{bmatrix} 0.1974 & 0.185 & 0.182 & 0.172 & 0.1008 & 0.1524 \end{bmatrix} \tag{5.14}$$

Step 6: Evaluation of Alternatives Once the criteria weight are available the subject matter experts of landing gear system can evaluate and provide the scores of the various alternatives under study for the door actuators of landing gear system. As previously mentioned in Chapter4, we use a simple weighting notation composed of only $++, +, +-, -, --$, with each sign representing one cardinal weight. The weighting notation can vary and each notation may correspond to an cardinal number. Table 5.15 shows the alternative evaluation matrix for landing gear actuators.

Table 5.15: Door Actuator Alternatives

Criteria	**Weight**	**Pneumatic**	**Electric**	**Hydraulic**
Functionality	**0.1974**	$+(74.4)$	$++(85,6)$	$+(72.4)$
Speed of Operation	32%	$++$	$+$	$+$
Weight	18%	$+$	$++$	$+-$
Endurance	30%	$-$	$++$	$+$
Power consumption	20%	$+$	$+-$	$+-$
Initial cost	**0.185**	$++(93)$	$++(87)$	$+-(54)$
Acquisition cost	65%	$++$	$+$	$-$
Delivery time	35%	$+$	$++$	$+$
Durability	**0.182**	$+-(50)$	$++(97)$	$++(90)$
Tol. to harsh conditions	30%	$--$	$++$	$+$
Tol. to water	15%	$++$	$+$	$++$
Tol. to electrical charges	20%	$+$	$++$	$+$
Tol. to shocks	20%	$--$	$++$	$++$
Duration of life	15%	$+-$	$++$	$+$
Dependability	**0.172**	$+(76)$	$++(86)$	$++(84)$
Rate of failure	40%	$++$	$+$	$+-$
Tol. to faults	30%	$+$	$+$	$++$
Potential hazards	30%	$-$	$++$	$++$
Maintainability	**0.1008**	$++(100)$	$++(88)$	$--(20)$
Time requirements	40%	$++$	$++$	$--$
Cost of maintenance	60%	$++$	$+$	$--$
Ease of integration	**0.1524**	$+(75)$	$++(85)$	$+(65)$
S/W requirements	50%	$++$	$++$	$+$
H/W requirements	25%	$+$	$+-$	$-$
Interface requirements	25%	$--$	$+$	$+-$
Net Score $\sum w$	100	75.57	87.26	67.4

Similarly, the analysis of alternatives for the landing gear extension and retraction actuators is carried out. Table 5.16 shows the the alternative evaluation matrix for the landing gear extension and retraction actuators.

Table 5.16: Landing Gear Extension and Retraction Actuators

Criteria	Weight	Pneumatic	Electric	Hydraulic
Functionality	**0.1974**	+(64.8)	++(79, 2)	+(78.4)
Speed of Operation	32%	++	+−	+
Weight	18%	+−	++	+−
Endurance	30%	−−	++	++
Power consumption	20%	+	+−	+−
Initial cost	**0.185**	+(80)	+(60)	+(80)
Acquisition cost	65%	+	+−	+−
Delivery time	35%	+	+−	++
Durability	**0.182**	+(70)	+(72, 6)	++(100)
Tol. to harsh conditions	30%	+−	+	++
Tol. to water	15%	++	+	++
Tol. to electrical charges	20%	+	++	++
Tol. to shocks	20%	+−	+	++
Duration of life	15%	+−	++	++
Dependability	**0.172**	+(70)	++(86)	++(92)
Rate of failure	40%	++	+	+
Tol. to faults	30%	+−	+	++
Potential hazards	30%	−	++	++
Maintainability	**0.1008**	+−(52)	+(80)	+−(44)
Time requirements	40%	++	+	+
Cost of maintenance	60%	−−	+	−−
Ease of integration	**0.1524**	+(75)	++(90)	+80
S/W requirements	50%	+	++	+
H/W requirements	25%	+	+	+−
Interface requirements	25%	+−	+	++
Net Score $\sum w$	100	69.04	76.51	80.92

5.4 Requirements Traceability

Requirement traceability information is necessary for various activities in the project life cycle. Requirement traceability benefits and its usage are explained in Chapter 3. In this chapter, we have not detailed the various stages of the project development, particularly aspects around the the V&V have not been presented. In our tool SysEngLab the traceability is provided by two means, implicit and explicit. Implicit traceability information is generated as the by-product of development process itself. Explicit traceability is provided by tool through allowing to create link between different types of artifacts explicitly.

5.4.1 Purposed Traceability

Previously, information regarding the perceived importance of goals by different stakeholders was elicited for the various goals in Eq.(5.2) and (5.3). The traceability needs of stakeholders can also be elicited in similar manner. Previously in Chapter 3, Eq.(3.4) mentions modeling the traceability preferences of stakeholders.

$$Str\_P = \begin{array}{c} \\ r \\ u \\ l \\ s \end{array} \begin{array}{c} \begin{array}{ccccc} G_1 & G_2 & G_3 & G_4 & G_5 \end{array} \\ \begin{bmatrix} 2 & -1 & 2 & 1 & -1 \\ 2 & 1 & 1 & 2 & -2 \\ 2 & 0 & 2 & 1 & -2 \\ -2 & -2 & 1 & 2 & 2 \end{bmatrix} \end{array} \tag{5.15}$$

With this stakeholder-traceability preference matrix, the stakeholders are provided with only the traceability information they are interested in. Researchers are given the traceability of $goal_1$, $goal_3$ with very fine granularity and $goal_4$ in coarser and $goal_2$ and $goal_5$ even more coarse. Similarly for University, Laboratory and students.

5.4.2 Cost-effective Traceability

Cost effective traceability necessitates creation of a matrix $St - R$ mentioned in Eq.(3.3).

$$weight * Str\_P = \begin{array}{c} \\ r \\ u \\ l \\ s \end{array} \begin{array}{c} \begin{array}{c} w \end{array} \\ \begin{bmatrix} 0.32 \\ 0.26 \\ 0.26 \\ 0.16 \end{bmatrix} \end{array} \begin{array}{c} \begin{array}{ccccc} G_1 & G_2 & G_3 & G_4 & G_5 \end{array} \\ \begin{bmatrix} 2 & -1 & 2 & 1 & -1 \\ 2 & 1 & 1 & 2 & -2 \\ 2 & 0 & 2 & 1 & -2 \\ -2 & -2 & 1 & 2 & 2 \end{bmatrix} \end{array} \tag{5.16}$$

$$weight * St\_R = \begin{array}{c} \\ r \\ u \\ l \\ s \end{array} \begin{array}{c} \begin{array}{c} w \end{array} \\ \begin{bmatrix} 0.32 \\ 0.26 \\ 0.26 \\ 0.16 \end{bmatrix} \end{array} \begin{array}{c} \begin{array}{ccccccc} R_1 & R_2 & R_3 & R_4 & R_5 & R_6 & R_7 \end{array} \\ \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array} \tag{5.17}$$

Already calculated weight of the matrix in Eq.(5.1) provides an estimate of the potential benefits that tracing of a particular of rationale may provide.

5.4.3 Pre-requirement traceability

Pre-requirement traceability is calculated by tracing back the requirements to the stakeholder and their rationales. Table 5.17, shows the traceability achieved between the stakeholders and the corresponding rationales and the leading goals. Table 5.18, shows the traceability achieved between the customer requirements depicted as objectives and the corresponding goals and stakeholders.

Table 5.17: Stakeholder Rationale Goal Traceability Matrix

$Sth/Rtnl - Goal$	R1	R2	R3	R4	R5	R6	R7	G1	G2	G3	G4	G5
ST_1	x	x	x	x				x		x	x	
ST_2						x	x					x
ST_3			x	x	x			x	x			
ST_4				x	x				x		x	

Table 5.18: Objectives Stakeholders Goals Traceability Matrix

Ob/Stb	ST_1	ST_2	ST_3	ST_4	G_1	G_2	G_3	G_4	G_5
O_1	x		x	x	x				
O_2	x		x	x	x				
O_3	x		x	x	x				
O_4	x		x	x	x				
O_5	x		x	x	x				
O_6	x		x	x		x			
O_7			x	x		x			
O_8			x	x		x			
O_9			x	x		x			
O_{10}			x	x		x			
O_{11}	x			x				x	
O_{12}	x			x	x			x	
O_{13}	x			x				x	
O_{14}	x			x				x	
O_{15}	x	x					x		x
O_{16}	x	x					x		x
O_{17}	x	x					x		x
O_{18}	x	x					x		x
C_1			x						
C_2			x						
C_3				x					

Table 5.19 the pre-requirement traceability achieved between the system requirements and the corresponding objectives representing the user/customer requirements, and stakeholders which account for the particular system requirement.

Table 5.19: Pre-requirement Traceability

SR/Obj_ST	O_1-O_5	O_6-O_{10}	O_{11}-O_{14}	O_{15}-O_{18}	ST_1	ST_2	ST_3	ST_4
_LG1			$O_{13,14}$		x			x
_LG2			O_{14}		x			x
_LG3			O_{14}		x			x
_LG4			O_{14}		x			x
_LG5			O_{14}		x			x
_LG6		O_{10}	O_{14}		x		x	x
_LG7		O_{10}	O_{14}		x		x	x
_LG8		O_{10}	O_{14}		x		x	x
_LG9		O_{10}	O_{14}		x		x	x
_LG10		O_{10}	O_{14}		x		x	x
_LG11		O_{10}					x	x
_LG12			O_{12}		x			x
_LG13		O_{10}	O_{12}		x		x	x
_LG14		O_{10}	O_{12}		x		x	x
_LG15		O_{10}	O_{12}		x		x	x
_LG16			O_{14}		x			x
_LG_Mi1	$O_{1,2}$		$O_{11,13}$		x		x	x
_LG_Mi2	$O_{1,2}$		$O_{11,13}$		x		x	x
_LG_Mi3	$O_{1,2}$		$O_{11,13}$		x		x	x
_LG_Mi4	$O_{1,2}$		$O_{11,13}$		x		x	x
_LG_Mi5	$O_{1,2}$		$O_{11,13,14}$		x		x	x
_LG_Mi6	$O_{1,2}$		O_{11}		x		x	x
_LG_Mi7	$O_{1,2}$		O_{11}		x		x	x
_LG_Mi8	$O_{1,2}$		O_{11}		x		x	x
_LG_Mi9	$O_{1,2}$		O_{11}		x		x	x
_LG_Mi10	$O_{1,2}$		O_{11}		x		x	x
_LG_Mi11	O_2		O_{11}		x		x	x
_LG_Mi12	O_2		O_{11}		x		x	x
_LG_Mi13	O_2		O_{11}		x		x	x
_LG_Mi14	O_2	O_{10}	O_{11}		x		x	x
_LG_Mi15	$O_{1,2}$	O_{10}	O_{11}		x		x	x
_LG_Mi16	$O_{1,2}$	O_{10}	O_{11}		x		x	x
_LG_Mi17	$O_{1,2}$	O_{10}	O_{11}		x		x	x
_LG_Mi18	$O_{1,2}$	O_{10}	O_{11}		x		x	x
_LG_Mi19	$O_{1,2}$	O_{10}	O_{11}		x		x	x
_LG_Mi20	$O_{1,2}$		O_{11}		x		x	x
_LG_Mi21	$O_{1,2}$	O_{10}	O_{11}		x		x	x
_LG_Mi22	$O_{1,2}$	O_{10}	O_{11}		x		x	x
_LG_Mi23	$O_{1,2,3}$	O_{10}	O_{11}		x		x	x
_LG_Mi24	$O_{1,2,3,4,5}$		O_{11}		x		x	x
_UI_LG1			$O_{11,13,14}$		x			x
_UI_LG2	O_5		$O_{11,13,14}$		x		x	x
_UI_LG3			$O_{11,13,14}$		x			x
_UI_LG4	O_5		$O_{11,13,14}$		x		x	x
_UI_LG5	O_5		$O_{11,13,14}$		x		x	x
_UI_LG6			$O_{11,13,14}$		x			x

Table 5.19 – continued from previous page

SR/Obj_ST	O_1-O_5	O_6-O_{10}	O_{11}-O_{14}	O_{15}-O_{18}	ST_1	ST_2	ST_3	ST_4
_UI_LG7			$O_{11,13,14}$		x			x
_UI_LG8			$O_{11,13,14}$		x			x
_UI_LG9			$O_{11,13,14}$		x			x
_UI_LG10	O_5		$O_{11,13,14}$		x		x	x
_UI_LG11			$O_{11,13,14}$		x			x
_UI_LG12			$O_{11,13,14}$		x			x
_UI_LG13			$O_{11,13,14}$		x			x
_UI_LG14			$O_{11,13,14}$		x			x
_UI_LG15			$O_{11,13,14}$		x			x
_UI_LG16	O_5		$O_{11,13,14}$		x		x	x

5.4.4 Post-requirement traceability

Post requirements traceability provides the links between the requirements and the other artifacts generated as part of the development activity. Table 5.20, shows the post requirement traceability for the landing gear system. The traceability matrix shows the landing gear system requirements matched against the design components of the landing gear system.

Table 5.20: Post-requirement Traceability

DE/SR_	LG1	LG2	LG3	LG4	LG5	LG6	LG7	LG8	LG9	LG10	LG11	LG12	LG13	LG14	LG15	LG16
Weight Sensor	x										x				x	
Landing set	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Wheels	x	x	x	x		x								x		
Electro valve	x	x	x	x	x	x	x									x
Hydraulic Pump	x	x	x	x	x	x	x									x
Ext-Ret Actuator	x	x		x	x	x		x								x
Brakes			x	x				x		x						
Steering	x			x						x						
Actuator																
LG controller		x	x	x	x	x				x	x	x	x	x	x	
Wheel Box									x		x					x
LG-Ret sensor						x					x	x	x			
Door-Open Sensor						x					x	x	x			x
Door-Close sensor						x					x	x	x			x
LG-Ext sensor						x					x	x	x			x
Door Actuator	x		x	x	x	x										x

5.5 Limitations and Conclusions

The approach proposed provides enthusiastic results, but its findings should be taken into account with the inherent limitations and assumptions made to realize this case study. As the IBIS project is in its first iteration of development, only the conceptual modeling has been carried out with a few of the simulations using the M&S platforms. The best criticism of the study could be that it is an academic project and only a very small team is in charge of its development. The members already are familiar with each other and have strong trust in each other's jurisdiction. Only minor conflict issues regarding the choice of aircraft systems capabilities rose in beginning, which were sorted out according the budget in hand. Another limitation regarding the results of the requirements traceability achieved is about the data set. The data set available for deriving traceability was available on a single workstation and not distributed and scattered like an actual work environment. The traceability generated remained limited to the first iteration of conception.

A case study on an aircraft simulator system called IBIS is presented, upon which various methods proposed in previous chapters are applied to demonstrate their ease of application. Goal model, responsibility models, strategy models and other requirements artifacts of the system are created using various diagrams of *CReML* through *SysEngLab* tool. User stories are elicited from the various potential stakeholders. High level rationale map is created using the rationales retrieved from the user stories of the stakeholders. Goals are derived from the thus obtained user stories. Goals are divided into smaller objectives which represent the customer requirements. Objectives representing the qualitative requirements can be rendered more measurable and quantified by proposing test cases against them. An ambiguous customer requirement can be treated by suitably using the negation or a test case with negation.

Traceability matrix of the IBIS project is generated automatically by the tool. Preference modeling over the traceability needs is also shown using the tool. Various objectives in which their is a possibility of conflicts are marked using the conflict relationship upon their analysis, the requirements are negotiated to resolve the conflicts. The decision for the various types of design solutions are made using the *URoW* technique. The results are encouraging but still there are few issues like: difficulty in judging precisely the definition of 'suitable' when using the negation, for instance developer may choose to use negation systematically for quality requirements, it is hard to tell where to stop. Choosing the weight generation function can be quite tricky and puts a big responsibility over the project manager. *CReML* on the other hand proves to be quite useful to define requirements and other artifacts. A comparative study of the *CReMl* and other Gore based languages needs to be made to precisely evaluate the advantages and disadvantages in deep. The ability to provide abstraction at different levels allows to model large number of requirements and still rendering them comprehensible. Judging the cost & effort involve in generation of traceability information is still an approximation and depends upon the judgement of requirement engineer involved.

Conclusion and Future Perspectives

The research conducted as part of this thesis develops a comprehensive methodology for the conception of the complex systems. This methodology aims to be the appropriate solution for the new paradigm of the highly complex systems demanded by the consumers. The consumer of these highly complex systems range from public to private organizations with systems ranging from medical & health systems, public transport systems, communication systems, defense and military systems, etc., covering nearly all aspects of modern lifestyle.

High degree of complexity and interdependency puts enormous pressure on the system architects and engineers. This pressure upon the architects, often leads to the poor conception or design flaws, owing to the lack of tools and methods apt to handle this situation. The complexity of the products come from the large numbers of requirements, which need to be satisfied by the system in order to be acceptable to the customer.

As mentioned in Chapter 2, requirement engineering activities are the foundation stone for the product development. They lone can determine the success or failure of products. A poorly organized requirements elicitation process might lead to faulty or incomplete set of requirements or hard to understand or unrealizable requirements. Their proper management can ensure the development of right product in right way, with everything which is indispensable to the customer. There are some issues linked to management of requirement mentioned in Chapter 3 such as, traceability of requirements, purposeful traceability, etc., which when addressed can help to handle this complexity by organizing it, and hence providing multiple benefits. Certain difficulties of the SE have origins in the weak decision making process, as mentioned in Chapter 4. Decision making process also needed to be improved to improve the quality of the product while precisely satisfying the customer requirements.

Various shortcomings and challenges were identified in the form of questions (Chapter 1) from the existing systems engineering activities and methodologies. Relevant axes were identified and researched extensively. For each identified challenge, holistic and pertinent solutions were identified. Appropriate techniques and methods were developed and integrated to form this holistic and comprehensive solutions. Previous chapters have detailed the researched presented. Requirements engineering, requirements management & traceability and decision making are researched and the finding and the contributions are presented.

Summary of Findings & Contributions

We presented a GORE based technique. The proposed graphical modeling language which is capable of functionalities typical to popular GORE techniques like i* and

KAOS and other functionalities which are of concern to systems engineers and other stakeholders. Proposed language and supporting tool allows to represent the preferences of the various stakeholders on the various goals and objectives. It allows to model both the core and optional features of the system under study. The goals can be traced back to the user stories which are linked to the goal modeling diagram. The responsibility and interaction among the agents is separately modeled and can be integrated if the developer wishes. The other interesting capability our tool provides is to model the rationales using view-points. The stakeholder rationales are projected and divided into various viewpoints from the very early stage, which allows to better understand the stakeholder requirements. The end-product of goal modeling leads to system requirements which can be allocated to the UML/SysML diagrams. Our tool supports a few of the diagrams of the UML/SysML notably Use-case diagram and Block definition diagram. This is to provide direct traceability throughout the V-cycle.

Aspects regarding the ambiguity and understandability of requirements are researched and a negation technique for writing quality or hard to understand requirements is presented. Restricted negation technique offers interesting aspects as a tool against the ambiguity and understandability problem in context of RE. We argued that for some notions such as quality, it is comparably easier to elaborate using negative sentences, than with purely affirmative ones. Our empirical findings have reaffirmed that for some concepts negation is more natural than affirmation. In the context of requirement engineering, it can be used successfully as it seems more natural and intuitive. Our examples have manifested that, a good blend of affirmative and negative sentences can provide in depth view implications of a quality requirement. Our interviews confirm that the user may prefer negative phrases and they may come intuitively. We argue that if something seems more natural, it is easily understood and accepted. Negation are natural in a natural language, so we should not hesitate to use them as a tool. Finally, we say that our technique improves the readability of requirement documents. The IBIS case study carried out confirmed our claims regarding negation, and their usage for design of test cases.

We present in this thesis a comprehensive approach for the requirement traceability for a systems engineering project. We give the distinction between the requirement traceability for software intensive system and hardware intensive systems. We use the existing literature for understanding the needs of comprehensive requirement traceability. We identify trace blocks to have minimum eight distinct types of values, which can provide the traceability, throughout the Vee-model of project development. We provide a platform to link these trace blocks with various relationships to the artifacts generated throughout the life-cycle and the requirements. We have seen that a comprehensive and reliable traceability link establishment is a two way process: forward and backward. Requirement traceability information remains a valuable entity for the enterprises involved in huge projects where new projects may find inspirations from the earlier projects. Requirement traceability information remains valuable even after the project ends and the product is removed from the service. Our approach for laying down the trace signs using the rationales

throughout the life cycle allows to link the artifacts in a systematic manner.

The basic idea of our methodology is to make the requirement traceability process more and more reactive and formal with precise semantics of relationships. We provided means to integrate our approach cohesively with tools supporting UML/SysML. We have seen that requirement traceability is iterative process. A semi-automatic approach is better choice for carrying out requirement traceability, with suitable human involvement. A carefully carried out requirement traceability activity is a huge source of knowledge for the entire enterprise, for the current and future projects including the entire life cycle of the product. We defined the semantic relationships between the various requirements artifacts. Our approach can be used as a tool for organizing complexity in the composite system design by providing ready to use traceability from previous step. Still, overcoming the social issues of traceability problem remains a challenging task and more research work needs to be done to address the traceability challenges.

This thesis also presents a new approach for traceability maintenance scheme, trying to address chief problems of current trace processes. The proposed traceability model emphasizes on maintenance with efficient maintenance schemes, developed tool implements our technique, and we have obtained a few of the results and observations which support our claims. Our technique provides interesting solution to the dangling trace problem, which can immensely help to reduce the tediousness of tracing process. Our solution offers a plausible solution to the information-loss problem as the information ever generated in the development process remains in system to provide the exact trace of evolution of the system. With the ease in trace maintenance process the cost of maintenance can be reduced noticeably as the dangling pointer problem is solved the effort in maintenance is reduced and hence less time and less human resources are engaged to do the same task. Usually in the system development process there are numbers of iterations before an artifact is finally accepted as a part of the system, our technique allows retaining the information regarding iterations and chronological evolutions and hence helps in better decision making.

We have shown that the different set of requirement traceability block can be used for different product management activities: configuration management, change impact management, maintenance, system up-gradation, product reuse, release management, system comprehension, etc. Such a use of set of trace blocks eases these tedious jobs, as the required data is easily available, and ultimately lowers resource consumption. Usually graph becomes large and hard to understand, our technique can be constrained to map intra-level traceability, reducing size and increasing the understandability of graph. Our technique can be evolved further to enable global distributed traceability.

In this thesis, we have also provided a systems theory of how the criteria weights can be obtained using the classical theory of preference modeling. Our technique Utility Rank Order Weighting (UROW) technique, organizes properly the influence and interest of stakeholders, seeks their preferences and before using them validates through a simulation. This approach provides multiple benefits with compared to

other existing approaches. Usually in systems engineering project, the engineers rely upon their intuition to provide weights, and later use other technique to combine the different decision makers' preferences. Our approach provides a formalism to this systems engineer intuition and hence provides the reasoning for the various weights achieved. Our approach is very easy to understand and use, and demands very low cognitive load from the engineers and stakeholders. It allows to formally provide the scores using the DM' drawn utility functions: risk prone, risk averse, or risk neutral; it provides a mechanism to combine them together to come up with a uniformly acceptable solution. In future, we look forward to link the simulation of the decision makers' preferences with the design library, in order to shorten the decision time. Our approach can easily be applied to the class of methods which require information on the attributes to carry out a decision analysis.

Other significant contribution of this thesis is about integrating the previously mentioned contributions in a way to provide a methodology to carry out the requirements engineering and decision making activities. The planned stepwise execution of the various tasks and their integration is provided and applied on a case study and the results are shared.

Future Research and Perspectives

Requirements engineering

As our research regarding the use of negation remained limited to only English and French language with more on occidental cultural boundary. More empirical studies need to be carried out to validate the use of negation in various other languages and cultures. We look forward to using this technique in a graduate level project at the university, involving multiple teams and multiple clients expressing their requirements.

Further research can be carried out in use of uniguous semantic language translation techniques, although it will demand expertise in both linguistics and informatics. RE can derive lot of benefits from the uniguous semantics translation as it could easily detect the multiple translations of one natural language text to another restricted vocabulary natural language or formal languages.

Further research can be carried out in representation of graphs, such that big and complicated graphs can be presented and understood by the user. Abstraction of information from the graphs is one solution for the scalability issues, but still it can be argued that abstraction from the highly complex graphs could be very difficult task itself.

Requirements Traceability

For our proposed requirements traceability scheme more empirical studies need to be conducted to validate the benefits from the various propositions made such as planning for purposed and cost effective traceability.

In spite of these facts there are other issues which need to be addressed like heterogeneous traceability schemes for capturing informal aspects.

Our traceability maintenance approach can be suitably coupled with event based approach as described by Cleland-Huang *et al.* [Cleland-Huang 2003]. As our case study is not based a collaborative environment, it needs to be further studied in a collaborative environment and integrated with the event based approach, which seems to be most appropriate for a collaborative maintenance.

We cannot still trace 100% of information as it is always difficult to trace the informal aspects of many artifacts. We advocate the usage of semi-automatic trace mechanism with event specific human intervention for the optimal benefits of traceability. Chapter 3 attempted to provide a small step towards approaching the traceability challenges. The fully reliable automation of comprehensive traceability in a systems engineering project remains a challenging activity yet for the requirement engineering. The information retrieval based traceability recovery policies still have issues with complete recall and a satisfactory precision. Traceability approach different then information needs to be researched, such as ontology based requirements traceability. More research on ontology based can traceability and its optimal integration to the various tools and platforms of the engineering needs to be researched more.

There are still issues like increasing the value of trace and methods to augment the usability of trace in organization and how to holistically link the various aspects of system development with the traces. Can we utilize traces for rapid development process? Can traceability patterns be used for product development? These are the numerous issues which need to be addressed by research communities.

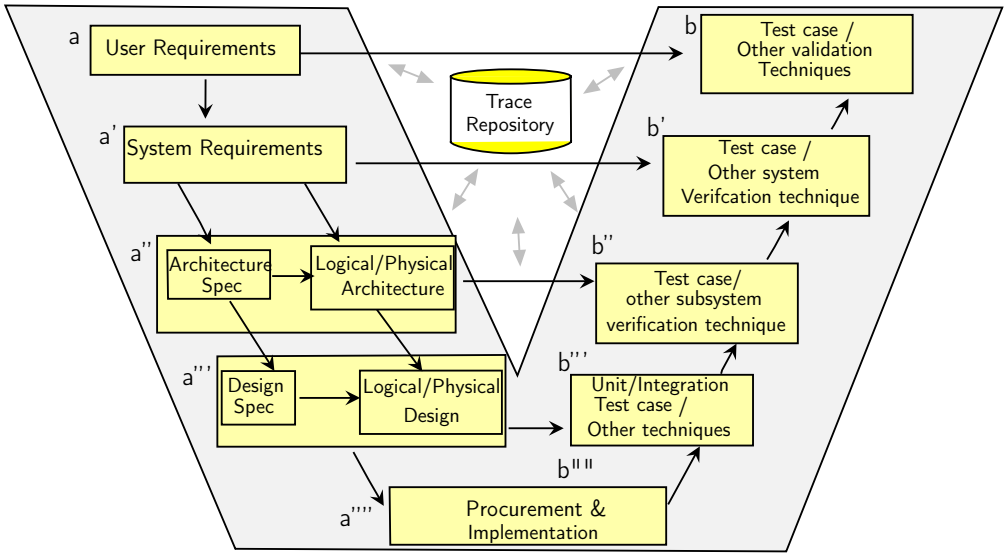
Requirements traceability research needs more empirical studies and data to determine and estimate the cost and effort of traceability, however we have some idea how to do this.

Estimating Effort & Cost of Traceability

Calculating cost and effort poured into the traceability need to be estimated or predicted before kick-starting the project. As nowadays computing memory and speed cost doesn't really bothers the projects. But the amount of time poured in for creating traces can be significant. A prescription of rationale-viewpoint based matrices is proposed for very fine granularity, but for certain projects more matrices can be envisaged and or reduced depending upon traceability needs. For estimating the effort of traceability, we need to provide a unit for it.

To actually able to estimate the cost and effort of traceability throughout a project, it is essential to take into account the efforts poured for linking the artifacts during the various stages of conception and development. Various factors need to be taken into account for estimating the traceability such as: number of tools/editors used to conceive and develop the system, number of stakeholders, number of user-stories/interview documentation available, number of requirement artifacts, test artifacts, expected granularity, size of code, etc. The traceability efforts estimation

needs to sum up all the efforts starting from requirements elicitation till system validation.



Requirements Traceability Effort Estimation

Estimating requirements traceability effort can be summarized in the above schema, where *a, a', a", a"', a""* represent the efforts poured in mapping traceability information vertically in the Vee-cycle, and *b, b', b", b"', b""* represent the effort poured in maintaining traceability horizontally.

Terms *a, b, a', b',* etc., are dependent on several factor and can be roughly estimated according to the project traceability needs such as granularity, views, etc., *a, b, a', b'* can be roughly estimated by the equations below:

$$a = (Number\_of\_tools).(Number\_of\_lines\_of\_user\_stories).(Number\_of\_requirements\_artifacts).(Number\_of\_stakeholders)$$

$$b = (Number\_of\_tools).(Number\_of\_lines\_of\_user\_stories).(Number\_of\_requirements\_artifacts).(Number\_of\_stakeholders).(Number\_of\_Validation\_Testcases)$$

$$a' = (Number\_of\_tools).(Number\_of\_requirements\_artifacts).(Number\_of\_system\_requirements\_artifacts).(Granularity).(Number\_of\_Validation\_Testcases)$$

$$b' = (Number\_of\_tools).(Number\_of\_requirements\_artifacts).(Number\_of\_system\_requirements\_artifacts).(Granularity).(Number\_of\_Verification\_Testscases)$$

Similarly, *a", b", a"', b"', a"", b""* can be roughly estimated. More works needs to be carried out in refining these equations according the traceability demanded. Working in this direction towards evaluating the effort needed to create the matrices, we define a *traceability cost unit* (TCU), as amount of effort needed to create a 10×10 sized $R - R$ matrix, i.e., *rationale-rationale* matrix. We used $R - R$ matrix for baseline because, rationales are govern by the law's of nature: laws of physics, laws of economics, laws of chemistry, etc., and hence relationship derived are logical ones. If one TCU is represented as ϱ, cost of creating $St - R$ or any other matrix can be given by the Equation below.

$$Cost = \mathbf{K} \times \varrho$$

where **K** is some constant, and depends upon factors such as: the type of matrix (who-why, where-how, etc.,), the quality of human resource participating, the number of rows and column.

The total cost of implementing traceability in a project would be sum of all cost incurred in establishing the matrices in the whole life cycle, i.e., project traceability cost $= \sum_l^L Cost_l$. This equation allows to compare the cost incurred by implementing different traceability policies in the project. The system engineers can then choose the right traceability policy according to their need and budget.

Decision Making

Although, we provided a methodology for multi criteria decision making. There are still issues on elicitation of preferences from the stakeholders such as: how much should be the difference between two degree of preferences and how to quantify them? how to elicit preferences of numerous stakeholders in a collaborative environment on-line ? Does elicitation in a collaborative environment impacts the preferences of the stakeholders ? Are these impacts beneficial for the group decision making or not ? These questions need to be answered with empirical studies. Decision making process cannot be a purely mathematical problem, it certainly needs help from other discipline such as social sciences and psychology. We have proposed simulation as a tool for validating the preferences and evaluation of macro family of solutions, but still there are issues such as whether , the simulation results lead to change in the preferences or perceived value by the stakeholders ? Some more empirical studies needs to be conducted in different scenarios to answer these questions.

Other Issues

Apart from them, few aspects of system design and modeling were also researched but kept aside from the main stream of the research. The development of reactive system modeling using UML/SysML needs to be researched in order to provide solution for the widely popular problem of rework in industry. We tried to seek insights this problem [Shukla 2012a], provided some methods, which are not mentioned in this thesis but are still relevant and need to be researched more in depth.

In future we look forward to implement and integrate all the structural and behavior diagrams of UML/SysML in our tool to make it more comprehensive and useful. We look forward to automating more and more the trace creation process using model transformations, as in model based systems engineering. In future we look forward to research the use of lattices for seeking extra benefits from the traceability information.

Appendix A

Tools Developed

A.1 SysEngLab

As previously mentioned, different tools were developed for implementing the theoretical developments. Tool Support is developed to encompass the various stages of Vee-life cycle. The platform upon which all the different developed tool modules are integrated is called ***SysEngLab***. A software module for supporting requirements and other RE activities is called ***RequirementLab***, which implements the CReML and the proposed traceability approach in Chapter 2 and 3 respectively. Another developed module is called DecisionLab which implements the proposed decision technique, ***UROW***, in Chapter 4. The platform is developed in Java and generates different kinds of reports in industrial and academic popular formats: Word, Pdf and Latex formats.

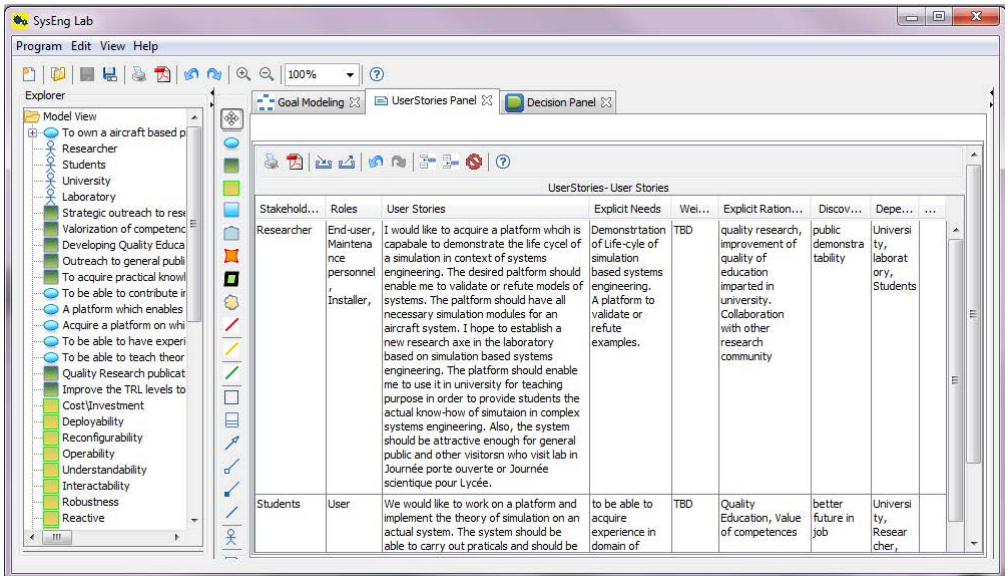


Figure A.1: SysEngLab Console

A.2 RequirementLab

RequirementsLab implements the CReML meta-model described in Chapter 2, it also allows to store different user-stories collected in tabular form. The requirement diagrams are implemented with ready to use requirement templates suitable

for different categories of requirements. The requirements traceability creation and maintenance approach is built in the tool, with additional tagging features to allow traceability. The traceability preference for different stakeholders can be taken into account, which allows to provide the appropriate demanded traceability. RequirementsLab is designed in a manner to support scalability issues with the graphs used during the requirements modeling. It equally generates the traceability matrices demanded by the particular stakeholder. Core and optional features can be recorded from the very early stage of the requirements elicitation. Conflicting requirements can be resolved using the weighting of the requirements. Its GUI is coded using the Java mxGraph library and documentation is generated using iText library.

A.3 DecisionLab

DecisionLab is integrated as a tool support for decision making process. A decision support system, which can function independently itself and in integrated mode with RequirementsLab. It is designed to analyze stakeholders, collect their preferences over different issues and make decisions. It supports storing the decisions made previously and hence supports the decision traceability. Its GUI is coded using the mxGraph and java Swing libraries, reports are generated using the iText library. Currently only the UROW technique is implemented in the tool, but other MCDM techniques are also being considered for integration in the tool.

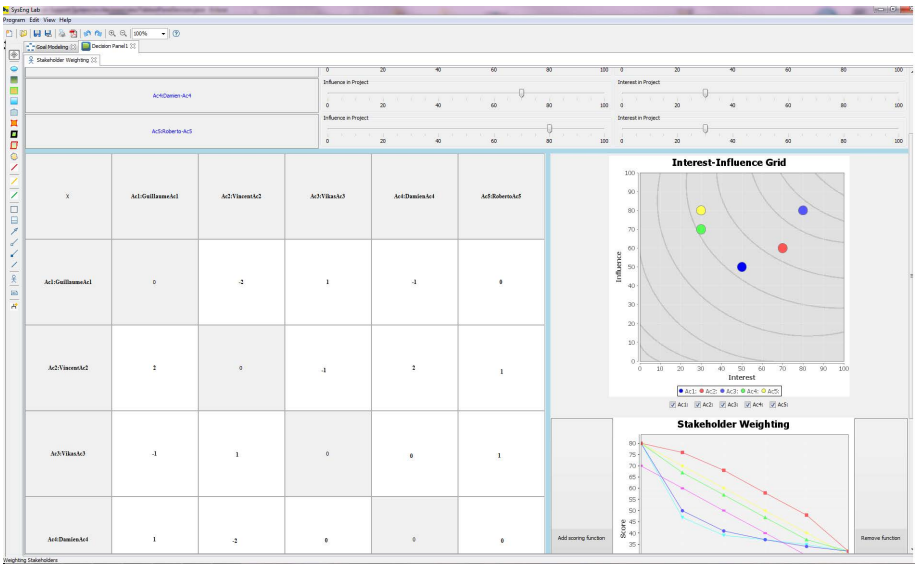


Figure A.2: DecLab module in SysEngLab Console

Appendix B

Résume en Français: Approche Globale d'Ingénierie Systèmes

Contents

B.1 Introduction 199
B.2 Systèmes Complexes 200
B.3 Ingénierie Systèmes 200
B.4 Ingénierie des Exigences 202
B.5 Gestion des Exigences et de la Traçabilité 204
B.6 La Prise de Décision et Résolution de Conflit 205
B.7 Approche Globale 207
B.8 Conclusion 209

B.1 Introduction

Récemment, il y a eu une forte augmentation de la complexité de notre environnement immédiat dans lequel nous vivons. Nous nous trouvons entourés de nombreuses technologies, pour mener à bien notre journée normale de tâches quotidiennes. Le taux d'induction de ces nouvelles technologies à notre environnement a nous pris par surprise. Notre routine quotidienne par rapport à nos prédécesseurs immédiats a énormément changé. A tout instant, nous pouvons accéder à l'information facilement, et nous sommes devenus à la fois consommateur et producteur de l'information. Avec les dernières innovations technologues rien ne semble impossible. Avec ces toutes nouvelles technologies radicalement innovantes, nous pouvons réaliser ce qui était autrefois considéré comme impossible. Ces nouvelles technologies ont donné naissance à de nouveaux besoins pour les systèmes complexes, nous avons besoin de systèmes plus capables, plus sûr, plus robustes... Contrairement à cela, nos processus actuels de développement de produits ne sont pas encore équipés pour exploiter complètement les avantages de ces nouvelles techniques et technologies afin d'offrir la complexité organisée exigé dans les produits.

En outre, avec augmentation de la complexité des produits demandés, les techniques traditionnelles utilisées pour les développer sont souvent jugés inaptes ou manquent en aspects multiples. Les techniques, les outils et les ressources humaines utilisées pour les développer restent sous une pression énorme. Il est évident que,

de nos moyens de développer ces systèmes très complexes sont loin d'être prêt ou apte à les développer. Nous devons apportée tout nouvel ensemble de mises à jour compétentes des méthodologies, des processus et des outils pour développer des nouveaux systèmes et technologies très complexes, qui sont exigés aujourd'hui et dans le proche avenir. Il y a un besoin urgent de réformer en profondeur nos méthodologies existantes de conception de produits pour bénéficier de ces technologies et d'offrir beaucoup plus, mieux et plus vite.

B.2 Systèmes Complexes

Les humains ont développé des machines pour faciliter leur vie quotidienne. Peu à peu, les humains ont appris à utiliser des groupes de ces machines ensemble pour résoudre les problèmes encore plus grands. Ce groupe de machines interaction est appelé système. Avec l'évolution, les humains continué d'apprendre à utiliser ces groupe de systèmes primitifs ensemble pour accomplir des tâches relativement plus difficiles. Moment où les humains entremêlés ces systèmes simples tel que résultant système n'était pas plus compréhensibles complètement, système complexe sont entrés en existence.

Le système complexe est en soi une notion relative, en fonction de l'observateur. Un système peut sembler complexe pour un observateur et non à l'autre. Une combinaison d'éléments en interaction, organisé pour atteindre un ou plusieurs objectifs déclarés. Un ensemble intégré d'éléments, sous-systèmes ou ensembles qui permettent d'atteindre un objectif défini. Ces éléments comprennent les produits (matériels, logiciels, firmware), des processus, des personnes, des informations, des techniques, installations, services et autres éléments de soutien. Un système complexe est celui qui de par leur conception ou la fonction ou les deux est difficile à comprendre et à vérifier.

Un système complexe comporte de nombreux éléments différents, ces composants sont souvent interdépendants, et il y a généralement des interactions non linéaires, c'est à dire de petits changements peuvent apporter de grands effets et les grands changements peuvent conduire à de petits effets. Souvent un système complexe agit sur les connaissances locales et les conditions, c'est à dire, ce n'est pas contrôlée centralement. Il a un comportement délibéré et autonome. Les systèmes complexes ont tendance à avoir l'organisation hiérarchique. A systèmes complexes résout des problèmes complexes, il remodèle sans cesse son avenir collectif, présente la nouveauté et s'adapte, il apprend de son expérience et affiche un comportement émergent et peut-être inattendu et imprévisible. Les systèmes complexes sont à la fois organisés et variés.

B.3 Ingénierie Systèmes

Le domaine multidisciplinaire concerné avec le cycle de vie complet des systèmes est souvent associée à l'ingénierie des systèmes (IS). Ingénierie Systèmes est reconnue

comme un mécanisme privilégié pour établir un accord pour la création de produits et services qui seront échangés entre les deux organismes ou plus. Il peut être appliqué à tout type de développement du système : pour un appareil électroménager, un avion, une centrale nucléaire, etc. Bonne application des principes et méthodes d'IS permet de maximiser les chances de succès d'un projet.

INCOSE manuel définit comme l'ingénierie des systèmes est une approche interdisciplinaire et des moyens pour permettre la réalisation de systèmes. Succès d'IS se concentre sur la définition des besoins des clients et la fonctionnalité requise au début du cycle de développement, la documentation des besoins, puis de procéder à la synthèse de la conception et du système validation tout en considérant le problème complet : les opérations, les coûts et calendrier, la performance, la formation et le soutien, l'essai, la fabrication et la retraite. IS considère à la fois aux besoins commerciaux et techniques de tous les clients avec l'objectif d'offrir un produit de qualité qui répond aux besoins des utilisateurs.

En termes simples, l'approche IS se compose de:

- Identification et quantification des objectifs du système,
- Création de concepts alternatifs de conception du système,
- Performance des métiers de la conception,
- Choix et mise en œuvre de la meilleure conception,
- Vérification que la conception est bien construit et intégré, et
- Evaluation de la mise en œuvre du message de la façon dont le système répond (ou atteint) les objectifs.

Dans cette thèse projet d'ingénierie système est utilisé pour représenter les systèmes intensifs de matériel. Le terme ingénierie logicielle ou d'un projet de logiciel est utilisé pour représenter un système intensif de logiciels. Toutefois, il est incontestable que, le terme IS est applicable à la fois pour les systèmes basés sur matérielles et logicielles. Nous considérons qu'un projet soit d'IS, quand il dispose d'une majorité de la partie en sous-systèmes autres que logiciels. Nous utilisons le terme système intensif non-logiciel pour un projet IS et d'un système intensif de logiciels pour le projet de génie logiciel. Le processus d'IS est utilisé par itération à chaque étape du cycle de développement afin de générer des descriptions plus détaillées du système.

Ces descriptions comprennent ce qui doit être atteint (exigences fonctionnelles et concepts des opérations), comment cela doit être atteint (exigences de performance), la manière dont cela est atteint (conception), et les résultats d'analyse et de tests sur la capacité de satisfaire les exigences (vérification) et de satisfaire l'utilisateur (validation). Le processus d'IS conduit à des descriptions plus détaillées pour chaque phase du cycle de développement. Chaque phase commence à partir d'un niveau supérieur de description du système et décompose les exigences du niveau supérieur en exigences de niveau inférieur pour chaque fonction.

B.4 Ingénierie des Exigences

Les activités d'ingénierie des exigences (IE) commencent dès que les projets s'énoncent. Elle est réalisée dans une phase précoce d'un programme de développement de système pour comprendre le client ou les besoins des utilisateurs finaux clairement. IE implique un ensemble d'activités qui permettent de comprendre les besoins réels des utilisateurs finaux et les transformer en un ensemble contractuel des besoins des utilisateurs ou des exigences de l'utilisateur. C'est la phase la plus importante d'un projet, s'il est sûr que l'équipe d'IE a suscité tout le bon ensemble des besoins des utilisateurs dans le tout début, et si on leur donne les ressources appropriées dans les prochaines phases, le projet est destiné à réussir. Bien au contraire, si même une exigence trivial est oubliée ou mal compris à ce stade, il est sûr de coûter cher aux parties prenantes impliquées dans les phases ultérieures. Le plus tard on découvre un problème, plus il en coûte pour eux. Pour la phase d'IE, nous avons donné deux contributions importantes: pour l'écriture exigences en utilisant CReML et l'amélioration de leur qualité dans le sens de l'ambiguïté.

Les Contributions sur le Théorie des Exigences et CReML

La théorie des exigences stipule que les exigences sont projection de justifications de parties prenantes en produit avec leurs convictions. En d'autres termes, avec son ensemble de croyances et de ses justifications partie prenante arrive à un besoin ou à une exigence des parties prenantes. Une justification à son tour se compose de multiples points de vue qui sont d'intérêt pour les parties prenantes, mais ce fait est souvent caché à lui-même. La théorie développée dit que, lorsque nous passons des exigences des parties prenantes vers les spécifications du système, le degré de clarté de l'information augmente progressivement à partir de points de vue très abstraits vers les points de vue très spécifiques. Les exigences des parties prenantes peuvent être considérées d'une abstraction de justifications intervenants, qui peuvent être connues ou inconnues de lui. Un intervenant au courant connaît ces raisons et peut facilement identifier plus précisément ses besoins. C'est la raison pour laquelle beaucoup de besoins dans une organisation viennent d'un petit groupe de participants souvent connu comme règle de Pareto 20-80. On observe que les justifications sont gouvernées par les lois de la nature : les lois de la physique, lois de l'économie, les lois de la chimie, etc., et donc les relations entre eux sont logiques. Ils peuvent être soit directement proportionnel, inversement proportionnel et quelques fois indépendant. Il y a toujours quelque chose qui est plus important à creuser– les croyances. Les croyances sont difficiles à susciter des intervenants, souvent, ils demeurent implicites dans le récit utilisateurs et des itérations coûteuses sont parfois nécessaires pour atteindre vraiment. C'est souvent le cas dans l'ingénierie des systèmes à base de logiciels.

Comme les exigences sont projection de justifications avec des croyances, des même justifications sont des projections de points de vue. Les justifications peuvent être décomposées en un ensemble de points de vue, avec un vecteur d'importance, qui

fournit la force de leur préférence détenues par un intervenant particulier. Ce vecteur d'importance est implicite à l'intervenant et assez difficile d'obtenir complètement.

Afin de prendre l'avantage à partir des formulations théoriques, un langage de la modélisation des exigences CReML est introduit, qui tente de surmonter les défis inhérents liés aux problèmes d'évolutivité de diagrammes d'exigences complexes. Nous avons proposé CReML pour utiliser en complément avec SysML pour prendre l'avantage de l'infrastructure existant de la modélisation du système. Le langage proposé de modélisation graphique qui est capable de fonctionnalités typiques des techniques GORE populaires comme i * et KAOS et d'autres fonctionnalités qui sont source de préoccupation pour les ingénieurs systèmes et d'autres parties prenantes. Proposé un libellé et un outil de support permet de représenter les préférences des différents acteurs sur les différents buts et objectifs. Cela permet de modéliser à la fois le cœur et les fonctionnalités optionnelles du système à l'étude. Les objectifs peuvent être retracés aux témoignages d'utilisateurs qui sont liés à la modélisation de but diagramme. La responsabilité et l'interaction entre les agents sont modélisées séparément et peuvent être intégrés si le développeur souhaite. L'autre fonctionnalité intéressante de notre outil offre est de modéliser les logiques à l'aide de point de vue. Les justifications des parties prenantes sont projetées et divisé en différents points de vue de la scène très tôt, ce qui permet de mieux comprendre les besoins des utilisateurs. Le produit final de la modélisation de but conduit à la configuration requise qui peuvent être allouées à la modélisation UML/SysML diagrammes. Notre outil prend en charge quelques-uns des schémas d'UML/SysML notamment diagramme de cas d'utilisation et diagramme de définition de bloc. Il s'agit de fournir une traçabilité directe tout au long du cycle en V.

Rédaction des Exigences avec Négation

Nous avons traité d'autres questions relatives aux exigences linguistiques naturelles telles que l'ambiguïté et compréhensible. Mis au point des méthodes pour éviter toute ambiguïté en utilisant négation et les ont utilisés pour la négociation des cas de test. Quelques-uns des enquêtes ont été menés pour renforcer la croyance dans les formulations théoriques et les résultats sont encourageants. Technique de négation restreint offre des perspectives intéressantes comme un outil de lutte contre le problème de l'ambiguïté et compréhensible dans le contexte de IE. Il est fait valoir que, pour les notions de qualité, il est comparativement plus facile d'élaborer en phrases négatives, que autrement avec seulement phrases affirmatif. Résultats avec les enquêtes ont confirmé que des concepts qualité exprimé en négation est plus naturel que une exprimé en que des phrase affirmatif. Dans le contexte de l'IS, il peut être utilisé avec succès comme il semble plus ou moins naturel et intuitif que ceux positive. Nos exemples ont manifesté que, un bon mélange de phrases affirmatives et négatives peut fournir en vue de la profondeur implications d'une exigence de qualité. Les entretiens confirment que l'utilisateur peut préférer des phrases négatives et ils peuvent venir plus intuitivement. Argument fondamental est que, si certaines phrases semblent être naturelles, elles sont facilement comprises

et acceptées. La négation est naturelle dans une langue naturelle, de sorte qu'ils devraient n'être pas bannis pour rédiger les besoins plutôt que de les utiliser comme un outil.

On voit également que, avec l'aide de la négation, le cas de test peut être plus facilement conçu pour les exigences difficiles à comprendre.

B.5 Gestion des Exigences et de la Traçabilité

Les activités de gestion des exigences (GE) concernent toutes les activités qui sont liées à leur manipulation, c'est à dire, la priorisation, la mise en œuvre, la négociation et la traçabilité. La gestion des exigences est tout au sujet de la façon dont les exigences sont effectuées tout au long du cycle de vie du projet. Les activités de gestion des exigences sont directement responsables à la qualité du produit final. Un processus de GE éprouvée permettra d'accroître la confiance des utilisateurs dans le produit. Il peut aider à réduire l'incertitude de la réussite d'un projet ou d'échec.

L'IE doit fournir la traçabilité des exigences, ce qui est l'une des activités recommandées pour le processus de développement du projet. La traçabilité des exigences dans les projets d'IS est beaucoup plus prisée en raison de leur longue durée de vie. Nous montrons que notre approche peut fournir la bonne traçabilité de l'activité d'ingénierie droite avec moins d'effort.

Les Contributions sur la Traçabilité

Le problème de la traçabilité est divisée en plusieurs sous-problèmes : la création, la production, la maintenance, l'estimation des coûts, etc., de la traçabilité des exigences. Solutions systématiques sont conçus dans le cadre de la traçabilité *pré-exigence* et *post-exigence* de traçabilité, les relations nécessaires et minimum sont identifiés pour fournir la traçabilité pour les *pré-exigence* et *post-exigence*. Mesures appropriées pour modéliser la politique de traçabilité avec but et le retour sur investissement sont fournis, qui peut ensuite être utilisée pour la prise des les décisions.

Nous avons fourni un processus de création de trace qui peut être facilement mis en œuvre semi-automatiquement. L'approche de la traçabilité et à la fois *pré-exigence* et *post-exigence* traçabilité, avec le niveau de granularité souhaité comme exigé par l'acteur ou de l'utilisateur. Une meilleure approche pour la maintenir de traçabilité est conçu. Le modèle de traçabilité proposé insiste sur l'entretien avec une maintenance efficace régimes. Technique de maintenir de trace permet de portée de solution intéressante au problème de la trace ballants, ce qui peut immensément contribuer à réduire la pénibilité du processus de traçage. La solution offre une solution plausible à la problème de la perte de l'information, comme les informations généré dans le processus de développement toujours reste dans le système pour fournir l'exacte retrace de l'évolution du système.

Avec la facilité dans le processus de maintien trace le coût de maintenir peut être réduit sensiblement. La problème de pointeur ballant est résolu, l'effort dans l'entretien est réduites et donc moins de temps et moins de ressources humaines

sont engagé à faire la même tâche. Le logiciel sur lequel il est mis en œuvre est une plateforme composite pour l'ingénierie des exigences, leur gestion, la modélisation de système et pour le processus décisionnel, appelé *SysEngLab*. Module de l'ingénierie des exigences est implémenté comme la *CReML* avec le module de traçabilité des exigences pour leur gestion.

Notre approche peut être utilisée comme un outil pour organiser la complexité dans la conception du système composite en fournissant des prêts à utiliser la traçabilité de l'étape précédente. Pourtant, surmonter les problèmes sociaux de problème de traçabilité reste une tâche difficile et plus de travail de recherche doit être fait pour relever les défis en matière de traçabilité.

L'idée de base de notre méthodologie est de rendre le processus de traçabilité des exigences de plus en plus réactive et formel avec sémantique précise des relations. Nous sommes impatients de fournir des moyens d'intégrer notre approche cohérente avec les outils de soutien du UML/SysML.

Les Contributions sur le Stratégie de Traçabilité

On a proposé des mesures appropriées pour modéliser la politique de traçabilité orientée des buts, et le retour sur investissement, qui peut ensuite être utilisée pour la prise des décisions.

La matrice de préférence des parties prenantes permet à l'ingénieur des systèmes d'investir convenablement vers les sensibilités des parties prenantes à la traçabilité des justifications et des exigences du cycle de vie. Cela permet d'associer les diverses exigences avec leurs utilisateurs finaux qui sont plus exigeants pour leur traçabilité correspondant, avant de commencer le projet. Le partage de cette matrice de préférence des parties prenantes avec d'autres organisations ou des projets peut aussi être utile et peut être utile de fournir des modèles de délibéré traçage.

Comme dans toute organisation de toutes les parties prenantes sont classés et pondérés avant le coup de départ du projet, le produit des poids des parties prenantes et des préférences des parties prenantes, matrices des parties prenantes et justifications, donne un aperçu du montant de l'avantage par rapport aux exigences en matière de traçabilité particulier fournira. Cette information peut être très utile lors de la planification du budget de la traçabilité.

B.6 La Prise de Décision et Résolution de Conflit

Dès que les conditions sont fixés, divers modèles de comportement du système sont proposées. Suivant, ces modèles de comportement sont allouées physiquement aux sous-systèmes et composants. À cette étape, différents problèmes se posent: comment choisir le meilleur ensemble de sous-systèmes et composants pour implémenter le comportement et les services demandés par les exigences. Comme il existe de nombreux ensemble de solutions fonctionnelles ou physiques proposées, il devient fastidieuse tâche d'analyser les différentes configurations possibles pour la conception du système. Souvent, c'est le cas de la décision multicritère. Dans une prise

de décision multicritère, cette étape de pondération des critères est très critique pour le choix du bon produit. Souvent, ces critères sont tracés de retour à un ensemble d'intervenants multidisciplinaires qui participent au processus d'évaluation. Habituellement, ces intervenants diffèrent sur leur pondération de cet ensemble particulier de critères avec d'autres parties prenantes.

Dans un projet d'IS, la prise de décision et les conflits sont phénomène omniprésent, les décisions sont nécessaires et prises, dès que le processus d'identification des parties prenantes commence.

Comme, Projet de développement IS implique une interaction entre plusieurs disciplines, chacun ayant plus ou moins vague compréhension de l'autre, ce qui conduit souvent à l'émergence de conflits au cours du projet à mesure qu'il progresse. Certains conflits ont besoin efforts tacites et sont résolus mutuellement, mais quelques-uns d'entre eux sont de graves conflits. Ces graves conflits doivent être résolus afin de passer par le développement du projet. De même, l'évaluation et le choix d'une architecture de droite ou du composant exige des efforts de plusieurs décideurs ou des intervenants.

Les besoins des utilisateurs ont été générés et transformés dans les exigences du système et plus tard dans l'architecture du système, les composants, sous-composants,... Souvent, les organisations suivant les principes de la IS, modèle plusieurs alternatives aux architectures de systèmes, qui sont ensuite comparés et évalués sur plusieurs critères pour choisir le meilleur. Divers intervenants ayant des différences de prioriser leurs besoins, présentent souvent sous conflit les uns avec les autres. Ces conflits entre les parties prenantes peuvent se produire à tout moment durant le cycle de vie du produit.

Contributions sur la Pondération des Critères

Nous avons proposé une technique de pondération sur le critères, qui permet d'assimiler les différents poids des critères des différentes parties prenantes à fournir un seul ensemble de critères poids qui peut être uniformément acceptée par tous les différents décideurs. L'approche s'appelle pondération de l'ordre au risque d'utilité ($UROW$). Les principales contributions sont les suivantes:

— Il fournit un moyen holistique pour intégrer les différents poids des critères des différentes parties prenantes à fournir un éventail de poids critère l'aide de la modélisation des préférences classique.

— Il montre que la façon dont toutes les parties sont uniformément satisfaits de la technique proposée.

Dans ce travail actuel, nous avons fourni une théorie des systèmes de la façon dont les poids des critères peuvent être obtenus en utilisant la théorie classique de la modélisation des préférences. Cette approche offre de nombreux avantages par rapport aux autres avec les approches existantes. Habituellement, dans le projet de l'ingénierie des systèmes, les ingénieurs s'appuient sur leur intuition pour donner poids et une

utilisation ultérieure autre technique de combiner les préférences des différents décideurs. Notre approche fournit un formalisme de cette intuition d'ingénieur systèmes et fournit donc le raisonnement pour les différents poids obtenus. Notre approche est très facile à comprendre et à utiliser et exige très faible charge cognitive des ingénieurs. Il permet de fournir officiellement les résultats en utilisant les fonctions d'utilité tirés du décideur, risque couchée, l'aversion au risque ou risque neutre, il fournit un mécanisme permettant de les combiner ensemble pour trouver une solution uniforme acceptable.

Contributions sur la Méthodologie de Prise de Décision

Nous avons fourni une méthodologie à utiliser la technique des critères de pondération holistique de prendre une décision basée sur la valeur du système prévu par la partie prenante à la clientèle (fournisseur de valeur du système) exécuté par les experts en la matière. Afin de fournir le maximum de satisfaction des caractéristiques et des services dont ils avaient besoin avec la qualité et les contraintes qu'ils imposées aux développeurs. Nous avons classé les différents acteurs en fonction des différents rôles qu'ils jouent dans la réalisation d'un projet d'ingénierie des systèmes décisionnels.

B.7 Approche Globale

Figure 5.1, indique la méthodologie globale proposée sous forme de diagramme de Gantt. Concepts développés précédemment sont utilisés et intégrés dans un ordre chronologique de leur utilisation au cours du cycle de vie du développement. La méthodologie proposée peut être divisé en huit sections principalement. Les six premiers phase concernent la IE, la conception et les méthodes de prise de décision et sont de cœur de notre méthodologie globale. Après l'étude de faisabilité de la première phase du projet, une fois le projet reçoit le signal vert de l'autorité nécessaire, le lancement du projet commence.

Première phase: Dans la première étape, principaux parties prenantes du système sont identifiées. Récit utilisateur, des interviews et autres déclarations de leur part sont rassemblés. Dans l'étape suivante, les autres parties prenantes secondaires sont identifiées et leurs déclarations, témoignages d'utilisateurs, etc., sont rassemblées. Suite à l'analyse des déclarations des parties prenantes de la modélisation de contexte du système en cours de conception est lancée. L'étape suivante consiste à la pondération des différents acteurs qui sont les clients potentiels du système.

Deuxième phase: La première étape de la deuxième phase prend en entrée les récit utilisateurs précédemment collectées et d'autres récits, différentes déclarations des intervenants et définit les objectifs (état qu'ils veulent réaliser) des différentes parties prenantes. Dans l'étape suivante, leurs justifications sont identifiées et cartographiées sous forme de graphique. Suite à cela, les intervenants sont invités à pondérer les différents objectifs identifiés par les ingénieurs des exigences. Dans l'

étape suivante, les besoins de traçabilité des acteurs sont identifiés en vue de formuler des politiques de traçabilité exigées par les acteurs et institutions impliqués dans le projet. Les politiques de traçabilité pour le projet sont conçus et configurés selon les besoins des parties prenantes. Le processus de traçabilité commence pour le projet conformément aux politiques souhaitées par les intervenants. Les besoins des artefacts produits jusqu'à maintenant sont tracées pour les parties prenantes et leurs témoignages d'utilisateurs, déclarations, etc.

Troisième phase: Objectifs préalablement identifiés des parties prenantes sont prises en entrée et un ensemble d'objectifs sont identifiés, qui, lorsqu'il est accompli permis d'atteindre les objectifs. Dans la prochaine étape, les objectifs fixés soient marqués comme facultatifs ou de base en fonction de leur importance perçue par les ingénieurs d'exigences et validées ultérieurement par les parties prenantes vis-à-vis des contraintes, sortant de leur côté. L'étape suivante consiste à identifier les points de vue qui sont d'intérêt pour le système, les ingénieurs d'exigence et les parties prenantes. Analyse des objectifs préalablement identifiés à la lumière des points de vue identifiés conduit à la détermination des objectifs potentiellement contradictoires. Dans l'étape suivante traçabilité besoins des parties prenantes vis-à-vis des points de vue identifié est suscité et les politiques en matière de traçabilité sont affinés pour chaque partie prenante. La dernière étape de la troisième phase se concentre sur l'identification des contraintes imposées au système par l'intervenant. Les objectifs et les points de vue déjà identifiés faciliter cette tâche et histoires d'utilisateurs recueillies précédemment et déclarations constituent la contribution à cette tâche.

Quatrième phase: Quatrième phase commence avec contrôles sur l'ambiguïté des artefacts des exigences et des déclarations déjà produites. Les exigences ambiguës sont traitées et rendues sans ambiguïté en utilisant des techniques appropriées, telles que les techniques de négation ou d'autres signalées dans la littérature. Conception de cas-tests peut être effectuée une fois que les objectifs et les contraintes sont traitées à l'ambiguïté. Un objectif ambigu serait difficile d'être associé à des cas-tests. Une fois le test-cas pour les objectifs et les contraintes ont été conçus, le processus de transformation des objectifs en exigences système démarre. Exigences du système sont établies à partir des différents points de vue, qui leur permettent d'être quantifiés, précis et mesurables. Modélisation du système peut être lancé une fois quelques-unes des exigences du système sont disponibles, en tenant entrées de la modélisation de contexte précédemment effectués de l'environnement du système. Cas-test pour les exigences du système sont conçus à ce stade, une fois un ensemble d'exigences de système sont disponibles.

Cinquième étape: Cette phase commence par l'identification des fonctionnalités exigées par les exigences du système précédemment identifiés. Il est suivi par la conception des architectures fonctionnelles différentes, qui peuvent fournir le accommoder les fonctionnalités souhaitées. Une fois les architectures fonctionnelles sont disponibles pour les ingénieurs d'exigence, leur évaluation est effectuée et une sélection est faite pour la mise en œuvre. Ensuite, les différentes architectures structurels potentiels sont identifiés pouvant supporter les architectures fonc-

tionnelles proposées. Il est suivi par l'évaluation des architectures de structure et d'une sélection de mise en œuvre finale. Enfin, des cas-tests pour les architectures fonctionnelles et structurelles sont identifiés.

Sixième étape: La conception détaillée des architectures fonctionnelles et structurelles est effectuée et les spécifications de conception précises sont établies. Les cas de tests pour les spécifications de conception des fonctions et structures sont décidées. Elle est suivie par l'identification d'alternatives pour la mise en œuvre physique des composants. Une fois que l'ensemble des solutions de rechange est disponible pour le concepteur et experts en la matière (PME), l'évaluation des solutions de rechange/composants peut être réalisée, et la sélection finale sont faites pour la mise en œuvre. Avant la mise en œuvre physique tests unitaires sont effectués sur les composants matériels.

Septième phase: Tous les composants nécessaires sont acquis ou fabriqués et la mise en œuvre du système est réalisée au cours de cette phase. Tous les sous-systèmes sont intégrés à réaliser l'ensemble du système. Tous les composants logiciels sont installés et le système est préparé pour les différentes épreuves.

Huitème Phase: Tests d'intégration prédéfinie est effectuée sur le système, suivis de la vérification, la vérification du système de sous-système et, enfin, les tests de validation du système sont effectuées. Après avoir passé avec succès toutes les procédures d'essai le système est prêt pour la livraison.

B.8 Conclusion

Nous avons présenté dans cette thèse une approche globale pour l'ingénierie des exigences et des activités de prise de décision, pour un projet d'IS. Une théorie des besoins a été présenté dans le but de mieux comprendre les exigences et les autres notions qui lui sont liées.

Nous avons traité d'autres questions relatives aux exigences en langue naturelles telles que l'ambiguïté et compréhensibilité. Mis au point des méthodes pour éviter toute ambiguïté en utilisant négation et les ont utilisés pour la négociation des cas de test. Quelques-uns des enquêtes ont été menées pour renforcer la croyance dans les formulations théoriques et les résultats sont encourageants. Technique de négation restreint offre des perspectives intéressantes comme un outil de lutte contre le problème de l'ambiguïté et compréhensible dans le contexte d'IE.

Nous avons présenté dans cette thèse une approche de la traçabilité des exigences pour un projet d'IS. Nous avons donné la distinction entre la traçabilité des exigences pour un système intensif de logiciels et de systèmes intensifs de matériel (systèmes cyber-physiques). Nous avons utilisé la littérature existante pour formuler une schéma et stratégie du traçabilité complète de l'exigence. Cette thèse a également présenté une méthodologie d'ingénierie des systèmes pour quelques-uns des problèmes de traçabilité existantes dans la littérature. Nous avons identifié de blocs des trace d'avoir huit types de valeurs distinctes, qui peuvent fournir la traçabilité tout au long de la V-modèle de développement du projet. Nous avons fourni une

plate-forme pour relier ces blocs de traces avec diverses relations avec les artefacts générés tout au long du cycle de vie et les exigences. Nous avons vu que la mise en place complète et fiable lien de traçabilité est un processus à double sens: vers l'avant et vers l'arrière. Informations de traçabilité des exigences demeure une entité précieuse pour les entreprises impliquées dans de grands projets où de nouveaux projets peuvent trouver inspirations des projets antérieurs. Informations de traçabilité des exigences reste valable même après la fin du projet et le produit est retiré du service.

Nous avons montré que les différents réglages du bloc de la traçabilité des exigences peut être utilisé pour différentes activités de gestion des produits: gestion de configuration, gestion des impacts du changement, la maintenance, le système de mise à gradation, la réutilisation des produits, la gestion des versions, la compréhension du système, etc. Une telle utilisation du jeu de blocs de traces facilite ces tâches fastidieuses, comme les données requises sont facilement disponibles, et réduit en fin de compte la consommation des ressources.

Nous avons formulé paramètres pour la modélisation de la politique de traçabilité purposed et retour sur investissement. Nous avons fourni un processus de création de trace qui peut être facilement mis en œuvre semi-automatique. L'approche de la traçabilité et à la fois pré-condition et post-exigence de traçabilité, avec le niveau de granularité souhaité comme exigé par l'acteur ou de l'utilisateur. Le logiciel sur lequel il est mise en œuvre est une plate-forme composite pour l'ingénierie des exigences, la conception du système, la simulation et la prise de décisions appelé *SysEngLab*.

L'automatisation entièrement fiable de traçabilité compréhensif dans un projet reste une activité difficile et loin d'être réalisable. Les politiques de relance de traçabilité à base de recherche d'information ont encore des problèmes avec rappel complet et une précision satisfaisante.

Nous avons vu que la traçabilité des exigences est un processus itératif. Une approche semi-automatique est un meilleur choix pour la réalisation de la traçabilité des exigences, avec la participation de l'homme convenable. Une activité de traçabilité des exigences soigneusement réalisée est une énorme source de connaissances pour l'ensemble de l'entreprise, pour les projets actuels et futurs, y compris l'ensemble du cycle de vie du produit. Nous avons l'intention de concevoir un algorithme qui permet de déterminer automatiquement les relations entre les diverses exigences.

Autre contribution importante de cette thèse est d'intégrer les contributions mentionnées précédemment de manière à fournir une méthodologie pour mener à bien les activités d'IE et prise de décision. L'exécution progressive prévue des diverses tâches et leur intégration est fourni et appliquée sur une étude de cas et les résultats sont partagés.

Appendix C

List of Publications and Reports

Chapter in Books

- G.AURIOL, C.BARON, V.SHUKLA, J.Y.FOURNIOLS, "System engineering method for system design, Systems Engineering - Practice and Theory," Boris Cogan (Eds), InTech, 354p., NřISBN 978-953-51-0322-6, Mars 2012, pp.201-216 .

International Journal

- V.SHUKLA, G.AURIOL, "A survey on contemporary requirement traceability techniques" demande de modifications mineures, INCOSE Systems Engineering Journal (Submitted), 15 pages.

- V.SHUKLA, G.AURIOL, K.W. HIPEL, "Comprehensive Multi Criteria Decision Making Methodology for Systems Engineering", IEEE Systems Journal, 11 pages.

International Conference with Proceedings

- V.SHUKLA, K.W.HIPEL, G.AURIOL, "Negotiation and conflict resolution in systems engineering: A prescriptive Approach," 13 International Conference on Group Decision and Negotiation, June-17-21, Stockholm, Sweden, 9 pages

- V.SHUKLA, G.AURIOL, C.BARON, H.DEMMOU, "Empowering graph model of game theory for system design," International Conference on Software and Systems Engineering and their Applicat ICSSEA 2012, 23-25 octobre 2012, Paris (France), 2012, 5 pages. ,

- V.SHUKLA, G.AURIOL, C.BARON, D.ESTEVE, J.C.PASCAL, P.ESTEBAN, M.MALBERT, "Intelligent system design tool: a comprehensive PDM/PLM tool," International Conference on Software and Systems Engineering and their Applications, ICSSEA 2012, 23-25 octobre 2012, Paris (France), 2012, 2 pages.

- V.SHUKLA, G.AURIOL, C.BARON, X.ZHANG, "Comprehensive requirement traceability information and relations in project life-cycle," INCOSE International Symposium 2012, Rome (Italie), 9-12 Juillet 2012, 15 pages.

- V.SHUKLA, G.AURIOL, C.BARON, "Integrated requirement traceability, multi-view modeling and decision-making," IEEE International Systems Conference, Vancouver (Canada), 19-22 Mars 2012, pp.406-410.

- V.SHUKLA, G.AURIOL, C.BARON, "Use of negation to improve understandability of natural language requirements," Complex Systems Design & Management (CSDM 2011), Paris (France), 7-9 Décembre 2011, 12 pages.

- V.SHUKLA, G.AURIOL, C.BARON, "A graph based requirement traceability maintenance model," International Conference on Software Engineering Advances (ICSEA 2011), Barcelona (Spain), 23-28 Octobre 2011, pp.161-165.

- X.ZHANG, G.AURIOL, C.BARON, V.SHUKLA "How to think about customer value in requirements engineering," International Conference on Software Engineering Advances (ICSEA 2011), Barcelona (Spain), 23-28 Octobre 2011, pp.483-486.

- G.AURIOL, C.BARON, V.SHUKLA, J.Y.FOURNIOLS, "Design and simulations of wireless sensors networks in a long range aircraft" WSEAS International Conference on Communications, Corfu (Greece), 14-17 Juillet 2011, pp.117-124

- V.SHUKLA, G.AURIOL, "Reinventing goal-based requirements modeling" poster CSD&M 2013,12 pages.

- V.SHUKLA, G.AURIOL, "Methodology for determining stakeholders' criteria weights in systems engineering" Poster CSD&M 2013, 13 page.

Reports LAAS CNRS

- V.SHUKLA, G.AURIOL, "Engineering for traceability," Dec 2012.

- V.SHUKLA, K.W.HIPEL, G.AURIOL, "Negotiation and conflict resolution: graph model for complex system design," Dec, 2012.

- G.AURIOL, C.BARON, V.SHUKLA, J.M.DILHAC, J.Y.FOURNIOLS, "Engineering wireless sensor network in an aircraft environment," Dec 2011.

Bibliography

[Abadi 2008] A. Abadi, M. Nisenson and Y. Simionovici. *A traceability technique for specifications.* In Program Comprehension, 2008. ICPC 2008. The 16th IEEE International Conference on, pages 103–112. IEEE, 2008. 72

[AFIS 2013] AFIS. *Association Francaise de Ingénierie Systèmes.* http://www.afis.fr, 2013. 83

[Albert 2012] Vincent Albert. Support de cours en: Modélisation et simulation des systèmes complexes. Université Paul Sabatier, Toulouse, France, September 2012. 10

[Albright 2009] S Christian Albright, Wayne L Winston and Christopher James Zappe. Data analysis and decision making with microsoft® excel. Cengage-Brain. com, 2009. 118

[Altenhofen 2010] Michael Altenhofen and Achim D. Brucker. *Practical issues with formal specifications: lessons learned from an industrial case study.* In Proceedings of the 15th international conference on Formal methods for industrial critical systems, FMICS'10, pages 17–32, Berlin, Heidelberg, 2010. Springer-Verlag. 52

[Ambriola 1997] V. Ambriola and V. Gervasi. *Processing natural language requirements.* In ASE '97: Proceedings of the 12th international conference on Automated software engineering (formerly: KBSE), pages 36–46, Washington, DC, USA, 1997. IEEE Computer Society. 47

[Anquetil 2010] N. Anquetil, U. Kulesza, R. Mitschke, A. Moreira, J.C. Royer, A. Rummler and A. Sousa. *A model-driven traceability framework for software product lines.* Software and Systems Modeling, vol. 9, no. 4, pages 427–451, 2010. 76

[Antoniol 2000] G. Antoniol, C. Casazza and A. Cimitile. *Traceability recovery by modeling programmer behavior.* In Reverse Engineering, 2000. Proceedings. Seventh Working Conference on, pages 240–247. IEEE, 2000. 65, 77

[Antoniol 2002] G. Antoniol, G. Canfora, G. Casazza, A. De Lucia and E. Merlo. *Recovering traceability links between code and documentation.* Software Engineering, IEEE Transactions on, vol. 28, no. 10, pages 970–983, 2002. 71, 72

[Aponte 2011] J. Aponte and A. Marcus. *Improving traceability link recovery methods through software artifact summarization.* In Proceedings of the 6th International Workshop on Traceability in Emerging Forms of Software Engineering, pages 46–49. ACM, 2011. 77

[Arrow 1963] K.J. Arrow. Social choice and individual values, volume 12. Yale university press, 1963. 27, 111

[Arunski 1999] Karl Arunski, Martin James, Phil Brown and Dennis Buede. *Systems engineering Overview.* In Presentation to the Texas Board of Professional Engineers. INCOSE, September 1999. 10

[Auriol 2008] G. Auriol, C. Baron and J.-Y. Fourniols. *Teaching requirements skills within the context of a physical engineering project.* In Requirements Engineering Education and Training, 2008. REET '08., pages 6 –11, sept. 2008. 50

[Ballejos 2008] Luciana C Ballejos and Jorge M Montagna. *Method for stakeholder identification in interorganizational environments.* Requirements Engineering, vol. 13, no. 4, pages 281–297, 2008. 18

[Bana e Costa 1994] Carlos A Bana e Costa and Jean-Claude Vansnick. *MACBETH–An interactive path towards the construction of cardinal value functions.* International transactions in operational Research, vol. 1, no. 4, pages 489–500, 1994. 113

[Bana e Costa. 2005] Carlos A. Bana e Costa., De Corte Jean-Marie and Vansnick Jean-Claude. *On the Mathematical Foundations of MACBETH.* In Multiple criteria decision analysis: state of the art surveys, pages 163–186. Springer, 2005. 113

[Barron 1996a] F Hutton Barron and Bruce E Barrett. *Decision quality using ranked attribute weights.* Management Science, vol. 42, no. 11, pages 1515–1523, 1996. 120

[Barron 1996b] F Hutton Barron and Bruce E Barrett. *The efficacy of SMARTER Simple multi-attribute rating technique extended to ranking.* Acta Psychologica, vol. 93, no. 1, pages 23–36, 1996. 120

[Bashar 2012] Md Abul Bashar, D Marc Kilgour and Keith W Hipel. *Fuzzy preferences in the graph model for conflict resolution.* Fuzzy Systems, IEEE Transactions on, vol. 20, no. 4, pages 760–770, 2012. 117

[Bell 1988] D.E. Bell, H. Raiffa and A. Tversky. Decision making: Descriptive, normative, and prescriptive interactions. Cambridge University Press, 1988. 26

[Belton 2002] Valerie Belton and Theodor J Stewart. Multiple criteria decision analysis: an integrated approach. Springer, 2002. 114

[Bernoulli 1738] Daniel Bernoulli. *"Specimen Theoriae Novae de Mensura Sortis," Commemarii Academiae Scientiarum Imperialis Petropolitanae 5, 175-192, Translated by Louise Sommer in Bernoulli, Daniel (1954). "Exposition of a New Theory on the Measurements of Risk.".* Econometrica, 1738. 25

[Berry 2003] Daniel M. Berry, Erik Kamsties and Michael M. Krieger. *From Contract Drafting to Software Specification: Linguistic Sources of Ambiguity*, 2003. Version 1.0. 47, 48, 50, 51, 53

[Berry 2004] Daniel M. Berry and Erik Kamsties. *Ambiguity in Requirements Specification.* In Julio Cesar Sampaio do Prado Leite and Jorge Horacio Doorn, editeurs, Perspectives on Software Requirements, pages 7–44. Kluwer, 2004. 47, 53

[Berry 2005] Daniel M. Berry and Erik Kamsties. *The Syntactically Dangerous All and Plural in Specifications.* IEEE Software, vol. 22, no. 1, pages 55–57, 2005. 47

[Boehm 1995] B. Boehm, P. Bose, E. Horowitz and M.J. Lee. *Software requirements negotiation and renegotiation aids: A theory-W based spiral approach.* In Software Engineering, 1995. ICSE 1995. 17th International Conference on, pages 243–243. IEEE, 1995. 27

[Boehm 1998] B. Boehm and A. Egyed. *Software requirements negotiation: some lessons learned.* In Software Engineering, 1998. Proceedings of the 1998 International Conference on, pages 503–506. IEEE, 1998. 27

[Boehm 2001] B. Boehm, P. Grunbacher and R.O. Briggs. *Developing groupware for requirements negotiation: lessons learned.* Software, IEEE, vol. 18, no. 3, pages 46 –55, may 2001. 27

[Bolchini 2003] D. Bolchini and J. Mylopoulos. *From task-oriented to goal-oriented Web requirements analysis.* In Web Information Systems Engineering, 2003. WISE 2003. Proceedings of the Fourth International Conference on, pages 166–175, 2003. 32

[Bottomley 2000] Paul A. Bottomley, John R. Doyle and Rodney H. Green. *Testing the Reliability of Weight Elicitation Methods: Direct Rating versus Point Allocation.* Journal of Marketing Research, vol. 37, no. 4, pages pp. 508–513, 2000. 119

[Brans 2005] Jean-Pierre Brans and Bertrand Mareschal. *PROMETHEE methods.* In Multiple criteria decision analysis: state of the art surveys, pages 163–186. Springer, 2005. 24, 113

[Bryson 2011] J.M. Bryson. Strategic planning for public and nonprofit organizations: A guide to strengthening and sustaining organizational achievement. Bryson on Strategic Planning. Wiley, 2011. 110, 127

[Buede 2011] D.M. Buede. The engineering design of systems: Models and methods. Wiley Series in Systems Engineering and Management. Wiley, 2011. 19

[Čančer 2012] Vesna Čančer. *Criteria weighting by using the 5Ws & H technique.* Business Systems Research, vol. 3, no. 2, pages 41–48, 2012. 121

[Capobianco 2009] G. Capobianco, A. De Lucia, R. Oliveto, A. Panichella and S. Panichella. *Traceability recovery using numerical analysis.* In Reverse Engineering, 2009. WCRE'09. 16th Working Conference on, pages 195–204. IEEE, 2009. 73

[Carroll 1995] John M Carroll. *Scenario-based design: envisioning work and technology in system development.* 1995. 36

[Castro 2002] Jaelson Castro, Manuel Kolp and John Mylopoulos. *Towards requirements-driven information systems engineering: the Tropos project.* Information Systems, vol. 27, no. 6, pages 365 – 389, 2002. 37, 38

[Chen 1992] Shu-Jen Chen and Ching-Lai Hwang. Fuzzy multiple attribute decision making methods. Springer, 1992. 113

[Chen 2011] X. Chen, J. Hosking and J. Grundy. *A combination approach for enhancing automated traceability:(NIER track).* In Software Engineering (ICSE), 2011 33rd International Conference on, pages 912–915. IEEE, 2011. 76

[CLAWS 2013] CLAWS. *http://www.comp.lancs.ac.uk/ucrel/claws.* http://www.comp.lancs.ac.uk/ucrel/claws, 2013. 75

[Cleland-Huang 2003] J. Cleland-Huang, C.K. Chang and M. Christensen. *Event-based traceability for managing evolutionary change.* Software Engineering, IEEE Transactions on, vol. 29, no. 9, pages 796 – 810, sept. 2003. 63, 82, 193

[Cleland-Huang 2005] J. Cleland-Huang, R. Settimi, C. Duan and X. Zou. *Utilizing supporting evidence to improve dynamic requirements traceability.* In Requirements Engineering, 2005. Proceedings. 13th IEEE International Conference on, pages 135–144. IEEE, 2005. 77

[Cleland-Huang 2012] Jane Cleland-Huang, Grant Zemont and Wiktor Lukasik. *A Heterogeneous Solution for Improving the Return on Investment of Requirements Traceability.* 2012 20th IEEE International Requirements Engineering Conference (RE), vol. 0, pages 230–239, 2012. 70

[Cramer 1728] Gabriel Cramer. *Letter from Cramer to Nicholas Bernoulli. Translated by Louise Sommer in Bernoulli, Daniel (1954). "Exposition of a New Theory on the Measurements of Risk.".* Econometrica, 1728. 25

[Cysneiros 2004] Luiz M. Cysneiros and Erik Yu. *Non-Functional Requirements Elicitation.* In Julio Cesar Sampaio do Prado Leite and Jorge Horacio Doorn, editeurs, Perspectives on Software Requirements, pages 115–138. Kluwer, 2004. 48, 52

[Dagenais 2007] B. Dagenais, S. Breu, F.W. Warr and M.P. Robillard. *Inferring structural patterns for concern traceability in evolving software.* In Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering, pages 254–263. ACM, 2007. 78

[Dawes 1974] Robyn M Dawes and Bernard Corrigan. *Linear models in decision making.* Psychological bulletin, vol. 81, no. 2, page 95, 1974. 118

[De Lucia 2010] Andrea De Lucia, Fausto Fasano, Rocco Oliveto and Genoveffa Tortora. *Fine-grained management of software artefacts: the ADAMS system.* Software: Practice and Experience, vol. 40, no. 11, pages 1007–1034, 2010. 82, 102

[Deerwester 1990] S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer and R. Harshman. *Indexing by latent semantic analysis.* Journal of the American society for information science, vol. 41, no. 6, pages 391–407, 1990. 72

[Denger 2003] Christian Denger, Daniel M Berry and Erik Kamsties. *Higher quality requirements specifications through natural language patterns.* In Software: Science, Technology and Engineering, 2003. SwSTE'03. Proceedings. IEEE International Conference on, pages 80–90. IEEE, 2003. 48, 49, 50

[Diakoulaki 1995] Danae Diakoulaki, George Mavrotas and Lefteris Papayannakis. *Determining objective weights in multiple criteria problems: the CRITIC method.* Computers & Operations Research, vol. 22, no. 7, pages 763–770, 1995. 118, 119

[do Prado Leite 1993] Julio Cesar Sampaio do Prado Leite and Ann Paula M. Franco. *A Strategy for Conceptual Model Acquisition.* In Proceedings of the First International Symposium on Requirements Engineering, pages 243–246. IEEE Computer Society Press, 1993. 48

[Doyle 1997] John R Doyle, Rodney H Green and Paul A Bottomley. *Judging relative importance: direct rating and point allocation are not equivalent.* Organizational Behavior and Human Decision Processes, vol. 70, no. 1, pages 65–72, 1997. 119

[Drivalos-Matragkas 2010] Nikolaos Drivalos-Matragkas, Dimitrios S Kolovos, Richard F Paige and Kiran J Fernandes. *A state-based approach to traceability maintenance.* In Proceedings of the 6th ECMFA Traceability Workshop, pages 23–30. ACM, 2010. 63, 82

[Dupré 1998] L. Dupré. Bugs in writing: a guide to debugging your prose. Addison-Wesley, 1998. 47

[Easterbrook 2005] Steve Easterbrook, Eric Yu, Jorge Aranda, Yuntian Fan, Jennifer Horkoff, Marcel Leica and Rifat Abdul Qadir. *Do viewpoints lead to*

better conceptual models? An exploratory case study. In Requirements Engineering, 2005. Proceedings. 13th IEEE International Conference on, pages 199–208. IEEE, 2005. 37, 39

[Eden 2013] C. Eden and F. Ackermann. Making strategy: The journey of strategic management. SAGE Publications, 2013. 109, 110, 127

[Edwards 1977] Ward Edwards. *How to Use Multiattribute Utility Measurement for Social Decisionmaking.* Systems, Man and Cybernetics, IEEE Transactions on, vol. 7, no. 5, pages 326–340, 1977. 119, 120

[Edwards 1994] Ward Edwards and F Hutton Barron. *SMARTS and SMARTER: Improved simple methods for multiattribute utility measurement.* Organizational Behavior and Human Decision Processes, vol. 60, no. 3, pages 306–325, 1994. 120

[Egyed 1997] A. Egyed, B. Boehm*et al. Analysis of System Requirement Negotiation Behavior Patterns.* In Proceedings of INCOSE, pages 269–276, 1997. 28

[Egyed 2001] A. Egyed. *A scenario-driven approach to traceability.* In Proceedings of the 23rd international conference on Software engineering, pages 123–132. IEEE Computer Society, 2001. 74

[Egyed 2002] A. Egyed and P. Grunbacher. *Automating requirements traceability: Beyond the record & replay paradigm.* In Automated Software Engineering, 2002. Proceedings. ASE 2002. 17th IEEE International Conference on, pages 163–171. IEEE, 2002. 74

[Egyed 2003] A. Egyed. *A scenario-driven approach to trace dependency analysis.* Software Engineering, IEEE Transactions on, vol. 29, no. 2, pages 116–132, 2003. 74

[Egyed 2004a] A. Egyed. *Resolving uncertainties during trace analysis.* In ACM SIGSOFT Software Engineering Notes, volume 29, pages 3–12. ACM, 2004. 74

[Egyed 2004b] A. Egyed and P. Grunbacher. *Identifying requirements conflicts and cooperation: How quality attributes and automated traceability can help.* Software, IEEE, vol. 21, no. 6, pages 50–58, 2004. 74

[Egyed 2005a] A. Egyed and P. Grünbacher. *Supporting software understanding with automated requirements traceability.* International Journal of Software Engineering and Knowledge Engineering, vol. 15, no. 05, pages 783–810, 2005. 74

[Egyed 2005b] Alexander Egyed, Stefan Biffl, Matthias Heindl and Paul Grünbacher. *A value-based approach for understanding cost-benefit trade-offs during automated software traceability.* In Proceedings of the 3rd international

workshop on Traceability in emerging forms of software engineering, TEFSE '05, pages 2–7, New York, NY, USA, 2005. ACM. 70, 84

[Egyed 2007] A. Egyed, G. Binder and P. Grunbacher. *STRADA: A tool for scenario-based feature-to-code trace detection and analysis.* In Software Engineering-Companion, 2007. ICSE 2007 Companion. 29th International Conference on, pages 41–42. IEEE, 2007. 74

[EIA632 2005] EIA632. *Processes for Engineering Systems.* Electronics Industry Association, 2005. 6, 8, 21, 33, 35, 36, 83

[Eng 2002] Jeremy Dick BSc Eng. *Doors: A tool to manage requirements.* In Requirements engineering, pages 187–204. Springer, 2002. 83

[Eveleens 2010] J.L. Eveleens and C. Verhoef. *The rise and fall of the Chaos report figures.* Software, IEEE, vol. 27, no. 1, pages 30 –36, 2010. 1, 36

[Fabbrini 2000] F Fabbrini, M Fusani, S Gnesi and G Lami. *Quality evaluation of software requirement specifications.* In Proceedings of the Software and Internet Quality Week 2000 Conference, pages 1–18, 2000. 48

[Fabbrini 2001a] F. Fabbrini, M. Fusani, S. Gnesi and G. Lami. *The Linguistic Approach to the Natural Language Requirements Quality: Benefit of the use of an Automatic Tool.* Software Engineering Workshop, Annual IEEE/NASA Goddard, vol. 0, page 97, 2001. 48, 50

[Fabbrini 2001b] F. Fabbrini, M. Fusani, S. Gnesi and G. Lami. *The Linguistic Approach to the Natural Language Requirements Quality: Benefit of the use of an Automatic Tool.* Software Engineering Workshop, Annual IEEE/NASA Goddard, vol. 0, page 97, 2001. 50

[Fang 1993] L. Fang, K.W. Hipel and D.M. Kilgour. Interactive decision making: The graph model for conflict resolution. Wiley New York, 1993. 27, 108, 117, 118, 125

[Figueira 2002] José Figueira and Bernard Roy. *Determining the weights of criteria in the ELECTRE type methods with a revised Simos' procedure.* European Journal of Operational Research, vol. 139, no. 2, pages 317–326, 2002. 121

[Fishburn 1970] P.C. Fishburn. *Utility theory for decision making.* Rapport technique, DTIC Document, 1970. 25, 27, 112, 125

[Flores 2010] Fernando Flores, Manuel Mora, Francisco ÃĄlvarez, Laura Garza and Hector Duran. *Towards a Systematic Service-oriented Requirements Engineering Process (S-SoRE).* In JoÃčoEduardo Quintela VarajÃčo, Maria-Manuela Cruz-Cunha, GoranD. Putnik and AntÃşnio Trigo, editeurs, ENTERprise Information Systems, volume 109 of *Communications in Computer and Information Science*, pages 111–120. Springer Berlin Heidelberg, 2010. 36

[Forsberg 1995] Kevin Forsberg and Harold Mooz. *The relationship of system engineering to the project cycle.* 1995. 16

[Foures 2013] D. Foures, V. Albert and A Nketsa. *Simulation validation using the compatibility between Simulation Model and Experimental Framework.* In proceedings of the Summer Simulation Conference (SCS 13), 2013. 153

[Freeman 2010] R.E. Freeman. Strategic management: A stakeholder approach. Strategic Management: A Stakeholder Approach. Cambridge University Press, 2010. 18

[French 2000] S. French and D.R. Insua. Statistical decision theory. Kendall's Library of Statistics Series. Arnold, 2000. 26

[Fuchs 1996] Norbert E Fuchs and Rolf Schwitter. *Controlled english for requirements specification.* IEEE Computer Special Issue on Interactive Natural Language Processing, 1996. 49

[Galvao 2007] I. Galvao and A. Goknil. *Survey of traceability approaches in model-driven engineering.* In Enterprise Distributed Object Computing Conference, 2007. EDOC 2007. 11th IEEE International, pages 313–313. IEEE, 2007. 76

[Glinz 2010] Martin Glinz. *Very lightweight requirements modeling.* In 18th IEEE International Requirements Engineering Conference, pages 385–386, 2010. 37

[Glinz 2011] Martin Glinz. *A Glossary of Requirements Engineering Terminology.* 2011. 21, 31, 32

[Goguen 1993] Joseph A. Goguen and C. Linde. *Techniques for requirements elicitation.* In Requirements Engineering, 1993., Proceedings of IEEE International Symposium on, pages 152–164, 1993. 19

[Gotel 1994] O.C.Z. Gotel and CW Finkelstein. *An analysis of the requirements traceability problem.* In Requirements Engineering, 1994., Proceedings of the First International Conference on, pages 94–101. IEEE, 1994. 62, 63, 70

[Gotel 2011] O.C.Z. Gotel and S.J. Morris. *Out of the labyrinth: Leveraging other disciplines for requirements traceability.* In Requirements Engineering Conference (RE), 2011 19th IEEE International, pages 121–130. IEEE, 2011. 62, 63

[Gotel 2012a] O. Gotel, J. Cleland-Huang, J.H. Hayes, A. Zisman, A. Egyed, P. Grunbacher and G. Antoniol. *The quest for Ubiquity: A roadmap for software and systems traceability research.* In Requirements Engineering Conference (RE), 2012 20th IEEE International, pages 71 –80, sept. 2012. 22, 70

[Gotel 2012b] O. Gotel, J. Cleland-Huang, J.H. Hayes, A. Zisman, A. Egyed, P. Grünbacher, A. Dekhtyar, G. Antoniol and J. Maletic. *The Grand Challenge of Traceability (v1. 0)*. Software and Systems Traceability, pages 343–409, 2012. 22, 64, 70

[Greco 2004] S. Greco. Multiple criteria decision analysis: State of the art surveys. International Series in Operations Research & Management Science. Springer, 2004. 27

[Gross 2007] H.G. Gross, M. Lormans and J. Zhou. *Towards software component procurement automation with latent semantic analysis*. Electronic Notes in Theoretical Computer Science, vol. 189, pages 51–68, 2007. 72

[Grossman 2004] D.A. Grossman and O. Frieder. Information retrieval: Algorithms and heuristics, volume 15. Springer, 2004. 70

[Grundy 1999] John Grundy. *Aspect-oriented requirements engineering for component-based software systems*. In Requirements Engineering, 1999. Proceedings. IEEE International Symposium on, pages 84–91. IEEE, 1999. 36

[Gutman 1989] M. Gutman. *Asymptotically optimal classification for multiple tests with empirically observed statistics*. Information Theory, IEEE Transactions on, vol. 35, no. 2, pages 401–408, 1989. 72

[Haimes 1974] Yacov Y Haimes and Warren A Hall. *Multiobjectives in water resource systems analysis: The surrogate worth trade off method*. Water Resources Research, vol. 10, no. 4, pages 615–624, 1974. 115

[Hamouda 2004] Luai Hamouda, D.Marc Kilgour and KeithW. Hipel. *Strength of Preference in the Graph Model for Conflict Resolution*. Group Decision and Negotiation, vol. 13, pages 449–462, 2004. 27, 117, 125

[Hansen 2008] Paul Hansen and Franz Ombler. *A new method for scoring additive multi-attribute value models using pairwise rankings of alternatives*. Journal of Multi-Criteria Decision Analysis, vol. 15, no. 3-4, pages 87–107, 2008. 114

[Haskins 2011] Cecilia Haskins and Kevin Forsberg. Systems engineering handbook: A guide for system life cycle processes and activities; incose-tp-2003-002-03.2. 1. 2011. 6, 8, 9, 10, 16, 18, 62, 113, 139

[Hayes 2003] J.H. Hayes, A. Dekhtyar and J. Osborne. *Improving requirements tracing via information retrieval*. In Requirements Engineering Conference, 2003. Proceedings. 11th IEEE International, pages 138–147. IEEE, 2003. 71

[Hong 2010] Youngki Hong, Minho Kim and Sang-Woong Lee. *Requirements management tool with evolving traceability for heterogeneous artifacts in the entire life cycle*. In Software Engineering Research, Management and Applications (SERA), 2010 Eighth ACIS International Conference on, pages 248–255. IEEE, 2010. 82

[Hooks 1993] Ivy Hooks. *Writing Good Requirements (A Requirements Working Group Information Report).* In Proc. Third International Symposium of the NCOSE, volume 2, 1993. 47

[Howard 1980] Ronald A Howard. *An assessment of decision analysis.* Operations Research, vol. 28, no. 1, pages 4–27, 1980. 106

[Hull 2010] E. Hull, K. Jackson and J. Dick. Requirements Engineering. Springer, 2010. 1, 31, 32

[Hull 2011] E. Hull, K. Jackson and J. Dick. Requirements engineering. Springer London, 2011. 36, 47, 148

[Hutton Barron 1992] F Hutton Barron. *Selecting a best multiattribute alternative with partial information about attribute weights.* Acta Psychologica, vol. 80, no. 1, pages 91–103, 1992. 120

[Huyck 2000] Christian R. Huyck and Feroz abbas. *Natural language processing and requiremnents engineering: a linguistics prespective.* Proc 1st Asia-Pacific Conference on Software Quality, 2000. 47, 51

[Hwang 1981] Ching-Lai Hwang, Kwangsun Yoon*et al.* Multiple attribute decision making: methods and applications: a state-of-the-art survey, volume 13. Springer-Verlag New York, 1981. 24, 113

[IEEE 1998] IEEE. *IEEE Recommended Practice for Software Requirements Specifications.* IEEE Std 830-1998, 1998. 8, 51

[IEEE 2005] IEEE. *IEEE Standard for Application and Management of the Systems Engineering Process.* IEEE Std 1220-2005 (Revision of IEEE Std 1220-1998), 2005. 6, 8, 22, 33

[IEEE 2008] IEEE. *ISO/IEC/IEEE Standard for Systems and Software Engineering - Software Life Cycle Processes.* IEEE STD 12207-2008, pages c1 –138, 2008. 8

[IEEE 2011] IEEE. *Systems and software engineering – Life cycle processes – Requirements engineering.* ISO/IEC/IEEE 29148:2011(E), pages 1–94, 2011. 18, 21, 32

[INCOSE 2012] INCOSE. *Guide for Writing Requirements, INCOSE-TP-2010-006-01,V.1, April 2012*, 2012. 47

[INCOSE 2013] INCOSE. *Brief History of Systems Engineering.* http://www.incose.org/mediarelations/briefhistory.aspx, 2013. 10

[INRIA] INRIA. *ATL, month = June, year = 2013, howpublished = http://eclipse.org/atl/.* 76

[ISO 2008] *Systems and software engineering System life cycle processes.* ISO/IEC 15288:2008(E) IEEE Std 15288-2008 (Revision of IEEE Std 15288-2004), pages 1–84, 2008. 8, 11, 16, 18, 22, 31, 33, 83

[Jiménez 2003] Antonio Jiménez, Sixto Rıŕos-Insua and Alfonso Mateos. *A decision support system for multiattribute utility evaluation based on imprecise assignments.* Decision Support Systems, vol. 36, no. 1, pages 65–79, 2003. 120

[Jiménez 2006] Antonio Jiménez, Sixto Ríos-Insua and Alfonso Mateos. *A generic multi-attribute analysis system.* Computers & operations research, vol. 33, no. 4, pages 1081–1101, 2006. 120, 121

[Jureta 2006] I.J. Jureta, S. Faulkner and P. Schobbens. *Justifying Goal Models.* In Requirements Engineering, 14th IEEE International Conference, pages 119–128, 2006. 21, 37

[Jureta 2009] I. Jureta, J. Mylopoulos and S. Faulkner. *Analysis of Multi-Party Agreement in Requirements Validation.* In Requirements Engineering Conference, 2009. RE '09. 17th IEEE International, pages 57 –66, 31 2009-sept. 4 2009. 27

[Jureta 2010] I.J. Jureta, A. Borgida, N.A. Ernst and J. Mylopoulos. *Techne: Towards a New Generation of Requirements Modeling Languages with Goals, Preferences, and Inconsistency Handling.* In Requirements Engineering Conference (RE), 2010 18th IEEE International, pages 115–124, 2010. 37

[Juristo 2000] Natalia Juristo, Ana Maria Moreno and Marta López. *How to use linguistic instruments for object-oriented analysis.* Software, IEEE, vol. 17, no. 3, pages 80–89, 2000. 48, 50

[Kahneman 1979] Daniel Kahneman and Amos Tversky. *Prospect theory: An analysis of decision under risk.* Econometrica: Journal of the Econometric Society, pages 263–291, 1979. 26

[Kamsties 2000] Erik Kamsties and Barbara Paech. *Taming Ambiguity in Natural Language Requirements.* In in Proceedings of the 13th International conference on System and Software Engineering and their Applications, volume 2, 2000. 47, 51, 53

[Kamsties 2001] Erik Kamsties, Daniel M. Berry, Barbara Paech, E. Kamsties, D. M. Berry and B. Paech. *Detecting Ambiguities in Requirements Documents Using Inspections.* In in Proceedings of the First Workshop on Inspection in Software Engineering (WISE'01, pages 68–80, 2001. 47, 51

[Kamsties 2003] Erik Kamsties, Antje von Knethen and Ralf Reussner. *A controlled experiment to evaluate how styles affect the understandability of requirements specifications.* Information and Software Technology, vol. 45, no. 14, pages

955 – 965, 2003. Eighth International Workshop on Requirements Engineering: Foundation for Software Quality. 50

[Keeney 1992] Ralph L Keeney. *On the foundations of prescriptive decision analysis.* Utility theories: Measurements and applications, pages 57–72, 1992. 25

[Keeney 1993] R.L. Keeney and H. Raiffa. Decisions with multiple objectives: Preferences and value trade-offs. Cambridge University Press, 1993. 24, 25, 27, 112, 120, 125, 128

[Keeney 2009] R.L. Keeney. Value-focused thinking: A path to creative decision-making. Harvard University Press, 2009. 106, 107

[Kovitz 1998] B.L. Kovitz. Practical software requirements: a manual of content and style. PowerBuilder Developer's Library. Manning, 1998. 47, 53

[Kovitz 2002] B. Kovitz. *Ambiguity and what to do about it [requirements engineering].* In Requirements Engineering, 2002. Proceedings. IEEE Joint International Conference on, 2002. 47, 51

[Kuhn 2007] A. Kuhn, S. Ducasse and T. Gírba. *Semantic clustering: Identifying topics in source code.* Information and Software Technology, vol. 49, no. 3, pages 230–243, 2007. 72

[Kukreja 2012] N. Kukreja and B. Boehm. *Process implications of social networking-based requirements negotiation tools.* In Software and System Process (ICSSP), 2012 International Conference on, pages 68–72. IEEE, 2012. 27

[Kwok 1990] KL Kwok. *Experiments with a component theory of probabilistic information retrieval based on single terms as document components.* ACM Transactions on Information Systems (TOIS), vol. 8, no. 4, pages 363–386, 1990. 72

[Lange 1997] D.B. Lange and Y. Nakamura. *Object-oriented program tracing and visualization.* Computer, vol. 30, no. 5, pages 63–70, 1997. 74

[Leffingwell 2003] Dean Leffingwell and Don Widrig. Managing software requirements: a use case approach. Addison-Wesley, 2003. 51

[Leite 2004] Julio Cesar Sampaio do Prado Leite and Jorge Horacio Doorn, editeurs. Perspectives on software requirements. Kluwer, 2004. 52

[Li 2004a] Kevin W Li, D Marc Kilgour and Keith W Hipel. *Status quo analysis in the graph model for conflict resolution.* Journal of the Operational Research Society, vol. 56, no. 6, pages 699–707, 2004. 27, 117

[Li 2004b] K.W. Li, K.W. Hipel, D.M. Kilgour and Liping Fang. *Preference uncertainty in the graph model for conflict resolution.* Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on, vol. 34, no. 4, pages 507 – 520, july 2004. 117

[Liaskos 2010] S. Liaskos, S.A. McIlraith, S. Sohrabi and J. Mylopoulos. *Integrating Preferences into Goal Models for Requirements Engineering.* In Requirements Engineering Conference (RE), 2010 18th IEEE International, pages 135–144, 2010. 37, 148

[Lichtenstein 2007] Sharman Lichtenstein, Lemai Nguyen and Alexia Hunter. *Issues in IT service-oriented requirements engineering.* Australasian journal of information systems, vol. 13, no. 1, 2007. 36

[Lormans 2005] M. Lormans and A. van Deursen. *Reconstructing requirements coverage views from design and test using traceability recovery via LSI.* In Proceedings of the 3rd international workshop on Traceability in emerging forms of software engineering, pages 37–42. ACM, 2005. 72

[Lormans 2006] M. Lormans and A. Van Deursen. *Can LSI help reconstructing requirements traceability in design and test?* In Software Maintenance and Reengineering, 2006. CSMR 2006. Proceedings of the 10th European Conference on, pages 10–pp. IEEE, 2006. 72

[Lucia 2007] A.D. Lucia, F. Fasano, R. Oliveto and G. Tortora. *Recovering traceability links in software artifact management systems using information retrieval methods.* ACM Transactions on Software Engineering and Methodology (TOSEM), vol. 16, no. 4, page 13, 2007. 82

[Mader 2009] Patrick Mader, Orlena Gotel and Ilka Philippow. *Enabling Automated Traceability Maintenance through the Upkeep of Traceability Relations.* In RichardF. Paige, Alan Hartman and Arend Rensink, editeurs, Model Driven Architecture - Foundations and Applications, volume 5562 of *Lecture Notes in Computer Science*, pages 174–189. Springer Berlin Heidelberg, 2009. 82

[Mahajan 2004] R. Mahajan, M. Rodrig, D. Wetherall and J. Zahorjan. *Experiences applying game theory to system design.* In Proceedings of the ACM SIGCOMM workshop on Practice and theory of incentives in networked systems, pages 183–190. ACM, 2004. 27

[Mahmoud 2011] A. Mahmoud and N. Niu. *Source code indexing for automated tracing.* In Proceedings of the 6th International Workshop on Traceability in Emerging Forms of Software Engineering, pages 3–9. ACM, 2011. 72

[Mahoney 2012] Joseph T Mahoney. *Towards a stakeholder theory of strategic management.* Towards a New Theory of the Firm. Barcelona: IESE Research Unit, forthcoming, 2012. 110

[Maletic 2003] J.I. Maletic, E.V. Munson, A. Marcus and T.N. Nguyen. *Using a hypertext model for traceability link conformance analysis.* In Proc. of the Int. Workshop on Traceability in Emerging Forms of Software Engineering, pages 47–54, 2003. 68

[Marcus 2003] A. Marcus and J.I. Maletic. *Recovering documentation-to-source-code traceability links using latent semantic indexing.* In Software Engineering, 2003. Proceedings. 25th International Conference on, pages 125–135. IEEE, 2003. 72

[Misra 2005] S. Misra, V. Kumar and U. Kumar. *Goal-oriented or scenario-based requirements engineering technique - what should a practitioner select?* In Electrical and Computer Engineering, 2005. Canadian Conference on, pages 2288–2292, 2005. 36

[Mitchell 1997] Ronald K. Mitchell, Bradley R. Agle and Donna J. Wood. *Toward a Theory of Stakeholder Identification and Salience: Defining the Principle of Who and What Really Counts.* The Academy of Management Review, vol. 22, no. 4, pages pp. 853–886, 1997. 108

[MITRE 2012] Corp. MITRE. Mitre systems engineering guide. The MITRE Corporation, Bedford, Massachusetts, USA, 2012. 18

[Mori 1997] R.D. Mori. Spoken dialogues with computers. Academic Press, Inc., 1997. 72

[Murphy 1995] G.C. Murphy, D. Notkin and K. Sullivan. *Software reflexion models: Bridging the gap between source and high-level models.* In ACM SIGSOFT Software Engineering Notes, volume 20, pages 18–28. ACM, 1995. 78

[Murta 2006] L.G.P. Murta, A. van der Hoek and C.M.L. Werner. *ArchTrace: policy-based support for managing evolving architecture-to-implementation traceability links.* In Automated Software Engineering, 2006. ASE'06. 21st IEEE/ACM International Conference on, pages 135–144. IEEE, 2006. 82

[NASA 2007] Headquarters NASA. Nasa systems engineering handbook (nasa/sp-2007-6105 rev1). Books Express Publishing, 2007. 9

[Nguyen 2005] T.N. Nguyen, C. Thao and E.V. Munson. *On product versioning for hypertexts.* In Proceedings of the 12th international workshop on Software configuration management, pages 113–132. ACM, 2005. 63, 82

[Obeidi 2005] Amer Obeidi, Keith W Hipel and D Marc Kilgour. *Perception and emotion in the graph model for conflict resolution.* In Systems, Man and Cybernetics, 2005 IEEE International Conference on, volume 2, pages 1126–1131. IEEE, 2005. 117

[Objetiver 2013] Objetiver. *http://www.objectiver.com/*, Juin 2013. 37

[Ohnishi 1994] Atsushi Ohnishi. *Customizable software requirements languages.* In Computer Software and Applications Conference, 1994. COMPSAC 94. Proceedings., Eighteenth Annual International, pages 5–10. IEEE, 1994. 48, 50

[OMG 2013] OMG. *QVT*. http://www.omg.org/spec/QVT/1.0/, 2013. 76

[Omoronyia 2011] I. Omoronyia, G. Sindre and T. Stålhane. *Exploring a Bayesian and linear approach to requirements traceability*. Information and Software Technology, vol. 53, no. 8, pages 851–871, 2011. 73

[Opricovic 1998] Serafim Opricovic. *Multicriteria optimization of civil engineering systems*. Faculty of Civil Engineering, Belgrade, vol. 2, no. 1, pages 5–21, 1998. 114

[Opricovic 2002] Serafim Opricovic and Gwo-Hshiung Tzeng. *Multicriteria Planning of Post-Earthquake Sustainable Reconstruction*. Computer-Aided Civil and Infrastructure Engineering, vol. 17, no. 3, pages 211–220, 2002. 114

[Opricovic 2007] Serafim Opricovic and Gwo-Hshiung Tzeng. *Extended VIKOR method in comparison with outranking methods*. European Journal of Operational Research, vol. 178, no. 2, pages 514–529, 2007. 115

[Öztürké 2005] M. Öztürké, A. Tsoukiàs and P. Vincke. *Preference modelling*. Multiple Criteria Decision Analysis: State of the Art Surveys, pages 27–59, 2005. 27

[Parnell 2011] G.S. Parnell, P.J. Driscoll and D.L. Henderson. Decision making in systems engineering and management. Wiley Series in Systems Engineering and Management. Wiley, 2011. 21, 26, 106

[Perini 2012] A. Perini, A. Susi and P. Avesani. *A Machine Learning Approach to Software Requirements Prioritization*. 2012. 28

[Pinheiro 1996] F.A.C. Pinheiro and J.A. Goguen. *An object-oriented tool for tracing requirements*. Software, IEEE, vol. 13, no. 2, pages 52–64, 1996. 82

[Pinheiro 2004] Francisco AC Pinheiro. *Requirements traceability*. In Perspectives on software requirements, pages 91–113. Springer, 2004. 42, 63

[Pyster 2012] A Pyster, D Olwell, N Hutchison, S Enck, J Anthony, D Henry and A Squires. *Guide to the Systems Engineering Body of Knowledge (SEBoK) version 1.0. Hoboken, NJ: The Trustees of the Stevens Institute of Technology© 2012*, 2012. 21

[Rao 1995] Anand S Rao, Michael P Georgeff*et al.* *BDI Agents: From Theory to Practice*. In ICMAS, volume 95, pages 312–319, 1995. 31

[Rashid 2008] Awais Rashid. *Aspect-oriented requirements engineering: An introduction*. In International Requirements Engineering, 2008. RE'08. 16th IEEE, pages 306–309. IEEE, 2008. 36

[Regev 2005] G. Regev and A. Wegmann. *Where do goals come from: the underlying principles of goal-oriented requirements engineering.* In Requirements Engineering, 2005. Proceedings. 13th IEEE International Conference on, pages 353–362, 2005. 37

[Riabacke 2012] Mona Riabacke, Mats Danielson, Ekenberg Love and Aron Larsson. *Employing Cardinal Rank Ordering of Criteria in Multi-Criteria Decision Analysis.* In Proceedings of the 10th International FLINS Conference on Uncertainty Modeling in Knowledge Engineering and Decision Making (FLINS), 2012. 120

[Richardson 2004] J. Richardson and J. Green. *Automating traceability for generated software artifacts.* In Automated Software Engineering, 2004. Proceedings. 19th International Conference on, pages 24–33. IEEE, 2004. 74

[Robertson 1977] S.E. Robertson. *The probability ranking principle in IR.* Journal of documentation, vol. 33, no. 4, pages 294–304, 1977. 72

[Robertson 1994] S.E. Robertson and S. Walker. *Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval.* In Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval, pages 232–241. Springer-Verlag New York, Inc., 1994. 72

[Robertson 2006] James Robertson and Suzanne Robertson. *Volere: Requirements specification template (2006).* The Atlantic Systems Guild, 2006. 47

[Robinson 1990] W.N. Robinson. *Negotiation behavior during requirement specification.* In Software Engineering, 1990. Proceedings., 12th International Conference on, pages 268–276. IEEE, 1990. 27

[Rolland 1992] Colette Rolland and Christophe Proix. *A natural language approach for requirements engineering.* In Advanced information systems engineering, pages 257–277. Springer, 1992. 48, 49, 50

[Roy 1968] B. Roy. *Classement et choix en présence de points de vue multiples (la méthode ELECTRE).* Riro, vol. 2, no. 8, pages 57–75, 1968. 24, 114

[Roy 1973] B. Roy and P. Bertier. *La Méthode ELECTRE II(Une application au média-planning...).* 1973. 24, 114

[Roy 1978] B. Roy. *ELECTRE III: Un algorithme de classement fondé sur une représentation floue des préférences en présence de critères multiples.* Cahiers du CERO, vol. 20, no. 1, pages 3–24, 1978. 24, 114

[Roy 1984] B. Roy and P. Vincke. *Relational systems of preference with one or more pseudo-criteria: Some new concepts and results.* Management Science, pages 1323–1335, 1984. 27, 125

[Roy 1991] B. Roy. *The outranking approach and the foundations of ELECTRE methods*. Theory and decision, vol. 31, no. 1, pages 49–73, 1991. 125

[(RWP) 2012] Requirements Working Group (RWP). Guide for writing requirements. INCOSE, 2012. 148

[Ryan 1993] Kevin Ryan. *The role of natural language in requirements engineering*. In Requirements Engineering, 1993., Proceedings of IEEE International Symposium on, pages 240–242, 1993. 47

[Saaty 1980] Thomas L Saaty. The analytic hierarchy process. McGraw-Hill, New York, 1980. 112, 120, 138

[Saaty 1990] Thomas L Saaty. *How to make a decision: the analytic hierarchy process*. European journal of operational research, vol. 48, no. 1, pages 9–26, 1990. 24, 112, 120, 138

[Saaty 1996] Thomas L Saaty. *Decision making with dependence and feedback: The analytic network process*. 1996. 112

[Sage 2000] A.P. Sage and J.E. Armstrong. Introduction to systems engineering. Wiley series in systems engineering. Wiley, 2000. 8, 19, 24, 106

[Sage 2009] A.P. Sage and W.B. Rouse. Handbook of systems engineering and management. Wiley series in systems engineering and management. Wiley, 2009. 108

[Salton 1975] G. Salton, A. Wong and C.S. Yang. *A vector space model for automatic indexing*. Communications of the ACM, vol. 18, no. 11, pages 613–620, 1975. 71

[Salton 1988] G. Salton and C. Buckley. *Term-weighting approaches in automatic text retrieval*. Information processing & management, vol. 24, no. 5, pages 513–523, 1988. 71

[Savage 1954] Leonard J Savage. The foundations of statistics. John Wiley & Sons, 1954. 25

[Schwarz 2010] H. Schwarz, J. Ebert and A. Winter. *Graph-based traceability: a comprehensive approach*. Software and Systems Modeling, vol. 9, no. 4, pages 473–492, 2010. 67, 82

[SEI 1994] CMU SEI. *The capability maturity model: guidelines for improving the software process*. US: Addsion Wesley Longman Inc, 1994. 22

[Seibel 2010] A. Seibel, S. Neumann and H. Giese. *Dynamic hierarchical mega models: comprehensive traceability and its efficient maintenance*. Software and Systems Modeling, vol. 9, no. 4, pages 493–528, 2010. 82

[Shannon 1949] Claude Elwood Shannon, Warren Weaver, Richard E Blahut and Bruce Hajek. The mathematical theory of communication, volume 117. University of Illinois press Urbana, 1949. 118, 137

[Shannon 2001] Claude E Shannon. *A mathematical theory of communication*. ACM SIGMOBILE Mobile Computing and Communications Review, vol. 5, no. 1, pages 3–55, 2001. 118, 137

[Sherba 2003] S.A. Sherba, K.M. Anderson and M. Faisal. *A framework for mapping traceability relationships*. In Proceedings of the 2nd International Workshop on Traceability in Emerging Forms of Software Engineering, pages 32–39, 2003. 77

[Shukla 2011] V. Shukla, G. Auriol and C. Baron. *A Graph-Based Requirement Traceability Maintenance Model*. In ICSEA 2011, The Sixth International Conference on Software Engineering Advances, pages 161–165, 2011. 63, 94, 95

[Shukla 2012a] V. Shukla, G. Auriol and C. Baron. *Integrated requirement traceability, multiview modeling, and decision-making: A systems engineering approach for integrating processes and product*. In Systems Conference (SysCon), 2012 IEEE International, pages 1–5. IEEE, 2012. 65, 196

[Shukla 2012b] V. Shukla, G. Auriol, C. Baron and X. Zhang. *Comprehensive requirement traceability information and relations in project life-cycle*. In 22nd Annual INCOSE International Symposium. INCOSE, 2012. 63, 98

[Shukla 2013] V. Shukla and G. Auriol. *Methodology for Determining Stakeholders' Criteria Weights in Systems Engineering*. 2013. poster in CSDM 2013, Paris. 147

[Simos 1990] Jean Simos. *Evaluer l'impact sur l'environnement: Une approche originale par l'analyse multicritère et la négociation*. 1990. 121

[Soley 2000] Richard Soley*et al.* *Model driven architecture*. OMG white paper, vol. 308, page 308, 2000. 76

[Sommerville 1997] Ian Sommerville and Pete Sawyer. Requirements engineering: A good practice guide. John Wiley & Sons, Inc., New York, NY, USA, 1st édition, 1997. 18, 19, 31, 47, 52

[Sommerville 2010] Ian Sommerville. Software engineering (9th ed.). Pearson Education, London, United Kingdom, 2010. 32

[Spanoudakis 2004] G. Spanoudakis, A. Zisman, E. Pérez-Minana and P. Krause. *Rule-based generation of requirements traceability relations*. Journal of Systems and Software, vol. 72, no. 2, pages 105–127, 2004. 75, 84

[Spanoudakis 2005] G. Spanoudakis and A. Zisman. *Software traceability: a roadmap*. Handbook of Software Engineering and Knowledge Engineering, vol. 3, pages 395–428, 2005. 63

[Stillwell 1981] William G Stillwell, David A Seaver and Ward Edwards. *A comparison of weight approximation techniques in multiattribute utility decision making*. Organizational Behavior and Human Performance, vol. 28, no. 1, pages 62–77, 1981. 118

[Supakkul 2010] S. Supakkul and L. Chung. *Visualizing non-functional requirements patterns*. In Requirements Engineering Visualization (REV), 2010 Fifth International Workshop on, pages 25 –34, sept. 2010. 48

[Svahnberg 2008] M. Svahnberg, T. Gorschek, M. Eriksson, A. Borg, K. Sandahl, J. Borster and A. Loconsole. *Perspectives on Requirements Understandability – For Whom Does the Teacher's Bell Toll?* In Requirements Engineering Education and Training, 2008. REET '08., pages 22 –29, sept. 2008. 48, 50

[Tjong 2008] Sri Fatimah Tjong. *Avoiding ambiguity in requirements specifications*. PhD thesis, University of Waterloo, 2008. 48, 50

[Tonnellier 2012] Edmond Tonnellier and Olivier Terrien. *Rework: models and metrics*. In Complex Systems Design & Management, pages 119–131. Springer, 2012. 36

[Triantaphyllou 2000] E. Triantaphyllou. Multi-criteria decision making methods: a comparative study, volume 11. Kluwer Academic Publishers Dordrecht, 2000. 24, 27, 112, 114, 115

[Tsoukias 1992] A. Tsoukias and P. Vincke. *A survey on non conventional preference modelling*. Ricerca Operativa, vol. 61, pages 5–49, 1992. 125

[Tzeng 2002] Gwo-Hshiung Tzeng, Sheng-Hshiung Tsaur, Yiou-Dong Laiw and Serafim Opricovic. *Multicriteria analysis of environmental quality in Taipei: public preferences and improvement strategies*. Journal of environmental Management, vol. 65, no. 2, pages 109–120, 2002. 114

[Ulrich 2008] K.T. Ulrich and S.D. Eppinger. Product design and development. McGraw-Hill, 2008. 19, 24, 130

[USDoD 1993] USDoD. *System Safety program requirements Appendix A: Guidance for Implementation of System Safety Program*. Department of Defense, 1993. 153

[Van Lamsweerde 2000] Axel Van Lamsweerde. *Requirements engineering in the year 00: A research perspective*. In Proceedings of the 22nd international conference on Software engineering, pages 5–19. ACM, 2000. 36

[Van Lamsweerde 2001] A. Van Lamsweerde. *Goal-oriented requirements engineering: a guided tour*. In Requirements Engineering, 2001. Proceedings. Fifth IEEE International Symposium on, pages 249–262, 2001. 32, 36, 38

[van Lamsweerde 2009] A. van Lamsweerde. Requirements engineering: From system goals to uml models to software specifications. John Wiley & Sons, 2009. 18

[Von Neumann 1937] J. Von Neumann and O. Morgenstern. Theory of games and economic behavior. Princeton university press, 1937. 25, 117

[Von Winterfeldt 1986] D. Von Winterfeldt and W. Edwards. Decision analysis and behavioral research. University Press, 1986. 120

[Walker 2009] Sean Bernath Walker, Keith W Hipel and Takehiro Inohara. *Strategic decision making for improved environmental security: coalitions and attitudes*. Journal of Systems Science and Systems Engineering, vol. 18, no. 4, pages 461–476, 2009. 117

[Walley 1991] Peter Walley. Statistical reasoning with imprecise probabilities. Chapman and Hall London, 1991. 120

[Wang 1970] Marilyn W Wang and Julian C Stanley. *Differential weighting: A review of methods and empirical studies*. Review of Educational Research, vol. 40, no. 5, pages 663–705, 1970. 119

[Weng 1999] Gezhi Weng, Upinder S Bhalla and Ravi Iyengar. *Complexity in biological signaling systems*. Science, vol. 284, pages 92–96, 1999. 8

[Werneck 2009] Vera Maria Bejamim Werneck, Antonio de Padua Albuquerque Oliveira and JCSdP Leite. *Comparing GORE Frameworks: i-star and KAOS*. In Workshop em Engenharia de Requisitos (WER 2009), Val Paraiso, Chile, 2009. 37

[Whittle 2001] J. Whittle, J. Van Baalen, J. Schumann, P. Robinson, T. Pressburger, J. Penix, P. Oh, M. Lowry and G. Brat. *Amphion/NAV: Deductive synthesis of state estimation software*. In Automated Software Engineering, 2001.(ASE 2001). Proceedings. 16th Annual International Conference on, pages 395–399. IEEE, 2001. 74, 76

[Wierzbicki 1998] Andrzej P. Wierzbicki and Gordon Macdonald. *Reference Point Methods in Vector Optimization and Decision Support*, 1998. 113

[Wilson 1997] William M Wilson, Linda H Rosenberg and Lawrence E Hyatt. *Automated analysis of requirement specifications*. In Proceedings of the 19th international conference on Software engineering, pages 161–171. ACM, 1997. 48, 50

[Wong 1991] S.K.M. Wong and YY Yao. *A probabilistic inference model for information retrieval.* Information Systems, vol. 16, no. 3, pages 301–321, 1991. 77

[Wong 1995] S.K.M. Wong and Y.Y. Yao. *On modeling information retrieval with probabilistic inference.* ACM Transactions on Information Systems (TOIS), vol. 13, no. 1, pages 38–68, 1995. 77

[Yu 1989] C.T. Yu, W. Meng and S. Park. *A framework for effective retrieval.* ACM Transactions on Database Systems (TODS), vol. 14, no. 2, pages 147–167, 1989. 72

[Yu 1995] Eric Siu-Kwong Yu. *MODELLING STRATEGIC RELATIONSHIPS FOR PROCESS REENGINEERING.* PhD thesis, University of Toronto, 1995. 32

[Yu 1997] E.S.K. Yu. *Towards modelling and reasoning support for early-phase requirements engineering.* In Requirements Engineering, 1997., Proceedings of the Third IEEE International Symposium on, pages 226–235, 1997. 37, 38

[Zadeh 1965] Lotfi A Zadeh. *Fuzzy sets.* Information and control, vol. 8, no. 3, pages 338–353, 1965. 114

[Zave 1997] Pamela Zave and Michael Jackson. *Four dark corners of requirements engineering.* ACM Trans. Softw. Eng. Methodol., vol. 6, no. 1, pages 1–30, Janvier 1997. 31

[Zeng 2004] Dao-Zhi Zeng, Liping Fang, KeithW. Hipel and D.Marc. Kilgour. *Policy Stable States in the Graph Model for Conflict Resolution.* Theory and Decision, vol. 57, pages 345–365, 2004. 27

[Zhang 2006] Y. Zhang, R. Witte, J. Rilling and V. Haarslev. *An ontology-based approach for traceability recovery.* In 3rd International Workshop on Metamodels, Schemas, Grammars, and Ontologies for Reverse Engineering (ATEM 2006), Genoa, pages 36–43, 2006. 78

[Zimmermann 2001] H.J. Zimmermann. Fuzzy set theory-and its applications. Kluwer Academic Publishers, 2001. 112

[Zisman 2003] A. Zisman, G. Spanoudakis, E. Pérez-Miñana and P. Krause. *Tracing software requirements artifacts.* In Proceedings of 2003 International Conference on Software Engineering Research and Practice (SERP'03), Las Vegas, Nevada, USA, pages 448–455, 2003. 75, 84

[Zowghi 2005] Didar Zowghi and Chad Coulin. *Requirements elicitation: A survey of techniques, approaches, and tools.* In Engineering and managing software requirements, pages 19–46. Springer, 2005. 19

Index

5Ws & H, 121

AFT, 106
Agreement Processes, 11
AHP, 112
ANP, 112
AORE, 36
Architectural Design Process, 14
Arrow's impossibility theorem, 111
Assumption, 31

Belief, 31

Card playing techniques, 121
Chaos Reports, 1
CiTL, 153, 172
Complexity, 8
Conflict, 26, 108
Constraints, 31
CReML, 41
Criteria, 106
Criteria weighting, 111
CRITIC method, 119
CROC, 120

Decision, 106
  Certainty, 108
  Conflict, 108
  Cooperation, 108
  Makers, 108
  Uncertainty, 108
Decision Analysis, 26
Decision analysis, 106
Decision Making, 23
  Descriptive, 25
  Normative, 25
  MCDM/MCDA, 24
  Prescriptive, 25
Desires, 31

Direct rating, 119
Disposal Process, 15

EIA-632, 33
Eigenvector method, 120
ELECTRE, 114
Entropy method, 118, 119
Equal weights method, 118
Ethnography, 20
Expected Utility Theory, 25

Fuzzy set theory, 114

Goal, 32
Goal model, 46
GORE, 36

HiTL, 154, 172

i* framework, 20
IBIS, 152
IEEE1220, 33
Implementation Process, 14
Integration Process, 14
Intentions, 31
Interval methods, 120
Interviews, 20
ISO/IEC 15288, 10, 13
ISO15288, 33

KAOS framework, 20

Least mean square, 118
Life-cycle, 10

MACBETH, 113
Machine, 7
MADM, 112
Maintenance Process, 15
MAUT, 112

MAVT, 112
MCDM, 106, 112, 115
MiTL, 153, 172
MODM, 112

NLR, 47

Operation Process, 15
Organizational project-enabling processes, 11

PAPRIKA, 114
Point allocation, 119
Power and interest grid, 109
Project Processes, 11
PROMETHEE, 113
prospect theory, 25

Rank exponents method, 118
Rank reciprocal method, 118
Rank sum method, 118
Rationale, 31
Reference point, 113
regression method, 118
Requirement, 18
Requirement Management, 4
Requirements
  Analysis Process, 13
  Boiler plates, 47
  Document, 47
  Documentation, 20
  Elicitation, 19
  Functional, 35
  Modeling, 20
  Modeling Languages, 20
  Negotiation, 21
  Non-Functional, 35
  Prioritization, 21
  Stakeholder, 31
  Templates, 47
  Traceability, 21
    Traceability Cost, 23
    Automatic Traceability, 23
    Manual Traceability, 23
    Semi-automatic Traceability, 23
    Value Based Traceability, 23
Requirements Engineering, 3, 18
Responsibility model, 46
ROC method, 120
Role, 32

SBRE, 36
Scenario, 32
Scenario generation, 20
Simulation, 153
SiTL, 153, 172
SMART, 119
SMARTER, 120
SMARTS, 120
SORE, 36
Stakeholder, 3, 18, 31
  Requirements, 18
  Identification Process, 4, 19
  Requirement Definition process, 13
Standard deviation, 119
Standard deviation method, 118
SWING, 120
SWT, 115
System, 1, 7
  Requirements, 19
  Complex System, 8
Systems Engineering, 2

Technical processes, 11
TOPSIS, 113
Trace Relationships, 86
  Allocation, 86
  Conflict, 87
  Dependency, 87
  derive, 86
  Overlap, 87
  Refine, 86
  Satisfy, 87
  Validate, 87
  Verify, 87
Trade off method, 120
Transition Process, 14

uniguous, 47
Use-Cases, 20

Utility theory, 112

Validation Process, 14
Variance method, 118
Vee model, 15
Verification Process, 14
VFT, 106
View, 31
View-Point, 31
Viewpoints, 46
VIKOR, 114
Volere, 47

WPM, 112
Writing Requirements, 47
WSM, 112

Comprehensive Methodology for Complex Systems' Requirements Engineering and Decision Making

Abstract: The primary goal of the systems engineering is the creation of a set of high quality products and services that enable the accomplishment of desired tasks and needs of the clients or user groups. A typical systems engineering project can be divided in to three phases: definition, development, and deployment. The definition phase involves the activities of requirement elicitation and refinement. By the end of system definition phase, we have all the system functional and non-functional requirements. One of the results of development phase is initial working model of the system. The deployment phase consists of activities of operational implementation, operational testing and evaluation, and operational functioning and maintenance. In a project life cycle there are numerous issues to be sorted out during the various phases to finally deliver a successful product. We proposed solution to the problems of requirements engineering & management, design conflict detection, and stakeholders conflict resolution. This thesis is based on the recent advances in industrial practices and research in the field of system design engineering.

The objective of this thesis work is to propose an innovative and holistic conception methodology taking into account the multidisciplinary environment and multiple stakeholders. We have proposed a requirements modeling language based on the GORE techniques. We have proposed a few of tools for reducing the ambiguity of requirements such as: using negation and test cases using negation for contracting difficult requirements. Requirement management techniques are proposed to provide better requirements traceability and aid for other systems engineering activities. Few guidelines have been designed to guide the design of traceability policies. Criteria weighting technique has been designed to better carry out the conflict resolutions, during the various life cycle stages. Using the same criteria weighting technique a flexible multi criteria multi participant decision methodology is proposed for various decision problems arising during the life cycle of systems engineering project.

Finally, a comprehensive prescriptive systems engineering approach is proposed using all the previously made contributions and an illustrative case study of a real ongoing project is presented developed using the supporting tool *SysEngLab*, which implements majority of the methods and techniques proposed during thesis.

Mots clés : Systems engineering, Requirements engineering, Requirements management, Decision Making, Comprehensive methodology

Auteur: Vikas SHUKLA
Titre: Approche globale de l'ingénierie des exigences et de la prise de décision pour les systèmes complexes
Directeur de thèse: Guillaume Auriol
Lieu et date de soutenance: Toulouse le 06 Janvier 2014

Résumé: L'objectif principal de l'ingénierie des systèmes est la création d'un ensemble de produits et des services de haute qualité qui permettent l'accomplissement de tâches pour répondre aux besoins des clients. Un projet typique d'ingénierie des systèmes peut être divisé en trois phases : la définition, le développement et le déploiement. La phase de définition comprend les activités de capture des exigences et de leur raffinement. À la fin de la phase de définition du système, nous avons toutes les exigences fonctionnelles et non-fonctionnelles du système. L'un des résultats de la phase de développement est le modèle de travail initiale du système. La phase de déploiement se compose des activités liées à (1) l'évaluation opérationnelle du système, à (2) l'utilisation du système et à (3) son entretien. Dans un cycle de vie du projet, il y a de nombreuses questions qui doivent être traitées au cours des différentes phases pour finalement livrer un produit.

Nous avons proposé une solution aux problèmes liés à l'ingénierie des exigences et aux techniques de la détection, de la gestion et de la résolution des conflits entre les parties prenantes. Cette thèse est basée sur les dernières avancées dans les pratiques industrielles et de recherche dans le domaine de l'ingénierie de conception du système.

L'objectif de ce travail de thèse est de proposer une méthodologie de conception novatrice et globale en tenant compte de l'environnement multidisciplinaire et de multiples intervenants. Nous avons proposé un langage de modélisation des exigences basé sur les techniques GORE. Nous avons proposé quelques outils pour réduire l'ambiguïté des exigences tels l'utilisation de phrases négatives et de tests á l'aide de négation lorsqu'il s'agit de traiter certaines exigences difficiles à comprendre avec les techniques classiques. Nous avons également proposé des techniques de gestion des exigences pour mieux assurer leur traçabilité. Concernant la résolution des conflits, nous avons proposé des techniques de pondération des critères au cours des différentes étapes du cycle de vie. En utilisant la même technique de pondération de critères, une méthode de décision multicritères et multi participants est proposée pour divers problèmes de décision survenant pendant le cycle de vie du projet d'ingénierie systèmes.

Enfin, une approche globale de l'ingénierie des systèmes est proposée pour intégrer toutes les contributions faites précédemment et est illustrée sur une étude de cas concernant un projet réel avec la présentation dŠun outil *SysEngLab* que nous avons développé pour mettre en œuvre la majorité des méthodes et des techniques proposées au cours de thèse.

Motclés: Ingénierie système, Ingénierie des exigences, Gestion des d'exigences, Theory de la prise de décision, Méthodologie globale

Discipline administrative: Informatiques et Génie Industriel
Intitulé et adresse du laboratoire: Laboratoire d'Analyse et d'Architecture des Systèmes - 7 avenue du Colonel Roche -F-31077 Toulouse, France