An Improved Method for Selection of COTS Components Based on Quality Requirements

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering
in
Software Engineering

Submitted By
Ravneet Kaur Grewal
(800931018)

Under the supervision of:
Ms. Shivani Goel
(Assistant Professor)

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

June 2011

CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled, *"An Improved Method for Selection of COTS Components"*, in partial fulfillment of the requirements for the award of degree of Master of Engineering in Software Engineering submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of Ms. Shivani Goel and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

Ravneet Grewal

(Ravneet Kaur Grewal)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

Shivani Goel

(Ms. Shivani Goel)
Assistant Professor,
Computer Science and Engineering Department
Thapar University
Patiala

Countersigned by

(Dr. Maninder Singh)
Head 3/1/11
Computer Science and Engineering Department
Thapar University
Patiala

(Dr. S. K. Mohapatra)
Dean (Academic Affairs)
Thapar University
Patiala

Acknowledgement

I would like to express my sincere gratitude to all who have made possible the fulfillment of this work.

Firstly, I would like to thank my guide, Ms. Shivani Goel, Assistant Professor, CSED, Thapar University, Patiala for the time, patience, guidance and invaluable advises she has given me not only while my thesis work but throughout the course. It was a great opportunity to work under her supervision.

Then I would like to thank Dr. Maninder Singh, Head of the Department, CSED, Thapar University, Patiala for providing all the facilities and environment. I would also like to thank all my Teachers for their support and invaluable suggestions during the period of my work.

I would also like to thank my parents for always supporting me in the tough and happy moments, for their never ending support and inspiration; and especially to my grandfather and my late grandmother for the love and care.

Finally, I wish to thank my brother, Arman Singh Grewal and my friends, Arpita Sharma, Ravneet Kaur Chawla, Vaneet Kaur Bhatia, Amandeep Kaur Johar, Vishonika Kaushal , Ipneet Kaur, Sonam Chawla, Aradhana Majithia and Aarti Sharma for being with me through the good and the bad.

Ravneet Kaur Grewal

(800931018)

Abstract

Commercial Off-The-Shelf (COTS) software products have received a lot of attention in the last decade. Selection of Commercial off the Shelf (COTS) software packages is currently a fundamental task in software engineering. Selection of inappropriate COTS may lead to irreplaceable loss of effort, time and much more. Most appropriate COTS selection is complicated because of a number of factors, the most important being the user quality requirements. Therefore, selection can be ameliorated by transforming user quality requirements into requirements expressed in terms of the quality model attributes.

Selection is basically composed of two main processes, namely: searching of candidate COTS from the marketplace based on the stakeholders' requirements and their evaluation with respect to the requirements. But most of the different existing methods for COTS selection focus their efforts on evaluation, letting aside the problem of relating the quality requirements of COTS while searching in the marketplace. Searching candidate COTS is not an easy task, having to cope with some challenging marketplace characteristics related to its widespread, evolvable and growing nature; and the lack of available and well-suited information to obtain a quality-assured search. Mapping of the system requirements into quality requirements is very important.

In this thesis work, the use of quality model in selection of COTS components is presented using the ISO/IEC 9126 quality standard using the hierarchical feature graph. A new method has been proposed that takes care of quality requirements of the stakeholders and an after evaluation mismatch handling.

Abbreviations

AHP	Analytic Hierarchy Process
BIN	Binary
CARE	COTS Aware Requirement Engineering
CBSD	Component Based Software Development
CBSE	Component Based Software Engineering
COTS	Commercial Off The Shelf
CRE	Cots Requirement Engineering
MCDM	Multi-Criteria Decision Making
MiHOS	Mismatch Handling Aware COTS Selection
MMV	Mismatch Value
NFR	Non- Functional Requirement
ORDN	Ordinal
OTSO	Off The Shelf Option
PORE	Procurement Oriented Requirement Engineering
SV	Satisfaction Value
SUD	System Under Development
WSM	Weighted Score Method

Table of Contents

List of Figures

List of Tables

CHAPTER 1
INTRODUCTION

This chapter introduces a description of the work presented in thesis. It gives a brief introduction of Reuse-based software development, Component Based Software Engineering (CBSE), the need of software reuse and challenges faced while implementing reuse based development.

1.1 Background

In the IT market, existing technologies get obsolete speedily than in any other industry. This rapid pace demands continuous increase in productivity and competitiveness for the computer market. Many industries like health care, transportation, retail depend on IT products and services. So, in order to deliver the products and services to these industries, product development life-cycles are continually decreasing.

In the last several years, software development has been upraised by reusing the existing software or the commercial-off-the-shelf (COTS) software products. So, the purpose of software reuse is to develop large systems by incorporating previously developed or existing software. By this way, software development organization is able to cut down development time, costs and effort required for the development of the same from the scratch. While developing the software with reuse, existing potential reusable softwares are needed to be evaluated whether they meet the requirements of the application or not. If not, then it requires the modification in the requirements or adaption in COTS itself. After this, compatibility of this reusable software is checked with the desired environment. In this way, the idea of reuse centered software development is known as reuse-based development.

1.2 Software Reuse

Krueger's general view of software reuse [1]: "software reuse is the process of creating software systems from existing software rather than building them from scratch." Reuse of software components has become very important in a variety of aspects of software engineering. Recognition of the fact that many software systems contain many similar or even identical components that are developed from scratch over and over again has led to

efforts to reuse existing components. This assumption has several outcomes for the system development lifecycle. First, the concept of development with reuse is entirely different from development for reuse. In the development with reuse methodology, the reusable components should already been developed and possibly used in other products when the system development process starts. But in development for reuse, components are developed for general purpose and reused in many applications that may or may not exist presently. Also, requirements and business ideas are not much application specific. Second, a new separate process will appear: Exploration and selection of the components are the additional tasks that demand much consideration in the development with reuse. Reuse based development is focused on the identification of reusable entities and relations between them, beginning from the system requirements and from the availability of already existing components. The difference between these two methodologies is well discussed through the V-model [2] given below:

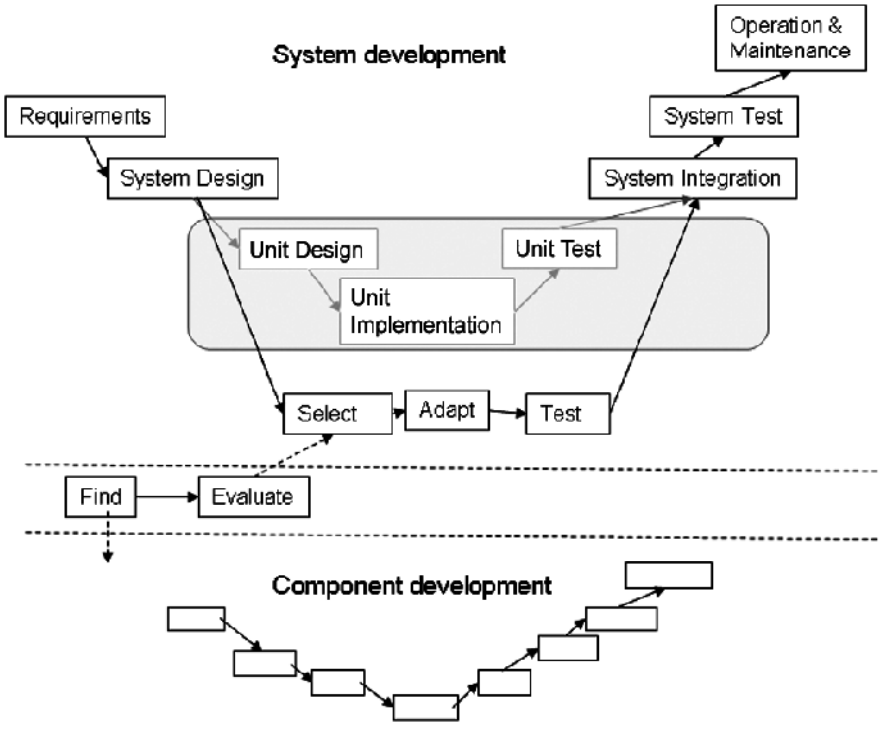

Figure 1.1: V- model for Reuse Based Development [2]

Instead of performing activities that often are time and efforts consuming like unit design and implementation, simply selection and evaluation of the appropriate components is performed and then integrated in the system. However, there are two problems here

which break this simplicity: (i) the availability of the component for the desired application, and (ii) the fitness of the selected components to our overall design. The first fact shows that we must have a process for finding components. This process includes activities for finding the components, and then the component evaluation. The second fact indicates for a need of component adoption and testing before it can be integrated into the system. But the process of finding, evaluation and selection of the components needs much consideration as it comes early in the cycle.

1.3 Component Based Software Engineering – Development with Reuse

"A component is a nontrivial, nearly independent, and replaceable part of a system that fulfills a clear function in the context of a well-defined architecture [3]."

Component-based software engineering (CBSE) is a sub-discipline of software engineering. CBSE is a process that emphasizes design and construction of a computer based systems using reusable software components.

Clements [3] describes CBSE as, "It is changing the way large software systems are developed. CBSE embodies the 'buy, don't build' philosophy espoused by Fred Brooks and others. In the same way that early subroutines liberated the programmer from thinking about details, CBSE shifts the emphasis from programming software to composing software systems. There is sufficient commonality in many large software systems to justify developing reusable components to exploit and satisfy that commonality".

Component based software development is understood to require reusable components that interact with each other and fit into system architectures. CBSE is primarily concerned with three functions [4]:

- Developing software from pre-produced parts
- The ability to reuse those parts in other applications
- Easily maintaining and customizing those parts to produce new functions and features

1.4 Need of Reuse Based Development

In today's world of competition, more and more companies compete in order to deliver the product faster and provide better innovative services to their customers. As a result of faster delivery, software organizations are not able to produce manageable, high quality, cost-effective software. This results in decreased business performance of the

organization. In order to improve the business performance, the software companies have to shorten the time required to bring a product to the market, reducing software development and maintenance costs, and increasing the quality of their software. To meet these business oriented goals, organizations are switched to reuse based development. Following are some of the benefits that reuse based development provide:

- Increased Product Trustworthiness as the reusable component are tried and tested in working environments.
- High maturity as COTS products go through multiple releases starting from beta versions to final versions.
- It reduces the risks involved in the development process as the cost of development is lesser as compared to traditional development.
- It reduces the risk of failure as the components are certified for their performance, quality and reliability.

1.5 Reusable Software Product – Commercial Off The Shelf (COTS)

Different researchers have attempted to define what we mean by a "reusable software product". In this thesis, we adopt the definition provided by Vigder et al. [5] which defines a reusable software product or COTS software product as: "a reusable software product: developed by a third party(who controls its ongoing support and evolution); bought, licensed, or acquired for the purposes of integration into a larger system as an integral part, i.e. that will be delivered as part of the system to the customer of that system; which might or might not allow modification at the source code level, but may include mechanisms for customization; and is bought and used by a significant number of systems developers."

Typical examples of COTS products are Notepad, Word Processors, Picture Manager, Internet Explorer provided by Microsoft, Adobe Photoshop etc.

1.6 Challenges to Software Reuse

The use of COTS software products provides various benefits stated in section 1.4. Besides the benefits, several challenges are also encountered during the software development with reuse. Many system failures occurred in past because the system failed to overcome these challenges. For example, the London Ambulance service fiasco in 1992, in which the system descended into chaos and reverted back to manual operation

partly because of inappropriate COTS selection which would be discussed in section 1.6 ([6], [7]). Following are the challenges to Reuse Based Development paradigm [8]:

- **Requirement engineering and COTS selection**

Some components don't suit exactly with the desired requirements. So, an analysis of the feasibility of requirements in relation to the available components is done which is followed by the reformulation of requirements or component adaptation sometimes.

- **Lack of tool support**

Purpose of software reuse is to build systems from reusable components simply and efficiently and to deliver the product in due time, and this can only be achieved with the help of tool support. For example, tool for component selection and retrieval, component test, tool for managing the component repositories.

- **COTS integration**

In intensive COTS based systems multiple COTS are selected. As a result, their interoperability issues arise which put a challenge to COTS integration.

1.7 COTS Selection

Building software systems from various commercial off-the-shelf (COTS), software components are very appealing because they can reduce the decreasing development risks and costs while increasing the functionality and capability of software systems. Selecting the components with the desired capabilities and lesser mismatches with the stakeholder's requirements is of much concern. The candidates components are gone under the evaluation phase which is the most crucial task of the selection cycle and resulting in various mismatches. Many COTS selection techniques are already proposed, but only few, like PORE (Procurement Oriented Requirement Engineering), MiHOS, have focused on the concept of mismatch handling, which resolve mismatches at prior. This reduces the time and effort spent over the integration of COTS in COTS based development.

1.7.1 COTS Selection Criteria

Following are some of the criteria on the basis of which COTS selection should be performed ([8], [9]):

- **Acquisition of changing requirements**

Requirements form main criteria for understanding of the system under development. This is because it is requirements that the concerned stakeholders have to communicate about, agree/disagree upon. They become criteria for evaluating and selecting of candidate components and are embedded in the legal contract (SRS). Requirements even provide acceptance criteria to check when the system is delivered that it meets customer's expectations. Most current methods and tools support systems design and integration but neglect the requirements engineering and product selection processes that must precede design and integration. In spite of this lack of focus on leads to requirements engineering, there is an emerging concept of iterative requirements engineering process that entertains the requirements change in component based development which leads to greater customer satisfaction.

- **Multiple stakeholders and the interaction**

The COTS selection process usually involves multiple stakeholders, each with his/her set of needs, preferences, and constraints, which may be in conflict with those of other stakeholders, thus leading to a negotiation problem. Moreover, the stakeholders have to establish a means for information sharing and for group representation of the problem in order to select COTS that satisfies the needs of every stakeholder as much as possible.

- **Presence of similar COTS components**

When COTS are procured from multiple vendors there arises a quite possible situation when two or more searched COTS components have the same functionalities but differ only in the name and their quality aspects. This leads to extra overhead while evaluating their characteristics.

- **Decision making analysis**

COTS selection is not based on single decision; it has to pass over a level of decisions that makes a hierarchical process. The hierarchical process comprises of the search for candidate components, evaluation of candidate list, and compliance of the evaluated components with non-functional aspects and so on. There are available many decision making techniques which facilitates DSS like Multi criteria Decision Making (MCDM), game theory etc [10].

- **Non-functional requirement description**

A complete COTS selection process includes considering the desired functionality together with the understanding of non-functionality aspects (performance, reliability, scalability, user friendly, etc). In general, vendors don't provide the non-functional requirement description. So, it emerges out as a challenging activity to take out a component with the required features and "lities" as well.

- **Cost and benefit analysis**

From the alternative list of COTS components for a desired functionality, cost and benefit analysis of the component in regard of the organization forms an elimination criteria. Cost estimation for the alternative components comprise of acquisition cost, adaptation cost and integration cost in the owner's environment.

- **Handling mismatches during component selection**

In CBSD, in order to decrease the development time, integration and testing time increases. There are various possibilities of mismatches, for example: (i) COTS products have extra capabilities that are not desired by stakeholders, and (ii) COTS products don't meet all specified requirements. So, in order to decrease the possibility of these kind of mismatches and integration time, integration issues (architectural and COTS mismatches) should be considered while the COTS selection.

1.8 Organization of the Thesis

Chapter 2 gives the overview of various COTS selection techniques and their drawbacks. Various decision making techniques for the evaluation purpose and mismatch taxonomies, proposed earlier, would be reviewed also.

Chapter 3 includes the problem statement and the scope of the thesis work.

Chapter 4 includes the proposed method for COTS selection explained with the help of Data Flow Diagram for various phases.

Chapter 5 includes the experimental results for validation of the proposed model with the help of the case study of selecting a web browser.

Chapter 6 includes the conclusion and future scope of the thesis.

CHAPTER 2
LITERATURE REVIEW

Previous chapter gave a brief introduction about reuse based development, its need and various challenges faced by the same. This chapter discusses the importance of COTS selection in component based software engineering, the general COTS selection method, various COTS selection techniques and the decision making techniques.

2.1Component Selection in Component Based Software Engineering

Software development process models used for writing the traditional programs cannot be used for assembling the application using reusable software components. A process model for this new approach should incorporate the activities to address the important aspects (such as component selection, customization and composition) required for building high quality component based software.

Overall, Component Based Software Development includes five activities [11, 12], as shown in figure 2.1, that are discussed below:

- **Requirements engineering**, which defines the desired capabilities and constraints, and helps establishing the COTS evaluation criteria. There are several proposals for defining COTS-oriented requirements engineering practices such as the Procurement-Oriented Requirements Engineering approach (PORE) [13] and COTS-Aware Requirements Engineering approach (CARE) [14]. These two approaches are described in Section 2.3.
- **COTS evaluation and selection**, which ensures selecting COTS products that perform the required functions, satisfy the defined constraints, and exhibit quality characteristics, e.g. performance.
- **COTS customization**, which involves tailoring the COTS product or modifying its source code to cover unsatisfied (or partially satisfied) requirements.
- **COTS integration**, which is the process of assembling a set of selected COTS products and components together to produce a system.
- **System evolution**, which includes maintenance issues such as updating the system with new COTS releases, adding new functionality to the system, and fixing errors.

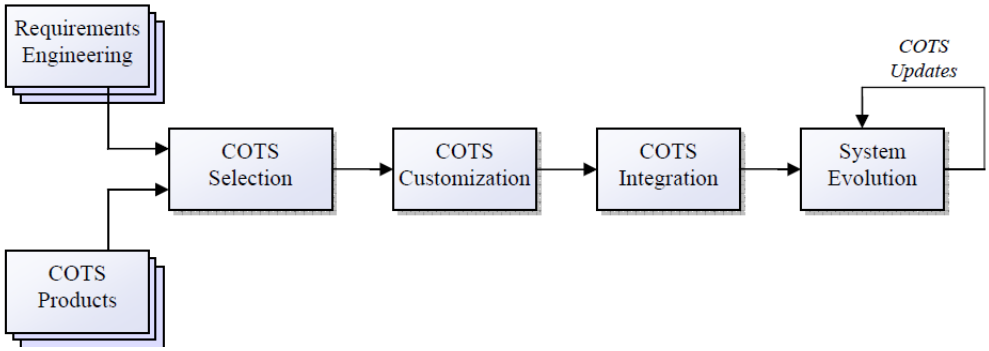

Figure 2.1: COTS-Based Development Model [17]

2.2 The General COTS Selection Method

COTS selection is the process of determining the appropriateness of the products developed by some third party to be used in a new system, and then selecting one or more products which are the most fit [15]. Even when there is no generally accepted method for COTS selection [16], all COTS selection methods share some key steps that can be overlapping. These steps are considered to be a part of the General COTS Selection (GCS) method which is described as follows:

Step1: Define evaluation criteria based on system requirements and constraints.

Step2: Search for COTS products.

Step3: Filter search results based on a set of 'must-have' requirements in order to define a short list of COTS candidates to be evaluated in more detail.

Step4: Evaluate COTS candidates on the short list.

Step5: Analyze the evaluation data (i.e. the output of Step4) and select the COTS product that best fits the criteria. Usually, either AHP or WSM is used for making the selection decision.

After Step 5 and according to the CBSD model described in Section 2.1, another activity is usually performed to resolve mismatches by customizing the selected COTS.

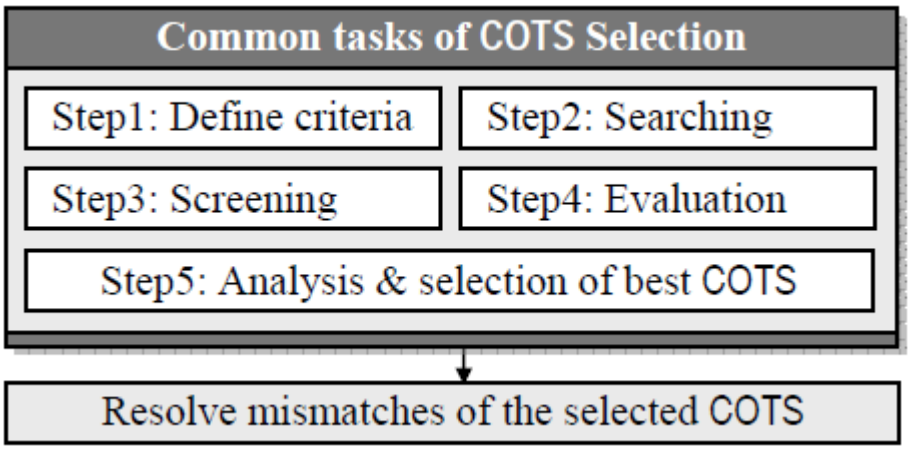

Figure 2.2: Activities in General COTS Selection Method [17]

2.3 COTS Selection Methods

This section describes the various COTS selection methods and their drawbacks.

2.3.1 Procurement Oriented Requirement Engineering (PORE)

The PORE approach has three main components [19]:

- A Process Model that identifies undertaken fundamental processes like requirement acquisition/validation, supplier selection, software package selection, package acceptance and management of system procurement.
- A Method Box that includes methods, techniques and tools that is available to help and achieve each of the processes.
- A Product Model that enables effective product evaluation and selection using the use case modeling, goals-based requirement methods and architecture modeling techniques.

PORE [19], which support the requirements engineering and product evaluation/selection processes for CBSE development process. PORE uses an iterative process of requirements acquisition and product evaluation/selection as its main approach as discussed by Fox et al. in [18] and in figure 2.3.

- It selects products by rejection [20], (i.e. the products that do not meet core customer requirements are selectively and iteratively rejected and removed from the candidate list).

- It makes use of various tools and techniques like card sorting & laddering for requirement acquisition, MCDM [10] & AHP [21] for decision making during component selection.
- It makes use of the Feature Analysis technique [8] for sorting the match between COTS features and desired requirements.

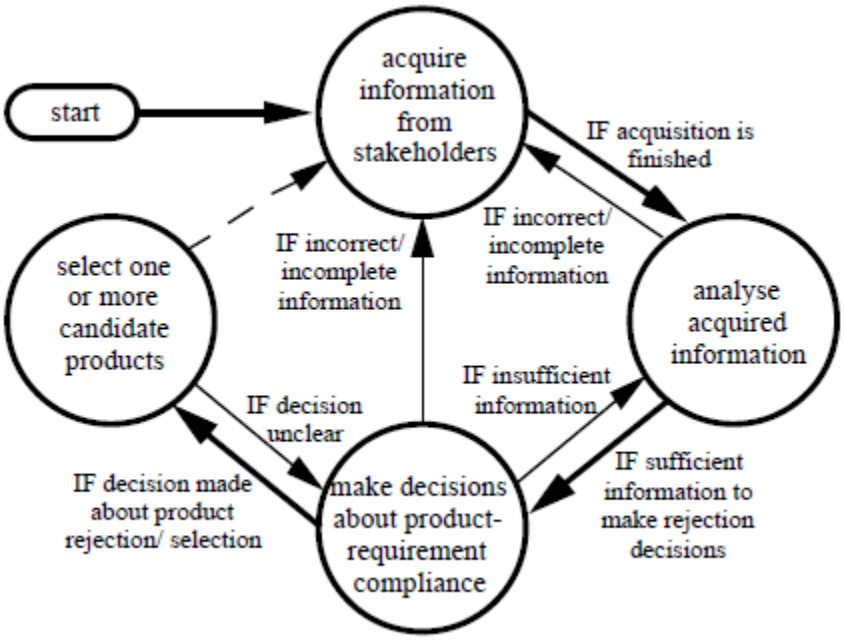

Figure 2.3: COTS Selection Process using PORE [23]

Drawbacks of PORE

- Iterative acquisition of requirements is a tedious and complex task and thus time & effort consuming.
- It partially addresses the mismatch problem. As it doesn't talk of the influence of the mismatch on System Under Development (SUD) and how to handle that mismatch.
- It doesn't support the resolution of the issue regarding the similar COTS components acquired from multiple stakeholders.
- Cost & Benefit analysis of the procured COTS is not available.
- Drawbacks of AHP limit its use.

2.3.2 COTS Aware Requirement Engineering (CARE)

The CARE approach is characterized as goal-oriented, agent oriented, knowledge based, and has a defined methodology or process. The Care process framework is based on five activities as shown in figure 2.4: Define Goals, Match Goals, Rank Components, Negotiate Changes and Select Components.

- It defines two types of requirement: Native (acquired from the stakeholders) and Foreign (acquired from the COTS capabilities).
- Knowledge Based: Because without the knowledge of the COTS components, they cannot be used [23]. Repository is maintained for storing the native requirements and the foreign requirements as well.
- Agent Oriented: The agents [25] are the various stakeholders involved in the SUD. An agent possesses characteristics including goals, beliefs and abilities that can make autonomous decisions, choices, and depends on other agents to accomplish goals, complete tasks, or furnish resources.
- Goal Oriented: The goals [26] are used to drive the development of the system. First, the goals are iteratively refined into system requirements which further refined into software, hardware, or interface requirements.
- It follows a well defined development process using COTS, so, it produces sustainable software. It is an iterative process for requirement acquisition, so it can welcome the changed objectives also.
- Maintenance of repositories, for knowledge based feature, of vendor's information that eases out the vendor selection problem.
- Use of NFR framework [27] leads to heavy documentation of goals given by the NFR. But it increases the efficiency of component selection for same goals in near future.
- It uses the elimination method by ranking the components based on the level of satisfaction of the requirements by the particular component.

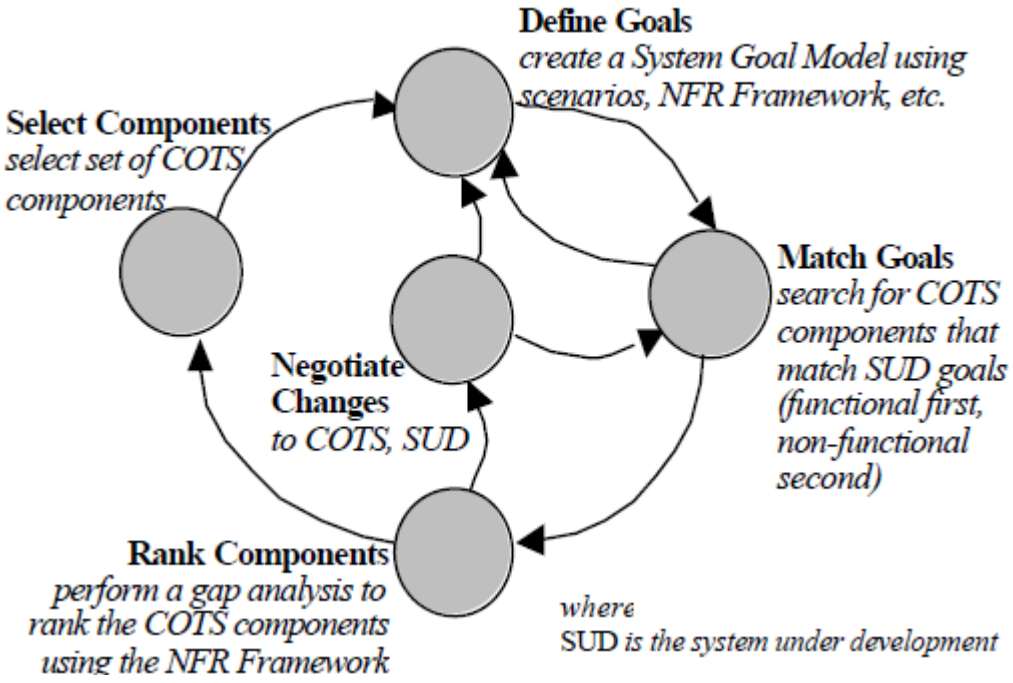

Figure 2.4: CARE – An Iterative process for Component Based Software Development [24]

Drawbacks of CARE

- Iterative nature of the elicitation, analysis, validation of requirements increases the time to deliver the frequent deliverables.
- Assumption of the characteristics of COTS components present in the repository puts restriction over the search of components for the desired goals of SUD.
- It partially addresses the mismatch problem as it doesn't talk of the influence of the mismatch on SUD and how to handle that mismatch.
- Drawbacks of AHP limit its use.

2.3.3 COTS Requirement Engineering (CRE)

CRE, as proposed by Alves et al. [28], is an iterative process for COTS component selection which aids the entertainment of changed requirements. It includes mainly 4 phases which goes iteratively.

- Requirement Acquisition
- Product Identification
- Product Description

- Product Acceptance

It is goal oriented, i.e. each phase is oriented by predefined goals. Each phase has a template that includes some guidelines and techniques for requirements acquisition/modeling and products evaluation. These templates describe the main goals as well as the final results of each phase. NFR framework [27] is focused as the main evaluation criteria for COTS as explained in figure 2.5 and thus selection of COTS component is based on rejection on the basis of time restriction, cost & benefit analysis, vendor reliability and many more "lities". Decision making analysis is done using the Multiple Criteria Decision Making (MCDM) techniques [10] such as Weighted Score Method (WSM). It aids component evaluation.

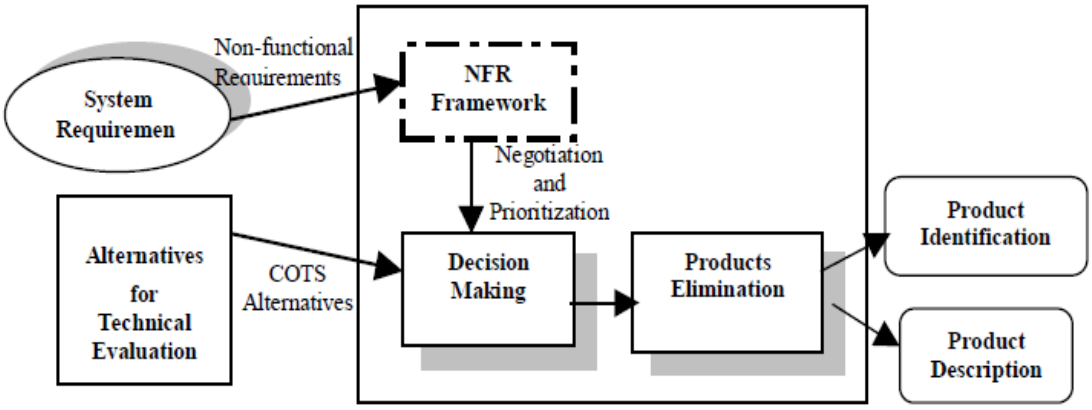

Figure 2.5: NFR Framework in Requirement Acquisition [29]

Drawbacks of CRE

- In case of large no. of COTS alternative and evaluation criteria, decision making process can be very complex due to the refinement of NFR's.
- Exit criteria from the evaluation of all NFR's are not cleared.
- Mismatch handling process of unsatisfied goal is absent.
- All drawbacks of WSM limit its use.

2.3.4 Off The Shelf Option (OTSO)

It was first COTS selection technique proposed in 1995, which presents a method (as in figure 2.6) for evaluating and selecting Off-The-Shelf software components to be reused in software development. Following describes the main motivation and principles of the method and provides a detailed description of it.

- It provides a well defined, systematic and detailed evaluation criteria definition ([30], [31], [32]) based on functional requirements, product quality characteristics, strategic concerns, and architecture compatibility.
- It provides a method for estimating the relative effort and cost-benefits analysis of different alternatives [31].
- Weighted Score Method (WSM) [10] is there for better decision making for the evaluation of potential alternatives.
- It reduces the time for COTS evaluation as it runs concurrently the evaluation criteria for the searched components.
- A method for comparing the "non-financial" aspects of alternatives, including situations involving multiple criteria [31].

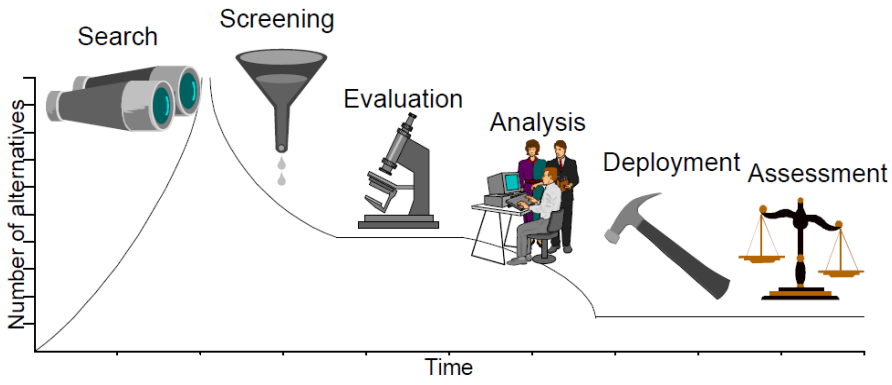

Figure 2.6: Main activities of the OTSO process [31]

Drawbacks of OTSO

- It directly starts with the evaluation criteria definition and fixed requirements as initially acquired.

- It doesn't identify mismatches that arise between the requirements and the COTS features and resolution action for them.

2.3.5 Mismatch-Handling aware COTS Selection (MiHOS)

By A. Mohammed et al. ([33], [34], [35], [36]), MiHOS (Mismatch-Handling aware COTS Selection) a COTS selection approach distinguished from other approaches by its capability to address COTS mismatches under limited resource constraints. The main activities involved are pointed out here and depicted in figure 2.7:

- As the name suggests, in this approach Mismatch handling is focused in which mismatches and their resolution actions are analyzed; a selected set of mismatches is then resolved.
- It addresses COTS mismatches (not architectural mismatches) during two main phases of the selection processes: (i) During COTS evaluation: It qualitatively handles mismatches and resolution will be just refining the requirement and rest mismatches are postponed to next phase. (ii) After COTS evaluation: mismatches are qualitatively addressed using matching level metric defined for each technical goal.
- COTS are selected on the basis of the maximum anticipated fitness value depending over the cost & effort required for resolving that mismatch.
- Weighted Score Method (WSM) [10] is there for better decision making for the evaluation of potential alternatives.
- Interactive Decision Support like Sensitivity Analysis ([34],[35]) of the COTS is also performed for the metric like anticipated fitness and DIFF metric [35].
- MiHOS-PT tool is used to help in applying IDS-SA.
- It also presents a taxonomy of COTS mismatches and Matching Level (ML) metric [33] which is described below:

$$ML = \begin{cases} 0 \; iff \; Zero \; Match \\ (0,1) \; iff \; Partial \; Match \\ 1 \; iff \; Exact \; Match \\ Undefined \; for \; over \; Match \; and \; Surplus \end{cases}$$

Drawbacks of MiHOS

- Non functional requirements are ignored.

2.4 Decision Making Techniques

When selecting a suitable COTS product, each COTS alternative should be ranked on how well it fits customers' requirements. Decision making techniques have been used in existing COTS selection methods for this purpose. The two most commonly used approaches are the Weighted Score Method and the Analytic Hierarchy Process.

2.4.1 Weighted Score Method (WSM)

The WSM method [10] calculates the overall fitness for each product against the evaluation criteria (i.e. the overall product's score) using the formula:

Total $Score_i$ for = Σ ($weight_j$ * $Score_{ij}$) for i=1 to n

Where $weight_j$ is the weight of the jth criterion, and $score_{ij}$ is the fitness score of the ith alternative in terms of the jth criterion. The weights are assigned by stakeholders, and the fitness score represents the compliance of the product with a specific criterion. An example is shown in Table 2.1, where the weights are represented on a 9-point scale:

- 1 indicates a criterion that is unimportant.
- 9 indicates a criterion that is extremely important.
- Any value between 1 and 9 indicates intermediate levels of importance.

COTS1 is selected as its total score, calculated using the formula given above, and has the maximum value. WSM technique is easy to apply and less time consuming.

Table 2.1: Example of Weighted Score Method

Criteria	Weight	COTS 1	COTS 2	COTS 3
Cost	5	1	0.5	0.5
Reliability	8	1	0.3	0.6
Performance	9	0.7	0.8	0.4
Usability	5	0.3	1	0.6
Security	9	0.8	0.6	0.7
Total Score		**28.0**	**22.5**	**20.2**

Drawbacks of WSM

- The results are represented by real numbers, which might be misinterpreted as difference between the candidates is very close.
- Estimation of the weights is difficult as it depends on the stakeholders' decision and also the number of criteria.
- Consolidation of the evaluation results into a single score is sometimes misleading because a high score in one attribute will hide a poor performance in another.

2.4.2 Analytic Hierarchy Process (AHP)

The AHP method [6] is based on arranging the evaluation criteria in a hierarchy, descending from the overall goal, the criteria, the sub criteria, and finally the COTS alternatives. The relative importance of the criteria at each level of the hierarchy is assessed by comparing them in pairs. Saaty et al. [21] introduced the following 9-point intensity scale which can be used in this comparison:

- 1 indicates that criteria C1 and C2 are of equal importance
- 9 indicates that criterion C1 is extremely more importance than C2
- Any value between 1 and 9 represents different levels of relative importance, while reciprocals means C2 is more important than C1.

The results of the comparison are then converted into normalized rankings using an Eigenvalue [21] technique on the comparison matrix. The normalized rankings represent the weights of the compared criteria. Similarly, the total weighted scores of different products can be estimated by comparing these products in pairs with respect to each criterion.

Table shows an example of applying AHP to weigh the four criteria at one level of the hierarchy. A pair-wise comparison is performed among the criteria, and the results are represented using the 9-point scale described above. For instance, criterion C1 in Table 2.2 is extremely more important than C2, while C4 is extremely more important than C1. The resultant normalized ranking (i.e. weight) of each criterion is shown in the right part of Table 2.2

Table 2.2: Example of Analytic Hierarchy Process

Criteria	C1	C2	C3	C4	Normalized Value
C1	1	9	3	1/9	.208
C2	1/9	1	5	1/9	.074
C3	1/3	1/5	1	1/7	.043
C4	9	9	7	1	.675

Drawbacks of AHP

- Comparison involves stakeholders' decision which might be misleading
- AHP involves many pair-wise comparisons, which would require a large amount of effort and time for a large number of criteria.

2.5 Mismatch Handling

Unlike traditional development, in Reuse Based Development, simultaneous tradeoffs are performed among the COTS features, User requirements, System architecture. These trade-offs are performed in order to detect the mismatches between the COTS features and user requirements or architectural differences between the desired system and of the COTS. According to A. Mohammed et al. [33], two types of mismatches are encountered during component selection: (i) Architectural Mismatches, which means difference in programming language, database types, calling procedures of different functions and (ii) COTS Mismatches, which means COTS features are not compatible with the stakeholder's requirements.

Yakimovich et al. [37] classified mismatches into two types: (i) wrong functionality, which means different functionality is incorporated in order to implement the required feature and (ii) missing functionality, which means the absence of a required feature.

Alves et al. [38] reformulated this work and investigated different types of issues caused by COTS mismatches. The 3 step framework by Alves, as shown in figure 2.7, classifies COTS mismatches into three types: (i) Differ: it means that there is partial match between a COTS feature and a stakeholder's requirement. For example, a requirement states that "the COTS product shall have spell check feature", but a COTS product does not support

this feature. (ii) Extend: it means that there are extra features present in the COTS when compared to the requirements. This excess might have helpful, neutral, or hurtful effects on the system. For example, a COTS product allows one user to open the files from remote location using FTP client but as not required. This extra feature might have a hurtful or helpful effect depending on the situation and (iii) Fail: this type occurs when there is a zero match between the COTS product and requirements. This type is similar to the missing functionality mismatch in Yakimovich's work [37].

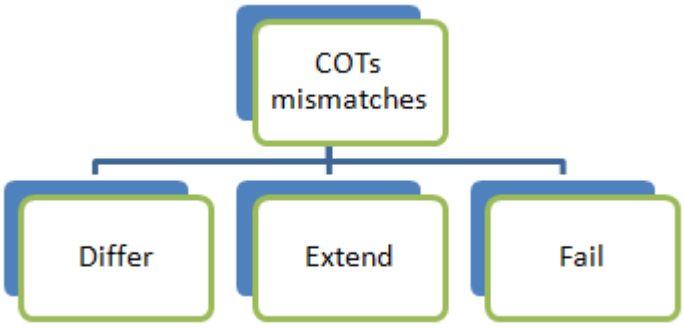

Figure 2.7: Mismatch taxonomy proposed by Alves

2.6 Quality in COTS selection

Presently quality assurance is an extended practice in many software engineering activities. One of these activities is Commercial Off-The-Shelf (COTS) components selection [40][41]. Assessing the quality of the COTS candidate components shortlisted for selection is a crucial issue. One of the approaches that have been used to analyze the quality of components is the definition of quality models [42] for representing the factors that impact on this quality, such as performance, integrity and interoperability. So a catalogue of quality factors is defined, either from the scratch or by refinement of a predefined one, and then the different COTS candidates are evaluated with respect to these factors. The ISO/IEC 9126[42] quality standard has been chosen for imparting quality into COTS selection. A structured quality model for given software provides a taxonomy of software quality features and also metrics for computing their value. The ISO/IEC[42] quality standard is chosen for the following reasons:

- It just fixes some general characteristics, and so the quality model may be tailored to any specific software domain. This is a crucial point, because quality models may dramatically differ from one domain to another.

- The standard explicitly recognizes the convenience of creating hierarchies of quality features, which is essential in order to build structured quality models.
- It is widespread.

ISO/IEC Software Quality Standard

A set of ISO/IEC standards are related to software quality, being standard number 9126, the most relevant one with respect to software selection [42]. The main idea behind this standard is the definition of a quality model and its use as a framework for software evaluation. A quality model is defined by means of general characteristics of software, which are further refined into subcharacteristics, which in turn are decomposed into attributes, yielding to a multilevel hierarchy; in fact, as mentioned by the standard, intermediate hierarchies of subcharacteristics and attributes may appear. At the bottom of the hierarchy there are the measurable software attributes, which values are computed by using some metric.

The ISO/IEC 9126 [42] standard fixes six characteristics:

- Functionality
- Reliability
- Usability
- Efficiency
- Maintainability
- Portability

The appropriated subcharacteristics and their attributes, and also the metrics for these attributes; attributes and even subcharacteristics will usually be organized as a hierarchy. Once this is process is completed, requirements over the domain, as well as package features, may be stated with respect to the resulting quality model. The framework can therefore be used to support the classical characteristics–requirement negotiation process during software package selection as shown in figure 2.8.

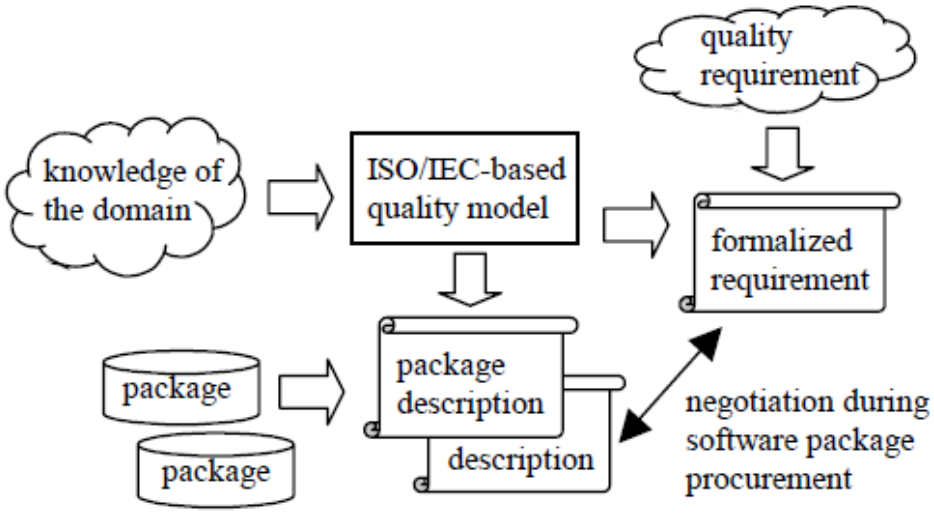

Figure 2.8: Using a quality model in software procurement [39]

This section describes the hierarchical feature graph that is used in our case study. The lowest level represents technical goals, while other levels represent strategic goals. Figure 2.9 shows the structure of hierarchical feature graph where Q_i, quality node is divided into a number of quality sub-feature nodes, S_i which are further divided into technical or functional feature nodes, F_i. The number of hierarchical levels depends on the system, but should be kept small to reduce complexity. The COTS features are evaluated against the functional feature nodes (F_i).

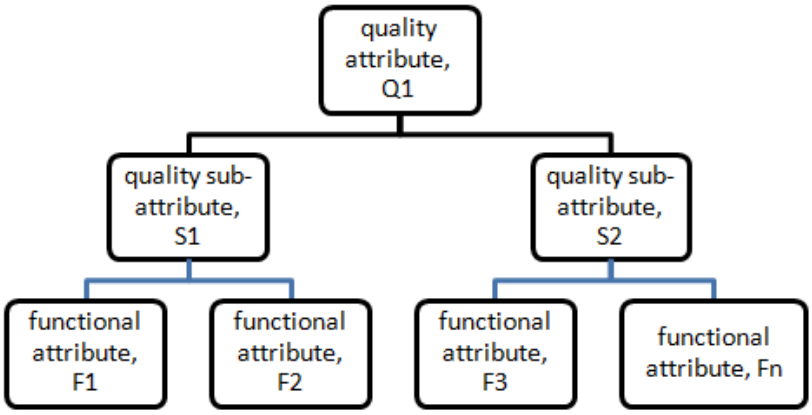

Figure 2.9: structure for hierarchical feature graph

CHAPTER 3
PROBLEM STATEMENT

In previous chapter we explored COTS selection processes for the development with reuse. We have also seen the advantages and limitations of all the techniques. But no technique is fully efficient. Because some of the process like CARE, PORE, supports the partial mismatch handling but the whole process becomes complex for large number of COTS alternatives whereas MiHOS supports mismatch handling in two phases but it leaves out the quality aspect while selection. COTS are represented in functional, non-functional and architectural attributes in CRE but the whole process is not well defines as it doesn't talk about the exit criteria from NFR framework. Following Table shows the limitations of various COTS selection techniques discussed in previous chapter.

Table 3.1: Limitations of current COTS selection techniques

COTS selection technique	**Limitations**	**Requirements for a new method**
PORE	Limitations due to use of AHP, iterative requirements engineering steps lead to time consumption	No single method for handling the mismatches as well as identifying the quality requirements.
CARE	Limitations due to use of AHP, iterative requirements engineering steps lead to wastage of time	
CRE	Refinement of NFR's is very tedious , mismatch handling is not present	
OTSO	Doesn't identify mismatches between requirements and COTS features	
MiHOS	Quality requirements are ignored	

In the previous chapter, feature analysis technique using hierarchical feature graph is explored in which the requirements are classified into hierarchies and this graph is then used to find out the satisfaction value between the COTS features and the features required by the system. Defining the feature graph is subjective and depends on the project context as well as stakeholders' needs and understanding. This means the results obtained in this case study might slightly differ if applied in a different context by different stakeholders. Metrics are used for evaluating the operational characteristics of the COTS components.

The aim of this thesis work is to propose an Improved Method for Selection of COTS components which is based on quality requirements. It also aims to gather and resolve maximum number of mismatches between the COTs features and the requirements which

would help in decreasing the time and effort in COTS integration. So, the objectives of this thesis are:

- To identify the limitations of various COTS selection techniques in handling Quality while selection
- To suggest a new COTS selection process taking care of Quality requirements.
- To validate the proposed method using a case study.

CHAPTER 4

PROPOSED COTS SELECTION METHOD

As discussed in the previous chapter, many COTS selection techniques are already proposed but they have some limitations too. Most of the techniques make use of AHP, WSM as a decision making technique for the evaluation of different COTS products. But these techniques come with drawbacks also. AHP involves many pair-wise comparisons, which would require a large amount of effort and time for large no. of components. Techniques like CARE, PORE handles mismatches partially. In order to overcome these aspects, the proposed model focuses on goal graph and WSM based feature analysis of COTS products in order to have the detailed evaluation of most promising candidates.

4.1 The systematic Process

The systematic process of the proposed cots selection has been divided into following phases. The whole process is described in figure 4.1.

- Requirement acquisition
- Searching
- Filtering
- Detailed evaluation using hierarchical feature graph and WSM
- Mismatch handling
- Selecting the component

4.1.1 Requirement Acquisition

Requirements form the basis for any effective COTS selection technique as they provide the evaluation criteria for any COTS. Requirements are elicited from various stakeholders on the basis of the various requirement elicitation techniques like interviewing and brain storming session. This phase ends with the software requirements specification (SRS) document.

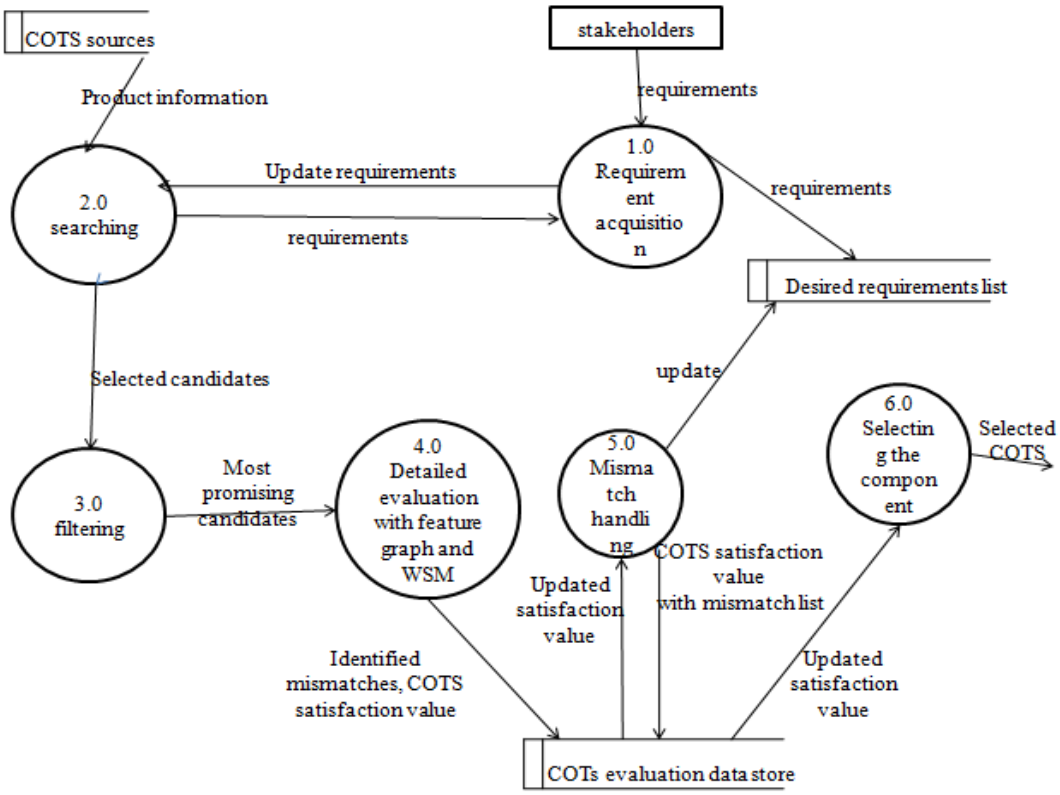

Figure 4.1: DFD of the proposed selection method

4.1.2 Searching

In this phase, potential reusable candidates are identified. In initial keyword based search is performed over many cots sources. These sources can be organization-specific in-house libraries or over the internet (for example, google.com, yahoo.com, infoseek.com) as they contain the updated information about the commercial, shareware and freeware software as well. Search results in the name of the potential components, their website address, vendor history, product licensing features and characteristics.

The search should be initiated as soon as main requirements or the basic features are known from the stakeholder's side. Search and requirement acquisition goes parallel and thus entertains the requirements discovered at later stage.

4.1.3 Filtering

In this phase, identified potential candidates are examined with the aim of getting most promising candidates. Basically, the list of searched candidates is filtered out so as to reduce the time and effort required for the detailed evaluation of each and every COTS. Filtering is performed using the criteria based on requirements like operating system

support, programming language difference and cost factor. But for ease of understanding the cots considered here are freeware. Moreover, the preparation of requirement checklist using the COTS information gathered in the search phase is also helpful for the filtration. COTS with maximum availing functionality are filtered out and act as most promising candidates.

4.1.4 Detailed evaluation

The aim of this phase is to evaluate the candidates that come out as a result of filtration, on the basis of their behavior captured during their dynamic analysis using pilots, trial version, hands on demonstrations or execution. The evaluation results are then documented. Dynamic analysis works well with COTS components as they are black box in nature and available with the .exe files, design specification without source code and trial demonstrations. So, the technique applies straightforwardly to COTS components, which are particularly difficult to address with the traditional testing and analysis techniques.

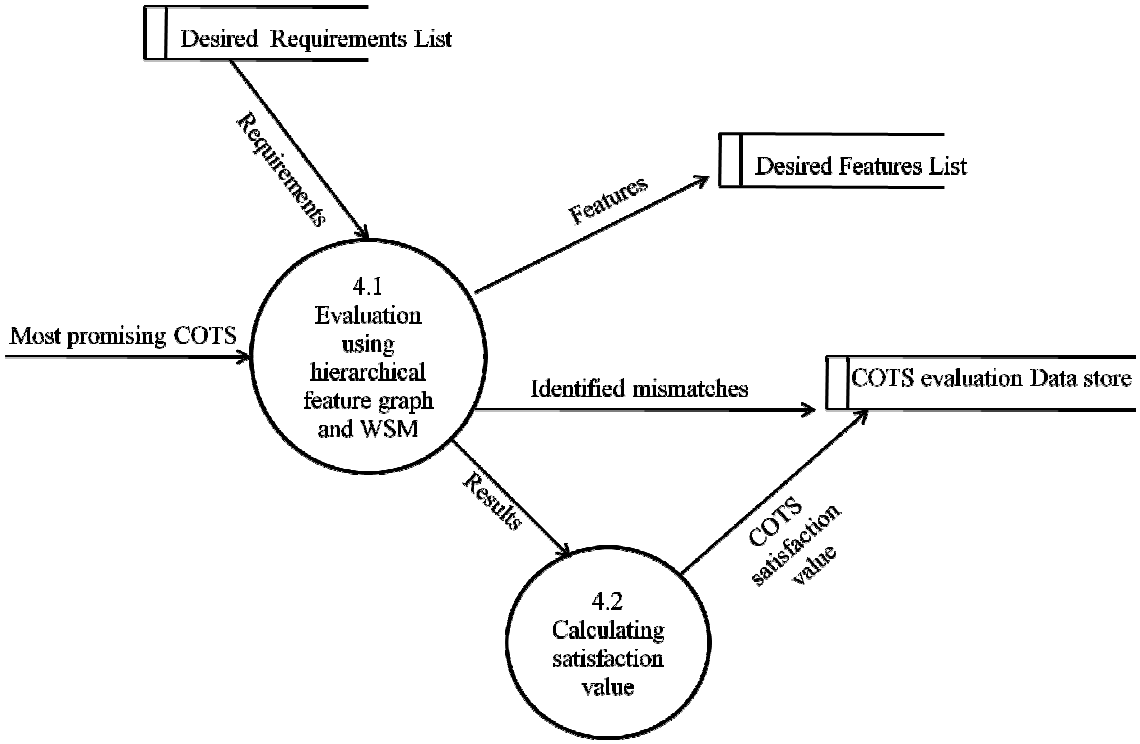

Figure 4.2: DFD of the detailed evaluation

Here detailed evaluation includes two steps, namely:

- **Creating hierarchical feature graph implementing ISO/IEC-9126 quality standard-** It overcomes the drawbacks incorporated by AHP for decision making in COTS selection. It includes the quality factor to the selection of COTS components that must be present. The hierarchical feature graph for the quality attribute, reliability is given below in table 4.1.

Table 4.1: Hierarchical Feature Graph for "Reliability"

Goal Structure		
Features	**Metric**	**Weights**
RELIABILITY		6
Recoverability		10
Reopen accidentally closed tabs	BIN	10
Tab Recovery with message & prompting when website times out	BIN	2
Vendor Support		10
Tutorials	BIN	4
Blogs	BIN	2

Once the features, sub features and their functional counterparts are identified, metrics are defined for each functional attribute for evaluation of the COTS products and able to get the satisfaction value of the different COTS. Two types of metrics defined for the measurement of the functional attributes: Ordinal metric and Binary metric. These metrics are explained in appendix A.

- **Calculating Satisfaction Value**- It includes assigning weights to the quality attributes and their sub-features in the hierarchical feature graph and calculating the weighted satisfaction values for each and every attribute using the WSM decision making technique. Weights are assigned to the features for calculating the satisfaction value using the WSM which is explained in section 2.4.1. To calculate satisfaction value of each promising COTS, normalization of the ordinal values assigned to sub-features of promising COTS in the ordinal scale should be done. For the normalization process, satisfaction function is defined which calculates the satisfaction values for all the features of each COTS candidate.

Satisfaction function

SV: set of ordinal values -> [0, 1] as

$SV(x_i) = a * x_i + b$ where the given initial conditions are:

$SV(0) = 0$

$SV(3) = 1$

$$SV(4) = \begin{cases} 1 \; if \; the \; impact \; is \; neutral \\ \qquad or \; helpful \\ 0 \; if \; the \; impact \; is \; hurtful \end{cases}$$

These initial conditions compute a=1, b=0 and SV = $x_i/3$ where xi is the ordinal value assigned as per the scale defined in appendix A.2.

$$SV(COTS) = \sum SV(x_i)$$

This COTS evaluation data is then properly documented into COTS evaluation data store.

4.1.5 Mismatch handling

Mismatch handling process is subdivided into 4 activities

- Classify mismatches
- Suggest mitigation action
- Refine feature
- Update mismatch value

$MMV(x_i) = 1 - SV(x_i)$ for each features of COTS defined in the template, where MMV (x_i) stands for Mismatch value.

Mismatches identified in the previous phase are analyzed and the decision about their handling is taken. On the basis of the taken decision, mismatches are into 3 categories:-

- **Resolvable mismatches** – in this category mismatches are resolved by customizing the COTS functionality using any mitigation action as described in sub-section 4.2.
- **Refineable mismatches**- this case arises when the corresponding feature is absent but the said feature is covered by another COTS capability. For example, in a web browser the desired feature “does the system provide customer support?” can be refined as it provides help through discussion forums.
- **Ignorable mismatches**- on the basis of the mismatch value and their impact over the application, mismatches are ignored. It is subjective to the stakeholders’ decision.

By suggesting the mitigation action for the respective resolvable mismatches or for the ignorable mismatches, the mismatch value is changed to zero. Corresponding mitigation action and new mismatch values (MMV_{new}) are updated for the respective COTS in the COTS evaluation data store.

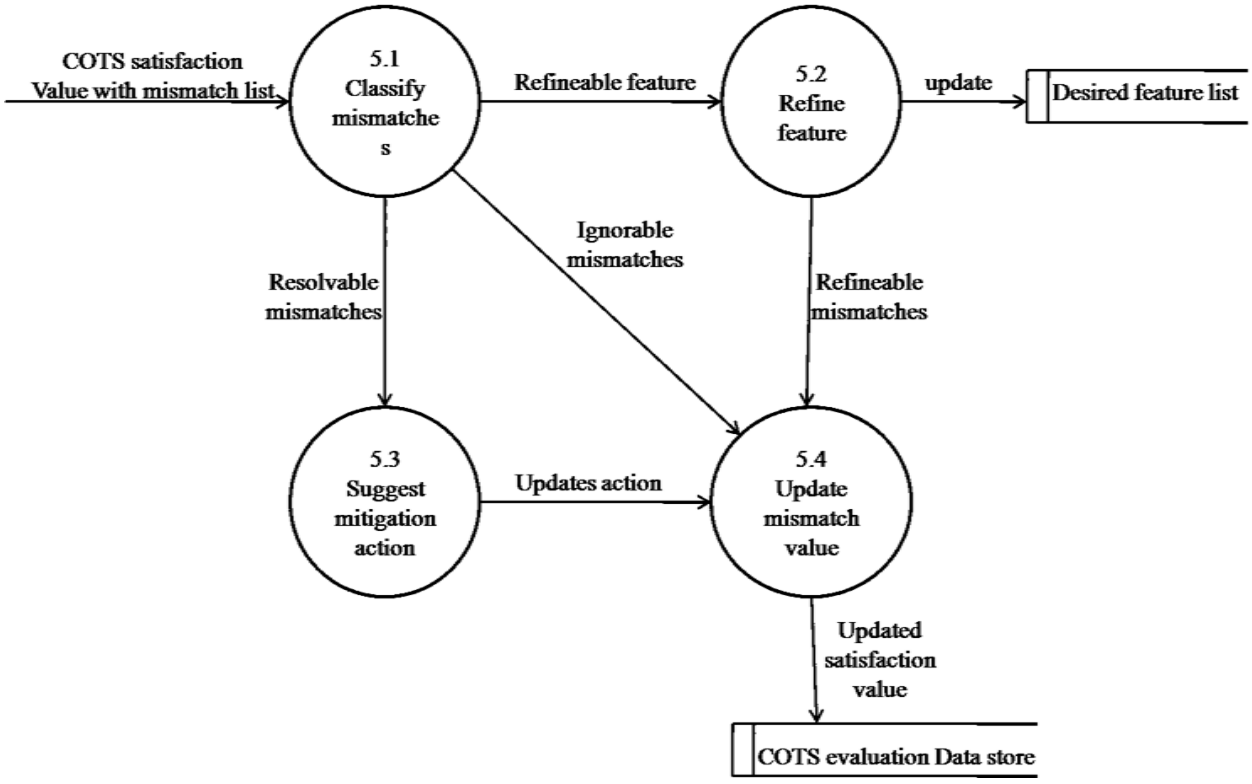

Figure 4.3: DFD for mismatch handling

4.1.6 Selecting the component

Calculate the new satisfaction value for each of the most promising candidate. Selection of COTS on the basis of the new satisfaction value after handling the mismatches is then done. The new satisfaction value is calculated as follows:

$$SV_{new}(COTS) = \sum (1 - MMV_{new}(x_i))$$

The COTS with the highest SV is selected.

4.2 Mismatch Mitigation Action

COTS mismatches mitigation includes two actions:

4.2.1 Refining requirements

This strategy suggests modification in the requirements, and thus making a compromise that reduces the mismatch. This action is taken when the desired feature is not present in the COTS but other attributes can satisfy the requirements. However, the current COTS selection methods only address the COTS mismatch problem during the requirements definition phase, but these approaches do not talk about the effect of mismatches on COTS selection, nor do they talk of mitigation action for the remaining violations. But in this thesis work this has been considered.

4.2.2 Resolvable mismatches

This strategy is used to resolve mismatches that are related to missing COTS functionality. According to Vigder et al. [7] this can be realized by several alternatives like:

- Acquiring additional plug-ins that add functionality to the COTS product,
- Writing custom code using a scripting language supported by the COTS, for example JavaScript, Visual Basic, and Perl.
- API – Application Programming Interface that is an interface implemented by a software program which enables it to interact with other COTS software.
- GUI based tailoring
- Modifying source code

CHAPTER 5
EXPERIMENTAL RESULTS

The specific goal of the case study is to validate the feasibility of the proposed model for the COTS selection. Web browsers are the applications for retrieving, presenting, and traversing information resources on the World Wide Web. An *information resource* is identified by a Uniform Resource Identifier (URI) and may be a web page, image, video, or other piece of content. Although browsers are primarily intended to access the World Wide Web, they can also be used to access information provided by web servers in private networks or files in file systems. Study shows how the process is used to select the best Web Browser having the maximum satisfaction value with desired set of requirements among the other alternatives.

5.1 Application of Proposed COTS Selection Process

This section presents the implementation of the proposed method to the case study. Each phase is described with respect to the case study under consideration.

5.1.1 Requirement acquisition

This phase includes the process of gathering user's requirements. It results in various functional requirements formulated in software requirement specification document as the output. For the proposed selection method the functional requirements are considered as way of implementing the quality requirements by adapting the ISO/IEC 9126 Software Quality Standard [42]. Table 5.1 shows the most basic requirements of the user that need to be there in the COTS selected.

5.1.2 Searching

In order to identify possible COTS candidates based on the gathered requirements, several information sources have been used such as online search engines like Google and Yahoo, as well as specialized websites that list and categorize various web browsers products, for example www.cnet.com. This resulted in identification of an initial set of 8 Windows based web browsers as shown in Table 5.2.

Table 5.1: Requirements specification for the Web browser case study

Requirements	Requirements description
User interface	The user interface should provide a tabbed browsing for moving between different pages at a time
Hardware requirement	Should demand minimum space on HDD and RAM
Privacy and security	Should provide pop-up blocking
Standards support	Should be compliant to the world wide standards
Added functionality	Should provide a dedicated download manager, auto-update function
Operating system	Must work on Windows

Table 5.2: List of COTS searched

COTS ID	Product Name	Version	Website	Vendor
COTS 1	Internet explorer	9.0	windows.microsoft.com	Microsoft
COTS 2	Lunascape	6.0	lunascape.tv	Lunascape corporation
COTS 3	Firefox	4.0	mozilla.com	Mozilla foundation
COTS 4	Opera	11.11	opera.com	Opera software
COTS 5	Chrome	10.0	google.com/chrome	Google
COTS 6	Seamonkey	2.1	seamonkey-project.org	Seamonkey council
COTS 7	Safari	5.0	apple.com/safari	Apple inc.
COTS 8	Konqueror	3.5	Konqueror.org	KDE

5.1.3 Filtering

Now the list of searched COTS is filtered according to the evaluation criteria defined in the SRS. These 5 promising COTS candidates namely, COTS1, COTS2, COTS3, COTS4 and COTS5, as shown in Table 5.4, are selected as a result of filtering on the basis of

non-compromising requirements of the stakeholders. Each of the selected COTS fulfills all the requirements listed in the Table 5.1.

Table 5.3: List of COTS filtered

COTS ID	Tabbed browsing	Ad filtering	Pop-up	Download mgmt.	Privacy mode	Auto update
COTS 1	✓	✓	✓	✓	✓	✓
COTS 2	✓	✓	✓	✓	✓	✓
COTS 3	✓	✓	✓	✓	✓	✓
COTS 4	✓	✓	✓	✓	✓	✓
COTS 5	✓	✓	✓	✓	✓	✓
COTS 6	✓	✓	✓	✓	X	✓
COTS 7	✓	X	✓	✓	✓	✓
COTS 8	✓	✓	✓	✓	X	X

Table 5.4: List of the most promising candidates after filtering

COTS ID	Product Name	Version	Website	Vendor
COTS 1	Internet explorer	9.0	windows.microsoft.com	Microsoft
COTS 2	Lunascape	6.0	lunascape.tv	Lunascape corporation
COTS 3	Firefox	4.0	mozilla.com	Mozilla foundation
COTS 4	Opera	11.11	opera.com	Opera software
COTS 5	Chrome	10.0	google.com/chrome	Google

5.1.4 Detailed Evaluation using Hierarchical Feature Graph and WSM

The objective of this phase is to evaluate the list of shortlisted candidates as an input to this phase from the previous phase. Evaluation of all the selected five candidates is done by executing all the five candidates and then scoring is done in parallel using the metrics defined in Appendix A.

Table 5.5: Sample of Hierarchical Feature Graph for "usability"

Goal Structure	**COTS 1 Matching**				
Features	**Metric**	**Weights**	**Value**	**SV**	**Type**
USABILITY		10		0.85	
Understandability		6		1.00	
Interface Languages Supported	ORDN	6	3	1.00	exact match
Learnability		10		1.00	
Tutorials	BIN	8	1	1.00	TRUE
Blogs	BIN	4	1	1.00	TRUE
Operability		10		0.82	
Tab Stacking	BIN	2	0	0.00	FALSE
Page Zooming	BIN	8	1	1.00	TRUE
Mouse Gesture	BIN	1	1	1.00	TRUE
Full Text Search Of History	BIN	7	1	1.00	TRUE
Voice Control	BIN	6	1	1.00	TRUE
Caret Navigation	BIN	4	1	1.00	TRUE
Password Management	BIN	8	1	1.00	TRUE
Spatial Navigation	BIN	3	0	0.00	FALSE
Attractiveness		10		0.63	
Search Toolbar	BIN	10	1	1.00	TRUE
Pdf Viewer	BIN	10	0	0.00	FALSE
Spell Check	BIN	8	0	0.00	FALSE
Themes	BIN	7	1	1.00	TRUE
Form Management	BIN	8	1	1.00	TRUE
Bookmark Management	BIN	6	1	1.00	TRUE

The evaluation is done on the basis of the 6 quality features that include 17 quality sub-features proposed based on the ISO/IEC-9126 quality standard using the hierarchical feature graph(as discussed in detail in Appendix B). In the graph the features are presented to the stakeholders and their importance is judged. Accordingly, weights are assigned to each feature and sub feature. So, WSM is used to find the summations of

weights for each COTS candidate. For example, the functional feature "spatial navigation" in Table 5.5 is evaluated as:

Metric type: BIN

Value: 0 means the feature is not present in COTS 1.

Type: FALSE

Weight: 3

Normalized Satisfaction Value: 0.00 indicated no satisfaction between the COTS1 capability and the required feature.

As calculated in Appendix C, from the detailed evaluation of the most promising candidate COTS, table 5.6 shows the two COTS with the highest satisfaction value that are selected. So, COTS1 and COTS5 are selected for further mismatch handling.

Table 5.6: Summary of Evaluation Results before mismatch handling

COTS ID	Satisfaction Value
COTS 1	0.93
COTS 2	0.42
COTS 3	0.75
COTS 4	0.77
COTS 5	0.89

5.1.5 Mismatch Handling

As per the results shown in the Table 5.6, selected COTS 1 and COTS 5 have the top most satisfaction value. As per the process, the identified mismatches have been classified into the resolvable, ignorable mismatches and the refineable features. First the mismatch value is calculated for all the promising candidates using:

$$MMV(x_i) = 1 - SV(x_i)$$

Mismatch Action is chosen in the stakeholder's discussion session for the respective mismatches. New mismatch value is assigned to the resolved mismatches or ignored mismatches as 0. Finally the new satisfaction value of each feature and then of respective COTS as follows:

$$SV_{new}(COTS) = \sum(1 - MMV_{new}(x_i))$$

For example, in COTS 5, mismatch of incompatibility with the older version web pages can be resolved by using an extension that is available for displaying such pages. Mismatch handling has been detailed in detail in Appendix C.

Table 5.7: Results after mismatch handling

COTS ID	New Satisfaction Value
COTS 1	0.96
COTS 5	0.98

5.1.6 Selecting the COTS

According to the new Satisfaction Values, COTS 5 i.e. Google Chrome 10.0 is selected because it comes with a higher satisfaction value when compared with the requirements.

CHAPTER 6

CONCLUSION & FUTURE SCOPE

The aim of this thesis is to propose a COTS selection method which helps in detection and resolution of the mismatches between the COTS features and the stakeholders' quality requirements during the COTS selection so as to decrease the time and effort spent over the COTS integration. This thesis proposes a six-step COTS selection, method in which the evaluation of COTS is based on hierarchical quality feature graph which implements the ISO/IEC-9126 quality standard. Mismatches are detected and Mismatch mitigation actions are suggested for the identified mismatches belonging to resolvable category. Refinement of the requirement is carried in case if the requirement is satisfied by another feature of the COTS.

6.1 Conclusion

As the evaluation of COTS in the proposed method is carried out on the basis of the hierarchical feature graph, which resolves the drawback of AHP by saving time spent on large number of pair-wise comparisons of requirements. Also, this method is an improvement in a way that it specifies the quality requirements along with mismatch handling. So, the major conclusions derived from the study are:

- Mismatches are not deferred till the integration phase as it detects most of them and suggests mitigation action for the same during the evaluation phase itself.
- It also covers quality aspects of the system by co-relating the functional and quality features using the hierarchical feature graph. So, it's an improvement over MiHOS.

6.2 Future Scope

The initial keyword based search can be replaced by any other COTS retrieval technique so as to retrieve more relevant components. As the process focuses on the single COTS software based system, so, multiple COTS selection need to be addressed for implementing the method to COTS-intensive systems. For this, issues like interoperability between COTS and architectural mismatches need to be explored. Applying the process to different case studies of various domains for a better evaluation. A tool may be developed to support the automation of this selection method.

REFERENCES

[1] Charles W. Krueger, Software Reuse, ACM Computing Surveys, Vol. 24, pp. 131-83, June 1992.

[2] I. Crnkovik, S. Larsen and M. Chaudron, “Component-based Development Process and Component Lifecycle”, 27th International Conference Information Technology Interfaces (ITI), IEEE, Cavtat, Croatia, June 2005.

[3] P. C. Clements, “From Subroutines to Subsystems: Component-Based Software Development”, American Programmer, vol:8, No:11, November 1995.

[4] G. T. Heineman, W. T. Councill, “Component-Based Software Engineering: Putting the Pieces Together”, Addison-Wesley Professional, 1st ed., May 2001.

[5] M.R. Vidger and J. Dean, “An Architectural Approach to Building Systems from COTS Software Components”, Proceedings of CASCON '97, **NRC Publication Number:** NRC 40221, Toronto, Ontario, Canada, 10-13 November, 1997. pp. 131-143

[6] M. Torchiano, M. Morisio, “Overlooked Aspects of COTS-Based Development, IEEE Software, vol:21, no:2, pp. 88-93, March 2004

[7] M.R. Vidger, M.W. Gentleman and J. Dean, "COTS Software Integration: State of the Art," NRC-CNRC Report, National Research Council, Canada, Jan. 1996.

[8] H. Mili, A. Mili, S. Yacoub, and E. Addy, “Reuse Based Software Engineering”, Wiley-Interscience Publication, USA, 2002.

[9] T. Wanyama and B. Far, “An Empirical Study to Compare Three Methods for Selecting Cots Software Components”, International Journal of Computing and ICT Research, Vol:2, No:1, pp. 34 – 46, June 2008.

[10] T. Wanyama and B. Far, “Towards Providing Decision Support for COTS Selection”, Electrical and Computer Engineering, 2005. Canadian Conference on, pp. 908-911, 1-4 May 2005.

[11] C. Abts, B. W. Boehm, and E. B. Clark, "COCOTS: a COTS software integration cost model model overview and preliminary data findings," in The 11th ESCOM conference, Munich, Germany, pp. 325-333, 2000.

[12] B. W. Boehm, D. Port, Y. Yang, and J. Bhuta, “Not All CBS Are Created Equally: COTS-Intensive Project Types”, 2580 ed. Berlin Heidelberg: Springer-Verlag, 2003.

[13] C. Ncube and N. A. Maiden, "PORE: Procurement-Oriented Requirements Engineering Method for the Component Based Systems Engineering Development Paradigm," in International Workshop on Component Based Software Engineering (in conjunction with ICSE’99) Los Angeles, CA, 1999.

[14] L. Chung and K. Cooper, "A COTS-Aware Requirements Engineering (CARE) Process: Defining System Level Agents, Goals, and Requirements," Department of Computer Science, The University of Texas, Dallas, TR UTDCS-23-01, Dec 2001.

[15] SEI: Software Engineering Institute in Carnegie Mellon Univ., available at http://www.sei.cmu.edu.

[16] G. Ruhe, "Intelligent Support for Selection of COTS Products," Web, Web-Services, and Database Systems, Lecture Notes in Computer Science, Springer, vol:2593, pp. 34-45, 2003.

[17] A.Mohamed, “Decision Support for Selecting COTS Software Products Based on Comprehensive Mismatch Handling”, Departmant of Electrical and Computer Engineering, Calgary, Alberta, April 2007.

[18] G. Fox, K. Lantner and S. Marcom, "A software development process for COTS-based information system infrastructure," Assessment of Software Tools and Technologies, Proceedings Fifth International Symposium on, pp.133-142, 2-5 Jun 1997.

[19] C. Ncube and N. Maiden, ‘Procuring Software Systems: Current Problems and Solutions’, Proceedings REFSQ97 workshop, CaiSE97, Barcelona, Spain, 16-17 June, 1997.

[20] C. Ncube and N. Maiden, “PORE: Procurement-Oriented Requirements Engineering Method for the Component-Based Systems Engineering Development Paradigm”, International Workshop on Component Based Software Engineering, 1999.

[21] Saaty T. L., 1990, “The Analytic Hierarchy Process”, New York, McGraw-Hill, 1990.

[22] B. Kitchenham and L. Jones, “Evaluating Software Engineering Methods and Tools: Part 5, The Influence of Human Factors”, Software Engineering Notes 22, vol:1, 1997.

[23] L. Chung and K. Cooper, "Knowledge-based COTS-aware requirements engineering approach," 14th International Conference on Software Engineering and Knowledge Engineering (SEKE'02) Ischia, Italy, 2002.

[24] L. Chung and K. Cooper, "A COTS-Aware Requirements Engineering (CARE) Process: Defining System Level Agents, Goals, and Requirements," Department of Computer Science, The University of Texas, Dallas, TR UTDCS-23-01, Dec 2001.

[25] L. Chung and K. Cooper, “Defining Agents in a COTS-Aware Requirements Engineering Approach”, in Proc. of7th Int. Australian Workshop on Requirements Engineering, 2002.

[26] L. Chung and K. Cooper, "Defining Goals in a COTS-Aware Requirements Engineering Approach," Systems Engineering journal, vol:7, issue:1, pp. 61-83, 2004.

[27] J. Mylopoulos, L. Chung, and B. Nixon, “Representing and using non-functional requirements: a process- oriented approach”, IEEE Transactions on Software Engineering, Vol:18, Issue:6, pp.483-497, June 1992.

[28] C. Alves, J. Castro, CRE: A Systematic Method for COTS Components Selection. XV Brazilian Symposium on Software Engineering (SBES) Rio de Janeiro, Brazil, October 2001.

[29] Alves, C., Castro, J., and Alencar, F., “Requirements Engineering for COTS Selection.” In Third Workshop on Requirements Engineering, Rio de Janeiro, Brazil, 2000.

[30] J. Kontio, OTSO: a systematic process for reusable software component selection, University of Maryland at College Park, College Park, MD, 1995.

[31] J. Kontio, “ A Case Study in Applying a Systematic Method for COTS Selection”, In Proc. Of ICSE-18, pp.201-209, 1996.

[32] J. Kontio and R. Tesoriero, "A COTS selection method and experiences of its use," in The Twentieth Annual Software Engineering Workshop Greenbelt, Maryland, 1995.

[33] Mohamed, A., Ruhe, G., Eberlein, A.:“MiHOS: an approach to support handling the mismatches between system requirements and COTS products”, Requirements Engineering Journal, 2007.

[34] Mohamed, A., Ruhe, G., Eberlein, A.,“ Decision support for customization of the COTS products and system requirements”, 6th international conference on COTS-based software systems (ICCBSS’07), Banff, Canada, pp. 63-72, 2007.

[35] Mohamed, A., Ruhe, G., Eberlein, A., “decision support for customization of the COTS selection process”, ACM SIGSOFT software engineering notes, in Proc. Of ICSE’05 workshop on models and processes for the evaluation of COTS components (MPEC’05), vol:14(4), pp. 1-4. 2005.

[36] Mohamed, A., Ruhe, G., Eberlein, A., “Integrating Mismatch Management into COTS Selection Process," in The 3nd Faculty of Engineering Annual Grad Conference (EGC'06) UofC, Calgary, AB, 2006

[37] D. Yakimovich, J. M. Bieman, and V. R. Basili, Software architecture classification for estimating the cost of COTS integration. Los Angeles, California, United States: IEEE Computer Society Press, 1999.

[38] C. Alves and A. Finkelstein, "Investigating Conflicts in COTS Decision-Making," International Journal of Software Engineering and Knowledge Engineering (IJSEKE), vol:13 (5), pp. 473-493, 2003.

[39] Xavier Franch, Juan P. Carvallo, “A Quality-Model-Based Approach for Describing and Evaluating Software Packages”, Proceedings of the IEEE Joint International Conference on Requirements Engineering (RE’02), 2002.

[40] J. Kontio. “A case study in applying a systematic method for COTS selection”. In Proceedings 18th International Conference on Software Engineering (ICSE’96), 1996.

[41] N. Maiden, C. Ncube. “Acquiring Requirements for COTS Selection”. IEEE Software, vol:15(2), 1998.

[42] International Organization for Standarization. ISO/IECStandard 9126: Software Engineering – Product Quality, part 1, 2001.

PAPERS PUBLISHED/COMMUNICATED

1. Ravneet Kaur Grewal, Shivani Goel, "Importance of Quality Models in Commercial Off The Shelf Components Selection", International Journal of Software Engineering (IJSE), Vol. 2, No. 2, July 2011. (Accepted)

APPENDIX A

TYPE OF MEASUREMENTS IN THE PROPOSED METHOD

A.1 Measurement Scales

Fenton and Pfleeger [90] define measurement as "The process by which numbers or symbols are assigned to attributes of entities in the real world in such a way as to describe them according to clearly defined rules". This means, different scales should be used to measure different data types. On this basis, the proposed method uses several types of measures defined on different scales in order to evaluate different types of COTS functionalities.

Also, Comella-Dorda et al. [17] suggest that when selecting COTS products, a consistent scale must be eventually used to represent results in order to understand the big picture when comparing different COTS products. This appendix describes the types of measures that have been used and their normalization rules.

In general, Fenton and Pfleeger [90] define five scales to be used in any measurement process:

- **Nominal**, in which categories are defined and attributes are characterized by these categories. The values of a nominal scale have no numeric meaning. For example, "color" might be measured on a nominal scale {"red", "blue", "green"}.
- **Ordinal**, in which a set of ordered values is used to measure an entity. Typically, the intervals between adjacent values are undetermined. For example, "technical support" might be measured on an ordinal scale {"good", "fair", "bad"}, where we know that "good" is better than "fair", but the distance between them is unknown.
- **Interval**, which preserves the order (as with an ordinal scale) as well as the differences; i.e. that the intervals between adjacent values are meaningful. However, the zero point on the scale is arbitrary, which means ratios are meaningless. For example, year 2006 is not as twice as much as year 1003, but the difference between year 2006 and 2000 is the same as the difference between year 1006 and 1000.

- **Ratio**, which is similar to interval scales except they have true zero point; i.e. it preserves the ratios between scale values. For example, a 40 year-old person is twice as old as a 20-year old one.
- **Absolute**, which simply counts the number of occurrences of an element in an entity. For example, number of line-of-code is measured by counting the number of lines in an application.

A.2 Metrics in Proposed Method

Two types of measures are assigned to functional features, and used when evaluating COTS products. For each type, a rule is defined for mapping the measured value to the ML range from 0 to 1. The following metrics are used:

- **Binary Metric (BIN)**

These metrics are the simplest, but yet the most common type of measures used in COTS evaluation. BIN measures are used to indicate the presence or absence of a COTS attribute. A measure x that is BIN is estimated on a nominal scale, x ∈ {True, False}. The rule used to map x to the ML's range is:

$$ML = \begin{cases} 0 \; iff \; x = FALSE \\ 1 \; iff \; x = TRUE \end{cases}$$

For example, a BOLN measure assigned to the technical-goal "support SSL protocol" means it can either be fully satisfied (i.e. "True", ML=1), or fully declined (i.e. "False", ML=0).

- **Ordinal metric (ORDN)**

These metrics are used for qualitative measurement. A metric x that is ORDN-based, is estimated on an ordinal scale, x ∈ {Zero Match, Very poor match, poor match, exact match, over match}. The scale assumes equal difference between the values.

$$Value = \begin{cases} 0 \; iff \; x = Zero \; Match \\ 1 \; iff \; x = Very \; Poor \; Match \\ 2 \; iff \; x = Poor \; Match \\ 3 \; iff \; x = Exact \; Match \\ 4 \; iff \; x = Over \; Match \end{cases}$$

For example, ORDN metric assigned to the question "the amount of RAM or HDD required for the browser to work?" can be evaluated using the ordinal scale.

APPENDIX B
HIERARCHICAL FEATURE GRAPH

This section describes the Hierarchical Feature Graph that is used in our case study. It presented in a tabular format for representing the feature, the sub-feature, the weight assigned to each of the features and the measuring unit to be used. The measure types have discussed in Appendix A. The following tables show the Hierarchical Feature Graphs for various Quality attributes according to the quality standard ISO/IEC-9126 [42].

Table B.1: Hierarchical Feature Graph for "Usability"

Feature	metric	weights
USABILITY		10
Understandability		6
Interface Languages Supported	ORDN	6
Learnability		10
Tutorials	BIN	8
Blogs	BIN	4
Operability		10
Tab Stacking	BIN	2
Page Zooming	BIN	8
Mouse Gesture	BIN	1
Full Text Search Of History	BIN	7
Voice Control	BIN	6
Caret Navigation	BIN	4
Password Management	BIN	8
Spatial Navigation	BIN	3
Attractiveness		10
Search Toolbar	BIN	10
Pdf Viewer	BIN	10
Spell Check	BIN	8
Themes	BIN	7
Form Management	BIN	8
Bookmark Management	BIN	6

Table B.2: Hierarchical Feature Graph for "Efficiency"

Feature	Metric	Weights
EFFICIENCY		8
Resource Utilization		6
RAM	ORDN	6
HDD	ORDN	8
Combined Search And Address Bar	BIN	6
Notification When Add-Ons Slow Performance	BIN	4

Table B.3: Hierarchical Feature Graph for "Maintainability"

Feature	Metric	Weights
MAINTAINABILITY		8
Changeability		10
Auto Update	BIN	8
Compatibility Mode For Older Websites	BIN	4
Testability		8
Acid Test 1	BIN	6
Acid Test 2	BIN	8
Acid Test 3	ORDN	10

Table B.4: Hierarchical Feature Graph for "Portability"

Feature	Metric	Weights
PORTABILITY		6
Adaptability		2
Operating System Supported-Linux	BIN	2
Installability		8
Automatic Configuration	BIN	10
User Manual Or Online Help	BIN	8

Table B.5: Hierarchical Feature Graph for "Functionality"

Feature	Metric	Weights
FUNCTIONALITY		10
Suitability		8
Per Site Security Configuration	BIN	10
Accuracy		8
Acid Test 1	BIN	1
Acid Test 2	BIN	2
Acid Test 3	ORDN	4
Security		10
SSL Enabled	BIN	10
Access Keys	BIN	8
Domain Name Highlight For Alerting Deception	BIN	2
Plug-Ins		10
ActiveX	BIN	10
RSS	BIN	9
Functionality Compliance		8
JavaScript	BIN	2
CSS	BIN	2

Table B.6: Hierarchical Feature Graph for "Reliability"

Feature	Metric	Weights
RELIABILITY		6
Recoverability		10
Reopen Accidentally Closed Tabs	BIN	10
Tab Recovery With Message & Prompting When Website Times Out	BIN	2
Vendor Support		10
Tutorials	BIN	4
Blogs	BIN	2

APPENDIX C

DETAILED RESULTS OF THE CASE STUDY

C.1 Snapshots of the most promising candidates

This appendix presents the snapshots of the 5 candidates filtered from the filtration phase as discussed in Section 5.1.

Figure C.1: Internet Explorer 9.0

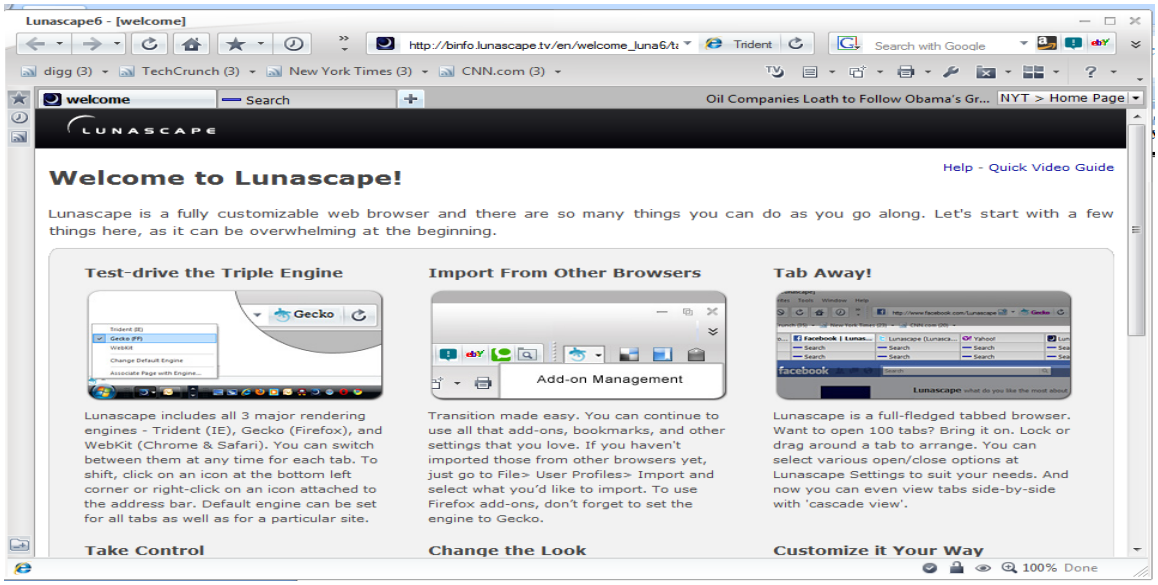

Figure C.2: Lunascape 6.0

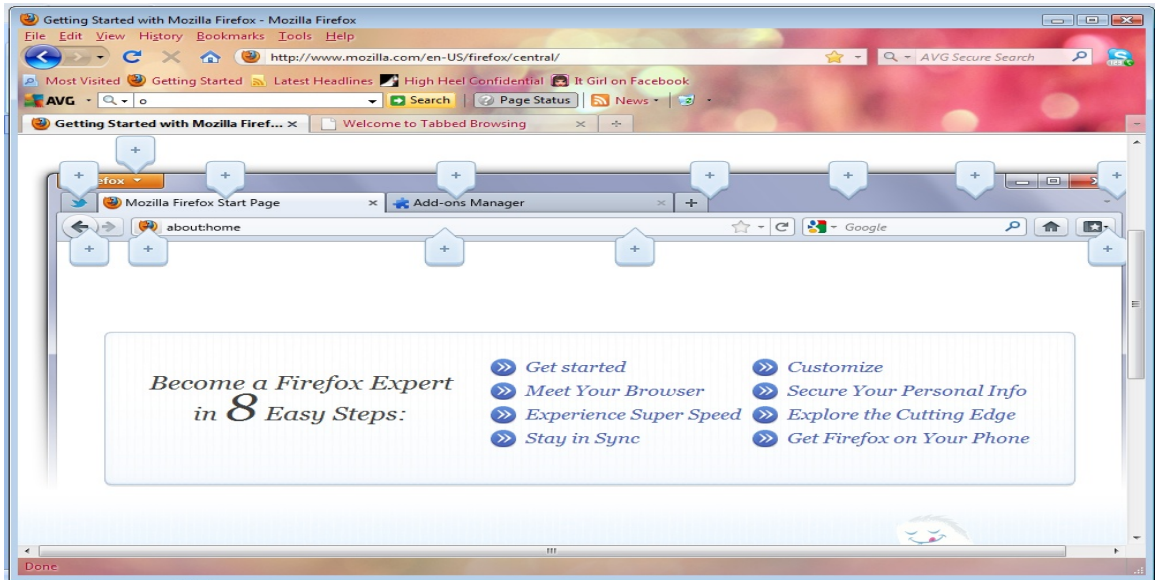

Figure C.3: Firefox 4.0

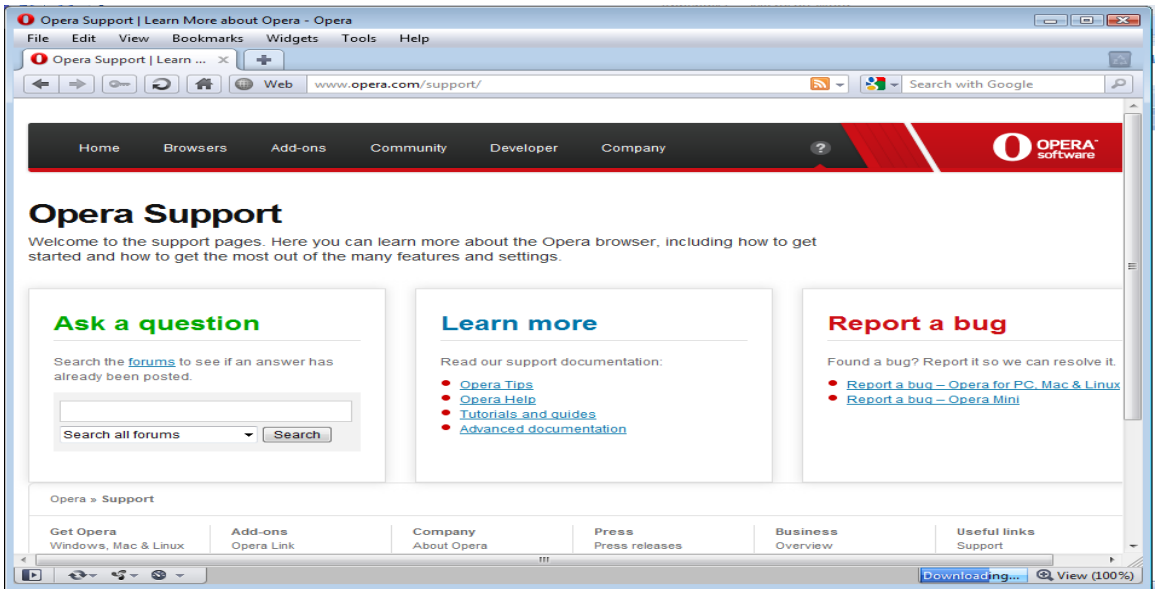

Figure C.4: Opera 11.11

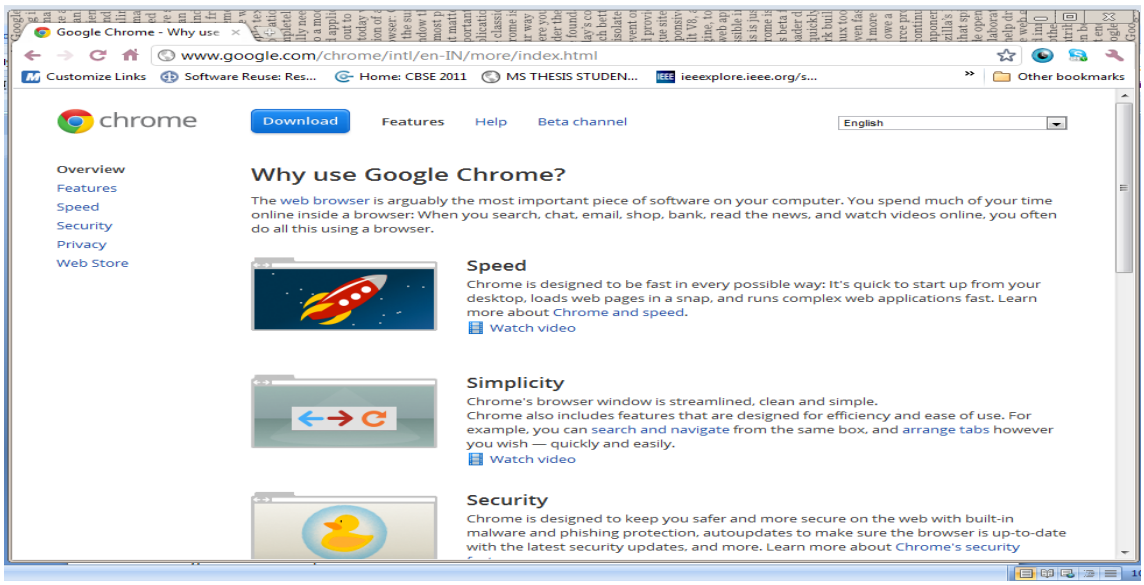

Figure C.5: Chrome 10.0

C.2 Detailed Feature Evaluation

This section presents the results of the evaluation of five most promising candidates using the Hierarchical Feature Graph listed in Appendix B. The results are shown in the terms of total satisfaction value of the COTS components.

The tables C.1 to C.6 show the detailed evaluation of the quality features using the Hierarchical feature Graph and calculating the satisfaction value using WSM for decision making.

C.3 Identified Mismatches and Mitigation Action

From the satisfaction values calculated in section C.2, the two most satisfying candidates, COTS 1 and COTS 5 are selected. This section lists the identified mismatches in these two COTS, its mismatch value. The new mismatch values are also listed after handling the mismatches using mitigation actions like plug-in.

The tables C.7 and C.8 show the mismatch values before and after mismatch handling.

Table C.1: Feature evaluation Results for “Usability”

Goal Structure	COTS 1 Matching			COTS 2 Matching			COTS 3 Matching			COTS 4 Matching			COTS 5 Matching		
Features	**Value**	**SV**	**Type**	**Value**	**SV**	**Type**	**Value**	**SV**	**Type**	**Value**	**SV**	**Type**	**Value**	**SV**	**Type**
USABILITY		0.85			0.57			0.87			0.90			0.9	
Understandability		1.00			0.67			1.00			1.00			1.00	
Interface Languages Supported	3	1.00	exact match	2	0.67	poor match	3	1.00	exact match	3	1.00	exact match	3	1.00	exact match
Learnability		1.00			0.33			1.00			1.00			1.00	
Tutorials	1	1.00	TRUE	0	0.00	FALSE	1	1.00	TRUE	1	1.00	TRUE	1	1.00	TRUE
Blogs	1	1.00	TRUE	1	1.00	TRUE	1	1.00	TRUE	1	1.00	TRUE	1	1.00	TRUE
Operability		0.82			0.82			0.73			1.00			0.73	
Tab Stacking	0	0.00	FALSE	0	0.00	FALSE	0	0.00	FALSE	1	1.00	TRUE	0	0.00	FALSE
Page Zooming	1	1.00	TRUE	1	1.00	TRUE	1	1.00	TRUE	1	1.00	TRUE	1	1.00	TRUE
Mouse Gesture	1	1.00	TRUE	1	1.00	TRUE	0	0.00	FALSE	1	1.00	TRUE	0	0.00	FALSE
Full Text Search Of History	1	1.00	TRUE	0	0.00	FALSE	0	0.00	FALSE	1	1.00	TRUE	1	0.00	TRUE
Voice Control	1	1.00	TRUE	0	0.00	FALSE	1	1.00	TRUE	1	1.00	TRUE	0		FALSE
Caret Navigation	1	1.00	TRUE	1	1.00	TRUE	1	1.00	TRUE	0		FALSE	1	1.00	TRUE
Password Management	1	1.00	TRUE	1	1.00	TRUE	1	1.00	TRUE	1	1.00	TRUE	1	1.00	TRUE
Spatial Navigation	0	0.00	FALSE	0	0.00	FALSE	0	0.00	FALSE	1	1.00	TRUE	1	1.00	TRUE
Attractiveness		0.63			0.49			0.80			0.65			1.00	
Search Toolbar	1	1.00	TRUE	1	1.00	TRUE	1	1.00	TRUE	1	1.00	TRUE	1	1.00	TRUE
Pdf Viewer	0	0.00	FALSE	0	0.00	FALSE	0	0.00	FALSE	0	0.00	FALSE	1	1.00	TRUE
Spell Check	0	0.00	FALSE	0	0.00	FALSE	1	1.00	TRUE	1	1.00	TRUE	1	1.00	TRUE
Themes	1	1.00	TRUE	0	0.00	FALSE	1	1.00	TRUE	0	0.00	FALSE	1	1.00	TRUE
Form Management	1	1.00	TRUE	1	1.00	TRUE	1	1.00	TRUE	1	1.00	TRUE	1	1.00	TRUE
Bookmark Management	1	1.00	TRUE	1	1.00	TRUE	1	1.00	TRUE	1	1.00	TRUE	1	1.00	TRUE

Table C.2: Feature evaluation Results for “efficiency”

Goal Structure	COTS 1 Matching			COTS 2 Matching			COTS 3 Matching			COTS 4 Matching			COTS 5 Matching		
Features	**Value**	**SV**	**Type**	**Value**	**SV**	**Type**	**Value**	**SV**	**Type**	**Value**	**SV**	**Type**	**Value**	**SV**	**Type**
EFFICIENCY		1.00			0.39			0.67			0.50			1.00	
Resource Utilisation		1.00			0.39			0.67			0.50			1.00	
RAM	2	1.00	exact match	2	0.67	poor match	3	0.67	poor match	3	0.67	poor match	3	1.00	exact match
HDD	2	1.00	exact match	2	0.67	poor match	3	1.00	exact match	3	1.00	exact match	3	1.00	exact match
Combined Search And Address Bar	1	1.00	TRUE	0	0.00	FALSE	0	0.00	FALSE	0	0.00	FALSE	1	1.00	TRUE
Notification When Add-Ons Slow Performance	1	1.00	TRUE	0	0.00	FALSE	1	1.00	TRUE	0	0.00	FALSE	1	1.00	TRUE

Table C.3: Feature evaluation Results for "Maintainability"

Goal Structure	COTS 1 Matching			COTS 2 Matching			COTS 3 Matching			COTS 4 Matching			COTS 5 Matching		
Features	**Value**	**SV**	**Type**	**Value**	**SV**	**Type**	**Value**	**SV**	**Type**	**Value**	**SV**	**Type**	**Value**	**SV**	**Type**
MAINTAINABILITY		0.94			0.71			0.42			0.82			0.82	
Changeability		1.00			0.67			0.67			0.67			0.67	
Auto Update	3	1.00	TRUE	2	1.00	TRUE	1	1.00	TRUE	2	1.00	TRUE	2	1.00	TRUE
Compatibility Mode For Older Websites	1	1.00	TRUE	0	0.00	FALSE	0	0.00	FALSE	0	0.00	FALSE	0	0.00	FALSE
Testability		0.86			0.75			0.86			1.00			1.00	
Acid Test 1	1	1.00	TRUE	0	0.00	FALSE	1	1.00	TRUE	1	1.00	TRUE	1	1.00	TRUE
Acid Test 2	1	1.00	TRUE	1	1.00	TRUE	1	1.00	TRUE	1	1.00	TRUE	1	1.00	TRUE
Acid Test 3	2	0.67	poor match	3	1.00	exact match	2	0.67	poor match	3	1.00	exact match	3	1.00	exact match

Table C.4: Feature evaluation Results for “Portability”

goal structure	COTS 1 Matching			COTS 2 Matching			COTS 3 Matching			COTS 4 Matching			COTS 5 Matching		
Features	**Value**	**SV**	**Type**	**Value**	**SV**	**Type**	**Value**	**SV**	**Type**	**Value**	**SV**	**Type**	**Value**	**SV**	**Type**
PORTABILITY		0.80			0.80			1.00			1.00			1.00	
Adaptability		0.00			0.00			1.00			1.00			1.00	
Operating System Supported-Linux	0	0.00	FALSE	0	0.00	FALSE	1	1.00	TRUE	1	1.00	TRUE	1	1.00	TRUE
Installability		1.00			1.00			1.00			1.00			1.00	
Automatic Configuration	1	1.00	TRUE	1	1.00	TRUE	1	1.00	TRUE	1	1.00	TRUE	1	1.00	TRUE
User Manual Or Online Help	1	1.00	TRUE	1	1.00	TRUE	1	1.00	TRUE	1	1.00	TRUE	1	1.00	TRUE

Table C.5: Feature evaluation Results for "Functionality"

Goal Structure	COTS 1 Matching			COTS 2 Matching			COTS 3 Matching			COTS 4 Matching			COTS 5 Matching		
Features	**Value**	**SV**	**Type**	**Value**	**SV**	**Type**	**Value**	**SV**	**Type**	**Value**	**SV**	**Type**	**Value**	**SV**	**Type**
FUNCTIONALITY		1.00			0.66			0.98			0.86			0.77	
Suitability		1.00			0.00			1.00			1.00			1.00	
Per Site Security Configuration	1	1.00	TRUE	0	0.00	FALSE	1	1.00	TRUE	1	1.00	TRUE	1	1.00	TRUE
Accuracy		1.00			0.67			1.00			1.00			1.00	
Acid Test 1	1	1.00	TRUE	0	0.00	TRUE	1	1.00	TRUE	1	1.00	TRUE	1	1.00	TRUE
Acid Test 2	1	1.00	TRUE	1	1.00	TRUE	1	1.00	TRUE	1	1.00	TRUE	1	1.00	TRUE
Acid Test 3	3	1.00	exact match	2	1.00	exact match	3	1.00	poor match	3	1.00	exact match	3	1.00	exact match
Security		1.00			0.55			0.90			0.90			1.00	
SSL enabled	1	1.00	TRUE	1	1.00	TRUE	1	1.00	TRUE	1	1.00	TRUE	1	1.00	TRUE
Access Keys	1	1.00	TRUE	0	0.00	FALSE	1	1.00	TRUE	1	1.00	TRUE	1	1.00	TRUE
Domain Name Highlight For Alerting Deception	1	1.00	TRUE	0	0.00	FALSE	0	0.00	FALSE	0	0.00	FALSE	1	1.00	FALSE
Plug-Ins		1.00			0.00			1.00			0.47			0.00	
ActiveX	1	1.00	TRUE	0	0.00	FALSE	1	1.00	TRUE	0	0.00	FALSE	0	0.00	FALSE
RSS	1	1.00	TRUE	0	0.00	FALSE	1	1.00	TRUE	1	1.00	TRUE	0	0.00	FALSE
Functionality Compliance		1.00			1.00			1.00			1.00			1.00	
JavaScript	1	1.00	TRUE	1	1.00	TRUE	1	1.00	TRUE	1	1.00	TRUE	1	1.00	TRUE
CSS	1	1.00	TRUE	1	1.00	TRUE	1	1.00	TRUE	1	1.00	TRUE	1	1.00	TRUE

Table C.6: Feature evaluation Results for "Reliability"

Goal Structure	COTS 1 Matching			COTS 2 Matching			COTS 3 Matching			COTS 4 Matching			COTS 5 Matching		
Features	**Value**	**SV**	**Type**	**Value**	**SV**	**Type**	**Value**	**SV**	**Type**	**Value**	**SV**	**Type**	**Value**	**SV**	**Type**
RELIABILITY		1.00			0.17			0.50			0.50			0.92	
Recoverability		1.00			0.00			0.00			0.00			0.83	
Reopen Accidentally Closed Tabs	1	1.00	TRUE	0	0.00	FALSE	0	0.00	FALSE	0	0.00	FALSE	1	1.00	TRUE
Tab Recovery With Message & Prompting When Website Times Out	1	1.00	TRUE	0	0.00	FALSE	0	0.00	FALSE	0	0.00	FALSE	0	0.00	FALSE
Vendor Support		1.00			0.33			1.00			1.00			1.00	
Tutorials	1	1.00	TRUE	0	0.00	FALSE	1	1.00	TRUE	1	1.00	TRUE	1	1.00	TRUE
Blogs	1	1.00	TRUE	1	1.00	TRUE	1	1.00	TRUE	1	1.00	TRUE	1	1.00	TRUE
COTS SATISFACTION VALUE		**0.93**			**0.42**			**0.75**			**0.77**			**0.89**	

Table C.7: COTS 1 Mismatch Handling

Features	SV	MMV	MMV_{new}	Mitigation Action
USABILITY	0.85	0.15	0.04	
Understandability	1.00	0.00	0.00	
Learnability	1.00	0.00	0.00	
Operability	0.82	0.18	0.13	
Tab Stacking	0.00	1.00	1.00	Not Resolvable
Spatial Navigation	0.00	1.00	1.00	Not Resolvable
Attractiveness	0.63	0.37	0.00	
Pdf Viewer	0.00	1.00	0.00	Resolve using Acrobat Reader
Spell Check	0.00	1.00	0.00	Resolve using Spell Check Plugin
EFFICIENCY	1.00	0.00	0.00	
Resource Utilization	1.00	0.00	0.00	
MAINTAINABILITY	0.94	0.06	0.06	
Changeability	1.00	0.00	0.00	
Testability	0.86	0.14	0.14	
Acid Test 3	0.67	0.33	0.33	Not Resolvable
PORTABILITY	0.80	0.20	0.20	
Adaptability	0.00	1.00	1.00	
Operating System Supported-Linux	0.00	1.00	1.00	Not Resolvable
Installability	1.00	0.00	0.00	
FUNCTIONALITY	1.00	0.00	0.00	
Suitability	1.00	0.00	0.00	
Accuracy	1.00	0.00	0.00	
Security	1.00	0.00	0.00	
Plug-Ins	1.00	0.00	0.00	
Functionality Compliance	1.00	0.00	0.00	
RELIABILITY	1.00	0.00	0.00	
Recoverability	1.00	0.00	0.00	
Vendor Support	1.00	0.00	0.00	
	0.93	**0.07**	**0.04**	

Table C.8: COTS 5 Mismatch Handling

Features	SV	MMV	MMV_{new}	Mitigation Action
USABILITY	0.93	0.06	0.06	
Understandability	1.00	0.00	0.00	
Learnability	1.00	0.00	0.00	
Operability	0.73	0.23	0.23	
Tab Stacking	0.00	1.00	1.00	Not Resolvable
Mouse Gesture	0.00	1.00	1.00	Not Resolvable
Voice Control		1.00	1.00	Not Resolvable
Attractiveness	1.00	0.00	0.00	
EFFICIENCY	1.00	0.00	0.00	
Resource Utilisation	1.00	0.00	0.00	
MAINTAINABILITY	0.82	0.18	0.00	
Changeability	0.67	0.00	0.00	
Compatibility Mode For Older Websites	0.00	1.00	0.00	Resolve using chrome Extension IE Tab
Testability	1.00	0.00	0.00	
PORTABILITY	1.00	0.00	0.00	
Adaptability	1.00	0.00	0.00	
Installability	1.00	0.00	0.00	
FUNCTIONALITY	0.77	0.23	0.00	
Suitability	1.00	0.00	0.00	
Accuracy	1.00	0.00	0.00	
Security	1.00	0.00	0.00	
Plug-Ins	0.00	1.00	0.00	
Activex	0.00	1.00	0.00	Resolve using chrome Extension IE Tab
Rss	0.00	1.00	0.00	RSS subscription extension can be deployed
Functionality Compliance	1.00	0.00	0.00	
Reliability	0.92	0.09	0.09	
Recoverability	0.83	0.17	0.17	
Tab Recovery With Message & Prompting When Website Times Out	0.00	1.00	1.00	Not Resolvable
Vendor Support	1.00	0.00	0.00	
	0.89	**0.10**	**0.02**	