Designing RIMCOTS Model for Risk Identification and Mitigation for COTS-based Software Development

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

Master of Engineering

In

Software Engineering

Submitted By

Amandeep Kaur

(Roll No. 800931002)

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, "***Designing RIMCOTS Model for Risk Identification and Mitigation for COTS-based Software Development***", 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.

Amandeep Kaur

(Amandeep Kaur)

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 30/6/21
Computer Science and Engineering Department
Thapar University
Patiala

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

ACKNOWLEDGEMENT

First of all, I want to thank Almighty, who has always guided me to work on the right path of life and for giving me strength to do this thesis work. Many people have shared their time and expertise to help me accomplish my goal.

I wish to express my deep gratitude to my guide, Ms. Shivani Goel, Assistant Professor, CSED, Thapar University, Patiala for her constant support and guidance. Her instant responses to my countless inquiries have been invaluable and motivational. 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 stimulating discussion 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. With their blessings, inspiration and love every work in the world seems possible.

Last but not the least, I wish to thank my brother Gurpreet singh Johar and my friends Vaneet, Ravneet Grewal, Ravneet Chawla, Arpita, Aarti, Ipneet, Aradhna, Sonam, Varun, lakhi and Harman for their immense love and encouragement without which it would not have been possible to complete this work.

Finally, my special thanks go to the authors whose work I have consulted and quoted in this thesis.

Amandeep Kaur

(800931002)

ABSTRACT

COTS-based software development (CBSD) shifts the emphasis from programming software to composing software systems from off-the-shelf components. Component based software development is often considered as a low risk development strategy, which provides a simple and rapid mechanism for developing software systems from pre-existing and proven components. In reality COTS-based software development carries significant risks throughout the software development life cycle. These risks can be due to the nature of COTS components, development process, component technologies and vendor support. The part of thesis work involves an approach to design a RIMCOTS model that incorporates risk identification and risk management mechanism for mitigation of risks in the COTS-based software development.

The purpose of this thesis work is to identify, analyze and explore critical risks in various stages of COTS-based software development that can affect long-term viability of software systems.

The Proposed RIMCOTS model includes techniques for evaluating and mitigating risks associated with deploying COTS components in Component based software systems and serves as a guide for how to manage multiple COTS software components in complex COTS-based systems. We validate this proposed RIMCOTS model by using CURE methodology.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER-1
INTRODUCTION

This chapter introduces a description of the work presented in the thesis. It gives a brief introduction of Component-based software development (CBD), COTS components, risks related to COTS components and the overview of work done.

1.1 Background

In the last several years, software development has been upraised by reusing the existing software or the commercial-off-the shelf components. So, the purpose of software reuse is to develop large systems by incorporating previously developed or existing software.

In the traditional custom based approach, within the budget constraints, all user requirements can be accommodated by a well understood development process from architectural design via detailed design to implementation, testing, verification and ending with validation. For a COTS-based approach the high level user requirements will remain the same, but a process is needed to match those requirements to the available COTS solutions and select the best fit.

Our industry has undergone a significant transition in our approach to the use of existing commercial-off-the-shelf (COTS) products in building systems [1]. By this way, software development organization is able to cut down development time, cost and effort required for the development of the same from scratch. The paradigm shift from complete custom developed systems to COTS-based systems requires new understandings about the COTS marketplace and how all engineering, business, and management activities must work together harmoniously to accommodate it [2].

1.2 Component-Based Software Development (CBD)

The primary role of component-based software development is to address the development of systems as an assembly of components, the development of components as reusable entities, and the maintenance and upgrading of systems by customising and replacing such components. Developing components is one kind of developing for reuse, while developing systems of reusable components is developing with reuse [3].

For a common understanding of component-based software development, the starting point is an agreement of what a component is and what it is not.

1.2.1 Component Specification

A *component* is defined as a piece of executable software with a published interface. Acc. to Szyperski's definition:

"*A software component is a unit of composition with contractually specified interface and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parts*" [4].

In the SEI's report [5] on technical aspects of CBD, a component is defined as:

- An opaque implementation of functionality.
- Subject to third-party composition.
- Conformant to component model.

1.2.2 Purpose of Component-based Software Development (CBD)

The purpose of CBD is to develop large systems, incorporating previously developed or existing components, thus cutting down on development time and costs. It can also be used to reduce maintenance associated with the upgrading of large systems. We assume that common parts (be it classes or functions) in a software application only need to be written once and re-used rather than being re-written every time a new application is developed.

1.2.3 COTS vs Open Source

Today's off-the-shelf market place is flooded with software components for organizations to select from when developing a system. Understanding the fundamental differences between COTS and open-source components is the first step in selecting off-the-self components.

Overview of COTS

As an increasing number and variety of commercial-off-the-shelf (COTS) become available, it is important to understand the costs, benefits, and risks entailed in using these components.

COTS Components

The Carnegie Mellon University/Software Engineering Institute defines a COTS product as one that is

- Sold, leased, or licensed to the general public

- Offered by a vendor trying to profit from it
- Supported and evolved by the vendor, who retains the intellectual property rights
- Available in multiple, identical copies
- Used without modification of the internals[6]

Vidger et al. [7][8] define COTS as pre-existing software products; sold in many copies with minimal changes; whose customers have no control over specification, schedule, and evolution; access to source code as well as internal documentation is usually unavailable; complete and correct behavioural specifications are not available.

Open Source software (OSS)

Open Source components are components in which the source code is made available for viewing and modifying and can distributed freely or for a fee. However the Open Source Initiative (OSI) provides a more in-depth definition that defines Open Source Software as something much more than components distributed with modifiable source code.

Trade Offs

The trade off between COTS and OSS comes down to cost, support, and modifiability. Due to the fact the OSS is free; there is no support center that the organization can contact when issues arise. The time and cost to resolve these issues then falls back on the organization.

However, if changes within the component are required such as: architectural, interface, or feature modifications; the source code is available and can be freely modified to meet the needs of the organization. COTS components have reduced modifiability because the source code is normally not provided however; the cost associated with COTS covers the support provided by the vendor. By having a vendor responsible for initial development and continued support, the organization has more resources available to focus on the intellectual property of their product, which in most cases, is the marketable component of the system being developed.

While choosing between COTS and OSS components is a business decision an organization will face if they decide to utilize off-the-shelf components, they both require a means of evaluation that can be incorporated into the organizations development life cycle [9].

1.2.4 COTS-based software development (CBSD)

CBSD approach is based on the idea to develop software systems by selecting appropriate commercial off-the-shelf components and then to assemble them with a well-defined software architecture [10]. COTS-based software development (CBSD) is being proposed as a low risk development strategy which provides a simple and rapid mechanism for increasing the functionality and capability of a system [11].

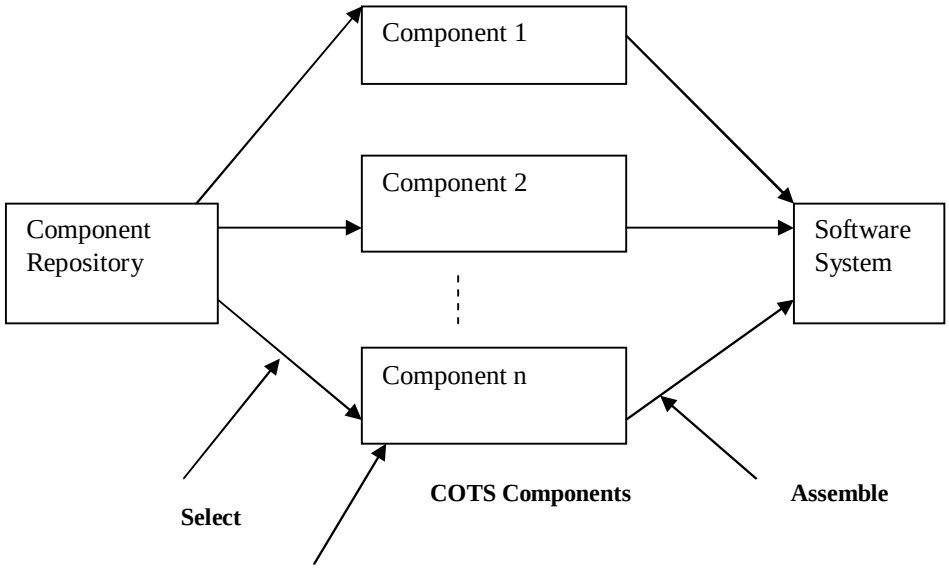


Figure 1.1: COTS-based software development [10]

COTS-based software development promises instant productivity gains, accelerated time to market and lower development cost.

1.3 Advantages and Disadvantages of COTS components

Advantages

- **Shorten Development Cycles** - The addition of a given piece of functionality will take days rather than months or even years for developing same from scratch.
- **Greater ROI** - Significant savings can be gained through purchasing software components rather than developing the same functionality in-house.
- **Enhanced Functionality** - In order to use a component containing a given piece of functionality you only need to understand its nature, not its internal details or how it can be implemented as a piece of software.

Using COTS components promises faster time-to-market and increased productivity. At the same time, COTS-based software introduces many risks [12].

Disadvantages

Though development using COTS reduces time-to market, increases productivity, it has many disadvantages as given below:

- Dependency on Vendor
- Modification to COTS components by developer is impossible
- Integration not always trivial; incompatibilities among different COTS
- Unnecessary features compromise usability, performance
- Reliability often unknown/inadequate
- No control over upgrades/maintenance

So, "COTS" is high risk because we are dependent on someone else. When a system integrator relies upon COTS software, trust is placed in unknown, black-box components [13]. So, there is need of a process that helps in evaluation and mitigation of the risks."

1.4 Steps for Successful COTS Implementation

A successful implementation of COTS intensive software system can save program's cost if you have the right solution and understand the potential risks involved. The following six Step methodology highlights the important activities that should take place during a COTS implementation:

- Analyze Software Requirements
- Evaluate and Select COTS Solution(s)
- Negotiate purchase / lease arrangement with vendor
- Implement the COTS based solution
- Maintain and upgrade the software solution
- Maintain license, subscription and royalty fees [14]

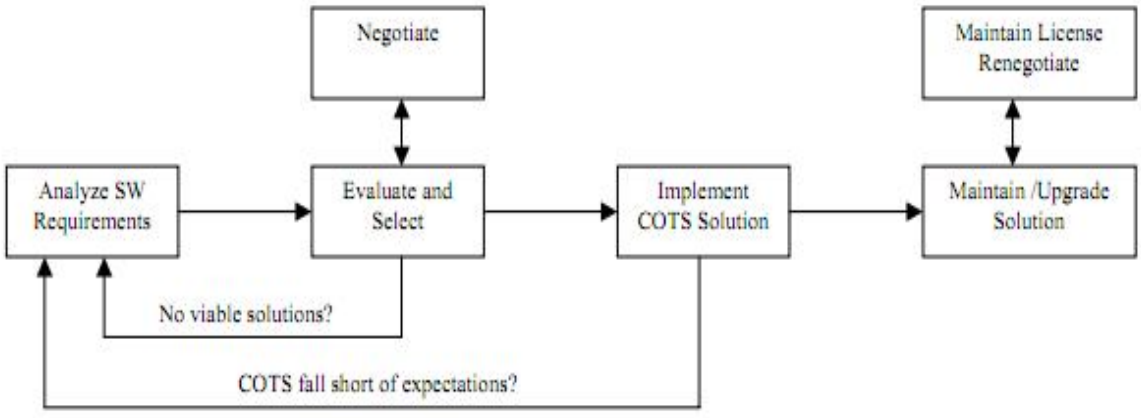


Figure 1.2: Overview of six steps [14]

Analyze Software Requirements

Software requirements analysis is a critical part of the software development process. The requirements analysis process is necessary to determine what functionality is necessary to deliver the capability required by the eventual end user(s). There are two general areas that need to be explored when determining and documenting requirements for a software system: end user requirements and technical requirements. The business analysts or requirements engineers ask the end user what they expect from the software. Once end user requirements have been gathered, the business analysts or requirements engineers restate those requirements and present back to the end user to ensure proper understanding. Technical requirements can be gathered through discussions with engineers who understand the technical nature of the problem being solved [14].

Evaluate and Select COTS solution(s)

Once a decision to pursue a COTS alternative is made, the first step is to determine the availability of COTS solutions that have the potential to provide needed functionality and evaluate these solutions. The evaluation needs to be focused on more than just product characteristics such as functionality, maturity, technology, architecture and long-term viability. There should also be a focus on vendor characteristics such as maturity, stability, cooperation and ability to provide adequate support, training and documentation. The selection process involves Progressive filtering of available COTS components and then chooses the best fit [14].

Negotiate terms with COTS vendor

It is important to negotiate the best deal possible when working with one or more vendors to craft a solution. Understand the impact of these negotiations and their timing on the eventual success or failure of your project. The end result of this negotiation should be a clear picture of the non-recurring and recurring costs associated with the system being developed. A non-recurring cost is a one-time fee associated with acquiring the COTS solution. Recurring costs are costs generally based on usage or time of use, such as annual licensing fees [14].

Implement the COTS Based Solution

Once an analysis, evaluation, and selection of COTS-based solution is complete, implementation can commence. The following activities may be required to ensure successful implementation:

- Tailoring of COTS Solution
- Modification of COTS software
- Design, Code and Test of Glue Code
- Integration and Test of COTS components with other COTS or custom components [14].

Maintenance and upgrade of the COTS based solution

Once the software is deployed, there are two on-going activities required to keep it operational and keep end users happy:

- Evaluation and inclusion of updates and upgrades from the vendor
- Bug Fixes [14].

Maintain license, subscription and royalty fee

License or maintenance fees need to be paid in order to ensure updates and upgrades as well as continuing support of the COTS components. Long-term analysis of the differences between annual subscription fees (if subscription is an option) versus paying for upgrades on an individual basis should be done. This analysis should include upgrade policies, vender stability, and frequency of releases. License and royalties should be an important part of the initial negotiation process. Renewal periods offer an opportunity to revisit the terms of the negotiation to determine whether the vendor is meeting support and upgrade commitments [14].

1.5 COTS-Trade off's

Development time can be reduced, but often at the cost of increased software component integration work. Component developers, application assemblers, and customers must all know CBD advantages and disadvantages before developing component-based applications [15].

1.6 Organisation of Thesis

The chapters in this thesis are organized as follows –

Chapter 2 – This chapter describes in detail the literature survey. It covers the details of COTS-based software development, its importance and type of COTS related risks.

Chapter 3 – This chapter describes the problem statement of the thesis. It describes the gap analysis and the need for designing risk management model.

Chapter 4 – This chapter describes the solution of problem. It covers the design of solution i.e. proposed RIMCOTS model.

Chapter 5 – This chapter gives the Experimental results i.e. Implementation of the proposed model.

Chapter 6 – This chapter describes the conclusion and future work that can be carried based on the work presented in this thesis.

CHAPTER-2
LITERATURE REVIEW

This chapter describes in detail the literature survey, what is COTS-based software development, its importance, rules for COTS-based software development and type of COTS related risks.

2.1 Motivation for component-based software development

In order to maintain a competitive edge in the technology driven, fast paced environment of today, many organizations are increasingly utilizing commercial off-the-shelf (COTS) components within their systems. The idea being that delivery dates, effort, cost, and the amount of custom development can all be reduced by integrating COTS components into a system [9]. With COTS software, the functionality you desire can be:

- Accessed immediately.
- Obtained at significantly lower prices than otherwise.
- Developed by someone who is an expert in that functionality.

2.2 Rules for COTS-based software development

The new rules apply to all COTS-based systems, whether they consist basically of one large product or product suite (called COTS-solution systems) or are made up of many COTS products from a variety of vendors (called COTS-aggregate systems). These rules must be recognized and observed for successful Component-based software development.

The new rules tell us that COTS-based software development

- is an act of composition
- is shaped by the realities of the COTS marketplace
- occurs through simultaneous definition and tradeoffs[2]

COTS-based Software Development is an Act of Composition

The first new rule for COTS-based systems is that the development of a custom system is essentially an act of creation, whereas the development of a COTS-based system is ultimately an act of composition and reconciliation. Custom development starts with the system requirements and developers create a system that meets them—

we are producers. However, COTS-based system development starts with a general set of requirements and developers then explore the offerings of the marketplace to see how closely they match the needs—we are consumers, who then integrate the products we buy into a system that meets the need. The nature, timing, and order of the activities done and the processes used differ accordingly [2].

COTS-based Software Development is shaped by the realities of the COTS Marketplace

The second rule of COTS-based software development concerns the effect of the marketplace on the nature and evolution of a COTS-based system. Eight inherent characteristics of the marketplace help determine the future of a COTS-based system endeavour:

- *COTS products and the marketplace change frequently and continuously.*
  The marketplace turns over products continuously.
- *COTS products are driven by the marketplace, not one system's need.*
  Continuous change is driven by what the companies perceive as being in their best (bottom-line) interest, not by whether or not it is right for any particular system.
- *Products have built-in assumptions about how they will be used.*
  Each product is built around some idea of how it will be used, whether those ideas are explicit or implicit; if those process ideas do not match the end-users' ideas, then clashes result.
- *Licensing and data rights are involved.*
  Attention to these may well be new to an organization, particularly government ones; for many COTS-based systems, licensing is now present on a scale that organizations have not usually seen before.
- *Programs have limited control of the frequency or content of COTS releases.*
  Users have no say over when new releases come out or what will be added or sometimes dropped—between one release and the next.
- *Programs have limited visibility into COTS product source code and behaviour.*
  Most off-the-shelf products are owned by someone else, and their internal content (i.e. software source code) is not shared with the purchaser.

- *Products are built on architectural assumptions that may vary across system components.*

  Just as products have built-in processes, they also have built-in architectural assumptions that could conflict with the evolving system architecture.
- *COTS components will have interdependencies.*

  Since components often depend on one another, a change to one may cause a ripple effect throughout the system [2].

Component-based Software Development Occurs through Simultaneous Definition and Tradeoffs

The third rule of COTS-based systems is really a consequence of the first two: there is a fundamental change required in the approach to system development for COTS-based systems, as shown in Figure 2.1. On the left is a traditional custom-development approach in which requirements (referred to as system context) are identified, then architecture is defined, and then (custom) implementation is undertaken.

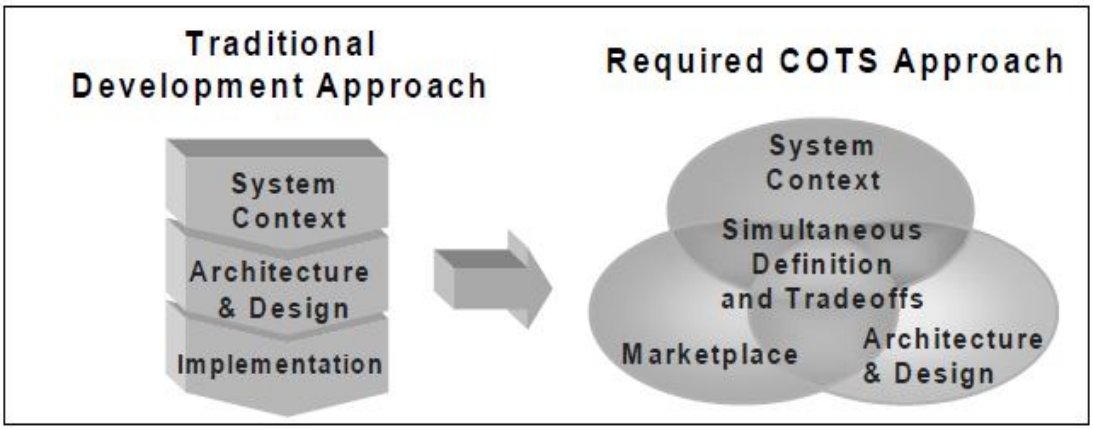


Figure 2.1: Traditional Versus COTS-Based Approach [2]

In case of COTS-based systems, it is necessary to consider system context, architecture, and the marketplace *simultaneously*. Any of these three considerations may have impacts on the other two, so none can proceed without knowledge and accommodation of the other two. Further, the activities that are performed for COTS-based systems are cyclic in nature. These tradeoffs will be repeated frequently throughout the lifetime of the system [2].

2.3 COTS-based software development life cycle Phases

COTS-based development has become more and more important in software and system development. Using COTS components promises faster time-to-market and increased productivity. At the same time, COTS-based software introduces many risks. This means that unknown quality properties of the chosen COTS components can be harmful for the final product and a COTS vendor may terminate the maintenance support of its COTS components. The use of COTS components introduces new system issues, which require revised software development processes [12].

COTS life cycle phases:

The COTS-based software development mainly involves COTS identification, COTS selection and COTS integration activities.

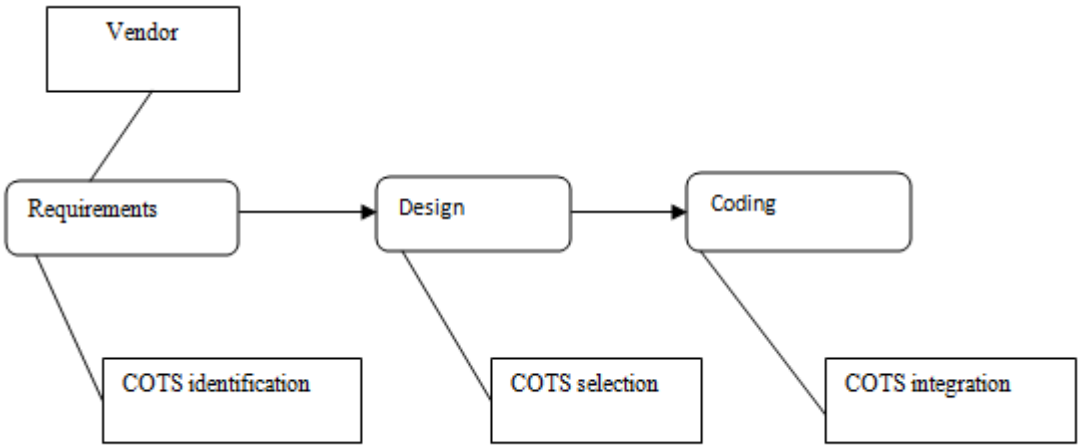


Figure 2.2: COTS-based development process [12].

Typically, a COTS-based development process consists of five phases, comprising:

- COTS component selection
- COTS component integration
- COTS development process
- COTS component implementation and testing
- Maintenance of COTS components in the system[12]

There is consensus that the use of a COTS component implies changes in the software process [12].

2.3.1 COTS component Selection phase

COTS component selection is regarded as the most crucial phase. Several formal selection processes and decision making methods have been proposed to support the selection and evaluation of COTS components. The candidates components are gone under the evaluation phase which is most crucial task of the selection cycle and resulting in various mismatches. The procurement-oriented requirements engineering (PORE) technique is an iterative technique that supports the evaluation and selection of COTS components.

The PORE process model identifies four goals that need to be performed in a thorough COTS selection process (refer to Figure 2.3). The four goals are acquiring information from the stakeholders, analyzing the information to determine if it is complete and correct, making the decision about product requirement compliance if the acquired information is sufficient, and selecting one or more candidate COTS components [16].

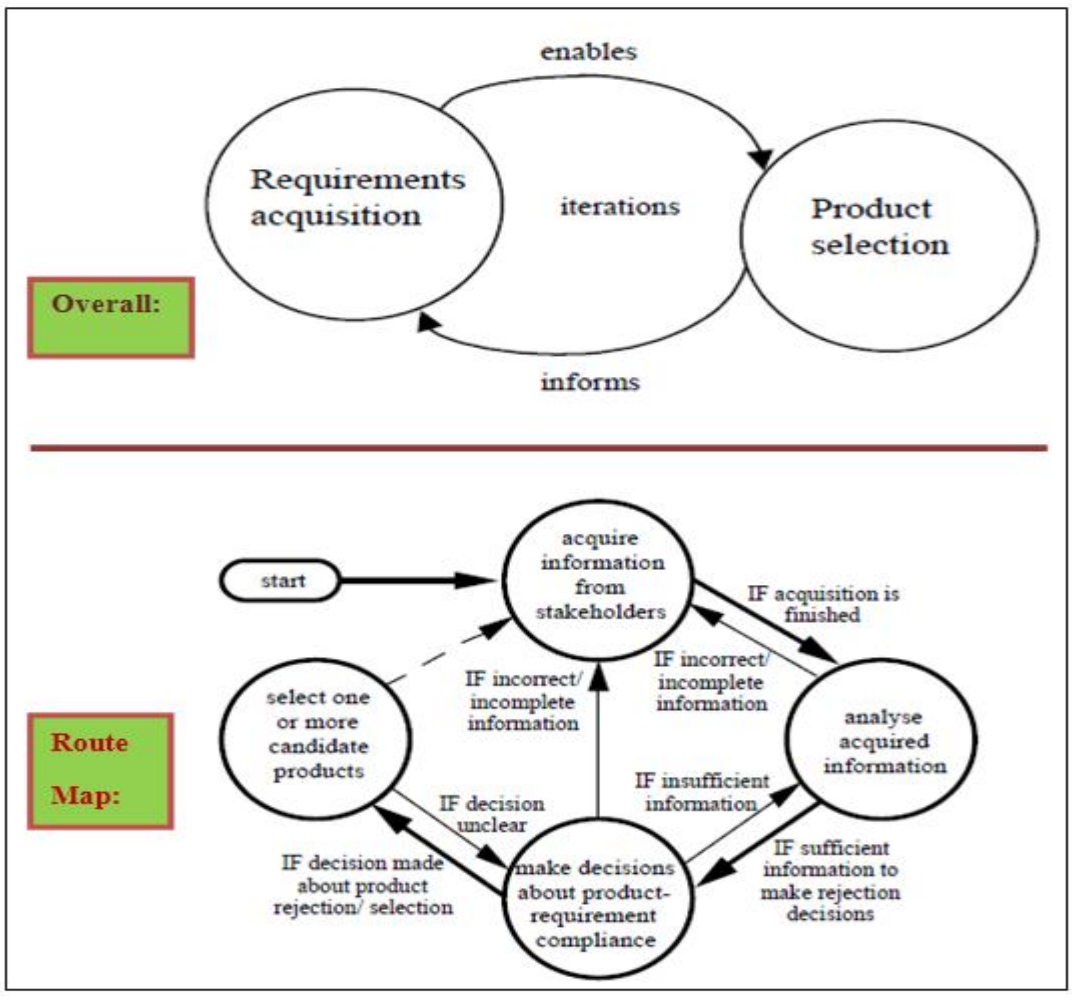


Figure 2.3: PORE process during decision making [16]

2.3.2 COTS component integration:

Integration gradually puts the pieces together - COTS, glueware and traditionally developed software - to assemble the final application [17].

The integration step varies a great deal from project to project, depending on which and how many COTS products are being used [18]. The challenges of component-based development are how to integrate various components in software systems. Identify glueware and Integration Requirements. Glueware and interfaces as dictated by the system architecture, operating system and hardware are identified [18]. Without the glue code, the components would be un-integratable and COTS-based systems can be difficult to comprehend, less evolvable than intended, and less reliable than the original components [19]. Most COTS products are not designed to interoperate with each other and most COTS vendors do not support glue code (sometimes called glueware and binding code). So, most software development teams that use COTS components have difficulties in estimating effort and schedule for COTS glue code development and integration of them into application systems.

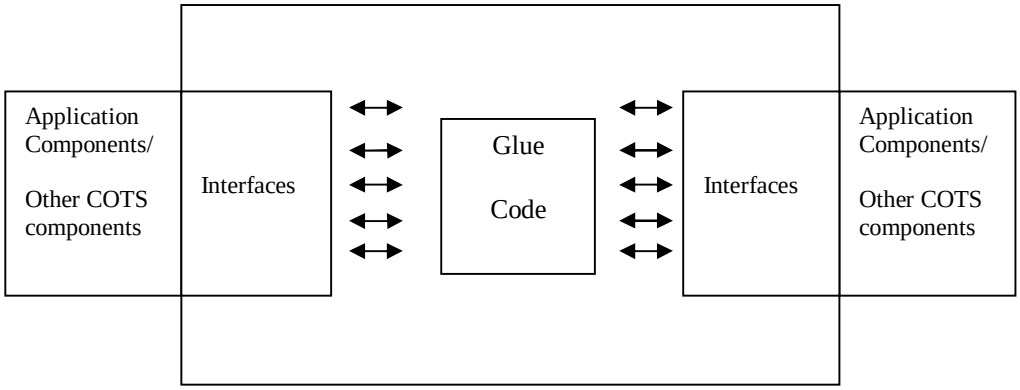


Figure 2.4: Glue Code Configuration [20]

2.3.3 COTS component development process

COTS-based development process is not a totally new process. It needs a customization (change or add new activities) on the traditional development processes. The process design could include two elements [12]:

- First, decide the main development processes based on the project profile. The main development process could be decided based on risk considerations.

- If the main development process is waterfall or evolutionary, the process could be customized based on the actual main development process and project members' familiarity with relevant COTS components as following Figure 2.5.

For each customization, some possible risks must be identified and managed.

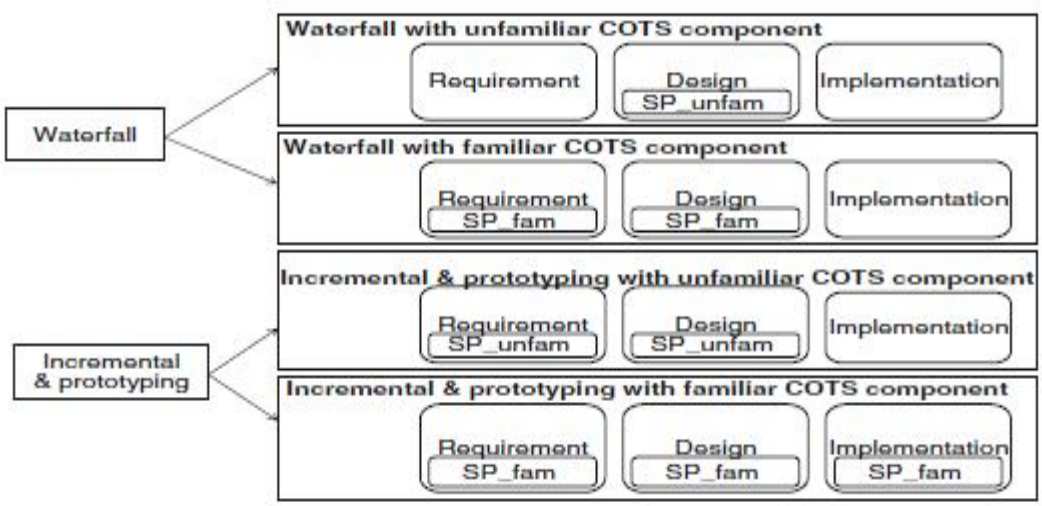


Figure 2.5: Possible COTS-based development process customizations [12]

SP_fam: Familiarity-based selection process. SP_unfam: Internet search, trial-based selection process

Waterfall with Unfamiliar COTS Components

it is difficult to use an unfamiliar COTS component successfully if the general process is waterfall. It will be difficult to find COTS components to fully satisfy the requirements after the requirements and design have been decided upon. Therefore, our recommendation is that the build vs. buy decision and the COTS components selection should be performed before the implementation phase. The COTS components selection process could be unfamiliar, and the ease of integration should be given much consideration in the build vs. buy decision. If the candidate COTS components do not appear to be easy to integrate, it is better to develop them in-house. Another recommendation is to inform the customer; if possible, about using COTS components after the build vs. buy decision. It may give the developers leeway to (re)negotiate the requirements if later limitations of COTS components are found [12].

Waterfall with Familiar COTS Components

If the project members are quite familiar with relevant COTS components, the build vs. buy decision and COTS component selection could be performed in the requirement phase, because the integrators know how to integrate them, and which functionality could be provided by the COTS components [12].

Incremental & Prototyping with Unfamiliar COTS Components

Here, the risk of using unfamiliar COTS components is relatively low, compared to waterfall projects. The integrator can select the COTS components based on the completeness of required functionalities and build a prototype in a short time. The customers requirements can be agreed upon by evaluating the prototype. For this customization, the recommendation is to integrate the unfamiliar COTS components first, if there are different phases to implement the system incrementally [12].

Incremental & Prototyping with Familiar COTS Components

If the project members have enough experience with relevant COTS components, they can easily build a prototype of the system with the COTS component. Here, the build vs. buy decision can be performed in the requirements and/or design phase. If there are different phases to implement the system incrementally, they do not need to integrate all COTS components upfront. In this case, COTS components could be considered as in-house built components. The process could be the same as a non-COTS one [12]

.

2.3.4 COTS component implementation and testing

This phase includes Target COTS-based system installation and acceptance test. Unlike the traditional life cycle, no formal acceptance testing or operational readiness reviews are mentioned by the teams. The development team installs the software on the target system [18].

2.3.5 COTS component maintenance

This maintenance phase includes extended development. The COTS component is upgraded to a newer version to meet current needs. The different customer-vendor evolution cycles may result in an uncertainty about how often COTS components in a system may have to be replaced and the extent of the impact of such a change on the rest of the system. This makes it difficult to plan and predict costs over the life cycle of a system [12].

2.4 Problems faced in Development with COTS components

COTS-based development (CBD) poses significant problems to organisations intending to adopt the technology. This is particularly critical for small organisations for which the failure of a project can result in the organisation going out of business. This section includes the various risks involved in CBD. These risks are mainly due to the black-box nature of COTS.

Using COTS components, systems are often hard to build, to support, and to maintain. The problems encountered may be related to the processes which an organization uses to build systems, in the technologies used to construct the system, and in the way systems containing COTS components evolve.

The problems stem from four main factors, which are a consequence of the component-ware paradigm:

- The black-box nature of COTS software (i.e. unknown design assumptions).
- Lack of information about source of components.
- The quality of COTS software.
- The lack of component interoperability standards
- The disparity in the customer-vendor evolution cycles[21]

The risks in using COTS are categorised on the basis of five key application development activities: *component evaluation, component integration, development process, system implementation* and *system evolution.*

Categorising CBD risks into different categories makes it possible to better appreciate their cross-cutting and overlapping nature. The risks are organised into 5 main categories: *selection, integration, development process, implementation and evolution risks.*

- **Selection risks** are associated with problems of evaluating and selecting off-the-shelf software for use in the system development.
- **Integration risks** related to the problems of composing systems from COTS software.
- **Process risks** are associated with the problems of using an inappropriate development process.
- **Implementation risks** may stem from the perceived quality of COTS components and various security issues.

- **Evolution risks** are related to the extended development and management of component-based applications.

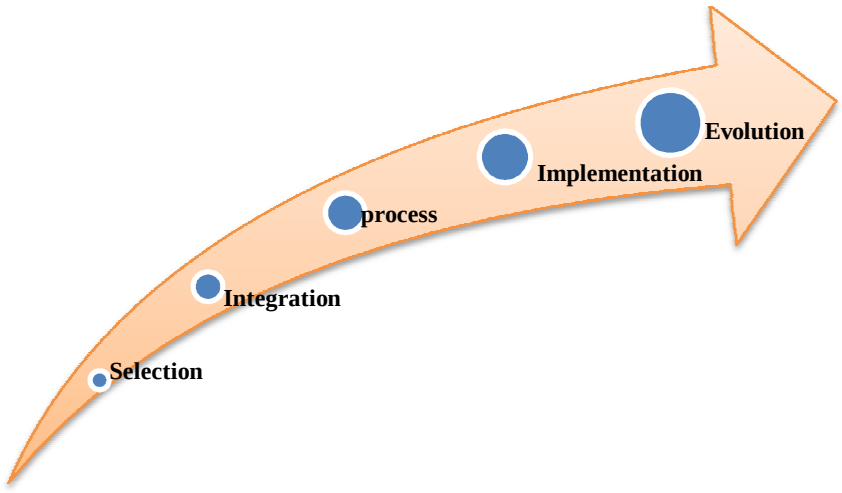


Figure 2.6: Categories of Risks

Some studies focus only on COTS components. Rose [22] classifies possible risks in COTS component-based development based on the different phases. Examples of possible risks are:

- Lack of skilled personnel
- Architecture mismatch
- Product obsolescence
- COTS components have asynchronous update cycles with the COTS-based system.

2.5 Security-Related COTS Products

Any type of COTS component might have an impact on the overall system security, depending on how it is used in the system. Therefore every type of COTS product could be security-related. Examples of COTS products intended to improve security include cryptographic software (and hardware), network firewalls, and antivirus tools [23].

2.5.1 Security Risks

Peter G. Neumann informally defines a risk as "*a potential problem, with causes and effects*" [24].

The security risks are defined as:

· The system, through misuse or by accident, experiences the introduction of security vulnerability [23].

(A security vulnerability is a flaw that could later be exploited to cause a loss of confidentiality, Integrity or availability) [23] .

2.5.2 Reasons for Security Risks

2.5.2.1 Component design

Some security risks originate from the design of the COTS components and are consequently beyond the control of the customers. Reasons for this may be:

- *Inadvertently flawed component design.* The components may have various types of bugs, some of which may affect security.
- *Intentionally flawed component design.* The components may contain intentional security flaws, such as backdoors, viruses, or Trojan horses.
- *Excessive component functionality.* A component may have many more features than the customer needs.
- *Open or widely spread component design.* Although most academic security researchers promote openness and public scrutiny for better security, a risk does exist if details of the component design are widely known outside the customer organization.
- *Insufficient or incorrect documentation.* The developer might not provide the customer with the documentation needed to correctly and securely integrate the component into the system [23].

2.5.2.2 Component procurement

There are also security risks associated with purchasing and delivering components:

- *Insufficient component validation.* A component purchase might not fully conform with the customer's *real* security requirements.
- *Delivery through insecure channel.* For example, in downloading a software component via the Internet, the product might be manipulated along the way by a third party (an intermediary attack) or the customer might be tricked into downloading a manipulated product from a site controlled by the attacker, instead of the real product from the vendor's site [23].

2.5.2.3 Component integration

Integrating components, which is a step in the design of the composed system, has the following risks:

- *Mismatch between product security levels.* A common problem when integrating different products is that the security level must be set to the lowest common denominator to make the products work together. For example, in the Microsoft Windows NT File System, user access to local system files and folders can be restricted to read-only permission to prevent accidental or intentional modification. However, for Microsoft Office 97 to work properly, the user must be given write permission for a number of system folders and files [25].
- *Insufficient understanding of integration requirements.* The integrators might not fully understand all of the preconditions for secure integration of the products, for example, that some Components must be physically protected.

2.5.2.4 Internet connection of system

When the system is connected to the Internet, a number of additional risks must be considered:

- *Increased external exposure.* By connecting the system to the Internet, exposure expands to a large number of potential external.
- *Executable content*. Many World Wide Web pages have executable content (for example, Java applets) that automatically downloads and executes on a user's computer when viewing the page in a Web browser. Credulous users might well run programs that attack their system.
- *Outward channel for stolen information.* The Internet connection constitutes a channel that can covertly and conveniently export information stolen from the system, for example, by internal attackers or by programs planted by external attackers [23].

2.5.2.5 System use

Some risks are related to how the users operate the system:

- *Unintended use.* The system can be used in an unintended way, for example, to store and process data that are more sensitive than the system was designed to handle or to attack other systems.
- *Insufficient understanding of function.* Users might not be able to judge their adherence to the security policy if they do not fully understand a function. For example, they might not know whether or not a particular program transmits passwords in the clear over the network [23].

2.5.2.6 System maintenance

Finally, there are risks involved in the maintenance of the system:

- *Insecure updating.* In the same way as the initial software delivery is risky if performed via an insecure channel.
- *Unexpected side effects.* Any changes made to components in the system can have unexpected side effects and might introduce new security vulnerabilities.
- *Maintenance backdoors.* The history of computer insecurity contains many cases in which developers left open backdoors for convenient testing and maintenance of their products. However, such backdoors can be misused by anyone who knows or finds out about them [23].

2.6 Evaluation Methods

A number of various methods have been established to aid in the development of COTS-based systems. These methods are aimed at specific activities such as integration of COTS-based systems, product evaluation, and risk evaluation. Three existing COTS evaluation methods:

- Evolutionary Process for Integrating COTS (EPIC)
- PECA (Plan, Establish, Collect, Analyze) Process
- COTS Usage Risk Evaluation (CURE) Method [9]

2.6.1 Evolutionary Process for Integrating COTS (EPIC)

EPIC is a process to help organizations build, field, and support solutions based on COTS and other pre-existing components [26]. EPIC is an iterative process in which, similar to the Rational Unified Process, is composed of four phases:

- Inception
- Elaboration
- Construction
- Transition

A quick overview of these four phases is provided as following:

Inception Phase

The purpose of the inception phase is to define the system context. The system requirements, constraints, and expectations are elicited from the system stakeholders and used to perform a quick evaluation of available COTS components that meet these requirements. In addition, the cost and schedule for the system are defined. The results of the inception phase should be at least one feasible, solution to evaluate. A summary of the possible solutions is created in which solutions that merit further examination are identified [26].

Elaboration Phase

Within the elaboration phase, the goal is to expand the level of knowledge associated with the feasible solutions identified in the inception phase. Through further definition of the system context, experimentation, and prototyping the system architecture begins to take shape as well as the integration of COTS components. By the end of the elaboration phase a single system solution should be selected as the baseline for the construction phase [26].

Construction Phase

The construction phase aims at creating a version of the system solution that is of production quality. This includes the generation of custom components, interfaces between components and the definition of any tests that need to be performed [26].

Transition Phase

The transition phase moves the system to the user community. The user community is made aware that the system has been released and customer support begins. Bug fixing, patches, and feature updates are to be expected as users begin to utilize the system. An important aspect of the transition phase is continuous monitoring of the market place. As COTS components change over time so too must the system. The organization must be aware of these changes and be able to support these changes within the deployed systems [26].

2.6.2 PECA

The PECA (Plan, Establish, Collect, Analyze) process provides a process that helps organizations make carefully reasoned and sound product decisions [27].

The PECA process also is composed of four activities:

- Planning the evaluation
- Establishing the criteria
- Collecting the data
- Analyzing the data

Planning the Evaluation

In planning for the evaluation process there are a handful of key decisions that must be made. An evaluation team must first be selected. The PECA process recommends a team that represents a diverse makeup of the organization. This includes a balance of power, as well as representation from different departments. The hope in creating a diverse evaluation team is that the opportunity for bias is reduced by the balance created by the diversity. Once an evaluation team has been selected a charter must be formed; establishing the goals, scope, and factors that are to be used during the evaluation. The other key decision that must be made in planning the evaluation is the selection of the approach to leverage during the evaluation. Proper criteria must be established according to the criticality of the component being selected [27]:

Establishing the Criteria

Establishing the criteria effectively maps the systems requirements to a prioritized set of evaluation criteria. This mapping occurs by performing the following steps [27]:

- Define the evaluation requirements.
- Define the evaluation criteria.
- Prioritize the criteria.

Evaluation requirements will stem from both system requirements and system context. Non-functional requires such as security and performance requirements are system requirements that will help guide the evaluation while context requirements like architecture, operational environment, and programming constraints are system context requirements that will feed into the evaluation.

Defining the evaluation criteria creates a means by which to verify that a COTS component can meet an evaluation requirement. “A good criterion needs both the capability statement and a measurement method” [27].In other words, a metric is established that can be used to verify that a COTS component performs to the specifications described by the evaluation requirements and therefore, will meet the need of the system.

Collecting the Data

The motivation behind collecting evaluation data on a particular COTS component is to verify it meets expectations, advertised capabilities, and system requirements. A variety of methods exist for performing this data collection from vender. Collect the metrics that verify the COTS component meets the criteria and requirements needed to provide functionality to the system [27].

Analyzing the Data

After collecting data from various COTS components this data must be analyzed so that a selection can be made. PECA provides various techniques for both the consolidation and analysis of the collected data. Upon consolidating and analyzing the data recommendations can be made. What the analysis provides is the trade-off within the recommendations. This trade space may span both the system requirements and criteria, but will provide adequate information to make a selection [27].

2.6.3 CURE

The COTS Usage Risk Evaluation (CURE) method is a risk evaluation method aimed at identifying risks related specifically related to COTS component. Following the basic method of the Software Risk Evaluation (SRE) technique, CURE gathers data utilizing a questionnaire followed by conducting an interview based on a discussion document. The questionnaire provides the evaluation team with insight into the system being developed that allows the team to focus the discussion topics to enable more effective risk identification [28].

The collected data is then analyzed to identify risk factors and risk conditions that complete condition-consequence statement templates. These condition-consequence statements identify a set of risks that are specific to COTS utilization and could have a significant impact within the system. The result of this analysis are areas in which the development team must be caution of potential risks that have yet to be identified as

well as risk areas where the development team has already been made aware of the potential risks and possibly has mitigated these risks [28].

2.7 Risk Management

Risk management is increasingly seen as one of the main jobs of project managers. It involves anticipating risks that might affect the project schedule or the quality of the software being developed and taking action to avoid these risks [29] [30].

Risk management for COTS and other reusable software components occurs in the context of overall program risk management. The process of risk management is not significantly different, but the risks that arise from the reuse of software components often are unique. This is due to the lesser degree of control that the program has over COTS and other reusable software components controlled by outside influences. COTS components are usually provided as binary code with natural language specifications of integration and deployment procedures [31].

For reusable software components, the kinds of risks and the techniques used to mitigate them both differ. In addition, the timing of risks associated with COTS and other reusable software components may not be as predictable as those that occur during the development of custom software code [32].

2.8 Awareness

An overarching risk assessment and subsequent risk management can only be successfully achieved when all parties involved in all stages of the life cycle of component-based software development i.e. in selection, integrating , development, deploying and finally maintenance of these systems, are aware of the imminent problems and willing to take responsibility in the risk management process [32].

CHAPTER-3
PROBLEM STATEMENT

In previous chapter we explored various phases for COTS-based software development. We have also seen the problems faced in development with COTS components. Using COTS components may help to shorten time-to-market and save development effort but using such external components introduces many risks. Although developing systems with commercial-off-the-shelf components is gaining more attention from both research and industrial communities, most literature on the COTS risks doesn't clearly identify the various types of risks that occur across component-based software development phases. In particular, the literature often lacks a definition of critical risks that needs to be mitigated first.

Before project managers decide to acquire an external component instead of building it in-house, they must evaluate and compare possible risks and benefits. Although several risks and risk management activities in COTS-based development have been identified from case studies, few empirical studies have subsequently verified their conclusions. As a result, software project managers have few effective and well-proven guidelines on using specific risk management activities to mitigate corresponding risks.

So, there is a need to design risk management model to identify the risks for each phase of COTS-based development. Next, there is a need to enhance the existing risk models to overcome the limitations of existing risk management model and incorporating evaluation methodology for identification and mitigation of critical risks.

CHAPTER-4
PROPOSED RIMCOTS MODEL

This section covers the process of finding a solution to the problem stated.

4.1 Designing of Risk identification and mitigation model (RIMCOTS)

RIMCOTS is a comprehensive risk identification and mitigation model for COTS that is based on sound theoretical principles, yet it has been designed to have sufficiently low overhead and complexity so that it can be used in real, time-constrained Applications. Designing of RIMCOTS performs following activities.

4.1.1 Risk Identification in COTS-based Development

The proposed RIMCOTS model will identify various COTS component risks that can occur during various application Development phases. We divide the risks into five main categories and then identify various risks for each individual risk category.

4.1.1.1 Risks during COTS Selection

COTS selection is defined as the process of searching candidates and evaluating them with respect to the system requirements. *Selection risks* are associated with problems of evaluating off-the-shelf components for use in system development. So, COTS selection activity involves COTS identification and COTS Evaluation. As COTS are identified, the evaluation and selection processes begin. Various Risks during this selection phase can be:

- Difficulty in predicting component behaviour due to black-box nature of COTS.
- Incomplete system Requirements.
- Poor mapping of user requirements to component-based architecture
- Lack of cooperation from users.
- Inappropriate COTS selection and procurement method used
- Required COTS is found costly as compared to in-house development cost
- Very less COTS components are available for selection

4.1.1.2 Risks during COTS Integration

Integration risks are associated with problems of composing systems from Pre-existing COTS components. Component integration process may suffer from inflexibility and poor component evaluation schemes. The risks can occur while integration of COTS components due to the following factors:

- Lack of interoperability standards among different COTS components.
- Occurrence of incompatible format among COTS components.
- Effort for integration may increase from what was estimated.
- Cost for integration may increase from what was estimated.
- Too much glue code criticality during COTS integration.
- Additional features than required may exist in selected COTS components which may cause difficulties in integration.

4.1.1.3 Risks during System Development

Development risks are associated with the problems of using an inappropriate development process. For example, both the waterfall model and evolutionary development are unsuitable for CBD because risk analysis phase is absent in these models. There can be following other risks during development phase:

- Risk analysis phase is not present in selected COTS based development process.
- Resources are insufficient for COTS -based development.
- Development time and cost exceeds the estimated time and cost.

4.1.1.4 Risks during System implementation

Implementation risks may stem from the perceived quality of COTS components and various security issues. So, risks during this stage can be further categorized into two categories: quality risks and security Risks.

Quality Risks

- Unknown design assumptions of COTS components reduce quality of testing.
- The perception of quality may vary across COTS software vendors and application domains.
- Effects system safety or security.
- Effects on system performance.

- Reduces system reliability.
- Users may feel uncomfortable while using the system.
- Users cannot retrieve relevant and needed information from the system.

Security Risks

Using COTS components poses serious threats to system security. There can be following Security Risks:

• System can be used in unintended way. So, sometimes use of COTS software introduces a vulnerability risk.

• Increased external exposure (By connecting the system to the Internet, exposure expands to a large number of potential external).

• Increasing vulnerability and attack options by integrating components with one another.

4.1.1.5 Risks during System Evolution

Evolution risks are related to the extended development and management of component-based applications. There can be following evolution risks :

- Occurrence of problems in system updation.
- Occurrence of problems in licensing arrangements of COTS.
- Difficulties in COTS replacements may occur.
- Incompatibility of new version of COTS with user requirements.
- Vendor may go out of business.
- Loss of support and continuity when COTS vendor cease to exist.
- The system is not properly modified to meet new user needs.
- Technical bugs are not overcome in allocated time.
- No enough technical support from system vendors.
- Probability of insufficient support from top management during post-implementation (i.e. maintenance).

4.1.2 Risk Analysis: In this proposed RIMCOTS model, the *COTS Usage Risk Evaluation* (CURE) methodology is adopted to perform Risk analysis. CURE is the approach to reduce the number of program failures attributable to COTS component.

This Evaluation consists of four activities:

- Preliminary data gathering
- Conduct survey by using questionnaire
- Analysis of data
- Presentation of results

Firstly the various risks during the COTS-based software development were identified phase-wise. Then the questionnaire was designed on the basis of theoretical risk ontology as shown in appendix A.1. For each risk item, 24 respondents were asked to provide their perceptions on

- ✓ the *probability of occurrence* of each risk identified (measured on a three-point Likert scale, ranging from high (3) to low (1);
- ✓ the *impact* of each risk event (measured on a three-point Likert scale, ranging from high (3) to low (1)

The risk that has a high probability of occurrence may not have a high impact and vice-versa. Therefore, when evaluating the importance of risks events, it was considered necessary to take into account these two risk aspects as in appendix A.2. Consequently, the following formula was developed:

Risk score of each risk= Probability * Impact

The RIMCOTS model includes a specific step for analyzing stakeholder interests and how they link to risks. These links are visualized in Figure 4.1. When risks are defined, their impact on the System is described. Each risk can be described by its Probability of occurrence and its potential impact on the agreed project goals.

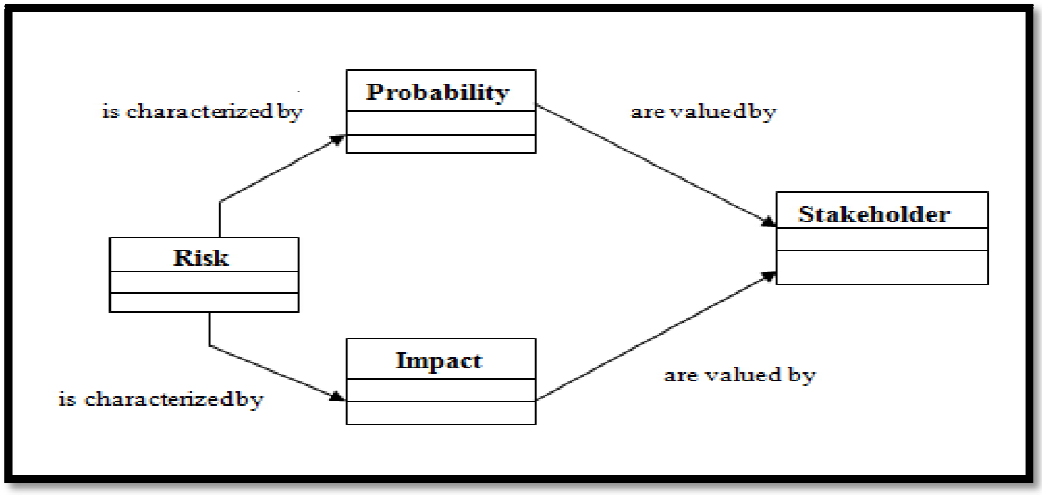


Figure 4.1: Definition of Risk within RIMCOTS model

In order to prioritize risks during Risk analysis, the most important risks have to be selected based on their probability and impact. To select the most important risks during each application development phase based on the combination of probability and impact, a specific *Ranking technique* is used. According to this ranking technique, risks that have high risk scores are considered as *critical risks* and for such critical risks, mitigation strategies will be developed. RIMCOTS model uses Analysis Graphs to describe and discuss risks.

CHAPTER-5
RESULTS AND FINDINGS

This section reports the findings of our observations from 24 respondents and achieves the objective of identification of the critical risks as stated in the problem statement. Then mitigation plans will be developed for critical risks. Moreover, it validates the solution.

5.1 Detailed evaluation results:

The results of the evaluations that have been conducted were summarized below:

Evaluation results for Selection Phase:

Table 5.1: Risk Scoring Method for Selection Phase

Risk Id	Risks Identified (in Selection Phase)	Risk Score
RS1	Difficulty in predicting component behaviour due to black-box nature of COTS.	115
RS2	Incomplete system Requirements.	197
RS3	Poor mapping of user requirements to component-based architecture.	170
RS4	Lack of cooperation from users.	40
RS5	inappropriate COTS selection and procurement method used.	180
RS6	Required COTS is found costly as compared to in-house development cost.	80
RS7	Very less COTS components are available for selection.	50

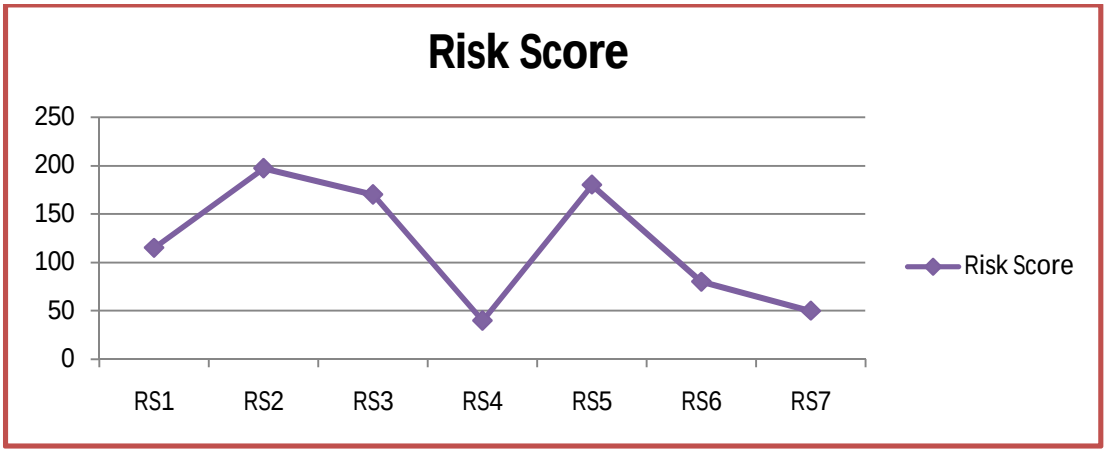


Figure 5.1: Risk analysis graph for Selection Phase

Average of Risk scores in selection Phase = 118.8571

Risks having value above this average were considered as *Critical risks*

So, RS2, RS3 and RS5 are critical risks.

Evaluation results for Integration Phase:

Table 5.2: Risk Scoring Method for Integration Phase

Risk Id	Risks Identified (in Integration Phase)	Risk Score
RC1	Lack of interoperability standards among different COTS components.	177
RC2	Occurrence of incompatible format among COTS components	137
RC3	Effort for integration may increase from what was estimated	118
RC4	Cost for integration may increase from what was estimated	75
RC5	Too much glue code criticality during COTS integration	178
RC6	Additional features than required may exist in selected COTS components which may cause difficulties in integration.	51

Figure 5.2: Risk analysis graph for Integration Phase

Average of Risk scores in Integration Phase =122.6667

So, RC1, RC2 and RC5 are critical risks.

Evaluation results for Development Phase:

Table 5.3: Risk Scoring Method for Development Phase

Risk Id	Risks Identified (in Development Phase)	Risk Score
RD1	What is the probability that selected development process may be unsuitable for COTS-based development?	106
RD2	What is the probability that risk analysis phase is not present in selected COTS based development process?	65
RD3	What is the probability that resources are insufficient for COTS based development?	142
RD4	What is the probability that development time and cost exceeds the estimated time and cost?	103

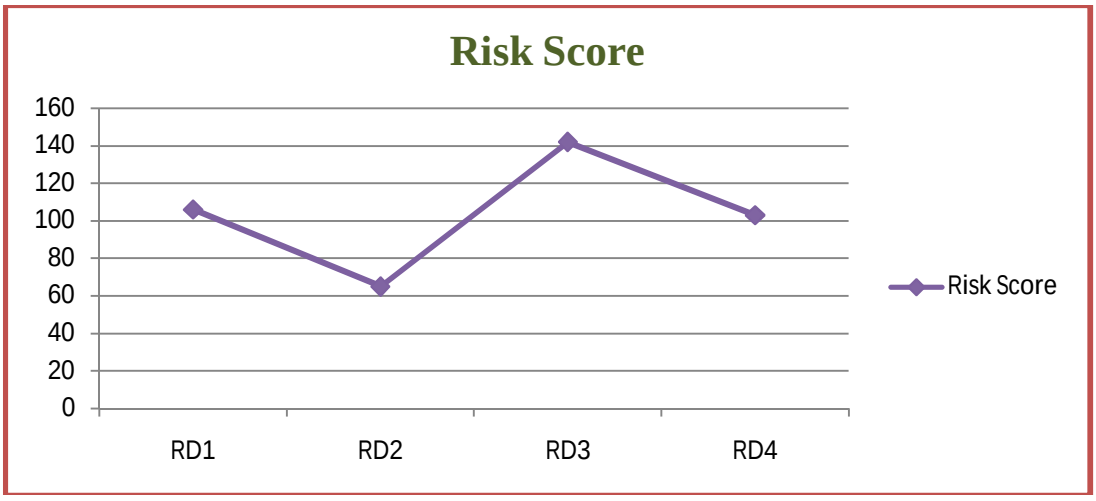


Figure 5.3: Risk analysis graph for Development Phase

Average of Risk Scores in Development phase=104
So, RD1 and RD3 are considered as critical risks.

Evaluation results for Implementation Phase:

Table 5.4: Risk Scoring Method for Implementation Phase

Risk ID	Risks Identified (in Implementation Phase)	Risk Score
RI1	Unclear design assumptions can reduce quality of testing.	186
RI2	Risks that may affects system safety or security.	167
RI3	Risks that may affects system performance.	118
RI4	Risks that may affects system reliability.	120
RI5	System can be used in unintended way.	85
RI6	Occurrence of external exposure.	93
RI7	Occurrence of vulnerability attacks.	148
RI8	Users may feel uncomfortable while using the system.	96
RI9	Users cannot retrieve relevant and needed information from the system.	126

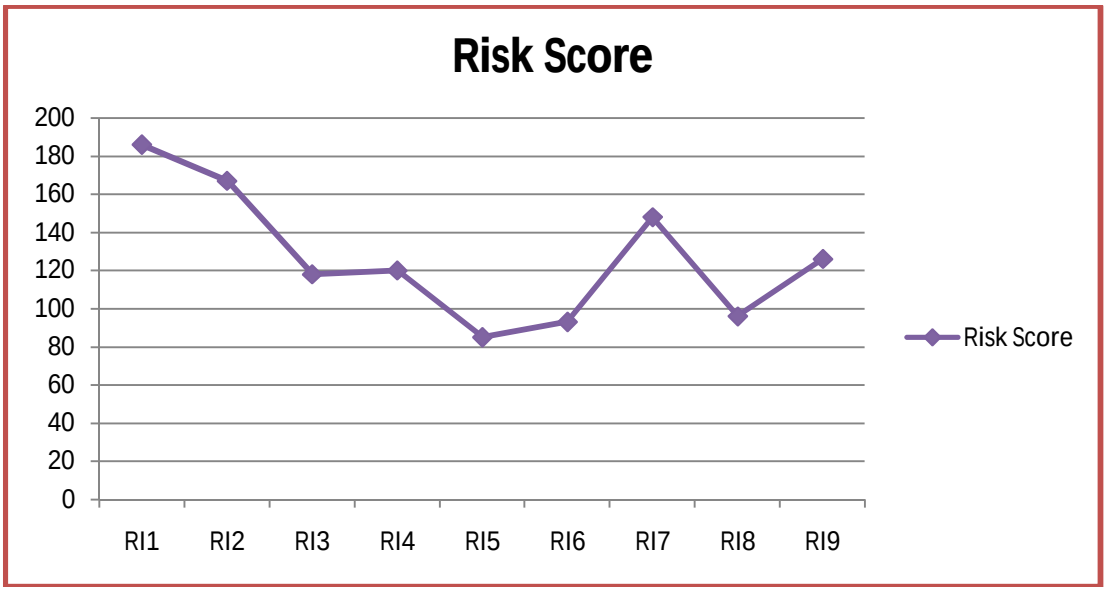


Figure 5.4: Risk analysis graph for Implementation Phase

Average of Risk Scores in Implementation phase=126.5556
So, RI1, RI2 and RI7 were considered as critical risks.

Evaluation results for Evolution Phase:

Table 5.5: Risk Scoring Method for Evolution Phase

Risk ID	Risks Identified (in Evolution Phase)	Risk Score
RE1	Problems in system updation.	152
RE2	Problems in licensing arrangements of COTS.	67
RE3	Difficulties in COTS replacements may occur.	184
RE4	Incompatibility of new version of COTS with user requirements.	202
RE5	Vendor may go out of business.	98
RE6	Loss of support and continuity when COTS vendor cease to exist.	101
RE7	System is not properly modified to meet new user needs.	162
RE8	Technical bugs are not overcome in allocated time.	90
RE9	There is no enough technical support from system vendors.	122
RE10	Insufficient support from top management during post-implementation (i.e. maintenance).	111

Figure 5.5: Risk analysis graph for Evolution Phase

Average of Risk Scores in evolution phase=128.9
So, RE1, RI3, RE4 and RE7 were considered as critical risks.

5.2 Risk Mitigation:

From the results obtained during risk analysis, the following graph shows the risk score percentile in various COTS-based Development phases.

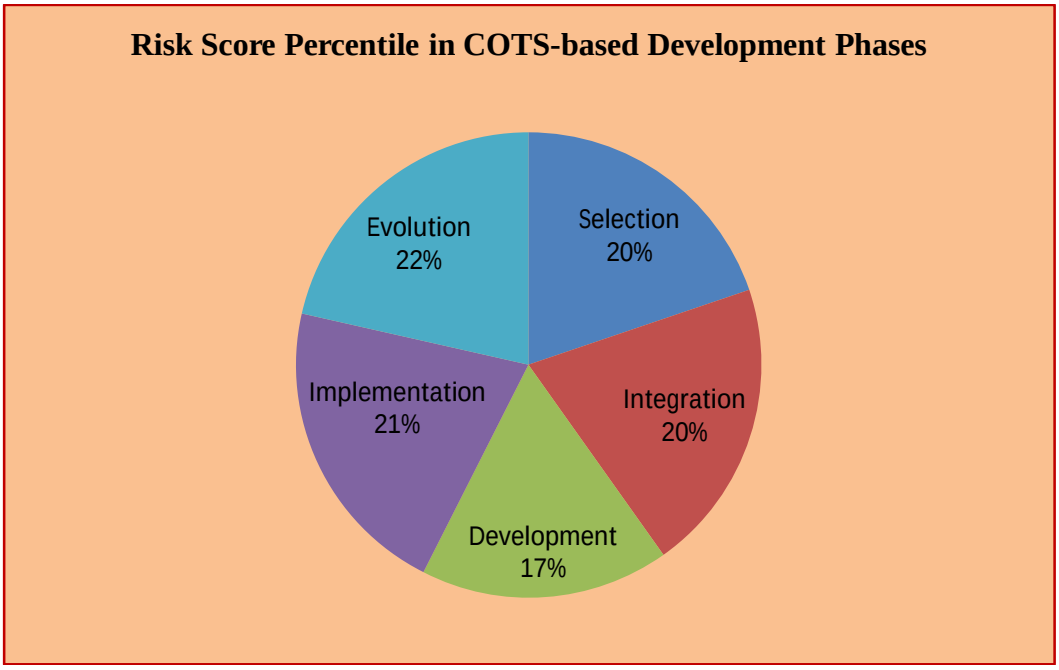


Figure 5.6: Risk Score percentile across COTS-based development phases

The most important and critical risks have been selected based on their probability and impact. Now, RIMCOTS model design mitigation strategies for identified critical risks.

Table 5.6: Risk Mitigation strategies for critical risks

Risks during Selection Phase
RS2: Occurrence of incomplete system requirements.   ***Mitigation:*** Proper System Requirements analysis should be done which include the Eliciting requirements, Analyzing requirements, Recording requirements steps.   ***RS3: Poor mapping of user-requirements to component based architecture.***   ***Mitigation:*** COTS-Aware Requirements Engineering and Software Architecting (*CARE/SA*) Framework should be used, which supports the iterative matching, Ranking, and selection of COTS components. COTS components are represented as an aggregate of their functional and non-functional requirements and architecture.   ***RS5: Inappropriate COTS Selection method used.***   ***Mitigation:*** Follow PORE (*Procurement Oriented Requirement Engineering*) Method during acquisition of requirements and selection of COTS components that satisfy these requirements.
Risks during integration Phase
RC1: Lack of Interoperability standards among different COTS Components.   ***Mitigation:*** Use *open Standard technologies* that are freely distributed data models or software infrastructure which provide basis for communication and enable

consistency among different COTS components.

RC2: Occurrence of incompatible format among COTS Components

Mitigation: Use of *Compositional wrappers* can provide a comprehensive treatment of component composition, including improved possibilities for automation of component.

RC5: Occurrence of too much glue code criticality during integration.

Mitigation: Choose exact match of COTS components with system requirements instead of approximate match of COTS components.

Risks during Development Phase

RD1: Selected Development Process is unsuitable for COTS-based development.

Mitigation: Developers should adopt dynamic, risk-driven process models, such as risk-driven spiral-type process models. Focus each spiral cycle on resolving the most critical risks

RD3: Resources are insufficient for COTS based development.

Mitigation: Proper Resource management is needed. Make sure in advance, the Human personnel with necessary skills, Hardware, Software resources are available for Development process.

Risks during Implementation Phase

RI1: Unclear design assumptions can reduce quality of testing.

Mitigation: There are four different testing scenarios to minimize this risk:

- *Prior to deployment:* the system integrator tests a new COTS component prior to deploying it in a larger system
- *Integrated system.* If a new component is added to the system or an older version replaced, the integrated system must be tested. Testing should also be done if the system configuration is altered.
- *Regression testing:* it should be performed critical system components whenever new versions of other constituent components are installed in the system
- *Non-functional testing.* Various kinds of non-functional testing on the system is required to ensure that the system meets the desired level of performance, dependability, stress and loading.

RI2: Occurrence of risks that may affects system safety or security.

Mitigation*:* Use trusted and reliable components. Always choose well-documented and certified components.

RI7: Probability of vulnerability attacks.

Mitigation: Choose Safe and trusted method for procurement of COTS components and a well-defined and relevant security policy should be used.

Risks during Evolution Phase

RE1: **Occurrence of problems in system updation.**

Mitigation: Perform the *upgrade management* process in general. Changes to components are going to occur and the program must be ready to effectively manage changing components and systems.

RE3: Difficulties in COTS replacements occur.
Mitigation: Perform *impact analysis* for the cost and difficulty of introducing a replacement for a COTS Product that is either a new product or newer version of an existing product that affect the software Configuration.
RE4: Incompatibility of newer version of COTS with user requirements.
Mitigation: *Quality control* should include fault identification, repair, and testing of installed and proposed COTS software.
RE7: System is not properly modified to meet new user needs.
Mitigation: *Configuration management* process should be followed. i.e. tracking and controlling the versions of all COTS products and custom software installed at all locations for the system
RE9: No enough technical support from Vendors.
Mitigation: During selection phase customer should analyze several components from competing vendors. They should analyze the costs involved in substituting a different component if needed.

5.3 Validation

To verify our results and suggested mitigation strategies, a feedback session with the respondents (i.e., the interviewees) was performed as shown in appendix B.2. The validation aimed at assessing the economical impact of the proposed RIMCOTS model.

This Characterize the *usefulness and adequacy* of the RIMCOTS model from the viewpoint of the risk management participants. To us usefulness and adequacy mean the advantages and drawbacks of risk management model with respect to (1) the RIMCOTS model's features (i.e., the techniques used within RIMCOTS), (2) performing explicit risk management in general. This validated risk management model will form the basis for future risk management activities.

CHAPTER 6

CONCLUSION & FUTURE WORK

COTS-based software development has a lot of promises. But it is not a silver bullet. The use of commercial off-the-shelf (COTS) software components is becoming an economic and strategic necessity for many organisations in a wide variety of different application areas including finance, defence, medicine, administration, manufacturing, and commerce. As the COTS software market develops, COTS users must face new challenges to successfully and effectively integrate commercial software components in applications and systems. Many organizations find that using COTS software carries a high risk and expense during development, deployment and maintenance of the system. To manage risks in COTS-based development, it is necessary to identify risks, evaluate them and design risk mitigation strategies.

6.1 Conclusion

We presented a RIMCOTS model that uses techniques for describing risks (Risk identification) and for selecting the most critical risks (Risk scoring technique) that are very effective for risk management for COTS-based software development. We have highlighted the various risks that cut across a component-based development cycle. We have categorised the risks into five categories and proposed a risk mitigation strategies for each critical risk. This identification and mitigation of critical risks brings COTS-based development a step closer to delivering on at an early stage. RIMCOTS model largely avoids COTS related risks by using ranking techniques that are appropriate for the type of information available. Our results showed that the RIMCOTS model is effective for Risk management, adds value to the project, and that its key concepts are understood and usable in practice.

6.2 Future Work

In Future, RIMCOTS model can be further extended to improve Risk Management so that the benefits of COTS-based development can be obtained to the larger extent. Our future research from this work could be extended into systematically refining and extending proposed Risk Mitigation Techniques.

REFERENCES

[1] C. L. Braun, “A Life Cycle Process for Effective Reuse of Commercial Off-the Shelf (COTS) Software”, In: Proceedings of the Fifth Symposium on Software Reusability, Los Angeles, California, 1999, pp. 29-36

[2] T. Oberndorf, L. Brownsword and C. A. Sledge, “An Activity Framework for COTS-Based Systems”, (CMU/SEI-2000-TR-010, ADA383836). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2000, pp. 3-5

[3] E. A. Karlsson, “Software Reuse, a Holistic Approach”, John Wiley & Sons, 1995.

[4] I. Crnkovic, “Component-based software engineering –new challenges in software development”, Software Focus, Vol. 2, No. 4, 2002, pp. 127–133.

[5] F. Bachmann, L. Bass, C. Buhman, S. Comella-Dorda, F. Long, J. Robert, R. Seacord and K. Wallnau, “Technical Concepts of Component-Based Software Engineering”, SEI Technical Report number CMU/SEI-2000-TR-008, Vol. 2, 2000, available at: http://www.sei.cmu.edu/

[6] R. John, Sr. Blanchette, “PROS AND CONS OF USING COTS PRODUCTS”, IEEE, 2005, pp. 472-476.

[7] M. Vigder, M. Gentleman and J. Dean, “COTS Software Integration: State of the Art”, Technical Report NRC No. 39190, 1996.

[8] M. Vidger and J. Dean, “An Architectural Approach to Building Systems from COTS Software Components”, Proc. of the 1997 Center for Advanced Studies Conference (CASCON 97), Toronto, Ontario, 1997.

[9] J. Peabody, “COTS Impact On Software Development Life Cycles”, Carnegie Mellon University, 2010.

[10] X. Cai, M. R. Lyu, K. F. Wong and R. Ko, “Component-based software engineering: Technologies, development frameworks, and quality assurance schemes”, in Proc. Asia-Pacific Software Engineering Conf., 2000, pp. 372–379.

[11] A. Rashid, G. Kotonya, “Risk Management in Component-based Development: A Separation of Concerns Perspective”, ECOOP Workshop on Advanced Separation of Concerns (ECOOP Workshop Reader), 2001: Springer-Verlag, Lecture Notes in Computer Science.

[12] L. Jingyue, F. O. Bjornson, R. Conradi and V. B. Kampenes, “An empirical study of variations in COTS-based software development processes in the Norwegian IT

industry", Springer Science and Business Media, 2006

[13] G. M. Kapfhammer, C.C. Michael, J. Haddox and R. Colyer. "An approach to identifying and understanding problematic COTS components", In Proceedings of the 2nd International Software Assurance and Certification Conference, Reston, Virginia, 2000.

[14] A. Minkiewicz, "Six Steps to a Successful COTS Implementation", Crosstalk Journal of Defense Software Engineering, 2005, pp. 17-21.

[15] P. Vitharana, "Risks and challenges of component-based software development", Commun. ACM, Vol. 46, No. 8, 2003, pp. 67–72.

[16] L. Chung, K. Cooper, and D.T. Huynh, "COTS-Aware Requirements Engineering Technique", submitted to ICCBSS, 2001.

[17] M. Morisio, C. Seaman, V. Basili, A. Parra, S. Kraft and S. Condon, "COTS-based software development: Processes and open issues", Journal of Systems and Software, Vol. 61,No. 3, 2002, pp.189-199.

[18] M. Morisio, C. Seaman, A. Parra, V. Basili, and S. Condon, "Investigating and Improving a COTS-Based Software Development Process", 22nd International Conference on Software Engineering, Limerick, Ireland, 2000.

[19] G. Fox, S. Marcom, K. Lantner, "A Software Development Process for COTS-Based Information System Infrastructure," CROSSTALK, 1998, pp. 20-25.

[20] W. K. Kim and J. Baik, "Dynamic Model for COTS Glue Code Development and COTS Integration", Department of Computer Science University of Southern California, pp.1-28.

[21] G. Kotonya and A. Rashid, "A Strategy for Managing Risk in Component-based Software Development", IEEE, 2001, pp.12-21

[22] L. C. Rose, "Risk Management of COTS Based System Development. Component-based Software Quality", Springer LNCS 2693, 2003, pp. 352-373

[23] U. Lindqvist and E. Jonsson, "A Map of Security Risks Associated with Using COTS", IEEE Computer, Vol. 31, No. 6, 1998, pp. 60-66.

[24] P. G. Neumann, "Computer-Related Risks", ACM Press, New York, 1995, pp. 2, 348

[25] Redmond W., "OFF97: Security Requirements when using NTFS Partitions", Article Q169387, Microsoft Corp., 1997.

[26] C. Albert, L. Brownsword, "Evolutionary Process for Integrating COTS-Based

Systems (EPIC): An Overview", (CMU/SEI-2002-TR-009). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2002, pp. 16-20.

[27] S. Comella-Dorda, J. Dean, G. Lewis, E. Morris, P. Oberndorf and E. Harper, "A Process for COTS Software Product Evaluation", (CMU/SEI-2003-TR-017, ADA443491). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003, pp. 86-96.

[28] D. J. Carney, E. Morris, J. Place and R. H. Patrick, "Identifying Commercial Off-the-Shelf (COTS) Product Risks: The COTS Usage Risk Evaluation", (CMU/SEI-2003-TR-023). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003.

[29] E. Hall, "Managing Risk: Methods for Software Systems Development", Readings, MA: Addision-Wesley, 1998.

[30] M. A. Ould, "Managing Software Quality and Business Risk", John Wiley & Sons, 1999.

[31] M. Vigder and J. Dean., "Building maintainable COTS based systems". In International Conference on Software Maintenance. IEEE, 1998, pp.132-138.

[32] W. Anderson; E. Morris, D. Smith and M. C. Ward, "COTS and Reusable Software Management Planning: A Template for Life-Cycle Management" (CMU/SEI-2007-TR-011). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2007.

LIST OF PUBLICATION/COMMUNICATED

[1] Amandeep Kaur Johar, Shivani Goel, “COTS components usage risks in Component Based Software Development", International Journal of Information Technology & Knowledge Management (ISSN 0973-4414) ,Vol. 2, No. 2, June 2011, pp. 573-575. (Published)

[2] Amandeep Kaur Johar, Shivani Goel, “Designing of RIMCOTS model for risk identification and mitigation for COTS-based software Development”, accepted in the Research Journal of Computer Systems and Engineering (for publication in July 2011).

Appendix A
Questionnaire for Probability and impact of Risks

A.1 Questionnaire for calculating risk score

Please rate the following risks in development using COTS components according to their probability of occurrence and impact during the life cycle phases.
Please tick (X) in the corresponding column.

Table A.1 Questionnaire for Probability of occurrence and impact of COTS risks

Risks during Component Selection phase (RS)		Low	Medium	High
RS1	What is the probability of difficulty in predicting component behaviour?			
RS2	What is the probability of occurrence of incomplete system requirements?			
RS3	What is the probability that poor mapping of user-requirements to component-based architecture may occur?			
RS4	What is the probability of lack of cooperation from users providing COTS from repository?			
RS5	What is the probability of inappropriate COTS selection and procurement method used?			
RS6	What is the probability that the required COTS is found costly as compared to in house development cost?			
RS7	What is the probability that very less COTS components are available for selection?			
Risks during Component Integration phase (RC)				
RC1	What is the probability of lack of interoperability standards among different COTS components?			
RC2	What is the probability of occurrence of incompatible format among COTS components?			
RC3	What is the probability that effort for integration may increase from what was estimated?			
RC4	What is the probability that cost for integration may increase from what was estimated?			
RC5	What is the probability of occurrence of too much glue code criticality during COTS integration?			
RC6	What is the probability that additional features than required may exist in selected COTS components which may cause difficulties in integration?			
Risks during System Development phase (RD)				
RD1	What is the probability that selected development process may be unsuitable for COTS-based development?			
RD2	What is the probability that risk analysis phase is not present in selected COTS based development process?			
RD3	What is the probability that resources are insufficient for COTS based development?			

RD4	What is the probability that development time and cost exceeds the estimated time and cost?			
Risks during implementation phase (RI)		**Low**	**Medium**	**High**
RI1	What is the probability that unclear design assumptions can reduce quality of testing?			
RI2	What is the probability of occurrence of risks that may affects system safety or security?			
RI3	What is the probability of occurrence of risks that may affects system performance?			
RI4	What is the probability of occurrence of risks that may affects system reliability?			
RI5	What is the probability that system can be used in unintended way?			
RI6	What is the probability of occurrence of external exposure?			
RI7	What is the probability of vulnerability attacks?			
RI8	What is the probability that users may feel uncomfortable while using the system?			
RI9	What is the probability that users cannot retrieve relevant and needed information from the system?			
Risks during System Evolution phase (RE)				
RE1	What is the probability of occurrence of problems in system updation?			
RE2	What is the probability of occurrence of problems in licensing arrangements of COTS?			
RE3	What is the probability that difficulties in COTS replacements may occur?			
RE4	What is the probability of incompatibility of new version of COTS with user requirements?			
RE5	What is the probability that vendor may go out of business?			
RE6	What is the probability of loss of support and continuity when COTS vendor cease to exist?			
RE7	What is the probability that the system is not properly modified to meet new user needs?			
RE8	What is the probability that technical bugs are not overcome in allocated time?			
RE9	What is the probability that there is no enough technical support from system vendors?			
RE10	What is the probability of insufficient support from top management during post-implementation (i.e. maintenance)?			

Appendix B
Evaluation of Risk Scores

B.1 Risk Score Calculation

This appendix presents the results of risk score evaluation for individual risk in each development phase.

Sr. No.	RS1	IS1	prod	RS2	IS2	Prod	RS3	IS3	Prod	RS4	IS4	Prod	RS5	IS5	Prod	RS6	IS6	Prod	RS7	IS7	Prod
1	2	2	4	3	3	9	3	3	9	2	1	2	3	3	9	1	2	2	2	1	2
2	2	2	4	3	3	9	2	2	4	1	1	1	2	3	6	1	2	2	1	1	1
3	3	2	6	3	3	9	3	3	9	1	1	1	2	3	6	1	2	2	2	1	2
4	3	2	6	3	3	9	2	3	6	1	1	1	2	3	6	3	2	6	2	1	2
5	3	2	6	3	3	9	3	3	9	2	1	2	2	2	4	1	2	2	1	2	2
6	2	2	4	3	3	9	3	3	9	1	1	1	2	3	6	2	2	4	2	1	2
7	2	2	4	3	3	9	1	3	3	2	1	2	3	3	9	2	2	4	3	1	3
8	2	2	4	2	3	6	3	3	9	1	2	2	2	3	6	1	3	3	2	1	2
9	2	3	6	3	3	9	2	3	6	1	1	1	3	3	9	1	2	2	2	1	2
10	2	2	4	3	2	6	3	3	9	1	1	1	3	3	9	1	2	2	2	1	2
11	2	2	4	2	2	4	2	3	6	2	1	2	3	3	9	1	2	2	3	1	3
12	2	2	4	2	2	4	3	3	9	1	2	2	3	3	9	2	2	4	1	1	1
13	1	3	3	2	3	6	2	2	4	2	1	2	3	3	9	1	2	2	2	1	2
14	1	3	3	3	3	9	2	3	6	1	1	1	3	3	9	1	2	2	1	1	1
15	3	3	9	3	3	9	2	3	6	1	1	1	2	3	6	2	2	4	2	1	2
16	2	3	6	3	3	9	2	3	6	2	1	2	3	3	9	1	2	2	2	1	2
17	2	2	4	3	3	9	2	3	6	3	1	3	3	3	9	2	2	4	3	2	6
18	3	2	6	3	3	9	2	3	6	1	1	1	2	2	4	3	2	6	2	1	2
19	3	2	6	3	3	9	2	3	6	2	1	2	2	2	4	3	3	9	2	1	2
20	2	2	4	3	3	9	3	3	9	1	2	2	2	3	6	2	2	4	1	1	1
21	2	2	4	3	3	9	2	3	6	2	1	2	3	3	9	1	2	2	3	1	3
22	2	2	4	3	3	9	3	3	9	3	1	3	3	3	9	1	2	2	3	1	3
23	3	2	6	3	3	9	3	3	9	1	1	1	3	3	9	2	2	4	1	1	1
24	2	2	4	3	3	9	3	3	9	2	1	2	3	3	9	2	2	4	1	1	1
Risk score			115			197			170			40			180			80			50

Figure B.1 Risk Scores for Selection Phase

Sr. no.	RC1	IC1	Prod	RC2	IC2	Prod	RC3	IC3	Prod	RC4	IC4	Prod	RC5	IC5	Prod	RC6	IC6	Prod
1	3	3	9	3	2	6	3	2	6	2	1	2	3	3	9	1	1	1
2	2	3	6	3	2	6	3	2	6	3	1	3	3	2	6	1	1	1
3	3	3	9	3	2	6	2	2	4	3	1	3	2	3	6	1	2	2
4	3	2	6	3	2	6	2	2	4	1	1	1	2	3	6	2	1	2
5	3	3	9	3	3	9	2	2	4	2	1	2	3	3	9	1	1	1
6	3	2	6	3	2	6	2	2	4	2	2	4	3	3	9	2	1	2
7	3	3	9	2	2	4	2	2	4	2	1	2	3	3	9	1	1	1
8	2	3	6	3	2	6	3	1	3	3	1	3	3	3	9	1	1	1
9	3	3	9	3	2	6	2	2	4	2	1	2	3	2	6	3	2	6
10	1	2	2	2	2	4	2	2	4	3	2	6	3	2	6	2	1	2
11	3	3	9	3	2	6	3	2	6	3	2	6	3	2	6	2	1	2
12	2	3	6	3	1	3	2	3	6	3	2	6	2	3	6	2	2	4
13	3	3	9	2	3	6	3	2	6	3	2	6	2	3	6	1	1	1
14	1	3	3	2	2	4	3	2	6	2	1	2	3	3	9	1	1	1
15	3	3	9	3	2	6	2	2	4	3	2	6	3	3	9	2	1	2
16	3	3	9	2	2	4	2	2	4	2	1	2	3	2	6	3	1	3
17	2	3	6	3	2	6	3	2	6	2	1	2	2	2	4	2	1	2
18	3	3	9	3	2	6	2	2	4	2	1	2	3	2	6	2	1	2
19	3	2	6	3	2	6	3	1	3	2	1	2	3	2	6	2	1	2
20	2	2	4	2	2	4	3	1	3	3	2	6	3	3	9	2	2	4
21	3	3	9	3	3	9	3	2	6	1	1	1	3	3	9	2	2	4
22	3	3	9	3	2	6	3	2	6	2	1	2	3	3	9	2	1	2
23	3	3	9	3	2	6	3	3	9	2	1	2	3	3	9	2	1	2
24	3	3	9	3	2	6	3	2	6	2	1	2	3	3	9	1	1	1
Risk Score			177			137			118			75			178			51

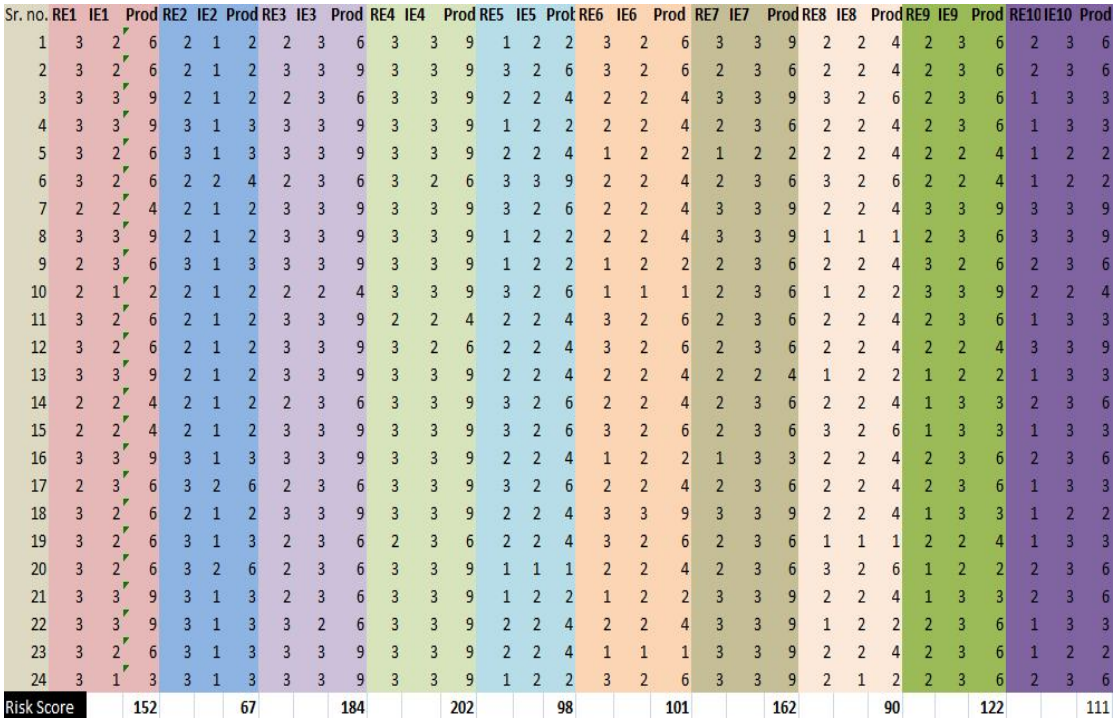
Figure B.2 Risk Scores for Integration Phase

Sr.no.	RD1	ID1	Prod	RD2	ID2	Prod	RD3	ID3	Prod	RD4	ID4	Prod
1	1	3	3	1	2	2	2	3	6	1	2	2
2	2	3	6	1	2	2	1	2	2	2	2	4
3	3	3	9	3	2	6	3	3	9	3	2	6
4	1	3	3	2	2	4	2	3	6	2	2	4
5	2	2	4	2	2	4	1	2	2	1	1	1
6	1	3	3	2	2	4	2	3	6	3	2	6
7	3	2	6	2	1	2	2	2	4	3	2	6
8	2	3	6	1	2	2	3	2	6	2	2	4
9	2	3	6	1	1	1	2	2	4	2	2	4
10	2	3	6	1	2	2	3	2	6	2	2	4
11	2	2	4	1	1	1	1	2	2	2	2	4
12	2	3	6	1	2	2	2	3	6	2	2	4
13	1	2	2	1	2	2	2	3	6	2	2	4
14	2	3	6	1	2	2	1	3	3	2	2	4
15	1	2	2	2	2	4	3	3	9	3	2	6
16	2	3	6	2	2	4	3	3	9	2	2	4
17	2	2	4	1	2	2	3	3	9	2	2	4
18	1	3	3	2	2	4	3	3	9	2	2	4
19	1	3	3	2	2	4	2	3	6	3	3	9
20	2	3	6	1	2	2	2	3	6	2	2	4
21	1	1	1	2	1	2	1	2	2	2	1	2
22	1	3	3	2	2	4	3	3	9	3	2	6
23	2	3	6	1	1	1	3	3	9	3	2	6
24	1	2	2	1	2	2	2	3	6	1	1	1
Risk Score			106			65			142			103

Figure B.3 Risk Scores for Development Phase

Sr. no.	RI1	II1	Prod	RI2	II2	Prod	RI3	II3	Prod	RI4	II4	Prod	RI5	II5	Prod	RI6	II6	prod	RI7	II7	Prod	RI8	II8	Prod	RI9	II9	Prod
1	3	3	9	2	3	6	2	2	4	2	2	4	1	3	3	1	2	2	2	3	6	3	3	9	2	3	6
2	3	3	9	3	3	9	2	2	4	2	2	4	1	3	3	3	2	6	3	3	9	1	3	3	1	2	2
3	3	3	9	3	3	9	2	2	4	2	2	4	2	3	6	2	2	4	1	2	2	1	3	3	3	3	9
4	2	3	6	3	3	9	3	2	6	3	2	6	1	3	3	2	2	4	2	3	6	1	2	2	1	2	2
5	2	2	4	3	3	9	2	3	6	2	2	4	2	3	6	1	2	2	1	3	3	2	2	4	2	3	6
6	2	3	6	2	2	4	3	2	6	2	2	4	1	2	2	3	2	6	2	3	6	1	2	2	2	3	6
7	3	3	9	3	3	9	2	2	4	2	2	4	1	2	2	3	3	9	2	3	6	2	2	4	2	3	6
8	1	2	2	3	3	9	3	2	6	2	2	4	2	2	4	1	2	2	1	3	3	2	3	6	3	3	9
9	3	3	9	2	2	4	2	2	4	3	3	9	2	2	4	1	2	2	2	3	6	1	3	3	2	3	6
10	2	3	6	3	3	9	3	3	9	2	2	4	2	3	6	1	2	2	1	2	2	2	3	6	1	3	3
11	3	3	9	2	2	4	3	2	6	3	2	6	2	3	6	2	2	4	2	3	6	2	3	6	2	3	6
12	3	3	9	3	2	6	2	2	4	2	2	4	1	2	2	2	2	4	1	3	3	2	3	6	2	3	6
13	3	3	9	2	3	6	2	2	4	2	3	6	1	3	3	1	2	2	2	3	6	1	3	3	2	2	4
14	2	3	6	3	3	9	1	2	2	3	2	6	2	3	6	2	2	4	3	3	9	1	2	2	2	3	6
15	2	3	6	3	3	9	3	2	6	2	2	4	1	3	3	1	2	2	3	3	9	2	3	6	2	3	6
16	3	3	9	3	3	9	2	2	4	2	2	4	3	2	6	2	2	4	2	3	6	1	2	2	2	3	6
17	3	3	9	2	3	6	3	2	6	3	2	6	1	3	3	2	2	4	3	2	6	2	3	6	3	3	9
18	3	3	9	2	3	6	2	2	4	3	2	6	1	2	2	2	2	4	3	3	9	1	3	3	2	3	6
19	3	3	9	2	3	6	2	2	4	3	3	9	1	2	2	2	2	4	3	3	9	1	3	3	2	3	6
20	3	3	9	2	2	4	2	2	4	2	2	4	1	2	2	1	2	2	2	3	6	1	3	3	2	3	6
21	3	3	9	3	3	9	2	2	4	2	2	4	1	2	2	2	3	6	2	3	6	1	3	3	1	3	3
22	3	3	9	2	3	6	3	3	9	2	2	4	1	3	3	2	3	6	3	3	9	1	2	2	1	3	3
23	3	2	6	2	2	4	2	2	4	3	2	6	1	3	3	2	2	4	3	3	9	1	3	3	1	2	2
24	3	3	9	2	3	6	2	2	4	2	2	4	1	3	3	2	2	4	2	3	6	2	3	6	1	2	2
Risk Score			186			167			118			120			85			93			148			96			126

Figure B.4 Risk Scores for Implementation Phase



Sr. no.	RE1	IE1	Prod	RE2	IE2	Prod	RE3	IE3	Prod	RE4	IE4	Prod	RE5	IE5	Prod	RE6	IE6	Prod	RE7	IE7	Prod	RE8	IE8	Prod	RE9	IE9	Prod	RE10	IE10	Prod
1	3	2	6	2	1	2	2	3	6	3	3	9	1	2	2	3	2	6	3	3	9	2	2	4	2	3	6	2	3	6
2	3	2	6	2	1	2	3	3	9	3	3	9	3	2	6	3	2	6	2	3	6	2	2	4	2	3	6	2	3	6
3	3	3	9	2	1	2	2	3	6	3	3	9	2	2	4	2	2	4	3	3	9	3	2	6	2	3	6	1	3	3
4	3	3	9	3	1	3	3	3	9	3	3	9	1	2	2	2	2	4	2	3	6	2	2	4	2	3	6	1	3	3
5	3	2	6	3	1	3	3	3	9	3	3	9	2	2	4	1	2	2	1	2	2	2	2	4	2	2	4	1	2	2
6	3	2	6	2	2	4	2	3	6	3	2	6	3	3	9	2	2	4	2	3	6	3	2	6	2	2	4	1	2	2
7	2	2	4	2	1	2	3	3	9	3	3	9	3	2	6	2	2	4	3	3	9	2	2	4	3	3	9	3	3	9
8	3	3	9	2	1	2	3	3	9	3	3	9	1	2	2	2	2	4	3	3	9	1	1	1	2	3	6	3	3	9
9	2	3	6	3	1	3	3	3	9	3	3	9	1	2	2	1	2	2	2	3	6	2	2	4	3	2	6	2	3	6
10	2	1	2	2	1	2	2	2	4	3	3	9	3	2	6	1	1	1	2	3	6	1	2	2	3	3	9	2	2	4
11	3	2	6	2	1	2	3	3	9	2	2	4	2	2	4	3	2	6	2	3	6	2	2	4	2	3	6	1	3	3
12	3	2	6	2	1	2	3	3	9	3	2	6	2	2	4	3	2	6	2	3	6	2	2	4	2	2	4	3	3	9
13	3	3	9	2	1	2	3	3	9	3	3	9	2	2	4	2	2	4	2	2	4	1	2	2	1	2	2	1	3	3
14	2	2	4	2	1	2	2	3	6	3	3	9	3	2	6	2	2	4	2	3	6	2	2	4	1	3	3	2	3	6
15	2	2	4	2	1	2	3	3	9	3	3	9	3	2	6	3	2	6	2	3	6	3	2	6	1	3	3	1	3	3
16	3	3	9	3	1	3	3	3	9	3	3	9	2	2	4	1	2	2	1	3	3	2	2	4	2	3	6	2	3	6
17	2	3	6	3	2	6	2	3	6	3	3	9	3	2	6	2	2	4	2	3	6	2	2	4	2	3	6	1	3	3
18	3	2	6	2	1	2	3	3	9	3	3	9	2	2	4	3	3	9	3	3	9	2	2	4	1	3	3	1	2	2
19	3	2	6	3	1	3	2	3	6	2	3	6	2	2	4	3	2	6	2	3	6	1	1	1	2	2	4	1	3	3
20	3	2	6	3	2	6	2	3	6	3	3	9	1	1	1	2	2	4	2	3	6	3	2	6	1	2	2	2	3	6
21	3	3	9	3	1	3	2	3	6	3	3	9	1	2	2	1	2	2	3	3	9	2	2	4	1	3	3	2	3	6
22	3	3	9	3	1	3	3	2	6	3	3	9	2	2	4	2	2	4	3	3	9	1	2	2	2	3	6	1	3	3
23	3	2	6	3	1	3	3	3	9	3	3	9	2	2	4	1	1	1	3	3	9	2	2	4	2	3	6	1	2	2
24	3	1	3	3	1	3	3	3	9	3	3	9	1	2	2	3	2	6	3	3	9	2	1	2	2	3	6	2	3	6
Risk Score			152			67			184			202			98			101			162			90			122			111

Figure B.5 Risk Scores for Evolution Phase

B.2 Validation Overview

Feedback questions are included in the Questionnaire and asked from risk management team members. For each question, respondents were asked to provide their satisfaction level.

Table B.1: validation questionnaire

S. No	Feedback Questions	Satisfaction level		
		L	M	H
V1.	Were the monitoring questions useful for determining risk and project status?			
V2.	Was the definition of risks by RIMCOTS model helpful for understanding of the risk?			
V3.	Do you think documentation of the risks in RIMCOTS model is appropriate?			
V4.	Did you have appropriate information to perform the comparison to identify the critical risk scenarios?			
V5.	Was the selection of most critical risks using ranking method comprehensible?			
V6.	Was the information on the risk sheet appropriate for risk analysis?			
V7.	Did this proposed RIMCOTS helps in better project management?			

V8.	Did the proposed mitigation strategies for critical risks effective over time?			
V9.	Was the data collection method for Risk analysis reliable?			
V10.	Were the Analysis graphs helpful in understanding of risk context and its consequences?			
V11.	Did you perceive RIMCOTS model for risk management as beneficial and practical?			
V12.	Did you think the controlling actions effectively controls the risk and develop the qualitative COTS-based system?			

B.3 Validation Results

Calculations were done on data filled by project and risk management team members to verify effectiveness of RIMCOTS model.

No.	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12
1	2	3	3	3	3	3	3	3	3	3	3	3
2	3	3	2	3	3	3	3	3	3	3	3	3
3	3	3	2	3	3	3	3	3	3	3	3	3
4	3	3	2	3	3	3	3	3	3	3	3	3
5	3	3	3	3	3	3	3	3	3	3	3	3
6	3	3	3	3	3	3	3	3	3	3	3	3
7	3	2	2	3	2	2	2	2	2	2	2	2
8	3	2	2	2	2	2	2	3	2	2	2	2
9	3	2	2	2	2	2	2	2	2	2	2	2
10	3	2	3	3	2	3	3	2	2	3	2	2
11	2	1	3	2	3	1	2	1	3	1	2	3
12	2	2	3	3	2	2	2	2	2	2	2	2
13	2	3	2	3	3	3	3	3	3	3	3	3
14	2	3	2	3	3	3	3	3	3	3	3	3
15	3	2	2	3	2	2	2	2	2	2	2	2
16	2	3	3	3	3	3	3	3	3	1	3	3
17	3	3	3	3	3	3	3	3	3	3	3	3
18	3	3	3	2	3	3	3	3	3	3	3	3
19	3	3	3	3	3	3	3	3	3	3	3	1
20	3	3	3	3	3	3	3	3	1	3	3	3
21	3	3	3	3	3	3	3	3	3	3	3	3
22	1	3	3	3	3	3	3	3	3	3	3	3
23	2	3	3	3	3	3	3	3	3	3	3	3
24	1	3	3	2	3	3	3	3	3	3	3	3
Satisfaction Level	61	64	63	67	66	65	66	65	64	63	65	64

Figure B.6 RIMCOTS usage satisfaction level

Calculations results into following:

%age Satisfactory level=773/864*100

=89.5

The satisfaction level of using RIMCOTS model is 89.5 %

Thus, we can say that RIMCOTS model is effective for Risk management for COTS-based Software development.