

**THE ROLE & IMPORTANCE OF
COMPONENT DOCUMENTATION IN
COMPONENT INTEGRATION PHASE:
A SYSTEM INTEGRATOR'S PERSPECTIVE**

BY

AZHAR SAEED KHAN

A Thesis Presented to the
DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

INFORMATION AND COMPUTER SCIENCE

JUNE, 2011

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by AZHAR SAEED KHAN under the direction of his thesis adviser and approved by his thesis committee, has been presented to and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of MASTER OF SCIENCE IN INFORMATION & COMPUTER SCIENCE.

Thesis Committee

Sajjad Mahmood June 11, 2011

Dr. Sajjad Mahmood (Adviser)

June 11, 2011

Dr. Muhammed S. Al- Mulhem (Member)

Alshayeb

Dr. Mohammad R. Alshayeb (Member)

عادل احمد 12/6/11

Dr. Adel Ahmed
Department Chairman

[Signature]
Dr. Salam Adel Zummo
Dean of Graduate Studies



15/6/11

Date

DEDICATION

This thesis is dedicated to my parents, siblings and my wife for their unconditional love, and trust. They have always supported me when I have needed it most. I could never have been able to pursue higher education without their encouragement and support.

Thank you very much.

ACKNOWLEDGEMENT

All praise is to Allah, the Almighty alone. May the Peace and Blessings of Allah be upon the Messenger of Allah (Salla Allahu alaihi wasallam), his family, and his companions (Radhi Allah Anhum).

I am grateful to the King Fahd University of Petroleum & Minerals for providing a great environment for research and academics. This thesis would not have been possible without the guidance and the help of several individuals who in one way or another contributed and extended their valuable assistance in the preparation and completion of this study. I first wish to extend my gratitude to my thesis adviser Dr Sajjad Mahmood for his continuous support, patience, and much needed encouragement. His extensive knowledge and experience was what shaped this research. I am also thankful to my thesis committee Dr. M. S. Al- Mulhem and Dr. M.R. Alshayeb for their time and useful comments.

I am also very thankful to all my friends for their help, support and valuable input to my studies. There are several other people who have helped me during my thesis in several

different ways. I am grateful to them all. Furthermore, I acknowledge all industrial practitioners who participated in this work for their support, contribution and cooperation during the industrial survey.

I would also like to thank my family. A specific appreciation goes to my parents for supporting and believing in me during all these years. I would also like to especially thank my wife for her continuous support in more than one way.

TABLE OF CONTENTS

DEDICATION	ii
ACKNOWLEDGEMENT	iii
TABLE OF CONTENTS.....	v
LIST OF FIGURES	x
LIST OF TABLES	xi
ABBREVIATIONS	xii
THESIS ABSTRACT	xiii
CHAPTER 1 INTRODUCTION	1
1.1 Overview	1
1.2 Problem Outline	4
1.3 Research Objectives	6
1.4 Research Motivations.....	6
1.5 Research Contributions	7
1.6 Thesis Structure.....	8
CHAPTER 2 BACKGROUND	9
2.1 Component-Based Software Development	9

2.2 Component Integration.....	16
2.3 Risk Management in Software Development.....	18
2.4 Risk Management in Component-Based Software Development.....	21
CHAPTER 3 LITERATURE SURVEY.....	23
3.1 Component Based Development.....	23
3.2 Component Integration.....	24
3.3 Component Documentation.....	26
3.4 Component Integration Risks.....	29
CHAPTER 4 RESEARCH METHOD	32
4.1 Data Collection.....	32
4.2 Data Analysis	35
CHAPTER 5 THE ROLE OF COMPONENT DOCUMENTATION IN COMPONENT INTEGRATION	38
5.1 Introduction	39
5.2 Research Questions and Hypotheses.....	40
5.3 Data and Result Analysis	44
5.3.1 Hypotheses testing phase-I.....	44
5.3.2 Hypotheses testing phase-II.....	46
5.3.3 Hypotheses testing phase-III	48

5.4 Discussion	51
CHAPTER 6 THE RISK FACTORS DURING COMPONENT INTEGRATION	55
6.1 Introduction	56
6.2 Research Questions and Hypotheses.....	57
6.3 Data and Result Analysis	60
6.3.1 Hypotheses testing phase-I	61
6.3.2 Hypotheses testing phase-II.....	64
6.3.3 Hypotheses testing phase-III	66
6.4 Discussion	68
CHAPTER 7 QUALITATIVE DATA ANALYSIS	74
7.1.1 Upgrading of components	75
7.1.2 Motivations of using commercial components.....	75
7.1.3 Determining the quality and reliability of the component.....	76
7.1.4 Limited control on commercial components	77
7.1.5 Ensuring that System Integrator has selected the right component.....	78
7.1.6 Difficulties faced in integrating components.....	78
7.1.7 Maintaining relationship with vendor.....	79
7.1.8 Testing the selected component for integrating.....	80
7.1.9 Difficulty in detecting defects and bugs in component base system.....	80

7.1.10 Component Certification	81
7.1.11 Component Repository	81
7.1.12 Strengths in integrating components	82
7.1.13 Weaknesses in integrating components	83
7.1.14 Further Observation	84
7.1.15 Avoid Component Risk:	86
CHAPTER 8 THREATS TO VALIDITY	88
8.1 Construct Validity	88
8.2 External Validity	89
8.3 Internal Validity	90
8.4 Conclusion Validity.....	91
CHAPTER 9 GUIDELINES.....	92
9.1 Guidelines.....	92
9.2 Validation of Proposed Guidelines	99
CHAPTER 10 CONCLUSION AND FUTURE WORK	102
10.1 Conclusion.....	102
10.2 Future work	104
APPENDIX A: INDUSTRIAL SURVEY	106
APPENDIX B: INDUSTRIAL SURVEY RAW DATA	126

REFERENCE.....	171
Vita.....	175

LIST OF FIGURES

Figure 1: Component-Based Software Development	11
Figure 2: CBS Development Life Cycle	12
Figure 3: Development with off-the-shelf components: actors and activities	18
Figure 4: Risk Management Framework	21
Figure 5: Theoretical Research Model.....	43
Figure 6: Theoretical Research Model.....	60
Figure 7: Proposed Guidelines.....	97
Figure 8: Mapping of Hypothesis & Open ended analysis with Guidelines.....	98

LIST OF TABLES

Table 1 : Correlation Coefficient Range	37
Table 2: Hypotheses testing using Pearson Correlation Coefficient	46
Table 3: Hypotheses testing using Spearman rank-order correlation coefficient	48
Table 4: Hypotheses testing using Partial Least Square Regression (PLS).....	50
Table 5: Comparison of results from different tests	51
Table 6: Hypotheses testing using Pearson Correlation Coefficient	63
Table 7: Hypotheses testing using Spearman rank-order correlation coefficient	65
Table 8: Hypotheses testing using Partial Least Square Regression (PLS).....	67
Table 9: Comparison of results from different tests	68

ABBREVIATIONS

CBS	Component-Based System
CBSD	Component-Based Software Development
CBSE	Component-Based Software Engineering
COM	Component Object Model (Microsoft)
COM	Component Object Model
EJB	Enterprise Java Beans (Sun)
CIMO	Component Integration Model
XML	Extensible Markup Language
ISO	International Standards Organization
COSO	Commission of Sponsoring Organizations
LOC	Lines of Code
COTS	Commercial-Off-The-Shelf
OTS	Off-The-Shelf
OSS	Open Source Software
PLS	Partial Least Square

THESIS ABSTRACT

NAME: AZHAR SAEED KHAN

TITLE: THE ROLE & IMPORTANCE OF COMPONENT DOCUMENTATION IN
COMPONENT INTEGRATION PHASE: A SYSTEM INTEGRATOR'S
PERSPECTIVE

MAJOR FIELD: COMPUTER SCIENCE

DATE OF DEGREE: JUNE 2011.

Component integration is widely recognized as a process which plays a central role in overall Component Based System (CBS) development. Due to the integration-centric nature of CBS development, a system analyst focuses on assembling existing components, developed by different parties, to build a software system CBS integration is a risk-prone process because it is rarely the case that components are perfectly matched and ready for 'plug and play'. The lack of detailed component documentation has been a key area of concern in CBS development due to its profound impact on the integration phase of a CBS development life cycle. It's difficult for system integrators to find out capabilities of components if component are not documented in a standard way. Component documentation can help in assessing the applicability and quality of a component. This research assists to understand the role of component documentation. It helps in identifying missing gaps in available component documentation and in identifying key risk factors which can take place during component integration process. In this research,

we report results of an industrial survey conducted among system integrators to understand role and importance of component documentation in the CBS and the impact of different component integration risk factors. The survey investigates whether the presence of component documentation helps a system integrator and its correlations with typical CBS integration success factors and identifies risk factors during integration process. We received data from 53 CBS integrators working in small to medium organizations. The result reinforces current perceptions of the significance of component documentation in CBS integration. Furthermore, based on the result analysis, we come up with component integration guidelines which can be used by any system integrator to avoid such risks during component integration phase.

MASTER OF SCIENCE DEGREE

KING FAHD UNIVERSITY OF PETROLEUM AND MINERALS, DHAHRAN,
SAUDI ARABIA

JUNE 2011

ملخص رسالة

الاسم : اظهر سعيد خان
العنوان : دور وأهمية الوثائق مكون في مرحلة الاندماج المكون : منظور تكامل النظام
مجال التخصص : علم الحاسوب
تاريخ الدرجة العلمية : يونيو 2011.

ومن المسلم به على نطاق واسع مكون التكامل كعملية التي تلعب دورا مركزيا في التنمية الشاملة (سي بي اس) على أساس العنصر النظام. نظرا لطبيعة التكامل تتمحور حول التنمية سي بي اس، وهو محلل نظام يركز على تجميع المكونات الموجودة، التي وضعتها مختلف الأطراف، لبناء شبكة سي بي اس نظام دمج البرامج هو عملية المعرضة للخطر لأنه نادرا ما يحدث أن تكون مطابقة تماما ومكونات استعداد ل'التوصيل والتشغيل. وكان عدم وجود وثائق مفصلة المكون مجالا رئيسيا للقلق في تطوير شبكة سي بي اس نظرا لتأثير عميق على مرحلة الاندماج في حياة سي بي اس دورة التنمية. من الصعب للتكامل النظم لمعرفة قدرات المكونات إذا لم يتم توثيقها عنصر في طريقة قياسية. يمكن مكون الوثائق مساعدة في تقييم مدى انطباق الجودة عنصر. هذا البحث يساعد في فهم دور وثائق المكون. وهي تساعد في تحديد الثغرات في عداد المفقودين في الوثائق المتاحة وعنصر في تحديد عوامل الخطر الرئيسية التي يمكن أن تحدث أثناء عملية التكامل عنصر. في هذا البحث، ونحن على تقرير نتائج المسح الصناعي أجريت بين الأنظمة المتكاملة لفهم دور وأهمية الوثائق عنصر في شبكة سي بي اس، وتأثير مختلف العوامل عنصر خطر الاندماج. المسح تحقق ما إذا كان وجود وثائق عنصر يساعد على تكامل النظام والارتباط مع شبكة سي بي اس نموذجي عوامل نجاح التكامل، ويحدد عوامل الخطورة أثناء عملية التكامل. تلقينا بيانات من تكامل شبكة سي بي اس 53 العمل في الشركات الصغيرة والمؤسسات المتوسطة. النتيجة يعزز التصورات الحالية لأهمية الوثائق عنصر التكامل في شبكة سي بي اس. علاوة على ذلك، استنادا إلى تحليل نتيجة لذلك ، جننا مع المبادئ التوجيهية التكامل عنصر والتي يمكن استخدامها من قبل أي تكامل النظام لتجنب مثل هذه المخاطر خلال مرحلة الاندماج المكون.

شهادة ماجستير علوم
جامعة الملك فهد للبترول والمعادن ، الظهران بالمملكة العربية السعودية
يونيو 2011

CHAPTER 1

INTRODUCTION

In this chapter, we briefly present overview of component based system and component integration in component based system. Furthermore, chapter describes the problem outline, research objectives, research motivation, and research deliverables and in the end the structure of the thesis.

1.1 Overview

The extensive uses of software have placed new demands on the software industry to enhance development productivity and reduce associated costs [1]. These expectations have led software engineers to re-establish the idea of reuse and focus on moving the

software industry away from developing each system from scratch [2], [3]. CBS development emerged in the mid-90s with the introduction of software component technologies such as Microsoft COM and COBRA. Recently, the next generation of CBS development tools such as Microsoft .NET, Enterprise Java Beans (EJB) etc. are available in the market. CBS has been successfully applied to a range of industrial applications such as web-based financial tools of a large European bank [4]. In CBS development, a component is a fundamental building block for a software application. Szyperski et al. [5] defines component as follows : *A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only.* A software component can be deployed independently and is subject to composition by third parties.

CBS development is a very fast growing trend in software industry and It's becoming increasingly important since it prompts reuse to higher levels of abstraction [1, 6]. The main idea of CBS development is to reuse the already developed components instead of developing them from scratch. There are different definitions on components whereas in this research, we adopt the definition from Brown and Short who define a component as "an independently deliverable set of reusable services"[7].

CBS development brings many advantages like saving lots of development time, efforts, reduces cost of development, increases productivity, quickens delivery of projects, improves quality, reusability, reliability and flexibility etc [1-2]. However, there are several disadvantages of using CBS development for instance it can increase complexities and efforts for integrating and developing reuse components, sometimes requirements are unclear and to build component, the requirements should be well defined, conflicts between usability and reusability, and component maintenance and complexity etc [1-2, 8].

CBS development is integration centric with a focus on assembling pre-existing software components that are either developed in-house or purchased off-the-shelf, to build a software system. Recent research [9] suggests that a system integrator plays an important role in the success of a CBS by putting together pieces developed by different parties who are usually unaware of each other [10]. Furthermore, individual components are usually designed for general purposes that might not satisfy all customer requirements and some of them may be unnecessary in a given system. Thus, the role of a system integrator becomes even more important because it is rarely the case that components are perfectly matched and system integration involves more than simply finding components, which, together perform the desired tasks, and connecting their interfaces [11].

The main idea of this research work is to investigate the role and importance of component documentation of a candidate component in component integration, relate the component documentation with CBS integration factors, identify the risk factors during component integration process, see how component documentation can help in avoiding such risks and developing some guidelines which can be used by any system integrator during component integration phase to avoid any risk causes system failure.

Our research is based on industrial questionnaire-based survey to better understand the role and importance of component documentation. This industrial survey helps to analyze and identify different risk factors during component integration in component-based systems. To ease the workload and streamline the data collection and validation process, we have developed a web interface to make questionnaire available to the respondents online. In the end, some statistical calculations using statistical software with some statistics tests will be performed.[12]

1.2 Problem Outline

CBS development process relies heavily on integrating individual components. System Integrators are responsible for assembling existing components, developed by different parties, to build a software system. However, such integration is still a risky process

because it's very rare that components are perfectly matched and ready for 'plug and play'.

The lack of comprehensive component documentation presents a potential risk for a system integrator during integration process. This research will help to understand the role and importance of component documentation in the CBS integration phase. It is essential to know whether the presence of available component documentation can help system integrators during component integration process. This research presents an evaluation of the impact of available component documentation, from a system integrator's perspective, on the overall success of a CBS integration process. There is a need to associate the component documentation with CBS integration factors and find out the missing gaps in available component documentation. Some of the best practices for system integrator will be picked and associated with component documentation to find out how component documentation can help to achieve these practices. To the best of our knowledge, none of the existing work investigates the impact of available component documentation for the integration phase, from the system integrator's perspective.

The other aim of this research is to identify risk factors which can occur during component integration process. It's important that we obtain information regarding negative aspects in component integration from a system integrator's point of view. We will identify risk factors which have high impact during integration process. There are

researches [7-10, 19, 23-24] made on risks in general component-based system but we hardly found research which focuses specifically component integration phase. It is essential to identify such integration risks and then find out how component documentation can help in avoiding such risks.

Furthermore, based on the results and analysis, we will come up with some guidelines or strategy which can be used by any system integrator to avoid such risk during component integration phase and ensure that they have integrated component successfully.

1.3 Research Objectives

The main objective of this Research is to investigate the role and importance of component documentation, analyze missing gaps in available component documentation, relate the component documentation with CBS integration factors, identify the risk factors during component integration process and see how component documentation can help in avoiding such risks and develop guidelines which can be used by any system integrator.

1.4 Research Motivations

This research helps to collect qualitative and quantitative data from industrial survey which will be conducted among different experienced companies in CBS. This helps to

understand the role and importance of component document from system integrator's perspective and help to identify risk factors which may occur during component integration phase. The collected data will help to derive research model which can further help in driving research questions and research hypothesis. Furthermore, the data will be analyzed and tested using different testing methods and statistical calculations to empirical evidence to support research questions and hypothesis.

1.5 Research Contributions

The contribution of our work is the industrial study to show the impact of current available component documentation during the integration phase for a CBS. We also analyze the relationship between the available component documentation and integration success factors to better understand the current industrial practices regarding use of component documentation during the integration process of a CBS. The results shows that system integrators in the industry found available component documentation useful for early evaluation of candidate components and learning new features about selected components. However, available component documentation is considered insufficient for integration effort estimation and performing integration and system testing of a CBS. Our work reinforces current perceptions about the significance of component documentation and the need for detailed component documentation standards to use commercial-off-the-

shelf components. Furthermore, we identified different risk factors which may occur during component integration phase. We investigate integration risk factors from system integrator's perspective. We will see how component documentation can help in avoiding these integration risks. We will develop some guidelines which can be used by any system integrator during component integration phase to avoid any risk or system failure.

1.6 Thesis Structure

The rest of the thesis is organized as follows. Chapter 2 presents the background of component based systems, chapter 3 presents the Literature Survey, chapter 4 presents research method use in this research, chapter 5 addresses the role of component documentation in component integration, chapter 6 is about risk factors during component integration phase, chapter 7 presents the threats to validity for the conducted industrial survey, chapter 8 presents the further discussion, analysis and the guidelines for the system integrators for implementing component in CBS and finally chapter 9 will presents the conclusion and future work.

CHAPTER 2

BACKGROUND

In this chapter, we present the background of component based system development including component based development life cycle and component integration in component based system. Furthermore, chapter describes general risk management in software development and then risk management specific to component based software development.

2.1 Component-Based Software Development

Component-based software development (CBSD) is a process of assembling components which are already developed and prepared for integration to build a software system. The

main concern of CBSD is assembly of preexisting software components into larger pieces of software. In CBSD, these software components are written and developed in such a way that they are generic enough for different systems. The main goal of CBS is to allow these parts or components of a software system to be replaced by newer components. CBS are flexible and easy to maintain due to the intended plug-and play nature of components. The COM+ from Microsoft and Enterprise JavaBeans (EJB) from Sun are few examples of components for software construction [13-14].

The main goal of this technology is independent deployment and assembly of components. This concept has been taken from manufacturing industry and civil engineering field like for instance; manufacturing of vehicles would have not been successful if their spare parts were not available. Software companies have used the same concept for developing software in parts. These software parts are developed by different companies who market and ship them in different forms. These software parts are components which later plug into any application. It's an independent part of the system having complete functionalities for a particular purpose. The major advantages of component-based software development are reusability, interpretability, upgradability, less complexity, time effectiveness, cost effectiveness, efficiency, reliability and improved quality [15].

CBS comprises of two parts: Component engineering and application engineering. Component engineering is concerned with the analysis of domains and development of generic domain-specific components whereas, application engineering process constructs application using off-the-shelf software components that is developed with reused components [16]. These components can be developed by different developers using different languages and different platforms. Figure 1 shows components can be taken from any component repository and integrated into a target software system [17].

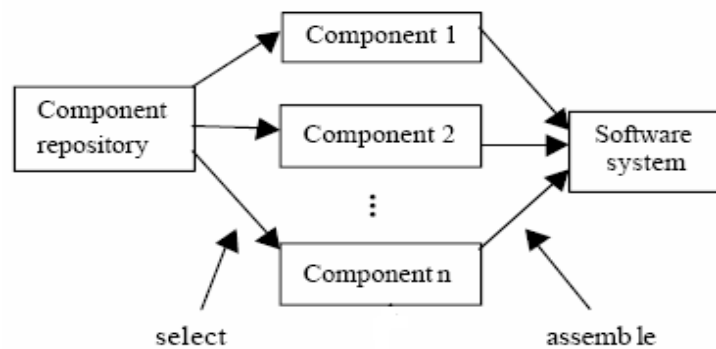


Figure 1: Component-Based Software Development

Figure 2 shows overview of the CBS development life cycle. The CBS development approach can be divided into three main phases, namely, selection, integration and maintenance phases. The first phase, component selection, starts with a process to identify suitable components with a potential to match stakeholder demands. After this process, candidate components are analyzed to compare and select suitable components

based on evaluation criteria. The second phase, component integration, focuses on adapting and assembling selected components through an architectural infrastructure. The third phase, component maintenance phase, handles continuous evolution of a CBS during its life cycle.

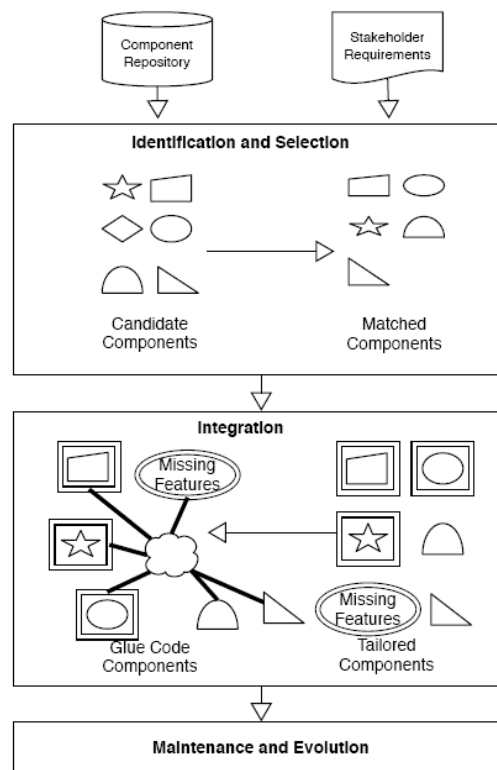


Figure 2: CBS Development Life Cycle

A typical CBS development process consists of seven phases as follows [11].

1) *Requirements Analysis and Definition*: The analysis activity involves identifying and describing the requirements to be satisfied by the system. In this activity, the system boundaries should be defined and clearly specified. Using a component-based approach, an analysis will also include specifications of the components that are to collaborate to provide the system functionality. To be able to do this, the domain or system architecture that will permit component collaboration must be defined. In CBS development, the analysis is an activity with three tasks. The first task is the capture of the system requirements and the definition of the system boundaries. The second task is the definition of the system architecture to permit component collaboration, and the third task is the definition of component requirements to permit the selection or development of the required components.

2) *Selection and Evaluation*: To perform a search for suitable components and make their identification possible, the components must be specified, preferably in a standardized manner. Again, this may often not be the case. The component specifications will include precisely defined functional interfaces, while other attributes will be specified informally and imprecisely (no method is developed for this) if specified at all. The components selected must therefore be evaluated. The process of evaluation will include several aspects of both a technical and nontechnical nature. Technical aspects of evaluation include integration, validation, and verification. Examples of nontechnical issues include

the marketing position of the component supplier, maintenance support provided, and alternative solutions.

3) *System Design*: System design activity typically begins with the system specification and the definition of the system architecture and continues from there. In traditional development, the design of the system architecture is the result of the system requirements, and the design process continues with a set of sequences of refinements (for example, iterations) from the initial assumptions to the final design goal. In contrast with traditional development, many decisions related to the system design will be a consequence of the component model selected.

4) *System Implementation*: In an ideal CBS development process, the implementation by coding will be reduced to the creation of the “glue code” and to component adaptation. Also, it may still be necessary to design and implement some components- those that are business critical or unique to a specific solution and those that require refinement to fit into a given solution.

5) *System Integration*: Integration is the composition of the implemented and selected components to constitute the software system. The integration process should not require great resources, because it is based on the system architecture and the use of deployment standards defined by the component framework and by the communication standard for

component collaboration. Moreover, one of the characteristics of many component-based systems is the ability to dynamically integrate components without interrupting system execution. This means that the integration activity in CBS development is present in several phases of the component-based system life cycle.

6) *Verification and Validation*: This last step before system delivery is similar to the corresponding procedures in a traditional development process. The system must be verified and validated. These terms can be easily confused although there is a clear distinction between them. Verification is a process that determines whether the system meets its specified functional and nonfunctional requirements (i.e. are we building the product right?). A validation process should ensure that the system meets customer expectations (i.e. Are we building the right product?).

7) *System Operation Support and Maintenance*: The purpose of the operational support and maintenance of component-based systems is the same as that of monolithic, non-component-based systems, but the procedures might be different. One characteristic of component-based systems is the existence of components even at run time, which makes it possible to improve and maintain the system by updating components or by adding new components to the system. This makes faster and more flexible improvement possible- it is no longer necessary to rebuild a system to improve it. In a developed component market it also gives end users the opportunity to select components from different

vendors. On the other hand, maintenance procedures can be more complicated, because it is not necessarily clear who is supporting the system-the system vendor or the component vendors.

2.2 Component Integration

The component integration in any component-based development is a very crucial and vital element. It's a process of assembling components together. It's basically a system integrator or application assembler who is responsible for taking care of this critical phase. The integration phase proceeds by integrating components, where each component satisfies some of the requirements of the intended system. These components are packaged in many different forms. The selected components are integrated through well-defined infrastructure and this infrastructure provides the binding that forms a system from the disparate set of selected components [10-11, 18]. The important issue when integrating components is to deal with the mismatches that occur when putting together software components developed by different parties [10]. Thus, it is important that component services are provided through a standard, published interface to ensure interoperability [1, 19]. An interface determines how a component can be used and interconnected with other components. Filters are perhaps the oldest component integration mechanism, only providing access to the data of the components without

considering their functionality. Application programming interfaces is another mechanism that provides individual external components with complete access to all data, functions and events. The concept of a shared data repository has also been discussed as a mechanism of component integration [20]. This method is based on the idea that multiple components share a common data repository, reading and writing the same data object.

The component integration process defines how well components plug and play. The system integrator must also follow some processes to implement these components into existing software system. The off-the-shelf (OTS) components such as commercial off-the-shelf (COTS) or open source software (OSS) assure to reduce development time and cost but it also complicates composition part which is handled by system integrators. System integrators have to be extra careful in integrating any component because any error during this process could cause the failure of the entire application. Another reason for failure is that the integrator doesn't know the functionality or specifications the component [21].

There are different industrial practices for system integrator in any OTS based development. These practices help to understand that how system integrator manages the processes and knowledge to ensure that they have integrated component successfully. There are different facts and role of a system integrator in any component-based

development. They are highlighted as different activities from system integrator perspective in Figure 3 [9].

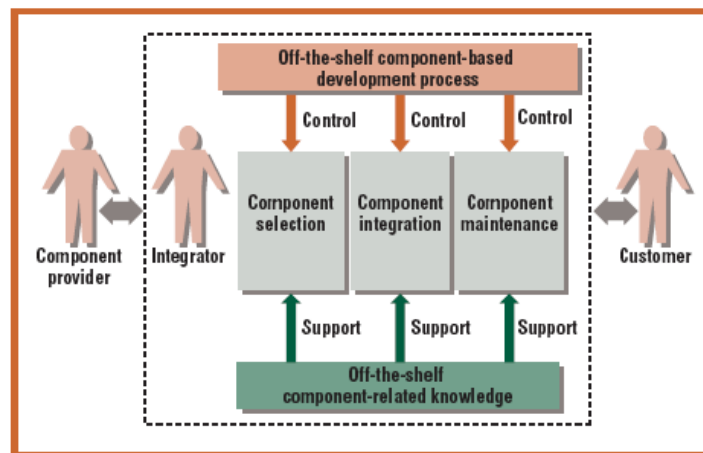


Figure 3: Development with off-the-shelf components: actors and activities

2.3 Risk Management in Software Development

Risk is defined as “the possibility of suffering harm or loss; danger.” [22] Unlike the hazards of daily living, the risks in the emerging field of software engineering must often be learned without the benefit of lifelong exposure. A more deliberate approach is required. Such an approach may involve studying the experiences of successful project

managers as well as academic research in the area. In [23] the following top 10 software risk items are listed:

- Personnel shortfalls
- Unrealistic schedules and budgets
- Developing the wrong functions and properties
- Developing the wrong user interface
- Gold-plating
- Continuing stream of requirements changes
- Shortfalls in externally furnished components
- Shortfalls in externally performed tasks
- Real-time performance shortfalls
- Straining computer-science capabilities

Software engineering involves risks that may harm a software project. Risk concerns the future of the project. Risks also involve changes and choices which have to be made in the process. In order to understand and prepare proactively to avoid or manage the risks, risk management is important [24].

Software risk management is a crucial part of successful project management. Software risk management, in general, involves two major steps: risk assessment and risk control, which are composed of various phases each. Risk assessment may constitute risk identification, risk analysis, and risk prioritization. One way to do risk identification could be to list all project specific risks in the form of a check list. A thorough analysis of the identified risks is followed in the second stage of risk assessment. It may include performance models, cost models, network analysis, and statistical decision analysis. The process also focuses on effects upon different quality attributes of the software project e.g. flexibility, reliability, availability and security. The final phase of risk assessment is to prioritize the identified and analyzed risk items using different techniques like risk exposure analysis, risk reduction leverage analysis, etc. Once risk assessment is done, the risk management process continues with its next step of risk control. Risk control includes risk management planning, risk resolution and risk monitoring. Risk management planning helps you in preparing to address each risk item. Risk resolution produces a situation in which risk items are eliminated or resolved and risk monitoring involves tracking the project's progress towards resolving its risk items and taking corrective actions where appropriate [21-23]. All these activities are shown in Figure 4.

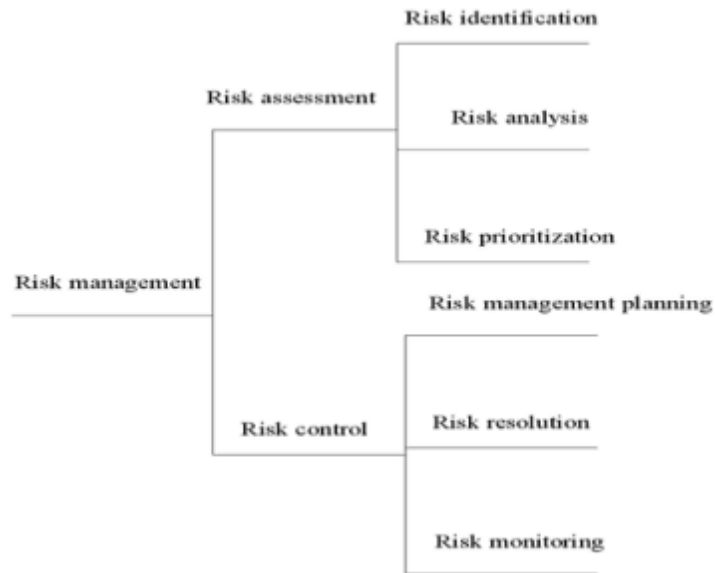


Figure 4: Risk Management Framework

2.4 Risk Management in Component-Based Software Development.

CBS development is a complex and risk-prone [25] process which needs careful risk assessment on behalf of a system integrator, to help achieve potential benefits of reduced time to market, increased productivity and the development of a quality system [11].

A large number of software projects are delivered over budget, late or with fewer features originally specified in the scope of the project. There is no doubt that the field of software engineering is fraught with software projects that have been unsuccessful or mismanaged.

Identifying and resolving the factors that cause these problems earlier in the life cycle can reduce cost and help to prevent software disasters. The ability to reliably predict the risk for a software project presents a significant advantage for a development team. It provides an opportunity to address high risk components earlier in the development life cycle, when their impact is minimized. Insights into risk management, development team skill, software process maturity, and software problem complexity are hypothesized as driving factors of software product quality [26].

Component-based development carries significant risks throughout the system life cycle. These risks are related to the nature of OTS software, the development process, component technologies and vendor support etc. Risk in component-based development can be determined by the quality of OTS software or how the given component performs in the current system. There are different factors which cause risk such as: the black box nature of the OTS software, lack of information about the source of component, the quality of OTS software, the lack of component interoperability standards, the disparity in the customer vendor relationship. These risks can be technology risk or business risk. Business risks are those risks which are associated with events that may result in loss of business through failure to deliver a system on time or within the budget. Technology risks are associated to specific technology used to build the system [16, 27].

CHAPTER 3

LITERATURE SURVEY

In this chapter, related work in area of component based development, component integration, component documentation and risk factors during component integration phase are presented.

3.1 Component Based Development

Component-based development (CBSD) has been a key research area in software engineering due to its promises to accelerate software development and to cut

development cost by assembling systems from already built-in components [28]. Component composition requires different expertise in system integrator in order to ensure the successful component integration in any component based system. System integrators are the one who are responsible for integrating components which are packaged in many different forms like functional libraries, frameworks or legacy application. The software development with off-the-shelf (OTS) component requires different practices for system integrator regarding selection and integration of a component [9, 16].

3.2 Component Integration

Rine et al. [29] proposed the use of adapters to integrate components. In this technique, each component has an associated adapter. Components request services from each other through their associated adapters. The associated adapter is responsible for solving the syntactic interface mismatch. The individual adapters communicate with each other to fulfill component interaction. Dietrich et al. [30] used active rules to design and generate wrappers to adapt components. The wrappers are automatically generated as enterprise java bean components and they act as proxy objects. These proxy objects intercept method calls and provide the functionality required by the overall component based system. Parrish et al. [31] proposed a formal model for identifying conditions under which various component deployment strategies are safe and successful. The framework

identifies two types of installation; (i) successful, and (ii) safe. Successful installations are the ones in which deployed applications work properly, while safe installations are the ones in which no existing applications are damaged by the installation. It then identifies conditions that are sufficient to guarantee both safe and successful installations. The framework at the current stage is mainly theoretical and needs to be applied to real life applications in order to provide a set of guidelines for software engineers to follow when building performing installations.

Xia et al. [32] proposed the Component Integration Model (CIMO) that facilitates the integration of components by presenting a component layer and a service layer. The component layer contains CIMO components which are based on Microsoft COM objects and the service layer which consists of CIMO configuration, manager, container and system. It is mainly responsible for supporting the management, communication and configuration of components. The CIMO platform provides the support for the setup and integration of the components in the CIMO application. CIMO platform has one CIMO system to coordinate the whole system and has a number of CIMO containers, which directly contains the CIMO components. Similarly, Depke et al. [33] used the concept of ports that enable the flexible and consistent integration of software components based on XML documents. These techniques help reduce CBS development process risks by considering component specifications and their interface properties, and increase flexibility by providing quantitative analysis of candidate components.

The concepts of aspect-oriented software development have also been incorporated into the CBS integration process. For example, Assman [34] presents the concept of invasive composition and uses self generated glue-codes to integrate components. Similarly, Suvee et al. [35] present the JAsCo language for CBS integration. The language is designed to be used with the Java Beans component model and introduces concepts of aspect beans and connectors.

3.3 Component Documentation

Jingyue [6, 36] presented the understanding of reuse components and shows that most developers are not satisfied with component documentation and they may get necessary information about the component through some other communication channels. One of his research areas in this thesis revolve around difficulty of component documentation and component knowledge management with reuse level of components. In his preliminary study he tried to indentified problems with the documentation. His result shows that around eighty six percent respondents believe that insufficient documentation is a problem. It means that documentation could be incomplete, or not updated or difficult to understand. He also present that component users acquire sufficient information about relevant component with different sources which shows the importance of having documentation or information about a component. His spearman test results shows that

developer will be less satisfied with component documentation with higher reuse levels. His research also presents that current component documentation technologies cannot describe all the required information such as performance, reliability, security. He emphasize in his conclusions that when companies move from lower to higher reuse level, more efforts should be spend on the component documentation and component knowledge management. He found that most of the developers are not satisfied with the component documentation and recommended the informal communication channels should be give more attention where component users can get necessary information and can share knowledge and their experience. Based on this research, we further investigate the association of component and component documentation in detail level instead of focusing components on general level. His research emphasizes the significance of having information about the component. We identify different component factors and relate each one of them with component documentation in order to show the importance of component documentation and knowledge. It helped us to identify the relationship and role of component documentation with each one of these factors.

The authors in [9] present different industrial practices of off-the-shelf (OTS) based development. These practices help to understand how the system integrator manages processes and knowledge to ensure that they have integrated component successfully. To achieve this goal, integrator must manage relationship with component provider, must be able to debug, detect and fix problems in a component and evaluate the component early;

they must be able to estimate correct efforts of a component for integration, ability to learn component features and modifications of a component. Authors have presented different facts and roles of a system integrator in any component-based development. They highlighted different activities from system integrator perspective. Based on this research, we picked some of the best practices for system integrator and associated them with component documentation to find out how component documentation can help to achieve these practices. System Integrators [9] usually don't care enough about technical details in the early stages, which could lead to implementation and integration problems. An OTS component's providers could release a new version so the system integrators would need to re-evaluate the component and redesign the system. In agile development processes, system integrators typically identify and document requirements using user stories and they set up the entire process such that they can easily make changes. In Design or the development iterations phase, integrator should think about how much effort is needed and how the refactoring of the system is possible.

Empirical investigation of CBS is still at an early stage. Jingyue et al [37] presented an empirical study evaluating the variations in CBS development processes. Their results indicated that CBS have two key activities: 'the build vs the buy decision' and 'component selection'. Further, components are usually selected based on system analyst experience and using 'hand-on trails'. Jingyue et al. [21] performed an industrial survey to analyze CBS common problems, most common risk reduction activities performed in

the industry and how successful they were in avoiding these risks. They identified (1) component understanding, (2) component quality evaluation, (3) component vendor reputation, (4) limiting the number of components in a project and (6) sharing company's knowledge as key risk-reduction activities to ensure the success of a CBS. Furthermore, they identify integration effort estimation and costly fault identification as two key challenges during CBS development. Recently Jingyue et al. [9] presented a list of ten facts about current CBS practices in the industry. They have observed a number of differences between academic theories and industrial practices in different areas of a CBS development life cycle ranging from adaptation of traditional development processes to a lack of use of formal component selection methods.

3.4 Component Integration Risks

Rashid and Kotonya [27] also highlighted six different categories for different phases in component-based development where one of them is integration. They highlighted the importance of a good understanding of a component for integration and deployment. They argue that for successful integration, a component should have adequate documentation, usage history, version details and test reports. Furthermore, lack of documentation standards and quality review procedures are serious risks for the integration process of a CBS. They also explain that one of the integration concerns is estimating schedules and resource requirements for a component before integration

process and also give importance to early evaluation of a component. Based on this research, we picked the addressed factors during integration phase and linked them with the component documentation to show how component documentation can help in solving and avoiding such factors.

According to Kotonya and Rashid [16], there are several risks in each phase of development. The composition or integration stage can encounter problems so the system integrator must be aware and overcome all these problems. Like for example few common problems they present is lack of information about the source component, the design of a component is unknown, the disparity in the customer vendor evolution cycle, components are packaged and delivered in many different forms like for example functional libraries, off-the-shelf application, frameworks etc. This may cause major difficulties for system integrator during component integration process. Most COTS software are not generally “plug and play”, significant effort is required to build wrapper classes or glue code between them in order to integrate the component in current situation. These wrappers have to be maintained over the period. Lack of interoperability standards could cause problems during component integration. Also COTS may offer similar functions but may have different system resource requirements that is memory and processor requirements; in this case this may affect the system operation.

According to Padmal [38], risk is involved with different stakeholders in component-based systems. The common risks which are associated with Application assembler or system integrator are requirement conformance, relationship with vendor, ownership and licensing, certified component, quality of component etc. Integrator deals with lack of visibility into the component-based development process. They highly rely on external developers and this place a significant risk due to limited control they have over the selected component. Also authors in [37] surveyed different risks using OTS components including efforts estimation for integration the component in component integration phase.

The summary of the related work presented in this section has exposed some component-based system key factors such as correct effort estimation for integrating component, maintaining relationship with component provider, learning new features of component and modification of a component, debugging and fixing bugs and testing in component based system and early evaluation of a component. It also presents the importance of knowledge and documentation of a component. We will use these key component-based system activities as a set of variables in the industrial investigation in order to construct the research model of our investigation. This related work also presents some key risk factors which can take place during component integration process, we will use these risk key factors and will correlate them with component documentation and construct the risk research model of component integration for our investigation.

CHAPTER 4

RESEARCH METHOD

This chapter explains the data collection method used in this research and methods for analyze the collected data and results.

4.1 Data Collection

The questionnaire-based survey was performed to better understand the role and importance of component documentation and to indentify the component integration risks. It was anonymous survey which helped us to analyze and identify different factors related to component-based systems and their association with component documentation. To ease the workload and streamline the data collection and validation

process, we enabled a web interface to make questionnaire available to the respondents online.

The survey was performed using a variant of snowball sampling [39], a technique where key practitioners in organizations serve as contact points for the study. The contact points were emailed the link for the web-based survey, which they can forward on to other potential respondents within their organizations. The contact points also reported the total number of respondents from each organization and functioned as a temporary checkpoint for the number of completed questionnaires.

Software developers with experience of using software components (both in-house and off-the-shelf Components) for more than three years were the target participants for our study. The participants belong to small to medium-sized companies from Australia, Pakistan, Saudi Arabia, United Arab Emirates and the United Kingdom. These companies provide a wide range of services such as software consultancy and off-shore software development. The some of the respondents belongs to well known companies like STC, SABIC, Ejada, Logica, Al-Falak, EMC, SunGard US, TATA, Wipro, CSC Aus, Unisys Aus, Si3. All the participants have either an undergraduate or a graduate degree in computer science or related fields. Furthermore, the participants' role in the organizations ranged from software developers to software architects with 5 - 7 years of experience in CBS. In total, 110 participants were contacted and 53 participants completed the survey.

The questionnaire had five major parts. The first part was introductory and general part to get any idea about participants and about participant's organization and participant experience in component base systems. The second part provided the key concepts of component integration as a component integrator and helps in understanding the importance of component documentation. This section includes some close-ended questions which are based on values (5-1) for adequate and inadequate rating where 5 represent maximum for adequate. Some of the outcomes are based on often and rare values using same adequate/inadequate scale. The third part was aimed in collecting and identifying risks factors during component integration process. The participants were asked to indicate the extent of their agreement or disagreement with statements using a five point scale that ranged from 'strongly agree' (5) to 'strongly disagree' (1) for each integration factor. Fourth part contained questions related to avoid and mitigate risks in component integration based on same scale which used in section three. The last part was aimed to collect more detailed information about component integration from system integrator point of view and provided an opportunity to participants to share their experience of CBS development; and discuss the strengths and challenges of component integration during a CBS development life cycle.

4.2 Data Analysis

A number of statistics analysis techniques are used to analyze the data collected during the study to validate each of the hypotheses which will be presented in next few sections. By analogy with Faheem and Capretz [40] [41], we have used both parametric (Pearson correlation) and nonparametric (Spearman correlation) statistical approaches. Since one technique is distribution dependent and other is not so these approaches helps to ensure the reliability of the results. We also used Partial Least Square (PLS) technique to cross validate the statistical outcomes [12] and increase the reliability of the results. These statistical calculations were performed using Minitab 14 software & Vassar correlation calculator [42].

A Correlation is a number between -1 and +1 that measures the degree of association between two variables e.g. X and Y. A positive value for the correlation implies a positive association which means large values of X tends to be associated with large values of Y and small values of X tend to be associated with small values of Y. A negative value for the correlation implies a negative or inverse associated which means large values of X tend to be associated with small values of Y and vice versa [43-44].

A hypothesis test is statistical procedure that is designed to test a claim. The p-value in statistical test measures how likely it was that you would have gotten your sample results if the null hypothesis were true. The smaller p-value will give more evidence against the null hypothesis to be true. All p-values probabilities between 0 and 1 [43] . P-value generally less than 0.05, you reject the null hypothesis and accept the alternate whereas value great then 0.05 allows you to reject alternate hypothesis and accept null hypothesis which means you don't have enough evidence to reject null hypothesis [43].

The cutoff points and resulting decision vary from researcher to researcher. In our research we use cutoff value of p-value is 0.05. This value will help us to justify or give confidence on our results. If p-value is less than 0.05 then it means we have confident to accept alternate hypothesis whereas if p-value is greater than or equal to 0.05 then it give us confidence to reject alternate and accept null hypothesis. The most statisticians like to see the correlation between +/- 0.6 or +/- 0.4 or +/- 0.8 [12, 43-45] but for coefficient above +/- 0.9 or +/-1 correlation against the real data can't be practical. In our research, we use following range for statistical correlation coefficient values. [44]

Table 1 : Correlation Coefficient Range

Range	Association
-1.0 to -0.7	Strong negative association.
-0.7 to -0.3	Weak negative association.
-0.3 to +0.3	Little or no association
+0.3 to +0.7	Weak positive association
+0.7 to +1.0	Strong positive association

CHAPTER 5

THE ROLE OF COMPONENT DOCUMENTATION IN COMPONENT INTEGRATION

This chapter briefly describes about role & importance of component documentation in component integration phase and presents the identified research questions, research hypothesis and theoretical research model. Furthermore, the chapter presents the data and results analysis along some statistical calculations. The results discussion is presented at the end of this chapter.

5.1 Introduction

Component documentation plays an vital role in the success of a CBS as it is the main source of information which is used to balance the conflicting interests between what is needed and what is available [46-47]. Cechich et al. [48] highlight that the standards on component documentation needs to be reinforced as information available at component repositories are usually unstructured and presented in the form of marketing brochures and natural language description. The good quality documentation facilitates communication between a component's creator and its users.

We identify that available component documentation usually consists of a list of features, reviewer comments, and price and, in some cases, trail versions. In addition to information about component interfaces, the integration process also needs information about component usage history, version control, test data and relevant quality attributes [27]. However, detailed component documentation is usually unavailable in the majority of component repositories. This lack of detailed component documentation introduces new challenges for system integrators as it increases ambiguity in the integration phase of a CBS.

In this chapter, we present an evaluation of the impact of available component documentation, from a system integrator's perspective, on the overall success of a CBS

integration process. The motivation of our work is to better understand how available component documentation is helping or hindering CBS practitioners in the integration process of a CBS. We analyze the relationship of available component documentation [6, 36] with five key integration success factors namely, integration effort estimation, early component evaluation, new features and modification information, integration testing and relationship with component vendors [5], [9], [24], [25], [26]. We focus only on these five integration success factors as they correspond to key information required by system integrators during the integration process of a CBS.

To date, empirical research work on CBS has been focused on identifying risks associated with component identification, selection and maintenance processes. However, to the best of our knowledge, none of the existing work investigates the impact and role of available component documentation for the integration phase, from the system integrator's perspective, of the CBS development life cycle.

5.2 Research Questions and Hypotheses

The objective of our study is to assess whether available component documentation is sufficient for integration process of a CBS development life cycle. We are also interested in investigating how available component documentation helps system integrator in

component integration for a CBS. In order to perform such analysis, we also need to understand the relationship between key integration success factors [46], [2], [38], [21], [9] and available component documentation. After an indicative literature survey of CBS integration techniques, we identify integration effort estimation, early component evaluation, learning new and modified component features, and integration testing and relationship with component provider as integration success factors. Hence based on the analysis we come up with following research questions. Figure 5 shows the theoretical model designed for this purpose. The main objective of this model is to answer following research questions.

RQ1: *Is it important to have essential information about the candidate component before integration process?*

RQ2: *Does the component documentation have an impact on different component based system factors?*

The hypotheses for assessing the effect of available component documentation on the key integration success factors are as follows:

HN1: There is no correlation between “available component documentation” and “integration effort estimation”.

HA1: There is a correlation between “available component documentation” and “integration effort estimation”.

HN2: There is no correlation between “available component documentation” and “early component evaluation”.

HA2: There is a correlation between “available component documentation” and “early component evaluation”.

HN3: There is no correlation between “available component documentation” and “new feature and modification analysis”.

HA3: There is a correlation between “available component documentation” and “new feature and modification analysis”.

HN4: There is no correlation between “available component documentation” and “integration testing”.

HA4: There is a correlation between “available component documentation” and “integration testing”.

HN5: There is no correlation between “available component documentation” and “relationship with component provider”.

HA5: There is a correlation between “available component documentation” and “relationship with component provider”.

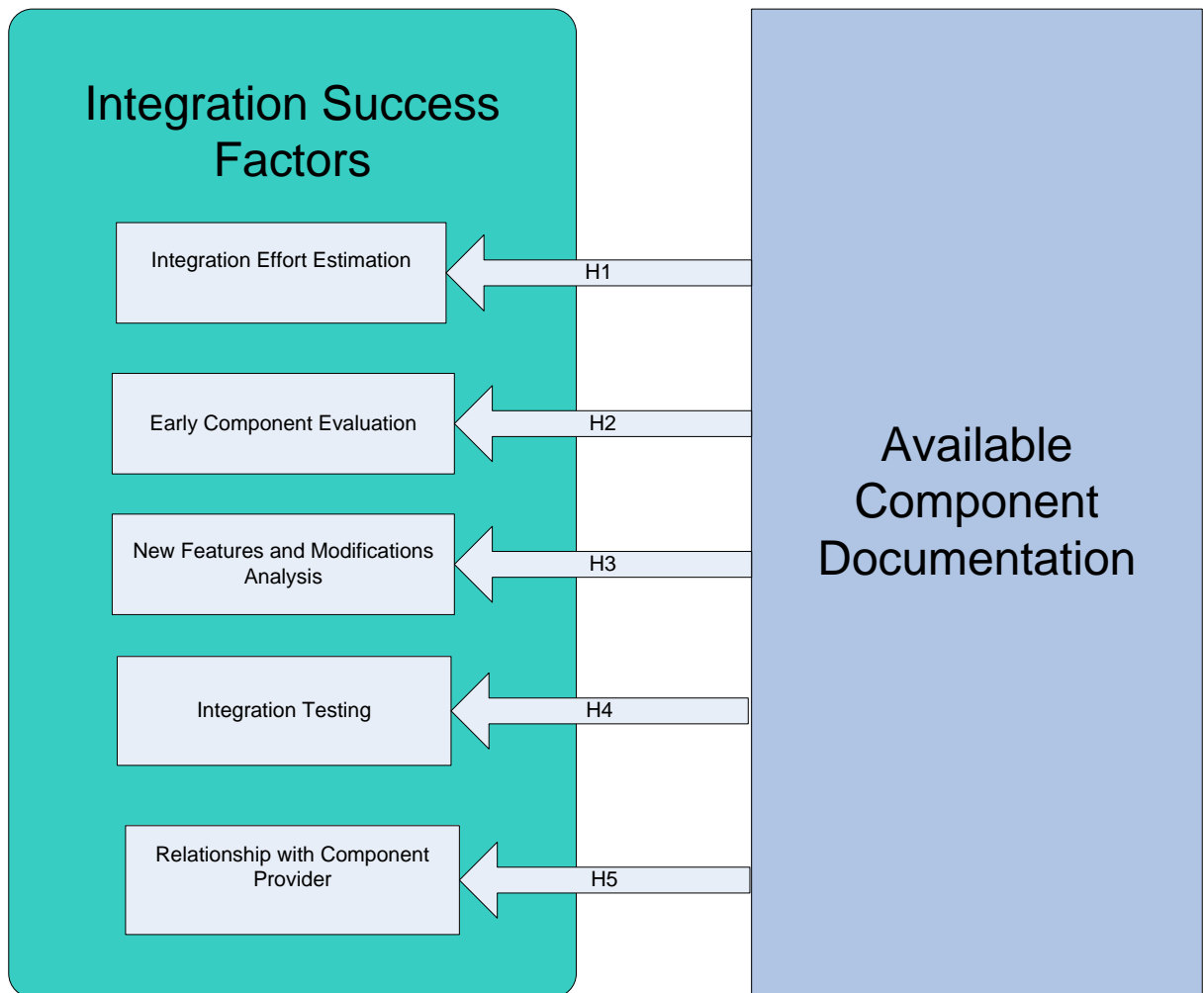


Figure 5: Theoretical Research Model

5.3 Data and Result Analysis

In this section, we will present our results in tabular form along with description. The Pearson correlation, Spearman correlation Partial Least Square (PLS) techniques are used to analyze the collected data of study to validate each of the hypotheses which have present in earlier section [12]. These statistical calculations were performed using Minitab 14 software.

5.3.1 Hypotheses testing phase-I

In order to test hypotheses H1-H5, we examined the Pearson correlation coefficient [12] between available component documentation and other factors presented in research model shown in Figure 5. The result of the statistical calculations for the Pearson correlation coefficient is reported in Table 2. The Pearson correlation coefficient between “available component documentation” and “integration efforts estimation” was -0.20 (little or no association) at $P < 0.05$ provided a justification to reject alternate hypothesis HA1 & accept null hypothesis HN1. The alternate hypothesis HA2 is accepted based on Pearson correlation coefficient 0.45 (weak positive association) at $P < 0.05$ between “available component documentation” and “early component evaluation”. The correlation coefficient of 0.41 (weak positive association) at $P < 0.05$ was observed between

“available component documentation” & “new features and modification analysis” and thus provided a justification to accept alternate hypothesis HA3. The Pearson correlation coefficient between “available component documentation” and “integration testing” was -0.20 (little or no association) at $P < 0.05$ provided a justification to reject alternate hypothesis HA4 & accept null hypothesis HN4. The correlation coefficient of 0.21 (little or no association) at $P < 0.05$ was observed between “available component documentation” & “relationship with component provider” and thus provided a justification to accept null hypothesis HN5.

Hence it was observed and is reported here that null hypothesis for H1, H4, H5 are accepted but their alternate hypothesis rejected whereas alternate for hypothesis H2, H3 are found significant and they have weak positive association.

Table 2: Hypotheses testing using Pearson Correlation Coefficient

H	Variable 1	Variable 2	Coefficient	p-value
H1	Available Component Documentation	Integration Effort Estimation	-0.20	0.156
H2	Available Component Documentation	Early Component Evaluation	0.45	0.001
H3	Available Component Documentation	New Feature and Modification	0.41	0.002
H4	Available Component Documentation	Integration Testing	-0.20	0.143
H5	Available Component Documentation	Relationship with Component Provider	0.21	0.133

5.3.2 Hypotheses testing phase-II

In phase-II we conducted non-parametric statistics using Spearman correlation coefficient [12] to test the hypotheses H1-H5. Table 3 also reported the observation made in this testing phase. The Spearman correlation coefficient between “available component documentation” and “integration efforts estimation” was -0.22 (little or no association) at

$P < 0.05$ provided a justification to reject alternate hypothesis HA1 & accept null hypothesis HN1. The alternate hypothesis HA2 accepted based on Spearman correlation coefficient 0.46 (weak positive association) at $P < 0.05$ between “available component documentation” and “early component evaluation”. The correlation coefficient of 0.37 (weak positive association) at $P < 0.05$ was observed between “available component documentation” & “new features and modification analysis” and thus provided a justification to accept alternate hypothesis HA3. The Spearman correlation coefficient between “available component documentation” and “integration testing” was -0.20 (little or no association) at $P < 0.05$ provided a justification to reject alternate hypothesis HA4 & accept null hypothesis HN4. The correlation coefficient of 0.20 (little or no association) at $P < 0.05$ was observed between “available component documentation” & “relationship with component provider” and thus provided a justification to accept null hypothesis HN5.

Hence it was observed and is reported here that null hypothesis for H1, H4, H5 are accepted but their alternate hypothesis rejected whereas alternate for hypothesis H2, H3 are found significant and they have weak positive association.

Table 3: Hypotheses testing using Spearman rank-order correlation coefficient

H	Variable 1	Variable 2	Coefficient	p-value
H1	Available Component Documentation	Integration Effort Estimation	-0.22	0.061
H2	Available Component Documentation	Early Component Evaluation	0.46	0.000
H3	Available Component Documentation	New Feature and Modification	0.37	0.003
H4	Available Component Documentation	Integration Testing	-0.20	0.078
H5	Available Component Documentation	Relationship with Component Provider	0.20	0.078

5.3.3 Hypotheses testing phase-III

In phase-III of hypotheses testing, we used Partial least square regression (PLS) technique [12] to overcome some of the associated limitations and to cross validate with the results observed using approach of Phase-I and Phase-II. Table 4 reports the result of PLS tests of the hypothesis with detailed observed values of coefficient and p-value

whereas Table 5 shows the cross validation of results. We tested the hypothesized relationships that are H1-H5 by examining their directions and significance. The PLS coefficient between “available component documentation” and “integration efforts estimation” was -0.20 (little or no association) at $P < 0.05$ provided a justification to reject alternate hypothesis HA1 & accept null hypothesis HN1. The alternate hypothesis HA2 accepted based on PLS coefficient 0.45 (weak positive association) at $P < 0.05$ between “available component documentation” and “early component evaluation”. The PLS coefficient of 0.41 (weak positive association) at $P < 0.05$ was observed between “available component documentation” & “new features and modification analysis” and thus provided a justification to accept alternate hypothesis HA3. The PLS coefficient between “available component documentation” and “integration testing” was -0.20 (little or no association) at $P < 0.05$ provided a justification to reject alternate hypothesis HA4 & accept null hypothesis HN4. The PLS coefficient of 0.21 (little or no association) at $P < 0.05$ was observed between “available component documentation” & “relationship with component provider” and thus provided a justification to accept null hypothesis HN5.

Hence null hypothesis for H1, H4, H5 are accepted but their alternate hypothesis rejected whereas alternate for hypothesis H2, H3 are accepted and they have weak positive association.

Table 4: Hypotheses testing using Partial Least Square Regression (PLS)

H	Variable 1	Variable 2	Coefficient	p-value
H1	Available Component Documentation	Integration Effort Estimation	-0.20	0.156
H2	Available Component Documentation	Early Component Evaluation	0.45	0.001
H3	Available Component Documentation	New Feature and Modification Analysis	0.41	0.002
H4	Available Component Documentation	Integration Testing	-0.20	0.143
H5	Available Component Documentation	Relationship with Component Provider	0.21	0.133

Table 5: Comparison of results from different tests

H	Pearson Correlation		Spearman Correlation		Partial Least Square	
H	Coefficient	p-value	Coefficient	p-value	Coefficient	p-value
H1	-0.20	0.156	-0.22	0.061	-0.20	0.156
H2	0.45	0.001	0.46	0.000	0.45	0.001
H3	0.41	0.002	0.37	0.003	0.41	0.002
H4	-0.20	0.143	-0.20	0.078	-0.20	0.143
H5	0.21	0.133	0.20	0.078	0.21	0.133

5.4 Discussion

From the statistical results, we make the following useful observations.

Integration Effort Estimation: The correct integration effort estimation is crucial for delivering a CBS on time and within the allocated budget. The coefficient in results shows that there is no or little association between available component documentation and integration effort estimations of a CBS and we have accepted null hypothesis. The system integrators participating in the study indicate that available component

documentation does not provide enough technical details of components and integration efforts estimates are done based on individual experiences and trails of the candidate components. Further, we believe that component documentation also needs to include data about component evolution and details of the changes.

Early Component Evaluation: The success of CBS [49], [50] [51] depends on the ability to select suitable components. Inappropriate component selection can lead to adverse effects, such as introducing extra cost, in integration and maintenance phases [50]. It is evident from our study that available component documentation helps in early evaluation of candidate components and it has weak positive association. The information about component features, version history and price helps a system integrator to analyze candidate components against system requirements and architectural constraints of the CBS-to-be. The feedback from the subjects also indicated the need for comprehensive documentation as it will help overcome glue-code and testing challenges later in the CBS development life cycle.

Learning New Features & modification: As coefficient value in our study shows there is a weak positive association between available component documentation; and learning new and updated feature details in new versions of components. This is due to the fact that component vendors are usually good at advertising new features and modifications in the new versions of their respected components. This helps system integrators in

assessing the changes needed at the architectural and integration code levels and take necessary steps to overcome compatibility risks.

Integration Testing: The coefficient in our result shows that there is no or little association between available component documentation and integration. In this association we are getting negative value which means that more testing information could cause problems in integration testing but due to value of confidence we have accepted null hypothesis and rejected the alternate hypothesis. Furthermore, we found from open data analysis that integration testing information is essential to have it and it will help during testing process. A component usually goes through traditional software testing at the developer's site [52]. However, the details of these individual component tests are rarely made available to the system integrator. Furthermore, the heterogeneous nature of components and deployment architectures introduce complexities in the integration phase of a CBS. Thus, there is a need to provide individual component testing details to assist the integration testing process of a CBS.

Relationship with component provider: The coefficient in results indicates that there is no or little association between available component documentation and maintaining relationship with component provider but due to p-value confidence we have accepted null hypothesis. This indicates that some of the key factors for system integrators such as details of technical support provided by vendors, customer reviews, component volatility

and personal contact details are usually lacking in the majority of current component documentation.

Other Integration Factors: The study evaluates five key integration success factors. There is a need to study the industrial common practices about handling requirements volatility, architectural mismatches, and lack of component support in the maintenance phase of a CBS. Furthermore, we need to analyze the importance of component certification and component knowledge for different phases of a CBS.

CHAPTER 6

THE RISK FACTORS DURING COMPONENT

INTEGRATION

This chapter presents risk factors in component integration phase and presents the identified research questions, research hypothesis and theoretical research model. Furthermore, the detail results discussion is presented at the end of this chapter.

6.1 Introduction

CBS development is a complex and risk-prone [40] process which needs careful risk assessment on behalf of a system integrator, to help achieve potential benefits of reduced time to market, increased productivity and the development of a quality system [12]. It doesn't matter which tools, techniques, and methodologies are used for component based development, it remains risk-prone process. Kotonya et al. [21] identify lack of source code and unknown design information; and disparity in component evolution cycles as the key risks for the integration phase of a CBS. An integration fault can be the result of incorrect understanding of a component or it may lie in one of the externally acquired components [26]. Similarly, Rashid et al. [34] highlight the importance of a good understanding of a component for integration and deployment. They argue that for successful integration, a component should have adequate documentation, usage history, version details and test reports. Furthermore, lack of documentation standards and quality review procedures are serious risks for the integration process of a CBS.

There are a number of risks and challenges associated with component Integration phase. We need to indentify different risk factors which can cause failure during component integration phase [1-2, 8, 21, 27, 53]. System integrators must manage some information to make sure that component is integrated successfully and CBS in risk free.

The aim of this chapter is to investigate the risk factors which can be faced by any system integrator during component integration phase. It's important that we obtain information regarding negative aspects in component integration from a system integrator's point of view to avoid any integration risk. We need to identify and correlate these risk factors and their impact during integration phase.

To date, empirical research work on CBS has been focused on identifying risks associated with component identification, selection and maintenance processes. However, to the best of our knowledge, none of the existing work investigates the risk factors in integration phase, from the system integrator's perspective, of the CBS development life cycle.

6.2 Research Questions and Hypotheses

We wish to gain insight regarding which risk factors have high impact during integration process. We discover from our study that identification of risks at early stages can minimized the impact during component integration. Figure 6 shows the theoretical model designed for this purpose. The main objective of this model is to answer following research question.

RQ1: What are the possible risks/problems during component integration phase?

The hypotheses to access the impact of risk factors during component integration is as follows.

HN1: There is no correlation between “Lack of Requirement Conformance” and “System becomes risk-prone”

HA1: There is a correlation between “Lack of Requirement Conformance” and “System becomes risk-prone”

HN2: There is no correlation between “Lack of Sufficient Testing” and “System becomes risk-prone”

HA2: There is a correlation between “Lack of Sufficient Testing” and “System becomes risk-prone”

HN3: There is no correlation between “Lot of Glue Code” and “System becomes risk-prone”

HA3: There is a correlation between “Lot of Glue Code” and “System becomes risk-prone”

HN4: There is no correlation between “Uncertified Components” and “System becomes risk-prone”

HA4: There is a correlation between “Uncertified Components” and “System becomes risk-prone”

HN5: There is no correlation between “Lack of Interoperability Standards” and “System becomes risk-prone”

HA5: There is a correlation between “Lack of Interoperability Standards” and “System becomes risk-prone”

HN6: There is no correlation between “Lack of Version Control Information” and “System becomes risk-prone”

HA6: There is a correlation between “Lack of Version Control Information” and “System becomes risk-prone”

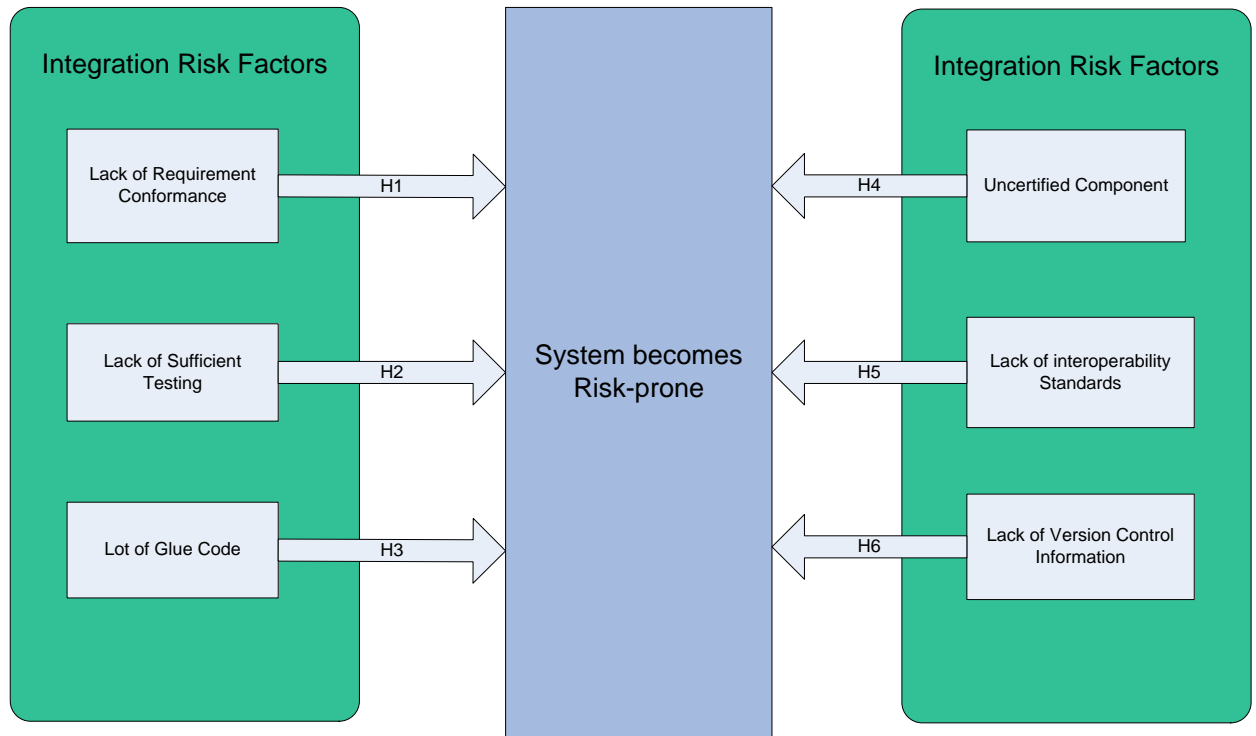


Figure 6: Theoretical Research Model

6.3 Data and Result Analysis

In this section, we present results in tabular form along with description. The Pearson correlation, Spearman correlation Partial Least Square (PLS) techniques are used [12] to analyze the collected data of study to validate each of the hypotheses which have present

in earlier section. These statistical calculations were performed using Minitab 14 software.

6.3.1 Hypotheses testing phase-I

In order to test hypotheses H1-H6 we examined the Pearson correlation coefficient [12] between different factors of the research model shown in Figure 6. The result of the statistical calculations for the Pearson correlation coefficient is reported in Table 6. The Pearson correlation coefficient between “system becomes risk-prone” and “lack of requirement conformance” was 0.50 (weak positive association) at $P < 0.05$ provided a justification to accept alternate hypothesis HA1 & reject null hypothesis HN1. The alternate hypothesis HA2 accepted based on Pearson correlation coefficient 0.62 (weak positive association) at $P < 0.05$ between “system becomes risk-prone” and “lack of sufficient testing”. The correlation coefficient of 0.50 (weak positive association) at $P < 0.05$ was observed between “system becomes risk-prone” & “lot of glue code” and thus provided a justification to accept alternate hypothesis HA3. The Pearson correlation coefficient between “system becomes risk-prone” and “uncertified components” was 0.52 (weak positive association) at $P < 0.05$ provided a justification to accept alternate hypothesis HA4 & reject null hypothesis HN4. The correlation coefficient of 0.41 (weak positive association) at $P < 0.05$ was observed between “system becomes risk-prone” & “lack of interoperability standards” and thus provided a justification to reject null hypothesis HN5 and accept alternate hypothesis HA5. The correlation coefficient of 0.01

(little or no association) at $P < 0.05$ was observed between “system becomes risk-prone” & “lack of version control information” and thus provided a justification to reject alternate and accept null hypothesis HN6.

Hence it was observed and is reported here that null hypothesis for H6 is accepted but alternate hypothesis rejected whereas alternate for H1, H2, H3, H4, H5 are found significant and they have weak positive association.

Table 6: Hypotheses testing using Pearson Correlation Coefficient

H	Variable 1	Variable 2	Coefficient	p-value
H1	Lack of Requirement Conformance	System becomes risk-prone	0.50	0.000
H2	Lack of Sufficient Testing	System becomes risk-prone	0.62	0.000
H3	Lot of Glue Code	System becomes risk-prone	0.50	0.000
H4	Uncertified Components	System becomes risk-prone	0.52	0.000
H5	Lack of interoperability Standards	System becomes risk-prone	0.41	0.002
H6	Lack of Version Control Information	System becomes risk-prone	0.01	0.915

6.3.2 Hypotheses testing phase-II

In this phase, we conducted non-parametric statistics using Spearman correlation coefficient [12] to test the hypotheses H1-H6. Table 7 also reported the observation made in this testing phase. The Spearman correlation coefficient between “system becomes risk-prone” and “lack of requirement conformance” was 0.50 (weak positive association) at $P < 0.05$ provided a justification to accept alternate hypothesis HA1 & reject null hypothesis HN1. The alternate hypothesis HA2 accepted based on Spearman correlation coefficient 0.62 (weak positive association) at $P < 0.05$ between “system becomes risk-prone” and “lack of sufficient testing”. The correlation coefficient of 0.54 (weak positive association) at $P < 0.05$ was observed between “system becomes risk-prone” & “lot of glue code” and thus provided a justification to accept alternate hypothesis HA3. The Spearman correlation coefficient between “system becomes risk-prone” and “uncertified components” was 0.48 (weak positive association) at $P < 0.05$ provided a justification to accept alternate hypothesis HA4 & reject null hypothesis HN4. The correlation coefficient of 0.40 (weak positive association) at $P < 0.05$ was observed between “system becomes risk-prone” & “lack of interoperability standards” and thus provided a justification to reject null hypothesis HN5 and accept alternate hypothesis HA5. The correlation coefficient of 0.02 (little or no association) at $P < 0.05$ was observed between “system becomes risk-prone” & “lack of version control information” and thus provided a justification to reject alternate and accept null hypothesis HN6.

Hence it was observed and is reported here that null hypothesis for H6 is accepted but alternate hypothesis rejected whereas alternate for H1, H2, H3, H4, H5 are found significant and they have weak positive association.

Table 7: Hypotheses testing using Spearman rank-order correlation coefficient

H	Variable 1	Variable 2	Coefficient	p-value
H1	Lack of Requirement Conformance	System becomes risk-prone	0.50	0.000
H2	Lack of Sufficient Testing	System becomes risk-prone	0.62	0.000
H3	Lot of Glue Code	System becomes risk-prone	0.54	0.000
H4	Uncertified Components	System becomes risk-prone	0.48	0.000
H5	Lack of interoperability Standards	System becomes risk-prone	0.40	0.001
H6	Lack of Version Control Information	System becomes risk-prone	0.02	0.445

6.3.3 Hypotheses testing phase-III

In phase-III of hypotheses testing, we used Partial least square regression (PLS) technique [12] to overcome some of the associated limitations and to cross validate with the results observed using approach of Phase-I and Phase-II. Table 8 reports the result of PLS tests of the hypothesis and **Table 9** shows the cross validation of results. The PLS coefficient between “system becomes risk-prone” and “lack of requirement conformance” was 0.50 (weak positive association) at $P < 0.05$ provided a justification to accept alternate hypothesis HA1 & reject null hypothesis HN1. The alternate hypothesis HA2 accepted based on PLS coefficient 0.62 (weak positive association) at $P < 0.05$ between “system becomes risk-prone” and “lack of sufficient testing”. The correlation coefficient of 0.50 (weak positive association) at $P < 0.05$ was observed between “system becomes risk-prone” & “lot of glue code” and thus provided a justification to accept alternate hypothesis HA3. The PLS coefficient between “system becomes risk-prone” and “uncertified components” was 0.52 (weak positive association) at $P < 0.05$ provided a justification to accept alternate hypothesis HA4 & reject null hypothesis HN4. The PLS coefficient of 0.41 (weak positive association) at $P < 0.05$ was observed between “system becomes risk-prone” & “lack of interoperability standards” and thus provided a justification to reject null hypothesis HN5 and accept alternate hypothesis HA5. The PLS coefficient of 0.01 (little or no association) at $P < 0.05$ was observed between “system becomes risk-prone” & “lack of version control information” and thus provided a justification to reject alternate and accept null hypothesis HN6.

Hence it was observed and is reported here that null hypothesis for H6 is accepted but alternate hypothesis rejected whereas alternate for H1, H2, H3, H4, H5 are found significant and they have weak positive association.

Table 8: Hypotheses testing using Partial Least Square Regression (PLS)

H	Variable 1	Variable 2	Coefficient	p-value
H1	Lack of Requirement Conformance	System becomes risk-prone	0.50	0.000
H2	Lack of Sufficient Testing	System becomes risk-prone	0.62	0.000
H3	Lot of Glue Code	System becomes risk-prone	0.50	0.000
H4	Uncertified Components	System becomes risk-prone	0.52	0.000
H5	Lack of interoperability Standards	System becomes risk-prone	0.41	0.002
H6	Lack of Version Control Information	System becomes risk-prone	0.01	0.915

Table 9: Comparison of results from different tests

H	Pearson Correlation		Spearman Correlation		Partial Least Square	
H	Coefficient	p-value	Coefficient	p-value	Coefficient	p-value
H1	0.50	0.000	0.50	0.000	0.50	0.000
H2	0.62	0.000	0.62	0.000	0.62	0.000
H3	0.50	0.000	0.54	0.000	0.50	0.000
H4	0.52	0.000	0.48	0.000	0.52	0.000
H5	0.41	0.002	0.40	0.001	0.41	0.002
H6	0.01	0.915	0.02	0.445	0.01	0.915

6.4 Discussion

From the industrial survey results, we make the following observations.

- *Lack of Requirement Conformance:* The software developer should not develop something which just accomplished the desired functions. For a successful software, the developers must develop software in such a way which best meets the needs and requirements of its clients. The coefficient in result analysis shows that there is a weak positive association between lack of requirement conformance and system becomes risk-prone and justifies accepting alternate hypothesis. This means that there is a chance of risk or system failure if requirements are unambiguous. The system integrators participating in the study indicate that they ensure that they have selected the right component by comparing their requirements with the features offered by a component. They make sure that they have clear requirements of a component and CBS before starting integration process. Therefore, we believe that it's important to map all requirements of the selected component with CBS to ensure risk free integration process.
- *Lack of Sufficient Testing:* Testing is any activity aimed at evaluating an attribute or capability of a program or system and determining that it meets its required results [54]. It is the process of executing a program or system with the intent of finding errors [54] which can further help in reducing the possibility of risk. The coefficient in result analysis shows that there is a weak positive association between lack of sufficient testing and system becomes risk-prone and we accepted alternate hypothesis. This means that insufficient testing can cause risk in CBS. It

is evident from our study that testing is required over and over using various approaches and scenarios. Furthermore, we found that testing helps to determine the quality and reliability of the component by giving sample data. We believe that there should be detail testing to make sure that integration process and CBS is risk free.

- *Lot of Glue Code:* Glue code or wrappers may require where one component wishes to make use of another, but there is an incompatibility between the providers and required interfaces of these components. The main concern of writing wrappers is to solve mismatches between the integrated components so they can communicate effectively. Well defined interfaces and with good component documentation can speed up building systems by integrating built components. If there are different providers of the components then the interfaces for that component usually not clearly defined. The coefficient in result analysis shows that there is a weak positive association between glue code and system become risk-prone and we have accepted alternate hypothesis. This means there is chance of risk while writing wrappers between two components during integration process. The participated respondents address the same issue that sometimes it becomes difficult to write or modify wrappers and could cause component integration failure. They highlight that it's not easy to maintain or debug wrappers

between different components. We believe that the use of wrappers during integration process should be minimal in order to avoid any risk in CBS.

- *Uncertified Components:* The coefficient in results shows that there is a weak positive association between uncertified components and system becomes risk-prone and we accepted alternate hypothesis. This means that components should be certified in order to make integration process and CBS risk free. The respondents in our survey indicate that system integrator must make sure that the selected component is certified before starting the integration process. They believe that there should be some methods or process to check & verify the component certification which further can go through to component testing process. Therefore, we believe that system integrator should make sure that component is certified before starting the component integration process.
- *Lack of Interoperability standards:* it's important to follow some standards while developing interfaces for any component for communication. The coefficient in our results shows that there is a weak positive association between lack of interoperability standards and system becomes risk-prone and we have accepted the alternate hypothesis. This means if components and its interface developed without any defined standards then there will be high chances of risk in CBS because it will be hard for any system integrator to understand the code which has

no standards or conventions and it will be difficult for system integrator while modifying the component and its interfaces if required or during writing wrappers or glue code between components. The system integrator participating in the study indicates that they follow international standards just like COSO and ISO and they have IT Compliance and Audit department for this purpose. They also indicates that it's not easy to understand code written by someone else with their own interface standards and declaration and it further make difficult when it comes to detecting defects or to modify the component interface in component based systems. Thus, we believe that it's important to have well defined development standards of component interface, must have well defined interfaces to communicate with other components, this could help in minimizing the chance of possible risk during component integration process.

- *Lack of Version Control Information:* The study indicates that there is no or little association between lack of version control information and system becomes risk-prone and we have accepted the null hypothesis. This means that maintaining versioning information of a give component has no high impact or risk during component integration phase. The participated respondents highly believe that components should have all features of previous version plus new ones and this information should be maintained and stored in component repository. They consider that for proper upgrades, proper versioning should be done and proper

builds should be maintained. After ensuring all of these functions, right version of component should be given to system integrator for component integration process. Thus, we believe that maintaining version control information for a component is important but it cannot be considered as high risk factor.

CHAPTER 7

QUALITATIVE DATA ANALYSIS

In this chapter, we present analysis based on respondent's response on open ended questions. It will help us in understanding further about component based system from system integrator's perspective. These chapters also presents the guidelines which can be consider by any system integrator during component integration process and finally we present our initial step for validating the proposed guidelines.

7.1.1 Upgrading of components

The upgrading of component may be required at any stage in CBS and our study shows that documentation plays a vital role during component upgrading process. The respondents participating in our survey emphasize that through proper documentation and by following some development standard, handling the upgrade of components can be easier. They believe that proper code design and architecture should be designed in such a way that it is flexible with upgrades of components. Some respondents emphasize that upgrade requires thorough impact analysis and testing and has to be planned properly and should be well documented i.e. proper upgrade information, proper versioning information, proper builds and technical specification should be maintained.

7.1.2 Motivations of using commercial components

We tried to find the motivation of using commercial components from experienced component engineers in order to gain the importance of using commercial component. Some of the respondents state that they use commercial component when something is beyond their skills or if something they try to develop is already available and different client is using it. Most of the times they find bug free commercial components which can be easily integrated with little modifications. One respondent believes that it saves a lot of development efforts, cost and its useless to reinvent the already developed component. Furthermore, they emphasize on commercial components to save time, they fill in the

gaps by making their own component and taking expertise from across the group. They believe commercial components are more reliable, better quality and most of them come with warranties or support.

7.1.3 Determining the quality and reliability of the component

The respondents stress the importance of quality and reliability of the component. They mentioned that they test each component separately and check the desired output by giving some sample data, which helps them to determine quality of it. For reliability, they check different clients using the same component or different product where the component is used. They also make sure that component is certified. One respondent mentioned that they perform proper analysis before using component. They always prefer known vendors whose solution is mature and reliable in market. They do not test the component because they believe that one advantage of using component is that they can pass through the testing which saves their cost of testing. One of the respondents believes that quality can be defined to fulfill the requirements. If the component is giving the required output without failing then they say that it has good quality and it has high reliability. Reliability can be taken further by saying that component's integration into the system shall not affect any other integrated components, i.e. it should be least cohesive. Some respondents think that by reading reviews and testing all business and technical scenarios can help in determining quality and reliability of the component. One

respondent believe that quality and reliability can be determined by the by the branding of the component vendor and by past history use of the component by others.

7.1.4 Limited control on commercial components

We found from our study that due to limited control over commercial components there are different shortcomings which can further raise the chances of risk in CBS. The main disadvantage of using commercial components is less control over component and source code is usually not available. Furthermore, debugging and testing is not that easy. One of the respondent experienced that it's not easy to handle bugs especially when you are using executable component and do not have the source code for it. Another drawback is replacement of a component, if Vendor Company goes out of business then it's hard to replace. In some contracts, you have to pay for every upgrade even for minor changes. The cost of maintenance becomes high and the organizations can't build an efficient learning experience since provider will install, configure and maintain the system. One respondent explains that when they want to deliver something highly customized to their customers, they face problem of highly restrictive actions that they can take in customizing commercial components. They face difficulty if the documentation is not good enough or worse, if the vendor created the component in such a way that it cannot be customized beyond a certain limit. In general, most of the respondents express that due to the nature of commercial component; they have less control or command on it. They are not aware of the complete source code of given component which makes it difficult

for them during debugging and testing process. They have very limited control on the given component and when there is any customization required then integrator always needs to contact component provider even for minor changes.

7.1.5 Ensuring that System Integrator has selected the right component

In order to know, how the system integrator ensures that the given component is the desired component or not, we found feedback from our study. Most of the respondents emphasize testing. They perform testing against the selected component by giving some sample data, by confirming the results and check if the results are in line with the requirements. Furthermore, they ensure it by comparing given requirements with the features component offers; they research for available choices and run a comparison, by doing thorough literature review related to selected component. One respondent believes that selecting the right component is driven by business requirements and the ratio of adaptability of that component towards the requirements.

7.1.6 Difficulties faced in integrating components.

We tried to find out difficulties faced by any system integrator during component integration process. The majority of our respondents state that most of the system integrators face difficulties in finding out the capabilities of components because components are not documented in standard way. Every component provider document

his component based on their standards and procedures which results inconsistent and insufficient documentation of a component. They believe that quality of documentation varies very much from one component provider to another. Furthermore, respondents complain that they faced problems when they worked across different platforms or design of the application changes, network workflow changes and dependency of the specific component on some other non-existent components. One respondent states that most of the time vendor specific code is the essence of component which poses technical challenges. Some respondents emphasize on Inadequate documentation, not receiving what was expected, lack of support from component vendor, too complex to understand because of inadequate and substandard coding and documentation, no proper versioning of control information. The respondents explain that they face problem in integrating heterogeneous components that do not understand each other, they need to create a wrapper layer for converting one component output into a format understandable by the other component or in other words they write interfaces for getting the components to talk to each other.

7.1.7 Maintaining relationship with vendor

We have found from component based system literature that the relationship with component provider is very important. The respondents in our study also emphasize that it's important to maintain a good relationship with component provider. They explain their different ways to retain it. They mentioned that they keep in touch with component

provider during each cycle of component integration phase. They prefer keeping clear terms and conditions in the contracts and keeping good relationships with the technical peers on vendor's side. Most of the respondents emphasize that they maintain the relationship through support contract, frequent interaction and through email correspondence.

7.1.8 Testing the selected component for integrating

The respondents in our study emphasize that testing of a component is very essential and they expressed their methods of testing. Some respondents use their own test cases and tools to test the component by giving some test data and then checking and verifying the results. The respondents state that they go through different features of a component and see if they work and then test its efficiency. By efficiency, it means it should not take very long for some fancy work. The respondents explain that they perform various testing e.g. unit testing, black box testing, integration testing etc.

7.1.9 Difficulty in detecting defects and bugs in component base system

Our study helps to understand the common problems and difficulties faced by system integrator during detecting defects, bugs or debugging in component based system. The participated respondents explains that debugging is not easy because you are not aware with the source code, usually you don't have source code or even if you have then it's not

easy to understand the code written by someone using their own development standards and declarations. One respondents believe that it depends upon the component vendor; if exception or handles are properly written then it's easy to detect bugs otherwise it's difficult to figure out proper meaning of certain errors. One respondent believe that it is difficult to locate bugs since the component appears as a black box, one can only check by providing various inputs and checking the output.

7.1.10 Component Certification

Our study shows that it's important to have certified component for component integration process. The respondents explain that they contact different clients and make sure that they have no problem using it and that the component is certified. They believe it's important in order to make sure that the selected component is reliable. They perform testing over and over again using various approaches and scenarios to make sure that the component is certified. Some respondents say that they follow international standards and their companies have IT Compliance and Audit department for this purpose.

7.1.11 Component Repository

The participated respondents in our study believe that it's important to have repository which will help maintaining and storing components. Most of the respondents have a repository in their organization for all project related artifacts, components, source code

etc. They believe that this is for proper configuration management and for back up purposes. One of the benefits of having component repository is that components can be reused in any new project and by any other development unit within the same organization.

7.1.12 Strengths in integrating components

We have tried from our study to investigate all the strengths and advantages from system integrator's perspective in CBS. By summarizing our results, we have found the following strengths based on system integrator's experienced. They believes that is good to have already built-in components, it saves lot of development time, it is easy to start integrating with an already built-in component instead of writing and developing everything from scratch and then integrating it. The already built-in component can save resources, reduce overall costs and time etc. They are more flexible and maintenance becomes easier, provides re-usability, rapid application development, fast delivery of projects, easily manageable, and avoids reinventing the wheel. The respondents believe that if component is reliable and efficient then it should save time in house development and testing or they can pass through the testing phase. If a component is properly developed based on some development standards then it's easy to track bugs and errors. The respondents believe that it's easier to ensure the quality of the code and maintenance because the entire application is divided in different isolated units. Each unit/component

is only concerned with its own implementation and how it is exposing its services to the client etc. If components are linked with well defined interfaces and with good component documentation then it can speed up building systems by integrating well built components.

7.1.13 Weaknesses in integrating components

We tried to identify weakness and disadvantages from system integrator's perspective in CBS. By summarizing our results we found that most of the respondents complain about lack of information about the source component, they do not have much control over it, and it is not easy to maintain a relationship with the provider of the component. They find complexity issues, external dependency, lack of control on source code, less flexibility, lack of proper information, unknown design and architecture because of inadequate documentation, inability to customize, difficulty in finding right components that meets user needs, lack of coding/development standards, support after component is integrated and moved into production, it requires high level of skills set in system integrator, maintenance cost could be high, lots of learning, not easy to detect and fix bugs, and lack of complete information about component behavior. If there are different providers of the components then the interfaces (how the component will interact with other components and vice versa) for that component usually not be clearly defined and it becomes very challenging. It can decrease security, increase dependency, may affect performance because of overhead in communication between components.

7.1.14 Further Observation

At the end of survey, we asked respondents to share their personal experience, their observations and suggestions which we may miss in our survey and it could help us in our research work. Based on the input given by different respondents we found that most of the respondents emphasize on low pricing of components, they emphasize on component security and performance, and they mentioned that component basic information and component execution information details should be known as well. We have summaries their suggestion following.

Low-Priced Component: The participated respondents believe that buying a cheaper component is major risk in component integration process. Cheaper component may provide all those features which other pricey components have but there is a high chance that you will end up writing more glue code and your own methods because such component doesn't come with proper documentation or with proper development kit or with proper interfaces methods. This will raise the chance of risk during integration process because you may need to write more complex interfaces for this component to communicate with other components. The methods and objects in such components are not very precise. Therefore, if you purchase expensive one, then there is very high chance that you will be writing minimal customize code to connect the component with other components. E.g. if you are asked to draw network diagram programmatically then

you may look for a component who provide API to place these objects. If you buy cheaper one then you may end up writing several line of code for just creating one node or line within one diagram which increase developing hours, efforts and may increase risk chance or possible error chance whereas in other hand if you buy much expensive one, they highly believe, it will come up with more precise methods and objects and it will be much easier to create one node or connect different nodes by writing one or two line of code.

Component Performance & Security: The respondents consider performance of a component is another major success factors for component integration. They believe, it should be consider before component integration process. They emphasize that they should know the size of the component which is basically line of code they have for that component, the capacity of component which is basically how much amount of work component can handle and perform. The throughput of the component, which is basically measures how much data the component can handle in a given time unit, the allocation time of resources which is basically the time component allocate the physical resource. The strategies to protect component against viruses, recovery methods in attack situations and methods to protect the data, it all comes under security of component.

Component Basic & Execution Environment: The respondents found that few of the components come with basic Information about the component. The component overview

information can be helpful in order to understand why the component has been developed or purchased. Other than that, they believe that component prerequisites information and component limit or constrains information should also be included in component documentation. Also component should have some unique name and it should be mentioned in document.

During component selection or evaluation it's very important to know about component execution environment. They have found this information missing in some of the available documentation of component. They believe that it is extremely important that the component execution information should be documented with all the important conditions. This information may include the component platforms, the interdependencies, physical resource requirements etc.

7.1.15 Avoid Component Risk:

We have designed the forth section of the survey in order to get just initial idea how risk can be best mitigated. This part of survey helps us in future work. We have learned from this section of survey that there should be frequent interactions and meeting with clients. They should be involved during discussions and during component integration cycle. We found that most of our respondents strongly agree that it not a good idea to write lot of glue code between components. It will be hard to maintain and handle. They emphasize testing over and over before and after component integration. Finally most of the

respondents believe that component integrator should have complete knowledge about the selected component which they are going to integrate.

CHAPTER 8

THREATS TO VALIDITY

In this chapter, we will present the possible threats for our study.

8.1 Construct Validity

We believe that our study has no serious construct validity threats because the research questions are based on the existing literature [1, 6, 9, 16, 21, 27, 36]. To ensure construct validity, the survey questionnaire was pre-tested internally with five colleagues to ensure that all questions are meaningful and their respective answers will help us in the result analysis. All terminologies used in the questionnaire are explained to provide clear definitions and avoid any misinterpretations. We also provided open ended questions to

ensure that respondents know the correct definition of component and their knowledge, concepts and understandings about component based systems are clear. Furthermore, time pressure is another threat to validity. We believe that the time allocated for the survey was sufficient as the subjects answered all the questions and no one complained about the lack of time in our communications. Thus, we believe that time is not a confounding factor in our study.

8.2 External Validity

The inherent limitation of survey-based studies lies in their external validity due to difficulty in achieving a true random sample of participants. We overcome the external validity threats in the study by using a variant of the snowball sampling technique [39] where key practitioners serve as contact points in the organizations involved. The contact points are then sent the questionnaire, and forward it on to other potential respondents. Another possible external threat is the location and size of the respondents company. Hence, it is not easy to generalize these results for all domains. In our study, the participants belong to Asian, European and Australian small to medium-sized companies and these companies work in different areas of IT industry. Furthermore, we ensured that all the potential participants had relevant experience in development of a CBS. We believe that the results of the study can be generalized for small to medium-sized companies where software developers are involved in component integration.

8.3 Internal Validity

The one possible threat to internal validity is our misunderstanding of respondent's answers. To avoid such misunderstanding, we have reviewed all answers and if we found any doubt or want to ensure that we have interpreted answers or comments correctly, we directly contacted to respondent for clarification. Furthermore, most of the respondents in our survey were bearing master degree and they were software developers, software architect, integrators and few of them were project managers with 5 - 7 years of experience in software development and in component based systems. Our survey results showed that they have taken interest in the survey and based on their experience, knowledge and skills in component based systems, we believe that they have answered the survey questions to the best of their ability. To avoid misunderstandings, we were available via email and phone during the study to clarify any ambiguities in the questions.

8.4 Conclusion Validity

In this study, we used standard statistical techniques [39] to either accept or reject the null hypotheses. We also mentioned the detailed output of the statistical techniques in Table 5 and Table 9. We used both parametric and non-parametric statistical techniques to validate our results. The Partial Least Square technique was used to cross validate the results and avoid the issue of small to medium data set size. Furthermore, survey took about 7-10 minutes to fill in completely and it was designed using standard scales that allow us to analyze feedback provided by subjects with 5 to 7 years of experience in software development.

CHAPTER 9

GUIDELINES

9.1 Guidelines

In this section, we propose guidelines. The main intention of these guidelines is to minimize the chances of risk, as much as possible, during component integration process and to make sure that everything is inline. The guidelines are designed based on earlier result analysis and open feedback and suggestion from experienced system integrators and further based on following analysis.

We concluded from our study that lack of requirements conformance is one of the major factors causing risk chances during component integration. So based on results and

studies we found that system integrator should make sure from their side that the selected component is the right component for the system. To achieve this, system integrators make sure that the selected component is the desired component by comparing system requirements with the features offered by a component. There should be frequent meetings with component provider and all open creative questions related to selected component must be asked and everything must be noted down for future reference. Afterwards, a team meeting must be conducted in which all notes are compared. The exact needs must be listed and then returned to the component provider if needed for clarification. Finally, after interactive meetings with component provider the selected component that meets the needs must be determined.

Our study shows that information about component features both technical and functional is important and they must be documented and checked very well. It may help in estimating component integration efforts, it also helps to evaluate and assist the component. We also observe from our study that good relationship with component provider can help to understand the component in more depth which will further help system integrator during component integration process. To accomplish this, make sure that component documentation must include component functional and technical specifications. It must have component history information; it must have well maintained versioning information, details of all upgrades even if it's minor should be maintained. The price of component, reviews of a component and details of new features offered by

component should be mentioned. The component performance, security and component execution environment information should be clear and explained. It should also include complete information of development kit along with explanation of all methods and objects used in the component. The details for design level changes should be documented; it will help in architectural changes and compatibility issues. The comprehensive information should be included about component application interfaces and wrappers. The source code of a component with proper exception handling and comments and explanations of each block of code should be included and in sync with documentation. The past history of component provider should be included; this will help in making sure that vendor's solution is mature and reliable in market. The personal contact details which may include email address, telephone number, fax numbers etc.

We also learn from our study that component should be trustworthiness. We are referring trusty component here with component certification and with component interoperability. In this study, we found that it's very important to have certified component. It is required to ensure that the published properties or specifications of a component have really been verified. Other than this, we also discover that lack of interoperability standards can cause risk during component integration. To achieve this, first the component certification should be verified by checking clients who have used this component previously and confirm component certification with third-party certifier. Second, make sure proper standards and conventions defined. The proper comments and explanation of

each block and well defined development standards followed and meaningful interfaces of component are defined which can be easily implementable and compatible with other components in CBS. Finally the auditing team should audit, verifies, test and ensure component and component interfaces.

In our study, we found that component testing is very vital process for successful component integration. Component testing helps to determine the quality of the component. We discovered that usually the information about integration testing is not present in available component documentation and lack of component testing raise chances of risk during integration process. Furthermore, we found that writing wrappers between components also increases the chances of integration risk. It's very hard to test the component if its interfaces information is not well documented in component documentation. In order to overcome this, make sure to have complete testing details of a candidate component, list down all offered features by component for testing. The component customizing details with information about wrappers/interfaces, methods and data objects should be included. The Information about component dependencies with diagrams should be well explained in document. The information about component internal methods and attributes through diagram should be explained. The information of component integration testing should be included, component Interfaces testing and other testing of technical and business scenarios information should be incorporated. The components should come with test sample data for testing with desired output and test

reports. The information about proper exception handling for debugging purpose should be included. Finally make sure testing results must be in line with business requirements.

Based on our study, further observations and above discussion we propose component documentation guidelines shown in Figure 7, We categorized the information into four different sections, named, basic information, detail information, testing and component provider information.

Basic Information	<ul style="list-style-type: none"> • Name • Overview • List of Prerequisites • Constraints • Cost • Reviews • Versioning • New Features • Certification Information
Detail Information	<ul style="list-style-type: none"> • Technical and Functional Specifications • Security • History • Performance • Execution Environment • Interoperability Standards • Development Kit • Design Level Details • Interface Information
Testing	<ul style="list-style-type: none"> • Test Details • Customizing Details • Dependency Information with Diagrams • Internal methods and objects information with diagrams. • Testing Scenario • Test Data • Reports • Debugging / Error handling
Component Provider Information	<ul style="list-style-type: none"> • Overview • History • Clients • Contact Details

Figure 7: Proposed Guidelines

Figure 8 presents the cross reference of guidelines factors with the different hypothesis and open ended analysis which were tested and presented in earlier sections.

Sections	Items	Documentation Hypothesis					Risk Factors Hypothesis						Open Ended
		H1	H2	H3	H4	H5	H1	H2	H3	H4	H5	H6	Analysis
Basic Information	Name												✓
	Overview												✓
	List of prerequisite												✓
	Constraints		✓										✓
	Cost		✓										✓
	Reviews		✓										✓
	Versioning											✓	✓
	New Features			✓									✓
	Certification Information									✓			✓
Detail Information	Technical and Functional Specifications	✓	✓				✓						✓
	Security												✓
	History		✓										✓
	Performance												✓
	Execution Environment												✓
	Interoperability Standards								✓		✓		✓
	Development Kit												✓
	Design Level Details								✓				✓
	Interface Information								✓				✓
Testing	Testing Details				✓			✓					✓
	Customizing Details												✓
	Dependency Information with												✓
	and Objects information with												✓
	Testing Scenario				✓			✓					✓
	Test Data				✓			✓					✓
	Reports												✓
	Debugging / Error Handling				✓			✓					✓
Component Provider Information	Overview					✓							✓
	History					✓							✓
	Clients					✓							✓
	Contact Details					✓							✓

Figure 8: Mapping of Hypothesis & Open ended analysis with Guidelines

9.2 Validation of Proposed Guidelines

In order to check the applicability of the proposed guidelines and component documentation template, we carried out an evaluation industrial survey against two type of audience. First we target those who participated in our initial survey and then in second survey we target the audience who were not part of our survey. This validation process is very basic and performed to just get initial idea or assessment about the proposed guidelines. In future work, we planned to expand our research in this area.

In the first survey, target audiences were the same who participated in our main initial survey. We filter out those participants who did not responded initially so basically we sent notification only those 53 participants who actively participated and we got responses from 35 of them. The main purpose of this survey was to obtain different perspective on guidelines from experienced system integrators. The structure of the document and the guidelines were assessed by different practitioners and most of them agree the content of the document and the flow of guidelines are comprehensive enough although the following improvements were proposed:

- Give examples of how to use component interfaces
- More detailed information for security and testing against component security.

- Expanding Reliability information.
- Detail information about how to modify and adopt component to a new environment.
- Information about how new features can be added to the component to expand the functionality of the component.

In the second survey, we target the audiences who were not part of main survey. Overall we found satisfaction but got following suggestions from few of them to improve it more.

- Instead of using term “debugging” use term “tracing” which is more common in component based terminology.
- One participant suggested use term “logging” instead of “debugging”.
- Instead of using term “Interface Information”, use term "application programming interface (API)" which is widely accepted term.
- Execution environment can be changed "supported environments".
- I see support/license information missing or you may have categories it into some other section.
- In basic information, for reviews, there should be link for open forum where different reviews posted for the specific component. There should be reference of review.

- Constrains should be part of details information with details of all kind of dependency and constrains.

CHAPTER 10

CONCLUSION AND FUTURE WORK

This chapter concludes the thesis and identifies possible areas for future research

10.1 Conclusion

This research presents the results of an industrial survey. The data was collected from 53 system integrators, with 5 to 7 years industrial experience as CBS integrators, working in Asian, Australian and European organizations. The aim was to assess the benefits of available component documentation during the integration process of a CBS and then finding missing gaps in available component documentation. We also analyzed the correlation between available component documentation with five integration success factors namely, early component evaluation, new features and modification information,

integration testing, relationship with component vendors and integration effort estimation. The participants of the survey found the available component documentation was useful in early component evaluation and discovering new features. However, it is important to note that on average available component documentation does not provide enough information to overcome the two most common CBS integration challenges of incorrect integration effort estimation and integration testing.

This research also presents some of the common risk factors which may occur during component integration process. These factors are lack of requirement conformance, lack of testing, lot of glue code or wrappers, uncertified components, lack of interoperability standards, lack of version control information. The results indicate that some of these risk factors have impact on making CBS system risk-prone. These factors were tested thoroughly and analyzed. At the end, research presents the guidelines based on the results from available component documentation factors, integration risk factors and from open analysis done by different experienced industrial system integrators. These guidelines can help system integrator in making sure the successful component integration.

The main idea of this work is to investigate the role and importance of component documentation, analyze missing gaps in available component documentation, relate the component documentation with CBS integration factors, identify the risk factors during

component integration process, see how component documentation can help in avoiding such risks and develop guidelines which can be used by any system integrator.

10.2 Future work

For future work, there is a need to conduct a series of more controlled experiments to investigate the fundamental question of whether more detailed component documentation yields practical benefits during the integration process of a CBS. We believe that there is a need to analyze integration process risks in a CBS development life cycle. In this thesis, we only consider five factors and relate them with component documentation but in addition to these five factors there could be more factors which can contribute and relate to component documentation. The further studies are required to understand the impact of other integration factors such as the importance of component design and quality attribute information during the integration process of a CBS. In addition to a system integrator's perspective, we also need to analyze integration process with reference to component vendors, requirements stakeholders and project managers. This will provide a detailed insight into how component documentation can be improved to help the successful integration of the selected components for a CBS.

Furthermore, future work involves expanding the survey to other countries. There is a need of further identification of more risk factors which could be faced by any application integrator during component integration phase and incorporate in industrial survey. There is also a need to study the industrial common practices about handling requirements volatility, architectural mismatches, and lack of component support in the maintenance phase of a CBS. In addition, we need to analyze the importance of component certification and component knowledge for different phases of a CBS.

Future work also includes more in depth study of component level testing, component interface and data objects testing, explore methods and models to verify the component certification and interoperability standards and synchronize the information of all these strategies with component documentation.

Furthermore, expand and modify the guidelines. More detailed and expand experiments and tests are needed to validate these guidelines. Based on the results, we may come up with maturity model for this purpose which will be sort of upgraded version of these guidelines.

All in all, a more thorough analysis of component documentation, component key factors, system integrator best practices and component integration risks are planned.

APPENDIX A: INDUSTRIAL SURVEY

An Industrial Study on the Importance of Component Documentation: A System Integrator's Perspective

We are conducting research at King Fahd University of Petroleum & Minerals (KFUPM) to better understand risks involved during integration process in Component Based Systems. It's an anonymous survey which will help us to analyze and identify integration risks in Component Based Systems.

Kindly spare some time to complete the given survey and add brief comments where needed. Your time and help in this matter will be highly appreciated.

Note: Please feel free to contact me in case you have any ambiguity or need clarification in any question.

SECTION – 1: GENERAL

1.1 Your current company is situated in?

- ☐ Asia
- ☐ Africa
- ☐ North America

- ☐ South America
- ☐ Antarctica
- ☐ Europe
- ☐ Australia

1.2 Size of your company (Number of Employees/Staff)

- ☐ 1000 +
- ☐ 500 - 1000
- ☐ 100 - 500
- ☐ 50 - 100
- ☐ < 50

1.3 Your company business area?

- ☐ IT Consultant
- ☐ Software Vendors
- ☐ Telecom Industry
- ☐ Higher Education
- ☐ Other

1.4 What is your Education Degree?

- ☐ Bachelor
- ☐ Master

- ☐ Ph.D
- ☐ Other

1.5 Your position in company

- ☐ Project Manager
- ☐ Software Architect
- ☐ Software Developer
- ☐ Other

1.6 How long have you been involve in component based development (Years)?

- ☐ 7 +
- ☐ 5 - 7
- ☐ 3 - 5
- ☐ 1 - 3
- ☐ < 1

SECTION – 2: AS A COMPONENT INTEGRATOR

2.1 How important is it to have adequate documentation of component before integrating it?

Adequate	Inadequate
<input type="radio"/> 5 <input type="radio"/> 4 <input type="radio"/> 3 <input type="radio"/> 2 <input type="radio"/> 1	

2.2 How important to have early evaluation of a component.

Adequate	Inadequate
<input type="radio"/> 5 <input type="radio"/> 4 <input type="radio"/> 3 <input type="radio"/> 2 <input type="radio"/> 1	

2.3 The Information in available component (i.e. Basic information, features, specifications, versioning, usage history, pricing etc) documentation which comes with component is not enough to start component integration process.

- ☐ Strongly Agree
- ☐ Agree Mostly
- ☐ Agree Somewhat
- ☐ Disagree
- ☐ Strongly Disagree

2.4 The information in available component documentation is not enough to in line all business requirements.

- ☐ Strongly Agree
- ☐ Agree Mostly
- ☐ Agree Somewhat
- ☐ Disagree
- ☐ Strongly Disagree

2.5 It's hard to learn new component features & modify from available component documentation?

- ☐ Strongly Agree
- ☐ Agree Mostly
- ☐ Agree Somewhat
- ☐ Disagree
- ☐ Strongly Disagree

2.6 It's hard to perform integration testing of selected component based on information in available component documentation?

- ☐ Strongly Agree

- ☐ Agree Mostly
- ☐ Agree Somewhat
- ☐ Disagree
- ☐ Strongly Disagree

2.7 The available component documentation is not enough to evaluate component?

- ☐ Strongly Agree
- ☐ Agree Mostly
- ☐ Agree Somewhat
- ☐ Disagree
- ☐ Strongly Disagree

2.8 How often you locate and fix the problem in individual component?

Often		Rare	
4	3	2	1

2.9 The information in available component documentation doesn't help in debugging component

- ☐ Strongly Agree
- ☐ Agree Mostly

- ☐ Agree Somewhat
- ☐ Disagree
- ☐ Strongly Disagree

2.10 How many times your effort estimation for integrating component is accurate?

Often		Rare	
4	3	2	1

2.11 The available component documentation doesn't help in estimating efforts for implementing component?

- ☐ Strongly Agree
- ☐ Agree Mostly
- ☐ Agree Somewhat
- ☐ Disagree
- ☐ Strongly Disagree

2.12 Do you agree that relationship with component provider should be maintained in order to better understand the functionality and other specs of component?

- ☐ Strongly Agree

- ☐ Agree Mostly
- ☐ Agree Somewhat
- ☐ Disagree
- ☐ Strongly Disagree

2.13 Do you agree that there should be frequent meetings with component provider in order to make sure that the selected component is the desired component?

- ☐ Strongly Agree
- ☐ Agree Mostly
- ☐ Agree Somewhat
- ☐ Disagree
- ☐ Strongly Disagree

2.14 Explain briefly about the common information in the documentation which comes with component.

--

2.15 How do you handle the upgrades of components?

--

2.16 Does your organization use commercial component or in-house component?

Commercial Component	In-House Component	Both
-------------------------	-----------------------	------

- a. If your answer is Commercial Component/both then what are the main motivations of using commercial components?

--

- b. If your organization use commercial component then what is the percentage?

100%	70%	50%	30% or less
------	-----	-----	-------------

- c. If 30% or less, then why?

--

2.17 How do you determine the quality and reliability of the component?

--

2.18 What risk do you see due to limited control on commercial components?

--

2.19 How do you ensure that you have selected the right component?

--

2.20 Please mention difficulties you faced in integrating components?

--

2.21 How do you maintain your relationship with vendor?

2.22 How do you test the selected component for integrating?

2.23 Do you agree that detecting defect locations (debugging) is difficult in component base system? (please explain either you agree or disagree)

2.24 What is the approach your organization use to certify component and why it's so important?

2.25 At your organization, do you have repository for storing and maintaining commercial components?

Yes	No
-----	----

(If yes, then what's the benefit of doing it? Please explain)

--

SECTION – 3: COMPONENT INTEGRATION RISK FACTORS

3.1 Having lot of glue code between components.

- ☐ Strongly Agree
- ☐ Agree Mostly
- ☐ Agree Somewhat
- ☐ Disagree
- ☐ Strongly Disagree

3.2 Lack of information about the source component could cause failure of integrating components.

- ☐ Strongly Agree
- ☐ Agree Mostly
- ☐ Agree Somewhat
- ☐ Disagree
- ☐ Strongly Disagree

3.3 Design assumptions or design of a component is unknown.

- ☐ Strongly Agree
- ☐ Agree Mostly
- ☐ Agree Somewhat
- ☐ Disagree
- ☐ Strongly Disagree

3.4 Lack of component interoperability standards.

- ☐ Strongly Agree
- ☐ Agree Mostly
- ☐ Agree Somewhat
- ☐ Disagree
- ☐ Strongly Disagree

3.5 There are high risk factors during component integration causes CBS risk prone.

- ☐ Strongly Agree
- ☐ Agree Mostly
- ☐ Agree Somewhat
- ☐ Disagree
- ☐ Strongly Disagree

3.6 Lack of version control information of a component.

- ☐ Strongly Agree
- ☐ Agree Mostly
- ☐ Agree Somewhat
- ☐ Disagree
- ☐ Strongly Disagree

3.7 Lack of requirement conformance.

- ☐ Strongly Agree
- ☐ Agree Mostly
- ☐ Agree Somewhat
- ☐ Disagree
- ☐ Strongly Disagree

3.8 Lack of sufficient component testing.

- ☐ Strongly Agree
- ☐ Agree Mostly
- ☐ Agree Somewhat
- ☐ Disagree
- ☐ Strongly Disagree

3.9 Selected Components are not certified

- ☐ Strongly Agree
- ☐ Agree Mostly
- ☐ Agree Somewhat
- ☐ Disagree
- ☐ Strongly Disagree

3.10 Lack of knowledge about the selected component and about the component based system.

- ☐ Strongly Agree
- ☐ Agree Mostly
- ☐ Agree Somewhat
- ☐ Disagree

- ☐ Strongly Disagree

3.11 As Project Manager / Team lead, do you agree that lack of technology and skills in Component Developers is one of the risk factor?

- ☐ Strongly Agree
- ☐ Agree Mostly
- ☐ Agree Somewhat
- ☐ Disagree
- ☐ Strongly Disagree

SECTION – 4: AVOIDING RISKS DURING COMPONENT

INTEGRATION

4.1 Client should be involved during discussion.

- ☐ Strongly Agree
- ☐ Agree Mostly
- ☐ Agree Somewhat
- ☐ Disagree
- ☐ Strongly Disagree

4.2 The wrappers or glue code should be well defined

- ☐ Strongly Agree
- ☐ Agree Mostly
- ☐ Agree Somewhat
- ☐ Disagree
- ☐ Strongly Disagree

4.3 Integration Testing is good approach for implement components?

- ☐ Strongly Agree
- ☐ Agree Mostly
- ☐ Agree Somewhat
- ☐ Disagree
- ☐ Strongly Disagree

4.4 Integrator should have complete knowledge of component?

- ☐ Strongly Agree
- ☐ Agree Mostly
- ☐ Agree Somewhat
- ☐ Disagree
- ☐ Strongly Disagree

SECTION – 5: STRENGTHS AND WEAKNESS

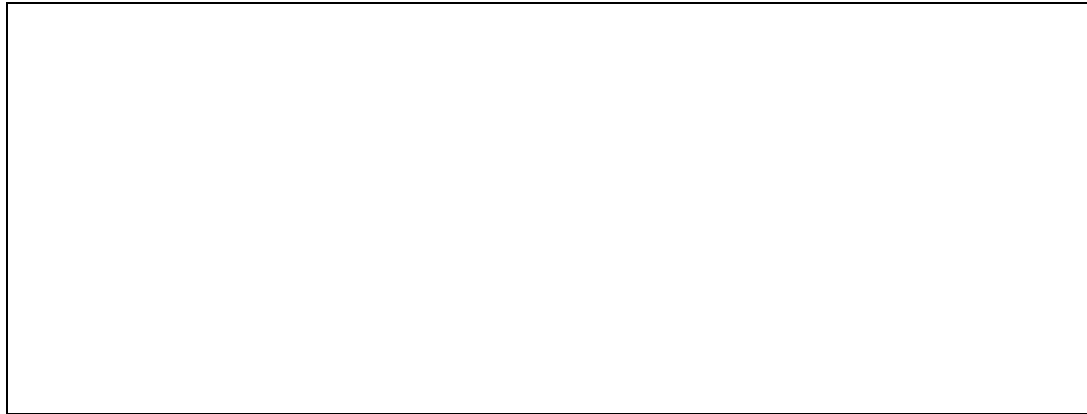
5.1 From your point of view, what are the strengths in integrating components?



5.2 From your point of view, what are the weaknesses in integrating components?



Suggestions: Please give your valuable input to improve our work. Let us know if we have missed any area, give us suggestion based on your experience in component based system, your observations etc which may help us in our research work.

A large, empty rectangular box with a thin black border, intended for users to provide suggestions or feedback.

APPENDIX B: INDUSTRIAL SURVEY RAW DATA

Quantitative Data

1.1	1.2	1.3	1.4	1.5	1.6
Asia	500-1000	IT Consultant	Bachelor	Software Developer	5-7
Asia	100-500	Higher Education	Master	Software Developer	3-5
Asia	100-500	Higher Education	PhD	Other	5-7
Asia	100-500	Software Vendors	Bachelor	Software Developer	5-7
Asia	100-500	Software Vendors	Master	Software Architect	5-7
Asia	50-100	Other	Bachelor	Other	1-3
Australia	500-1000	IT Consultant	Bachelor	Software Developer	3-5
Australia	500-1000	IT Consultant	Master	Software Architect	5-7
Asia	500-1000	Telecome Industry	Bachelor	Software Developer	3-5

Asia	500-1000	Telecome Industry	Bachelor	Software Developer	3-5
Asia	1000+	IT Consultant	Master	Software Architect	5-7
Asia	1000+	IT Consultant	Bachelor	Software Developer	5-7
Asia	1000+	IT Consultant	Master	Software Developer	3-5
Asia	1000+	IT Consultant	Master	Other	3-5
Asia	100-500	Higher Education	Master	Other	3-5
Asia	50-100	Other	Bachelor	Software Developer	1-3
Asia	50-100	Other	Bachelor	Project Manager	5-7
Asia	100-500	IT Consultant	Bachelor	Project Manager	5-7
Asia	50-100	IT Consultant	Bachelor	Software Developer	3-5
Asia	50-100	Other	Master	Software Architect	1-3
Asia	50-100	Other	Bachelor	Project Manager	1-3
Asia	100-500	Higher Education	Bachelor	Software	3-5

				Developer	
Asia	100-500	Higher Education	Master	Software Developer	3-5
Asia	100-500	IT Consultant	Bachelor	Project Manager	5-7
Asia	100-500	IT Consultant	Master	Software Architect	5-7
Asia	100-500	IT Consultant	Bachelor	Software Developer	3-5
Europe	1000+	Software Vendors	Master	Software Developer	3-5
Europe	1000+	Software Vendors	Master	Software Architect	7+
Asia	50	Other	Other	Software Developer	1-3
Asia	100-500	IT Consultant	Bachelor	Project Manager	7 +
Asia	100-500	IT Consultant	Bachelor	Software Developer	3-5
Asia	100-500	Software Vendors	Bachelor	Software Architect	5-7
Asia	100-500	IT Consultant	Bachelor	Software Developer	5-7

Asia	100-500	Higher Education	Master	Software Developer	3-5
Australia	500-1000	IT Consultant	Bachelor	Software Developer	5-7
Australia	500-1000	IT Consultant	Bachelor	Software Architect	7+
Asia	500-1000	Telecome Industry	Bachelor	Software Developer	5-7
Asia	100-500	Other	Master	Project Manager	5-7
Asia	50-100	Other	Other	Software Developer	3-5
Asia	50-100	Other	Other	Other	3-5
Asia	100-500	IT Consultant	Bachelor	Software Developer	5-7
Asia	100-500	Other	Bachelor	Software Developer	5-7
Asia	500-1000	IT Consultant	Other	Software Developer	5-7
Asia	500-	IT Consultant	Bachelor	Project Manager	5-7

	1000				
Asia	500-1000	IT Consultant	Master	Software Architect	5-7
Asia	100-500	IT Consultant	Master	Software Developer	5-7
Asia	500-1000	IT Consultant	Bachelor	Other	5-7
Asia	100-500	Higher Education	Bachelor	Other	3-5
Asia	500-1000	IT Consultant	Bachelor	Project Manager	5-7
Asia	500-1000	IT Consultant	Master	Other	5-7
Asia	100-500	IT Consultant	Master	Project Manager	5-7
Asia	500-1000	IT Consultant	Bachelor	Software Developer	3-5
Asia	50-100	Other	Bachelor	Software Developer	1-3

2.1	2.2	2.3	2.4	2.5	2.6
4	5	5	3	4	3
4	5	1	3	3	5
4	4	5	3	3	3
3	4	5	2	4	2
3	4	4	3	3	4
3	4	4	3	3	2
5	5	5	3	4	2
5	5	3	1	3	1
5	5	5	3	2	2
5	5	5	3	3	4
5	5	5	3	5	4
4	4	3	4	4	4
4	4	1	1	2	3
5	4	5	1	4	3
3	4	4	1	3	3
2	4	5	2	4	3

2	5	5	3	2	3
3	5	3	4	2	2
4	5	4	3	3	2
3	3	5	3	3	2
5	3	4	2	3	3
4	4	3	2	3	4
3	4	5	2	4	1
4	3	3	2	3	4
4	4	4	3	3	3
3	5	5	2	3	4
3	4	5	3	4	1
4	3	4	3	4	5
4	4	4	3	3	2
4	5	4	3	3	4
5	4	5	1	4	1
5	4	4	1	3	5
4	5	5	4	4	3

3	4	5	4	5	5
4	3	5	4	3	4
5	4	5	3	3	2
4	5	5	4	4	3
3	4	4	3	3	5
2	4	5	4	4	3
3	4	5	2	5	4
4	5	5	2	3	4
4	5	1	2	1	5
3	5	5	2	5	4
4	4	3	2	3	4
2	5	5	3	5	5
3	5	3	4	3	4
4	4	4	3	4	5
5	5	4	2	4	5
4	3	3	3	4	4
5	4	5	2	3	5

4	3	5	2	2	4
3	3	5	3	1	4
3	4	5	4	5	5

2.7	2.8	2.9	2.10	2.11	2.12	2.13
4	2	3	3	3	5	5
2	2	2	2	5	5	4
4	3	3	2	5	5	4
4	2	3	2	3	4	4
4	3	3	2	4	4	4
3	2	2	3	4	3	5
5	3	3	3	3	4	5
5	2	3	3	5	3	5
4	3	2	3	2	4	4
4	2	5	3	5	5	4
4	3	4	3	3	4	4

5	1	3	2	4	3	3
3	3	3	2	3	5	3
5	1	3	2	3	4	3
4	3	3	2	4	2	3
5	2	2	1	5	4	3
5	1	2	2	3	4	3
2	1	2	2	3	4	4
3	1	1	2	3	5	4
3	1	1	3	4	3	2
3	1	1	3	3	3	2
2	3	3	3	5	5	4
3	1	3	4	4	5	3
3	3	2	3	3	2	2
4	4	2	4	3	3	2
5	1	2	3	4	4	3
5	1	2	4	4	3	3
4	3	1	3	5	4	5

4	2	1	3	1	4	4
4	1	3	4	2	3	3
5	1	4	3	4	4	2
4	3	5	4	4	3	3
2	1	2	3	4	3	4
4	2	4	3	3	5	5
3	2	3	3	4	5	4
4	1	3	3	3	5	3
4	1	3	3	4	4	2
4	2	2	3	5	5	3
5	3	2	3	4	5	4
5	2	3	3	4	4	3
5	2	2	3	3	3	2
4	3	3	4	5	1	3
3	2	2	4	4	5	2
3	1	2	4	5	5	2
5	1	2	4	4	4	3

3	1	2	3	5	4	4
4	3	3	2	5	4	4
4	2	3	3	5	3	5
3	1	3	3	5	5	5
4	1	3	4	5	4	4
4	3	2	3	4	4	3
5	1	2	3	5	5	4
5	2	3	4	4	5	3

3.1	3.2	3.3	3.4	3.5	3.6
4	5	5	4	5	4
2	5	5	3	4	4
1	4	5	4	5	3
3	4	5	3	2	4
3	4	4	3	2	3
2	4	4	2	2	2

3	4	4	2	2	2
3	4	4	2	2	2
3	5	3	3	2	4
4	5	4	4	2	2
3	5	5	3	3	2
3	5	4	3	3	4
3	5	3	2	3	3
2	5	4	4	2	2
3	5	3	4	2	2
2	4	3	4	2	3
3	4	3	4	3	3
2	5	3	5	2	3
4	4	4	3	5	4
3	3	3	3	3	2
3	4	3	2	3	4
2	5	4	2	2	2
3	3	3	3	3	3

3	4	3	4	3	3
2	5	4	4	2	4
4	5	5	4	4	4
3	5	4	3	3	3
3	4	3	2	3	4
3	3	4	3	3	3
4	4	4	4	4	4
4	5	4	4	4	4
5	4	4	4	5	2
5	3	3	3	5	2
2	4	3	5	4	3
3	5	3	3	3	3
3	4	3	5	5	3
2	4	3	3	3	3
2	4	4	4	4	3
4	4	4	4	3	3
3	5	4	5	5	2

4	4	4	4	5	3
3	4	3	2	5	2
4	4	3	4	4	2
5	5	4	5	5	3
4	4	5	5	5	2
4	4	5	3	5	2
4	4	4	4	5	3
3	5	3	4	4	2
4	5	4	3	5	2
3	5	3	3	3	3
3	5	3	5	5	2
4	3	4	5	5	4
4	4	3	4	5	4

3.7	3.8	3.9	3.10	3.11
4	3	4	4	3

2	4	4	4	3
4	5	4	4	4
2	3	2	5	4
3	3	4	5	4
2	2	3	5	5
2	4	4	4	4
2	2	2	5	3
2	3	3	4	4
2	3	4	3	4
4	3	3	4	4
3	3	4	4	3
3	3	3	3	3
2	2	3	4	2
2	3	3	5	3
4	3	3	4	4
3	4	3	3	3
4	3	2	4	3

4	4	4	5	3
2	3	4	4	3
3	3	3	3	4
3	2	2	4	4
2	3	4	3	4
3	3	3	3	4
3	3	4	4	3
4	3	4	4	3
3	4	4	4	3
2	3	4	3	4
3	3	3	3	4
3	3	3	4	3
4	3	4	5	4
3	4	3	4	3
4	3	4	3	4
2	4	3	4	4
3	3	4	3	3

3	4	5	4	3
3	3	3	5	4
3	3	3	4	5
3	3	3	3	4
3	3	4	4	3
3	4	4	4	4
3	4	3	4	5
3	4	4	3	4
4	4	4	4	3
3	4	5	5	4
3	3	5	5	3
3	5	4	4	4
3	4	3	4	5
3	4	5	4	4
3	3	3	3	3
4	5	5	3	4
4	3	3	3	3

4	4	3	4	3
---	---	---	---	---

4.1	4.2	4.3	4.4
3	4	4	5
4	4	4	5
3	5	4	5
3	5	4	4
3	5	5	4
4	4	5	4
4	4	4	4
4	5	4	5
3	5	5	5
3	4	4	5
3	3	4	5
4	4	5	4
3	5	3	3
4	5	3	4

3	4	4	4
3	3	4	4
4	3	5	3
3	4	4	3
4	4	3	4
3	5	4	5
3	4	5	5
3	3	4	5
4	4	5	5
5	5	4	5
4	4	4	5
3	3	4	5
5	4	5	4
3	4	4	3
3	4	3	3
5	4	4	3
4	3	4	3

4	3	5	4
4	4	4	4
4	5	3	5
5	5	4	5
4	5	5	5
4	4	4	5
3	5	5	4
4	4	5	3
2	5	4	5
3	4	5	4
4	3	4	5
5	4	4	5
4	5	4	4
3	4	4	4
4	3	5	4
3	4	5	4
4	5	5	5

3	4	3	5
4	3	3	5
3	4	4	5
5	5	3	4
3	4	3	5

Qualitative Data

2.14	<p>Version information and History</p> <p>Very minimal information</p> <p>less information</p> <p>brief information come</p> <p>sometime depends on component by component but mostly history information ,</p> <p>version information, pricing information, basic technical specs</p> <p>and functional</p> <p>version inform</p> <p>most of the time it comes with basic function and technical specs</p> <p>very basic tech and functional specs</p> <p>basic overview of the component</p> <p>Most common is about component, functionality and sometime main important features listed.</p> <p>Very minimal information like name of component , or basic functionality</p> <p>Pricing and version information and basic information</p> <p>Very basic information with price , history , specific features</p> <p>Overview of component and about the provider</p>
------	--

	<p>Company information and component features</p> <p>Point to point information about component and about some upgrade information, version etc.</p>
2.15	<p>follow on development standard</p> <p>Just replace old one with new one and same time change your code to handle new change in new component.</p> <p>Through proper documentation</p> <p>Proper code design and product is architect so well that it is flexible with upgrades of components.</p> <p>Recrd new changes of new component.</p> <p>Components rule is they should have all features of previous version plus new ones. Up gradation is not a bottle neck.</p> <p>Unless it is absolutely necessary we don't upgrade.</p> <p>Upgrade requires thorough impact analysis and testing and has to be planned properly.</p> <p>For proper upgrades, proper versioning should be done and proper builds should be maintained.</p> <p>Technical specs should be followed while upgrading and specs should be updated afterwards.</p> <p>Just find a new one.</p>

	<p>By following the documentation provided by the vendor.</p> <p>Testing new upgrades in test environment and then migration towards production.</p> <p>ship both the old component for backward compatibility and the new one for new features.</p> <p>By consulting the vendor of component, studying the modifications, testing the upgrade and if successful, put it on live system.</p> <p>Projection of usage.</p> <p>Through research,</p> <p>Clearly define the interfaces (integration interfaces, not visual interfaces) of the component. Then implement the enhancement in the component's internal workings. Then test the component to pass the clearly defined interfaces for its interaction with the system in which it gets plugged in.</p> <p>Firstly upgrades are always tested on test instances and upon upgrade maturity; the upgrade is done on production environment.</p> <p>in touch with vendors and having good knowledge.</p> <p>By maintaining Component versions.</p> <p>It is usually plug n play.</p> <p>I let the provider's consultants worry about it.</p> <p>Keep the backward compatibility.</p> <p>through Version Control</p>
--	--

<p>2.16</p> <p>(a)</p>	<p>We use commercial component when something is beyond our skills or if something we try to develop is already available and different client using it so it means it's reliable.</p> <p>For Commercial Component we use creak copy.</p> <p>Reliable Support.</p> <p>Thhe Stability</p> <p>I think it saves lot of development efforts and why to re invented the already developed wheel.</p> <p>Most of the time we found bug frees Commercial Component which can be easily integrated with little modifications.</p> <p>No need to re invent the alrdy develop wheel.</p> <p>Saves time and effort as there's no need to reinvent the wheel.</p> <p>We have the enough budgets for so why develop them In-House.</p> <p>Commercial components mean they are build and tested. And if it's commercial they should have good support.</p> <p>Component is standard in industry make vs. buy analysis comes out to be in favor of buy.</p> <p>We are system integrators so that is why we emphasize on commercial components to save time. We fill in the gaps by making our own component and taking expertise from across the group.</p> <p>Technical support.</p>
------------------------	--

	<p>Lesser development time</p> <p>Easy to use.</p> <p>The reliable running system without having any last minute surprises at any level.</p> <p>Quality, reliability and warranties/support.</p> <p>Plug n Play.</p> <p>Standardization.</p> <p>Ease of use and getting functionality out-of-the-box rather than re-inventing the wheel.</p> <p>They are more reliable.</p> <p>Saving time.</p> <p>Reduce cost.</p> <p>Having bug-free components.</p> <p>Ease of Management.</p> <p>Least Risky.</p> <p>Tried and tested component.</p> <p>Saving time and money on component development.</p> <p>Time Saving for in-house development.</p> <p>Proven Record.</p> <p>Customization not in need.</p> <p>To avoid reinventing the wheel.</p>
--	---

	<p>To save time.</p> <p>To have external reliable support for the component.</p>
2.16 (C)	<p>I think I could not answer, it changes with requirements.</p> <p>Don't want to get too dependent on other vendors as applications become hard to debug if any issue arises and you can't always trust outside support on a tough deadline.</p> <p>Mostly each project is unique.</p> <p>Culture of development.</p> <p>Customized Solutions needed</p>
2.17	<p>We test them separately and check the desire output by giving some sample data, so it helps us to determine quality of it. For reliability, we check different clients using the same component or different product where the component is used.</p> <p>We make sure that component is certified.</p> <p>The feature that component provides and we do testing from that component.</p> <p>Proper analysis in done before using component.</p> <p>We always prefer known vendor whose solution is matured and reliable in market. We do not test the component because we believe that one of advantage of using component is that we can pass through the testing. This saves our cost</p>

	<p>of testing.</p> <p>Through trail testing.</p> <p>Component without reliability is not a component :). But would be impressed with number of features it offers and how much efficient it is.</p> <p>Research, forums and evaluation.</p> <p>Testing and lots of testing!</p> <p>Black and white box testing.</p> <p>Various testing.</p> <p>In this specific case, quality can be defined to fulfill the requirements. If the component is giving the required output without failing then we say that it is quality and reliability is good. Reliability can be taken further by saying that component's integration into the system shall not affect any other integrated components, i.e. it should be least cohesive.</p> <p>By reading reviews if available and lot of testing.</p> <p>Case studies and years in production.</p> <p>Workshops and through analysis.</p> <p>By testing all business and technical scenarios in Test Environment.</p> <p>Unit testing and building test cases for integration.</p> <p>Peer reviews.</p> <p>Market share.</p> <p>Diversity of the provider's clients.</p>
--	--

	<p>Internet professional reviews.</p> <p>Unit Testing.</p> <p>Security.</p> <p>By the branding of the component vendor. By past history use of the component by others. By our own pilot testing of the component in our own system.</p>
2.18	<p>Same time there bug come from the component that not easy to handle specially when use executable component and do not have the source code.</p> <p>Customizing the control to your own needs.</p> <p>Yes, this is biggest issue we have faced in commercial components. Since we do not we access.</p> <p>That's the bottle neck of using commercial components. It should be properly taken care of from the start that components are what we require.</p> <p>They can be hard to replace if the company goes out of business.</p> <p>Should have after delivery support or it can cause serious financial and relationship troubles with client.</p> <p>Less control over component.</p> <p>Source code not available.</p> <p>Have to pay for upgrades - even for minor changes.</p> <p>It might end up processing something which we don't want.</p> <p>Application failure leading to system restores and therefore wastes of man hours</p>

	<p>and service downtime.</p> <p>Operational and technical risks are involved but they can be different according to the environment.</p> <p>Cost of the components.</p> <p>High maintenance cost.</p> <p>Tech specs not there</p> <p>Unforeseen costs.</p> <p>Upgrades.</p> <p>Src code not there</p> <p>Testing not easy</p> <p>The organization can't build an efficient learning experience since provider will install, configure and maintain the system. There are employees who have a good level of knowledge about the component but not to the degree they are independent of the provider.</p> <p>Component owner support.</p> <p>Missing knwldge</p> <p>Technical details unknown</p> <p>Lack of Customization.</p> <p>Fewer Features.</p> <p>Debugging not easy.</p> <p>When we want to deliver something highly customized to our customers, we face</p>
--	--

	<p>problem of highly restrictive actions that we can take in customizing commercial components. If the documentation is not good enough, or worse, if the vendor created the component not to be customized beyond a certain limit.</p>
2.19	<p>By testing it using some sample data.</p> <p>If the component has the feature that we want i used it.</p> <p>Comparing my requirements with the features it offers.</p> <p>Research for available choices and run a comparison.</p> <p>Provider's background on that particular field and its portfolio.</p> <p>Reviews.</p> <p>By thorough literature review and by setting up the test environment for a specific component.</p> <p>By confirming the results and how they results inline with the requirements.</p> <p>Quality testing.</p> <p>If it matches most of the requirements.</p> <p>Selection of the right component is driven by Business requirements and the ratio of adaptability of that component towards the requirements.</p> <p>Research first before selecting the individual component.</p> <p>Check that the component fulfills the requirements.</p> <p>Demo, trial, features, and professional reviews.</p> <p>Through Testing.</p>

	<p>User Feedback.</p> <p>Test Cases.</p>
2.20	<p>We usually face when we require modifying it as per our system.</p> <p>Support Arabic language.</p> <p>Lack of training lack of support.</p> <p>Usually platform related if you are working across different platform.</p> <p>Application design change.</p> <p>Network workflow change.</p> <p>Lack of informant about the sourc component give us prb when we perform integration</p> <p>Dependency of the specific component on some other non-existent components.</p> <p>Conflicting DLLS</p> <p>Most of the time vendor specific code is the essence of component which poses technical challenges.</p> <p>Various converters or interfaces may differ from case to case.</p> <p>Inadequate documentation, received what is not expected, lack of support, too complex to understand (inadequate and substandard coding and documentation), no version control. Doesn't support the deployment environment (non interoperable).</p> <p>Documentation is not complete or sufficient</p>

	<p>Unkown design</p> <p>Not enough knowledge about the selected component.</p> <p>Mismatch</p> <p>Normally if we have 2 components from different vendors we have to write an ETL interface for getting the components to talk to each other. Similarly, components need an interface to talk to our legacy systems.</p> <p>Interfacing matching</p> <p>Dependency</p> <p>Especially in integrating heterogeneous components that do not understand each other, may need to create a wrapper layer for converting one components output into a format understandable by the other component.</p>
2.21	<p>We keep in touch during each cycle of component integration.</p> <p>By email.</p> <p>Don't you think it's his Job?</p> <p>Through channels department.</p> <p>We prefer to do a service contract for a define amount of time, say for 3 years etc so if they perform well they get the money.</p> <p>Mostly by the support contract.</p> <p>By keeping clear terms and condition in the contracts and keeping good relationships with the technical peers on vendor's side.</p>

	<p>Stringent service level agreements.</p> <p>Making sure they themselves have means to engage their clients like how do they provide ongoing support to their clients? Are they accessible?</p> <p>Pay them for their job.</p> <p>Telephone contact</p> <p>support</p> <p>Frequently interactions.</p> <p>License and support agreement.</p> <p>Through Email Correspondence.</p> <p>Through Personal Relations</p> <p>eMails</p>
2.22	<p>We have some our own test cases and tool use to test the component by giving the some test data and then check and verify the results.</p> <p>Go through different features and see if they work, then test its efficiency. BY efficiency I mean it should not take forever for some fancy work.</p> <p>POC.</p> <p>Integration testing</p> <p>Testing over n over</p> <p>Unit testing and integration testing.</p> <p>Various tests.</p>

	<p>Testing against sample data</p> <p>Make Test Cases.</p> <p>Automated Testing Tools</p> <p>Running it on the test instance.</p> <p>By using any available testing tools available for the particular type of component.</p> <p>Pilot or demo phases are conducted.</p> <p>Sample environment, Component, string and UAT etc.</p> <p>All testing is done on Test environments, which are replicas of Production environment. We have to write UAT scripts to cover all integration aspects for testing.</p> <p>Black box testing.</p> <p>Professional reviews.</p>
2.23	<p>Yes its not easy, because you are not aware with the source code, usually you don't have source code or even if you have then its not easy to understand the code written by someone using their some own standards and declarations.</p> <p>Now a days No.</p> <p>Yes it is.</p> <p>I think it's easier, because you can isolate problem first on functional basis and then target that particular component.</p> <p>Can be in some conditions.</p>

	<p>It depends upon the component vendor. If exception and/or handles are properly written then it's easy otherwise it's difficult to figure out proper meaning of certain errors.</p> <p>I disagree. It is much easier to track and isolate the problem at component level by confirming the results. Another advantage is that it is easier to upgrade the single component as compare to the complete system.</p> <p>Agree, the actual scenarios may contain unforeseen obstacles.</p> <p>It depends how adequate you have written your Test Plans.</p> <p>This is not necessarily true. It depends on the technology on which the component is developed. Generally speaking, components are executable binaries which cannot be debugged if the component is vendor based.</p> <p>Yes it is difficult since the component appears as a black box, one can only check by providing various inputs and checking the output.</p> <p>True, as they are trusted black boxes.</p> <p>I agree because you can attach any process for debugging.</p>
2.24	<p>Features and efficiency.</p> <p>We don't have any certifying process in our firm.</p> <p>Quality assurance.</p> <p>We buy from reliable vendor, no need for certification check.</p> <p>Usually the engineers raise the requirement to deploy certain component and the</p>

	<p>KPIs then justify the continuity of it.</p> <p>Digital certificates are used for login components. The obvious reason for the security is management requirement.</p> <p>Testing over and over again using various approaches and scenarios.</p> <p>Not really.</p> <p>Contact different clients and make sure that they have no problem using it and it's important in order to make sure that selected component is reliable.</p> <p>No need</p> <p>Get trusty one.</p> <p>We follow international standards like COSO and ISO. We also have a IT Compliance and Audit department for this purpose.</p> <p>No certification.</p> <p>Though SM Security</p>
2.25	<p>There is no central repository.</p> <p>FWe have a repository for all project related artifacts, components, source code etc. this is for proper configuration management.</p> <p>Back up purposes.</p> <p>The components are be reused in any new project and by any other development unit. That is why the components are maintained in a central repository.</p> <p>Find the difference.</p>

5.1	<p>Flexibility.</p> <p>Maintenance becomes easier.</p> <p>Error Isolation.</p> <p>It's good to have already built-in components, it save lot of time of development something from scratch and then integrate, you can save resources, cost and time etc.</p> <p>It saves lot of development effort.</p> <p>We can pass - through the testing phase.</p> <p>Maintenance cost is saved.</p> <p>Saves time and effort.</p> <p>Provides re-usability.</p> <p>Rapid application development.</p> <p>Avoids reinventing the wheel.</p> <p>Can reduce overall cost.</p> <p>Saves time and money. Considering if the component is reliable and efficient, it should save your in house development and testing time.</p> <p>Fast delivery of projects. No need to re invent the wheel.</p> <p>Management becomes easy. If properly developed then errors are easy to track.</p> <p>Less costly</p> <p>NO need to develop from scratch</p>

	<p>Easy</p> <p>Save time and cost</p> <p>Standard design and confirmation of output.</p> <p>Knowledge should be at par in regards of the technology that you are integrating for a client.</p> <p>Ready to use</p> <p>Ability to purchase what is required and avoid what is not required. Require a standard body which can establish standard protocols for integration.</p> <p>No need to spend time to develop something which is already there in market</p> <p>Less cost more output.</p> <p>Re-usability, portability, easy maintenance, better quality. Write once, user everywhere. It's easier to ensure the quality of the code and maintenance as the entire application is divided in different isolated units. Each unit/component is only concerned with its own implementation and how it is exposing its services to the client. etc.</p> <p>Integrated components can be reused as per business scenario.</p> <p>Reusability.</p> <p>Fast</p> <p>Ease of use Reuse of component Ease of functionality.</p> <p>Code reusability results in decreased development time consistency.</p> <p>We achieve a high level of abstraction since the designer makes the</p>
--	--

	<p>programmer's job easier by reasoning about individual concerns in isolation from.</p> <p>Save time</p> <p>Reliable</p> <p>They are reliable. They are fast to implement if the organization dedicates proper resources.</p> <p>Component based implementations drive No-Single-Point-Of-Failure approach, are easier to manage and highly scalable.</p> <p>Time and cost saving. Reliable and tested components doing work. Reuse of code.</p> <p>Save efforts</p> <p>Efficiency and fast</p> <p>No need to reinvent.</p> <p>Less time for development.</p> <p>If the components are well defined with interfaces, with good documentation, it can speed up building systems by integrating well built components. It helps avoid rebuilding the wheel.</p> <p>Verify functional, performance, and reliability.</p>
5.2	<p>Complexity</p> <p>External dependency.</p> <p>Lack of control.</p>

	<p>Lack of information about the source component, not much control over it, relationship with the provider of the component etc.</p> <p>Lack of proper information.</p> <p>Unknown architecture.</p> <p>Inability to customize.</p> <p>Lack of full access to source code.</p> <p>No support from vendor</p> <p>Poor after sales support.</p> <p>Can't be customized to business specific needs.</p> <p>Finding the right components that meets your need may be difficult.</p> <p>Later you may find out that you made a mistake selecting the component.</p> <p>Less flexibility.</p> <p>As it's not your own code, it might give you bottle necks later.</p> <p>Support after production. Changes required. No source code.</p> <p>I don't see any.</p> <p>Require high level skill set, standards can change for component development.</p> <p>Lack of skills and not able to see the whole picture while integrating the components.</p> <p>Complex structure</p> <p>Hard to get vendor support after buying component</p> <p>If particular standard body doesn't exist then it could easily fail the whole system.</p>
--	---

	<p>Maintenance Cost to maintain components, vendors etc which can easily blow out the costs.</p> <p>Lots of learning.</p> <p>Not much as long as one component can work independent of another. If there are different providers of the components then the interfaces/contract (how the component will interact with other comps and vice versa) for that component must be clearly defined. Could be hard to find an issue if it is closed source. etc.</p> <p>None.</p> <p>Vendor problem</p> <p>Success of integration depends upon whether the component has been built well or not. Limitations in component may not match our requirement of the system we are trying to build using the components.</p> <p>Iteration and Impact Analysis</p> <p>Lack of control on the functionality.</p> <p>Customization problem.</p> <p>Lack of control over source code. Decrease in security. Increase in dependencies. may affect performance , because of overhead in communication between components.</p> <p>Separation of crosscutting concerns provides the software with a loose coupling between the different concerns, achieving the usability of a single concern.</p> <p>Complexity</p>
--	--

	<p>Less control</p> <p>The know-how is not built as it should be.</p> <p>Documentation.</p> <p>Dependence on the knowledge of internal working of a component.</p> <p>Lot of glue code needed sometimes to fit components in related applications.</p> <p>Lack of complete information about component behavior.</p> <p>Security Issues.</p> <p>Conflict of Components.</p>
	<p>More quantitative and with more point to point question</p> <p>Nice one but make survey little short,</p> <p>Too many questions</p> <p>Good to start</p> <p>Should be face to face interview as well for these questions</p> <p>Send me end results</p>

REFERENCE

1. Mahmood, S., R. Lai, and Y.S. Kim, *Survey of component-based software development* IET Software 2007. **1 Issue:2** (April 2007): p. 57 - 66.
2. Crnkovic, I. and M. Larsson, *Challenges of component-based development*. The Journal of Systems and Software, 2002. **61**(3): p. 201 - 212.
3. Mahmood., S., *The impact of acceptance tests on analyzing component-based systems specifications: An experimental evaluation*, in *10th International Conference on Computer and Information Technology*. IEEE Computer Society 2010. p. 241 – 248.
4. Kotlarsky, J. and I. Oshri, *Managing Component-Based Development in Global Teams*, ed. P. Macmillan. 2009.
5. Szyperski, C., D. Gruntz, and S. Murer, *Component Software - Beyond Object - Oriented Programming*, ed. Addison-Wesley. 2002.
6. Li, J., *Process Improvement and Risk Management in Off-The-Shelf Component-Based Development*, in *Department of Computer and Information Science*. 2006, Norwegian University of Science and Technology.
7. Brown, A.W. and K. Short, *On Components and Objects: The Foundations of Component-Based Development*, in *5th International Symposium on Assessment of Software Tools*. 1997.
8. Crnkovic, I., *Component-based Software Engineering - New Challenges in Software Development*, in *Information Technology Interfaces, 2003. ITI 2003. Proceedings of the 25th International Conference on 2003*. p. 9 - 18.
9. Li, J., et al., *Development with Off-the-Shelf Components: 10 Facts*. IEEE Computer Society 2009. **26**(2): p. 80-87.
10. Alejandra Cechich, M.P., and Antonio Vallecillo, *Assessing component based systems*, in *Component Based Software Quality*. 2003.
11. Crnkovic, I. and M. Larsson, *Building Reliable Component-Based Software Systems*. 2002: Artech House Publishers.
12. Mendenhall, W. and T. Sincich, *Statistics for Engineering and the Sciences*. 5th ed. 2006: Prentice Hall. 1060.
13. Capretz, L.F., M.A.M. Capretz, and D. Li, *Component-Based Software Development*, in *The 27th Annual Conference of the IEEE Industrial Electronics Society*. 2001.
14. Mahmood, S., et al., *A survey of component based system quality assurance and assessment*. Information and Software Technology 47, 2005. **47**(10): p. 693-707.

15. Qureshi, M.R.J. and S.A. Hussain, *A reusable software component-based development process model*. Advances in Engineering Software, 2008. **39**(2): p. 88-94.
16. Kotonya, G. and A. Rashid, *A Strategy for Managing Risk in Component-based Software Development*, in *27th Euromicro Conference*. 2001.
17. Asokan, S., *Component Based Software Engineering*. 2007, Cochin University of Science and Technology.
18. Wallnau, A.W.B.a.K.C., *The current state of CBSE*. IEEE Software, 1998. **15**(5): p. 37 - 46.
19. Heineman, G.T. and W.T. Councill., *Component-Based Software Engineering: Putting the Pieces Together*. 2001: Addison-Wesley Professional.
20. Rader., J.A. *Mechanisms for integration and enhancement of software components*. in *Proceedings Fifth International Symposium on Assessment of Software Tools and Technologies*. 1997.
21. Li, J., et al., *A State-of-the-Practice Survey of Risk Management in Development with Off-the-Shelf Software Components*. IEE Transactions on Software Engineering, 2008. **34**(2): p. 271 - 286.
22. Dedolph, F.M., *The Neglected Management Activity: Software Risk Management*. Bell Labs Technical Journal, 2003. **8**(3): p. 91-95.
23. Boehm, B.W., *Software Risk Management: Principles and Practices*. IEEE Software, 1991. **8**(1): p. 32 - 41.
24. Pressman, R., *Software Engineering: A Practitioner's Approach*. 6th ed. 2005: McGraw-Hill.
25. Lewis, G.A., S.A. Hissam, and R.C. Seacord. *Building Systems from Commercial Components*. in *International Conference on Software Engineering, Proceedings of the 24th International Conference on Software Engineering*. 2002.
26. Beaver, J.M., G.A. Schiavone, and J.S. Berrios. *Predicting Software Suitability Using a Bayesian Belief Network*. in *Proceedings of the Fourth International Conference on Machine Learning and Applications*. 2005.
27. Rashid, A. and G. Kotonya, *Risk Management in Component-based Development: A Separation of Concerns Perspective*, in *ECOOP Workshop on Advanced Separation of Concerns (ECOOP Workshop Reader)*. 2001.
28. Mahmood., S., *The impact of acceptance tests on analyzing component-based systems specifications: An experimental evaluation*, in *10th International Conference on Computer and Information Technology*. 2010. p. 241 – 248.
29. Rine, D., N. Nada, and K. Jaber, *Using adapters to reduce interaction complexity in reusable component based software development*, in *Proceedings of the 1999 symposium on Software reusability*, ACM Press. 1999.
30. Dietrich, S.W., et al., *Component adaptation for event-based application integration using active rules*. Journal of Systems and Software, 2006. **79**(12): p. 1725 – 1734.
31. Parrish, A., B. Dixon, and D. Cordes, *A conceptual foundation for component based software deployment*. Journal of Systems and Software,, 2001. **57**(3): p. 193 – 200.

32. Xia, Y., A.T.S. Ho, and Y. Zhang. *Cimo - component integration model*. in *Proceedings of Seventh Asia-Pacific Software Engineering Conference, APSEC 2000*. 2000.
33. Depke, R., et al. *Process-oriented, consistent integration of software components*. in *Proceedings of 26th Annual International Computer Software and Applications Conference, COMPSAC 2002*. 2002.
34. Assman, U. *A component model for invasive composition*. in *ECOOP 2000 Workshop on Aspects and Dimensions of Concerns*. 2000. Cannes, France.
35. Suvee, D., W. Vanderperren, and V. Jonckers. *Jasco: An aspect-oriented approach tailored for component based software development*. in *Proceedings of the 2nd International Conference on Aspect-Oriented Software Development, ACM Press*. 2003. Boston, USA
36. Li, J., et al., *Product Focused Software Process Improvement. A Study of Developer Attitude to Component Reuse in Three IT Companies*. Vol. Volume 3009/2004. 2004: Springer Berlin / Heidelberg.
37. Li, J., et al. *An empirical study on decision making in off-the-shelf component-based development*. in *International Conference on Software Engineering, Proceedings of the 28th international conference on Software engineering*. 2006.
38. Vitharana, P., *Risks and challenges of component-based software development*. Communications of the ACM, 2003. **46**(8): p. 67 - 72
39. Kitchenham, B.A. and S.L. Pfleeger, *Principles of survey research: Parts 1 - 6*. ACM SIGSOFT Software Engineering Notes 2001 - 2002.
40. Ahmed, F. and L.F. Capretz, *The software product line architecture: An empirical investigation of key process activities*. Information and Software Technology 2008. **50**: p. 1098–1113.
41. Ahmed, F. and L.F. Capretz, *Managing the business of software product line: An empirical investigation of key business factors*. Information and Software Technology, 2006. **49**: p. 194 - 208.
42. Lowry, R. *Spearman Rank-Order Correlation Coefficient*. 1998-2011; Available from: http://faculty.vassar.edu/lowry/corr_rank.html.
43. Rumsey, D., *Statistics For Dummies*. 2003 John Wiley & Sons
44. Simon, S. *What is a correlation*. 2008; Available from: <http://www.childrensmency.org/stats/definitions/correlation.htm>.
45. StatSoft. Available from: <http://www.statsoft.com>.
46. Boehm, B. and C. Abts, *COTS integration: Plug and Pray?* IEEE Computer, 1999. **32**(1): p. 135-138.
47. Alves., C., *Cots based requirements engineering. Component Based Software Quality*. 2003.
48. Piattini, A.C.a.M., *Early detection of cots component functional suitability*. Information and Software Technology 2007. **49**(2): p. 108-121.

49. Maiden, N.A. and C. Ncube, *Acquiring cots software selection requirements*. Software, IEEE 1998. **15**(2): p. 46 - 56.
50. Leung, K.R.P.H. and H.K.N. Leung, *On the efficiency of domain-based COTS product selection method*. Information and Software Technology, 2002. **44**(12): p. 703 - 715.
51. Alves, C. and A. Finkelstein, *Investigating conflicts in cots decision-making*. International Journal of Software Engineering and Knowledge Engineering, 2003. **13**: p. 1-21.
52. Gao, J.Z., H.-S.J. Tsao, and Y. Wu, *Testing and Quality Assurance for Component Based Software*. 2003: Artech House.
53. Ding, Y. and N. Napier. *Measurement Framework for Assessing Risks in Component-based Software Development*. in *Proceedings of the 39th Hawaii International Conference on System Sciences*. 2006.
54. Pan, J., *Software Testing*, in *Dependable Embedded Systems*. 1999.

Vita

Name: Azhar S. Khan

Nationality: Pakistani

Born: 5th June 1981

Summary:

- Part-time graduate student of ICS at KFUPM.
- BS degree in computer science from Iqra University (IU), Karachi, Pakistan.
- 8 years of industrial experience in software engineering.
- Area of interest includes Component Based Software Development, Software Outsourcing, and Microsoft Technologies, ERP systems & SharePoint, Database Design, Analysis and Optimization.

Current Address: P.O.BOX 75021, Khobar 31952, Dhahran, K.S.A

Permanent Address: Malir Cant, Near Jinnah International Airport, Karachi, Pakistan.

Email: azharkhan5@gmail.com

Linkedin: <http://www.linkedin.com/in/azharsaeedkhan>