

**ASSESSMENT OF SOFTWARE ARCHITECTURE AND  
DESIGN FOR OFFSHORE PROJECTS**

Tanya Kudchadker

Nivedita Angadi

August 2010

---

# Table of Contents

**Assessment of the Software Architecture and Design for Offshore Projects vi**

<b>Abstract</b>	<b>vii</b>
<b>Acknowledgement</b>	<b>1</b>
<b>1. Introduction</b>	<b>3</b>
1.1 Motivation	5
1.2 Background Information	5
1.3 Research Question	6
1.4 Document Overview	7
<b>2. Offshoring : A Capgemini perspective</b>	<b>9</b>
2.1 Purpose	9
2.2 Onsite-Offshore model	10
2.2.1 Benefits	11
2.2.2 Challenges	11
<b>3. The standards, models and techniques used</b>	<b>14</b>
3.1 LaQuSo Software Certification Product Model (LSPCM)	14
3.2 Capgemini Accelerator for Software Architecture (CASA)	16
3.3 Rational Unified Process (RUP)	17
3.4 Analysis and Design Artifacts	18
3.5 Related Work	22

<b>4.</b>	<b>Case studies</b>	<b>29</b>
4.1	<i>Introduction</i>	29
4.2	<i>Approach</i>	29
4.3	<i>Case study 1 : An Electronic ticket printing application</i>	31
4.3.1	Case description	31
4.3.2	Assessment Results	31
4.3.2.1	LSPCM	32
4.3.2.2	CASA	38
4.3.3	Remarks on LSPCM	50
4.3.3.1	High Level Design	50
4.3.3.2	Detailed Design	56
4.3.4	Remarks on CASA	57
4.3.4.1	Software Architecture Document (SAD)	57
	Use case realization	60
4.4	<i>Case Study 2: A Banking application</i>	61
4.4.1	Case description	61
4.4.2	Assessment Results	61
4.4.2.1	LSPCM	62
4.4.2.2	CASA	68
4.4.3	Remarks on LSPCM	74

4.4.3.1	High Level Design	74
4.4.3.2	Detailed Design	76
4.4.4	Remarks on CASA	78
4.4.4.1	Software Architecture Document	78
4.4.4.2	Use case realization	80
4.4.4.3	Analysis Model	80
4.4.4.4	Design Model	80
4.4.4.5	Implementation Model	81
4.4.5	Recommendations for the case studies documentation	81
<b>5.</b>	<b>Observations</b>	<b>83</b>
5.1	<i>LSPCM</i>	83
5.1.1	High Level Design	83
5.1.2	Detailed Design	89
5.2	<i>CASA</i>	91
<b>6.</b>	<b>Conclusion</b>	<b>93</b>
6.1	<i>High Level Design</i>	93
6.1.1	Problems & Impact	93
6.1.2	Solution	94
6.2	<i>Detailed Design</i>	95
6.2.1	Problems & Impact	96

6.2.2	Solution	96
6.3	<i>Result</i>	97
6.4	<i>Benefits to LaQuSo (Laboratory of Quality Software)</i>	97
6.5	<i>Benefits to Capgemini</i>	98
6.6	<i>Future work</i>	98
<b>7.</b>	<b>Bibliography</b>	<b>100</b>
<b>Appendix A.</b>	<b>List of Supplements</b>	<b>103</b>

---

# Assessment of the Software Architecture and Design for Offshore Projects

Tanya Kudchadker

Nivedita Angadi

Eindhoven University of Technology(TU/e)

Department of Mathematics & Computer Science

Software Engineering & Technology Group

P.O.Box 513, 5600 MB Eindhoven, The Netherlands

---

## Abstract

Lowering the software development cost has always been a prime focus of the software services providing company. Due to this, IT projects and software development outsourcing has become the most popular and the most successful business process, enabling the companies to create highly competitive solutions with considerable cost reduction and in a shorter project time and other business benefits which offshore offers. In this project we consider the software architecture and analysis & design for offshored projects. The documentation during this phase remains the vital step. We analyze the quality of these documentation for the offshore projects.

To assess the quality of software architecture and analysis & design discipline the following approaches were used : the LaQuSo Software Product Certification Model(LSPCM) developed by Laboratory of Quality Software(LaQuSo) for assessing product quality, Capgemini Accelerator for Software Architecture (CASA) developed by Capgemini for Software architecture and design. The research question is *“How can we improve the quality of software architecture and design for offshoring”*. For this we check the suitability of methods used for software architecture and design i.e. RUP and CASA and quality assessment model i.e. LSPCM for offshoring.

Thus, in this research study we critically analyze LSPCM and CASA by conducting detailed case study evaluation for offshore projects undertaken by Capgemini. We focus mainly on two project areas High Level Design and Detailed Design as the software architecture and analysis & design is mapped to High Level Design and Detailed Design in LSPCM terms.

During the analysis of these case studies, we identified the limitations of LSPCM and CASA for offshore projects. In order to address these limitations, we propose suggestions for

improvements for both LSPCM and CASA. In order to verify these suggestions, it has been implemented on these case studies and in turn we have suggestions for the improvement of the artifacts of these industrial projects undertaken by Capgemini.

---

## Acknowledgement

No creation in this world is a solo effort. Neither is this master thesis project. There have been numerous people associated with this piece of work. We, Tanya and Nivedita thus consider this opportunity to thank each and everyone individually.

First and foremost we would like to thank Eindhoven University of Technology(TU/e), Eindhoven, The Netherlands and Manipal University, Manipal, India for providing us with the opportunity to be a part of this dual master degree program.

Time just flies when one is in the company of cheerful and friendly colleagues and supervisors at work. We encountered the similar situation during our internship with Capgemini and Laboratory of Quality Software(LaQuSo). We would like to extend our sincere gratitude to our guide at ir. Martijn Klabbers for his immense support and valuable guidance. His help was a guiding path at every stage during the project right from the structuring the project proposal to reviewing this master thesis report. The weekly presentation sessions conducted by him were indeed helpful for us to improve the performance in a big way.;

The graduation supervisor Prof. Dr. Mark van den Brand for his valuable help and guidance. His comments on the project proposal as well as the presentations were a boon.;

Dr. Alexander Serebrenik for devoting his valuable time for attending the progress presentations and providing critical comments as well as guidelines during these sessions.

We would like to thank our supervisor Drs. Cornelly Spier. We are indeed indebted to her for help during our stint with the organization. We further extend our gratitude to Rob Ista for his valuable inputs during the initial stages of the project.

We are thankful to Mr. Arjan Hendriksen for his guidance and help throughout the duration of the project. In spite of his busy schedule he devoted time for helping us by addressing all the queries with the practices followed within Capgemini.

We would also like to thank Mr. Rob Boomsma for his valuable help with the all the administrative and legal matters with the organization.

We would also extend our sincere gratitude to Mr. Evert-Jan de Raadt, Mr. Jochem Smit, Mr. Aditya Jha, Mr. Merlijn Sluis, Mr. Shylesh Chowla and Mr. Sicko Hempenius for their help during the project. We also thank the members of the CASA team Mr. Sybren de Hartog, Mr. Herre Kuijpers for helping us with quick understanding of the CASA technique and its usage.

We are grateful to our supervisor from Manipal University Dr.(Prof.)Manohara Pai M.M, Dr. Radhika M. Pai for providing the necessary guidance. We would also like to thank our guide Mr. Preetam Kumar for his help and support.

We would like to thank Dr. Prof. Mark van den Brand, Dr. Prof. Jos Baeten, Drs. Cornelly Spier, ir. Martijn Klabbers for being the members of the graduation committee.

Last but not the least we would like to thank Ms. Soundarya Moorthy and Mr. Jeevan Prabhu for their critical comments on the presentation slides and the documentation during the progress presentations held at LaQuSo. We are thankful to all the people from Capgemini and LaQuSo who have helped us successfully complete this project.

We also thank all our friends for their help and support during this project.

---

# 1. Introduction

*Software architecture* is an abstraction of a system. It defines the system elements and how they interact. The right architecture paves the way for system success whereas the wrong architecture usually spells some form of disaster.

To begin with the definition, The *software architecture* of a computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them[9]. It does not concern itself with the details of the system, but rather how components relate to one another. *Software Design* follows the decisions made during the *software architecture* phase.

Software quality is an important factor to be considered throughout the software development life cycle(SDLC) phases. The software quality is the totality of features and characteristics of a software product that bear on its ability to satisfy stated or implied needs[14]. When the software quality is in consideration, it entails specification of attributes of code quality, design quality and documentation quality.

The quality and longevity of a software system is primarily determined by its architecture[25]. The software architecture of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them[9]. Hence, it is essential to ensure that the quality is preserved in the software architecture and design artifacts when the software specifications are converted to design decisions.

There is no official definition for “high-quality” artifacts. However as per [19], the quality with respect to design is defined in the following manner.

A “high-quality” design is one that is

- Understandable, consistent and complete
- Modular, simple and traceable
- Technically adequate

Hence we have followed this definition in the assessment to ensure the presence of quality in artifacts. Considering the future and growing attention to IT offshoring, it is interesting to investigate how companies handle most of the design related aspects for the offshore projects.

Offshoring is practiced by most of the western countries by offshoring the work to the lower wages countries. The quality should be consistent even when the software has been developed in the country other than the original one. Hence a sound design of software is a critical factor especially in the scenario where two nations with disparate cultures are involved. Globalization and rapid improvements in the Information and Communication Technologies have resulted in closely integrated global labor and capital market, where companies have a greater access to human capital scattered around the globe[8].

As a part of this research project we focus on the software architecture for the offshored projects wherein two different countries and their diverse cultures etc. are involved.

## 1.1 Motivation

The research conducted in [26], which focused on requirements gathering and analysis formed as the basis for our research. In this, case studies were considered in which the requirements are gathered onsite and design and development stages have been offshored. Hence in our research work, we analyze the LSPCM model in comparison with company standards for design documents. i.e. CASA(Capgemini Accelerator for Software Architecture) and consider the case studies in which the High Level Design was initiated onsite and continued at offshore, whereas the Detailed Design was done at the offshore end followed by the implementation stages. Since the certification has distinct criteria for High Level Design(HD) and Detailed Design(DD), it can be applied on the design documents for offshore projects.

## 1.2 Background Information

The studies carried out by WSR Consulting group[20] state that 60% of the outsourced projects fail due to the miscommunication between the onsite and offshore locations.

A Software project is considered failed in the following scenario:

- The software did not meet the customer requirements.
- The software did not meet the delivery deadline.
- The actual cost of the software is significantly higher than its estimated cost.
- The software was delivered with too many errors.

This indicates that the project artifacts should be of “high-quality” as described in the previous paragraphs before being handed over from onsite to offshore locations.

A good software architecture and design is the most significant means to ensure reliability of the system. It involves the use of more abstract and general means of specifying the parts of the software and its functionality in the High level design, whereas the same details are made more concrete during the detailed design phase.

Common impediments in achieving architectural success according to [25] are:

- Lack of adequate architectural talent/experience.
- Insufficient time spent on architectural design and analysis.
- Failure to identify the quality or non-functional requirements and design for them.
- Failure to properly document and communicate the architecture
- Failure to ensure that the architecture directs the implementation
- Failure to evolve the architecture and maintain documentation that is up to date.

The documentation related to the software architecture and design plays an important role in the software architecture. These act as the basis for the further stages for the development. Hence the presence of errors in these artifacts will be taken further in the implementation phases leading to the project failure.

### 1.3 Research Question

The goal of our master thesis assignment is structuring and verifying the suggestions for improvement for software architecture and design for offshored projects. The verification of the suggestions for its correctness is done on the projects undertaken by Capgemini and presented as case studies so as to minimize the defects caused due to miscommunication of software architecture and design decisions. In order to achieve this goal, as an initial step we

have considered analyzing the certification model of LaQuSo (i.e. LSPCM) and Capgemini techniques for analysis & design discipline (CASA). The second step involves assessment of the projects undertaken by Capgemini using LSPCM and CASA. Finally, based on the assessment results we formulate suggestions for LSPCM as well as CASA for improving the quality for the analysis & design discipline. These suggestions are verified on the industrial cases undertaken by Capgemini and the suggestions are in turn provided for improvement of these project documentation based on the assessment.

To assess the analysis & design discipline we have considered the following software product areas as categorized by LSPCM :

- High Level Design
- Detailed Design

Assessment of High level design involves verification of the quality of the High Level Design artifacts. Recommendations will include suggestions for improving LPSCM and CASA for improving quality of High Level Design.

The Detailed Design focuses on checks for the detailed design artifacts and focuses on assessing the quality of each of the design. Hence we have chosen these two product areas as it covers the software architecture and design completely.

## 1.4 Document Overview

Chapter 2 discusses the offshoring concept with Capgemini point of view. It details the handling of the software development tasks within Capgemini for offshored projects and various terminologies involved with the practice. Chapter 3 provides details about the related work and introduces the specific techniques used for assessing the software

architecture & design quality. Chapter 4 shows the details on the case studies and also highlights the major defects encountered in these project artifacts. Chapter 5 depicts the observations about the techniques used i.e. LSPCM and CASA and the suggested improvements for them. Chapter 6 concludes the research work with problem and its impact, solution, result and future work. Sections related to **High Level Design** are owned by Nivedita whereas the section related to **Detailed Design** are written by Tanya. The rest of the contents of the report is a combined effort.

---

## 2. Offshoring : A Capgemini perspective

This chapter describes offshoring practice from Capgemini point of view. It details on the various terminologies used within the organization where offshoring of the tasks is involved.

There is no standard definition for the term Offshoring . The location where the work is offshored is referred to as an “Offshore” location or “Offshore Development Centre”(ODC). Offshore can be any country outside the home country. It is referred to as shifting of tasks to low-cost nations, rather than to any destinations outside the country. Low-cost nations are those that fall into the category of “developing nations”. For example: A British software firm does not to its US software research centre as an “offshore site”. Different companies use different terms for offshoring such as Best shore by EDS, Offsource by HCL Technologies and Rightshore by Capgemini.

At Capgemini, the onsite location is referred to as an Accelerated Development Center(ADC). The ADC performs the client communication related activities. Moreover, the project related documentation standards i.e. templates etc, delivery approaches are decided here. Capgemini takes into consideration the various skill set and the other resources available before an offshore location is chosen for development. Moreover, Capgemini is also involved in the research over the offshore locations whereby it analyses the available prospective offshore locations based on the skills and domain knowledge available.

### 2.1 Purpose

In today’s challenging economic environment, businesses are seeking to reduce costs and enhance growth. They need business and IT solutions that promote innovation and

transformation of user needs to a acceptable working system, while helping to create and sustain a competitive advantage. Businesses can achieve this goal with a scalable approach to global delivery and sourcing that combines quality, efficiency, talent and collaboration.

Rightshore<sup>®</sup>, Capgemini's global delivery model, helps add value while using resources more effectively. The best talent in combination with right balance on onshore, nearshore and offshore locations are identified to work as a unified team.

## 2.2 Onsite-Offshore model

There are no predefined rules on which of the software development can be done offshore. However, there are certain activities that are a better fit at offshore locations while others are better done at onshore locations.

Over the past 40 years, Capgemini has developed into a global multicultural company. During this time more than 500 projects have been delivered using the Rightshore approach. For each of the project executions, Capgemini analyzes which activities are suitable for offshoring based on the skill set available. Activities that tend to stay are those that require customer interaction, customer proximity, deep domain knowledge and cultural knowledge. Thus requirements gathering activities early in the development cycle are conducted onshore because of the need to be close to customers, which involves meeting them and talking to them in their own language.

The design phase is carried on both at onsite and offshore end. High-level design is often done onshore while the detailed design is offshored. The coding phases are completely carried on offshore whereas testing is offshored in part.

*\*Rightshore<sup>®</sup> is a trademark of Capgemini*

Testing phases are also carried on at offshore locations, final system integration testing is however carried on at onshore location to monitor quality. For intermediate work products a good review procedure from both locations helps keeping mutual understanding.

### 2.2.1 Benefits

Cost reduction is the primary focus of offshoring work to the developing nations. However, [6] states while cost reduction is the primary strategic focus of most of the companies that are offshoring, it is not the only advantage of offshoring the work to the low wages countries. The abundant supply of skilled human resources which offshore development centre offers companies provides greater agility to assign a large number of engineers to a problem and respond to a business need within days instead of months[6]. Most of the developing nations possess skilled personnel and also the population in these nations are English speaking which makes them one of the most effective key factors for work being offshored.

### 2.2.2 Challenges

Companies may choose to offshore their work due to the potential savings in wage and other benefits involved, but companies cannot turn a blind eye to the range of management, people and other issues it raises. There are a numerous challenges posed by offshoring. However [6] has listed the following most challenging factors which prove to the most dominant hurdles in the tasks which are offshored.

### **Communication breakdown**

Offshoring involves working with people far away, with whom communication can be conducted via channels such as eye-catchers and C-ports are mainly used in communications which involve more than just text. An eye-catcher is a videophone which combines eye contact and a picture of a person on the other side, as if the person on the other side is in the same room. Similarly, a C-port is a further step than conference calls or a video conferencing room. C-Port systems use mirrors & some 3D effects to make the meeting as natural as possible. At project start up at offshore location, it is recommended that the project manager, system analyst and software architect from the onshore location visit the project team on offshore location as the most effective kick-off. Despite of all the modern communication tools and visiting each other once a while from onsite and offshore side, it is mutual trust that is the hard part when you are not sitting on the same desk, you must rely on your counterpart and keep the team feeling.

### **Cultural barriers**

Offshoring also involves working with a “foreign” culture. Each culture has different beliefs, values, communications. As a result of these differences, it can leads to misinterpretation of messages. The team leads on both locations can do a good job if they manage to reach the one team feeling since that helps in understanding and accepting each other’s cultures. Rather than a front- and back office approach were people act sometimes more as enemies than as friends.

### **Documentation and Handover points**

The documentation and handover moments pose the largest challenge to the offshore projects. The solution to the challenge of handover is to document the details appropriately. For example: Using a standard language or maintaining the translation document for the documents when two different teams belonging to two different nationalities are involved.

### **Getting questions answered**

The final issue that troubles the offshore teams is the response time required on posing questions to people who operate in radically different time zones.

---

## 3. The standards, models and techniques used

This section explains the various standards, models and techniques considered during the research for assessment of high level and detailed level design artifacts.

### 3.1 LaQuSo Software Certification Product Model (LSPCM)

The Laboratory for Quality Software (LaQuSo) has defined a Software Certification Model that is LaQuSo Software Product Certification Model[1] for verification and validation of software systems and their intermediate artifacts.

LSPCM is rule-based software product certification model for major class of deliverables of software development. Major class of deliverables are called Product Areas. In LSPCM, product areas are Context description, User Requirements, High Level Design, Detailed Design, Implementation and Tests[1](pp.9). There are three Certification Criteria, which hold for all Product areas : areas must have completely (and formally) specified, uniform and conformant.

LSPCM Certification Criteria [1] (pp. 11-14):

- Completeness – All required elements in the product area should be present and as much as possible formalized.
- Uniformity – The style of the elements in the product area should be standardized.
- Conformance-All elements should conform to the property that is subject of the certification.

Input for the software certification model are one or more software artifacts and one or more properties of these artifacts that are to be certified.

The properties can be one of the following categories[1] (pp.6):

- Consistency : do the different (parts of) software artifacts conform each other?
- Functionality : does input to the system produce the expected output?
- Behavioral : does the system meet general safety and progress properties like absence of deadlocks or are constraints on the specific states of the system met?
- Quality : do the artifacts fulfill non-functional requirements in the area of e.g. performance, security, and usability?
- Compliance: do the artifacts conform to standards, guidelines and legislation?

The Model has five certification levels for each product area, but only four are relevant for awarding certification for product area[1] (pp.43) . The first level is the entry level for the certification process.

- Initial : This certification level is the entry level for certification process. This level is given when all the required elements of the product area are present and uniform.
- Manually Verified : This level is achieved, when all elements, relationships and properties have been verified manually.
- Automated Verified: This level is awarded when all the elements, relationships and properties have been verified with tool-support.
- Model Verified : This level is achieved when a model representing elements, relationship and properties has been constructed by an assessor and verified with mathematical methods.

- Formally Verified: This level is achieved when elements, relationship and properties have been verified with mathematical methods without explicit model construction step.

In our thesis project suggestion for High Level Design and Detail Level Design criteria for off shoring has been provided.

### 3.2 Capgemini Accelerator for Software Architecture (CASA)

Capgemini Accelerator for Software Architecture (CASA) is an initiative of mixed team of architects (cross-technology, cross-country and cross-culture) from Capgemini.

CASA focuses on setting standards for and further detailing the Analysis and Design and Software Architecture discipline. These standards can be reused as packages of artifacts within projects.

The CASA accelerator consists of guidelines, templates and checklists that focuses on the Analysis and Design discipline. On top of the standard documentation used within project , CASA guidelines, template and check lists will help to create systematic, detailed documents in the Analysis and Design discipline. CASA is used mainly by Designers and Software Architects of Capgemini on IT-Projects .

The CASA initiative is the logical successor to Capgemini practices that is SEMBA and IRMA.

SEMBA stands for Structured Expert Method for Business Analysis is a Capgemini method for business analysis.

IRMA stands for Integrated Requirement Management Approach. IRMA supports system analyst in gathering and elicitation of system requirement from the business requirements.

CASA supports Rational Unified Process (RUP) process as well as DELIVER SAP, a method for implementation of SAP packages. In our research, focus has given on CASA for RUP.

CASA provides template, guidelines and checklist for analysis and design artifacts, e.g.:

- Software Architecture Document (SAD)
- Use case Realization (UCR)
- Models – Analysis, Design, Implementation and Deployment Model.

Here, templates provide outline structure for the information that has to be filled in and maintained in an artifact. Guidelines explain required information that has to be completed and maintained in artifacts. Checklist are maintained to verify an artifact for followed CASA template.

In our research CASA template and checklist have been used as assessment tool for quality analysis of Software Architecture.

### 3.3 Rational Unified Process (RUP)

Rational Unified Process(RUP)[10] is an iterative software engineering process. It is developed and maintained by IBM Rational Software. In our project, the focus has been given for CASA for RUP and one of the case studies considered has followed RUP, considering these facts RUP study has been considered.

Iterative process has been organized into four phases[10] (pp.62-75) and each phase is concluded by a major milestone.

- Inception Phase: Inception phase involves the formulation of scope of the system, that is, capture the context and the most important requirements and constraints for the

system, plan and prepare a business case which includes the business context, project plan, staffing and trade-offs among cost, schedule, and portability has been defined. This phase is concluded by the lifecycle objective milestone.

- **Elaboration Phase :** Elaboration phase is to analyze the problem domain, establish a sound architectural foundation, develop the project plan, and eliminate highest-risk elements. This phase is concluded by the lifecycle architecture milestone.
- **Construction Phase:** Construction phase involves building the product and evolving the vision, the architecture, and the plans until the product is ready for delivery to its end user community. Construction phase is concluded by the initial operational capability milestone.
- **Transition Phase:** In this phase involves delivery of product to end users, which includes manufacturing, training, supporting, and maintaining the product until user satisfied. It is concluded by the product release milestone.

### 3.4 Analysis and Design Artifacts

In the software development, next step to the requirement discipline is analysis and design. The purpose of analysis and design discipline is to translate the requirements into a specification that describes how to implement the system [11].

Analysis and Design discipline involves defining the software architecture and design of the system.

Software architecture can be defined as, "Software architecture is the set of significant decision about the organization of a software system, the selection of the structural elements and their interfaces by which the system is composed, together with their

interfaces by which the system is composed, together with their behavior as specified in the collaborations among those elements, the composition of these structural and behavioral elements into progressively large sub-systems, and the architectural styles that guides this organization, these elements and their interfaces, their collaborations, and their composition." [Booch, Rumbaugh and Jaccobson 1999], (p.31)

Architecture focuses on how the major elements and components within an application are used by, or interact with, other major elements and components within the application.

The selection of data structures and algorithms or the implementation details of individual components are design concerns. Architecture and design concerns are very often overlap [18].

LSPCM has High Level Design and Detailed Level Design as product areas. High Level Design consists of software architecture of the system and Detailed Level Design covers design part of analysis and design discipline of software development. In case of CASA, we do not have clear distinction between High Level Design and Detailed Level Design. Based on the CASA artifacts and discussion with software architects in Capgemini, for analyzing the artifacts we categorized CASA artifacts into High Level Design artifacts and Detailed Level Design artifacts. Thus, we have **Software Architecture Document(SAD), Analysis model** categorized as under High Level Design artifacts for assessment. The **Use Case Realization (UCR), Design, Implementation and Deployment models** are considered as Detailed Design artifacts for assessment based on its individual functionality. The following paragraph briefly explains these artifacts.

**High Level Design(HLD)** : High Level Design, also called software architecture is the first step to analyze and consider all requirements for software and attempt to define a structure

which is able to fulfill them. It captures and conveys architectural decisions made to support functional and non-functional requirements of the system [19].

**Detailed Design (DD):** The process of refining and expanding software architectural designs to more detailed descriptions of the processing logic, data structures and data definitions. This continues until the design is sufficiently complete to be implemented. Detailed Level Design also called low-level design [ANSI/IEEE Std. 610.12-1990].

In Capgemini, analysis and design documents consists of Software Architecture Document (SAD), Use Case Realization (UCR) and Models.

**Software Architecture Document(SAD):** Software Architecture Document provides a comprehensive architecture overview of the system, using different architectural views to depict different aspects of the systems. It is intended to capture and convey the significant architectural decisions which have been made on the system for supporting both functional and non-functional requirements[15](pp. 7). SAD helps in shaping the design and construction of the system. A well written SAD forms the foundation for more detailed design. This artifact falls under High Level Design part of assessment.

Use Case Realization (UCR): Use case realization is defined as the set of model elements that show the internal behavior of the software that corresponds to the use case.

Models : CASA provides modeling guidelines for RUP models. Modeling guidelines describes the standard folder structure for RUP models , which will be supported in the standard tools(RSM/RSA/Enterprise Architect).This modeling guidelines addresses the following RUP models:

**Analysis Model:**

Analysis model is the first step towards design. This model describes the structure of the system or application. It consists of class diagram and sequence diagrams that describes the logical implementation of the functional requirements. In this model, functionalities are modeled in terms of boundary, control and entity classes. Sequence diagrams realize use cases by describing the flow of events in the use cases when they are executed. These artifacts are categorized as High Level Design artifacts for assessment.

**Design Model:**

Design model builds on analysis model by describing, in greater detail, the structure of the system and how the system will be implemented. A Design model consists of design classes structured into packages and subsystems with well-defined interfaces. This model describes the clear relationship between the design classes, and implementation elements. Implementation elements are directories and files including source code, data and executable files. These artifacts are analyzed for to the Detailed Design.

**Implementation Model:**

The Implementation Model represents the physical composition of the implementation in terms of Implementation Subsystems, and Implementation Elements (directories and files, including source code, data, and executable files). This artifact is considered in the assessment of the Detailed Design.

### **Deployment Model:**

Deployment model describes the deployment units of a system and one or more physical network (hardware configurations on which these units are deployed). This artifact belongs to the Detailed Design part of an assignment.

### **3.5 Related Work**

The offshore software development approach induces several changes as well as challenges to the development process. One of the changes are development process is distributed over country borders to low wages countries. One of the main challenges faced in this approach are quality of artifacts developed by different teams (Onshore and offshore teams). This division of responsibilities requires a stricter quality assurance. Software Product Quality can be assured and improved by imposing certification. There are, however, many approaches to software certification, that mostly rely on formal verification, expert reviews or software metrics to determine product quality.

In the industry, there are few software product quality verification and validation approaches like LSPCM[1], SCfm\_Prod model[12], software product maturity model by Nastro(1997)[13] etc. In our thesis project, LSPCM certification model has been used as for quality analysis and improvement for software architecture and design.

The following sections describe the various software product quality models available. We have used these to compare the various features provided by these which helped us choose the most feasible certification model for our research.

**SCfm\_Prod Model** : SCfm\_Prod model[12] is one of the software product certification models. This model is a result of empirical study conducted in Malaysia for demand of

software product quality certification model. Model has been developed based on end-product quality approach for certification. This model consists of four main entities that is Pragmatic Quality Factor (PQF), assessment team, certification representation method and product certification repository.

In the first step of building this model main characteristics consideration for assessing software products has been listed. Based on the survey and software quality characteristics according to ISO 9126 model main characteristics were functionality, efficiency, integrity , maintainability and reliability .

To construct model, they defined the certification representation method. This method involves weighted scoring method, certification level and decision process for certification level[12](pp. 4-5). This model contains four distinct certification levels : excellent, good, basic, acceptable and poor.

#### **Observation in SCfm\_Prod model :**

LSPCM also focuses on certification for Software Product Quality. SCfm\_prod model differs from LSPCM model because it only awards certification for end product and not for the intermediate software artifacts like requirements, high-level design and detailed design etc.

LSPCM has specific checks for quality in terms of completeness, uniformity and conformance which were selected as a result of continuous research in this field from theory and practice.

In case of SCfm\_prod model the quality attributes defined in ISO 9126 model form the baseline of the assessment matrix.

SCfm\_prod model defines two sets of quality attributes, which are by means of the behavioral and the impact attributes. The behavioral attributes consists of high level software quality characteristics which include usability, efficiency, functionality,

maintainability, portability, integrity and reliability. Integrity is not included in ISO 9126 model but is included in this model because of the requirement from the literature and empirical studies they have performed. The impact attributes indicates the conformance in user requirements, expectations and perception.

**Software product maturity model by Nastro (1997)** : In Software Product Maturity model [13], a core set of elements that constitute the basis of a product maturity calculation. Core elements of Product maturity model are product capability, product stability and product maintainability. Two sub elements are product repeatability and product maintainability.

Product capability refers to the performance of the product against the specifications. It can be measured by examining the success or failure of the product against the applicable qualification tests. These tests represent the validation of the derived requirements specification or equivalent.

Product stability refers to the impact of modifications to a baseline software product over a specified period. These modifications are usually indicated as software problem or change reports. Product stability can be measured by interpreting the software changes per period, then weighting the changes by impact or severity. The results can then be interpreted through evaluation against a previously established benchmark that may be the result of an industry standard, a project specification or historical data of a similar project.

Product maintainability refers to the ability of a software organization to understand, maintain, use and upgrade software.

Product repeatability refers to the ability of a software product to provide consistent results from execution to execution.

Product compatibility refers to the performance of the software product against the interface specification.

#### **Observation in Software Product Maturity Model:**

Unlike LSPCM model, Software Product maturity model measures the properties of the end product.

In this model more focus is on tracking the development progress (i.e., comparison of builds within one project) than for the objective measurement of the product quality.

#### **Architecture Tradeoff Analysis Method (ATAM)**

In the area of software architecture, Architecture Tradeoff Analysis Method (ATAM)[14] is one of the analysis method for software architecture. It evaluates how well an architecture satisfies particular quality goals (such as performance or modifiability). Goal of ATAM is to understand the consequences of architectural decisions with respect to the quality attribute requirements of the system.

Input for the evaluation process are statement of quality requirements and specification of the architecture with a clear articulation of the architectural decisions.

#### **ATAM analysis process:**

Identify architectural approaches : Architectural approaches are identified by the architect.

Architecture approaches define the important structures of the system and describe the ways in which the system can grow, respond to changes, integrate with the other systems.

Generate quality attribute utility tree: The quality factors that comprise system "utility" (performance, availability, security, modifiability, etc.) are elicited, specified down to the level of scenarios, annotated with stimuli and responses, and prioritized.

Analyze architectural approaches : Based on the high priority factors identified in step 2, the architectural approaches that address those factors are elicited and analyzed (for example, an architectural approach aimed at meeting performance goals will be subjected to a performance analysis). During this step architectural risks are identified.

Brainstorm and prioritize scenarios: Based upon the exemplar scenarios generated in the utility tree step, a larger set of scenarios is elicited from the entire group of stakeholders. This set of scenarios is prioritized via a voting process involving the entire stakeholder group.

Analyze architectural approaches: This step reiterates step3, but here the highly ranked scenarios from Step 4 are considered to be test cases for the analysis of the architectural approaches determined thus far. These test case scenarios may uncover additional architectural approaches, risks which are then documented.

Present results: Based upon the information collected in the ATAM (styles, scenarios, attribute-specific questions, the utility tree, risks) the ATAM team presents the findings to the assembled stakeholders and potentially writes a report detailing this information along with any proposed mitigation strategies.

#### **Observations in ATAM :**

ATAM is subjective analysis method that is more related to Software architecture process followed in the software development.

ATAM analysis method involves brainstorm and prioritize scenarios which involve greater number of stake holders i.e. users, developers, project manager, ATAM assessment team etc.

In case of LSPCM – High Level Design quality analysis process is not a scenario based check. It, not only concentrates on evaluation of quality requirements of the system but also checks

for completeness, uniformity and conformance of the design for considered properties of the system[1](pp.6).

In ATAM, focus is only on software architecture discipline of software development. But, LSPCM divides the software life cycle phases into its product areas and can provide certification for individual product areas. i.e. user requirements, high level design, detailed design etc.

In our thesis project for analysis and improving quality of Software Architecture and Design, we have considered the LSPCM : LaQuSo Software Product Certification Model considering following points with comparison of other methods:

LSPCM model is comprehensive and flexible enough to allow certification of software products in all life-cycle stages, with the applicable rigor for the criticality of the software, up to formal verification for the most critical products.

LSPCM model will help both software developer and auditor. Since, it is not only used to certify products after their creation, but it also indicates which elements and relations should be present in the product when it is being created.

Certification process in LSPCM is systematic. Since, LSPCM has three certification criteria, each criteria has sub criteria in it and based on the achievement levels of each criteria final certification of the product has been decided.

LSPCM is rule-based certification method, i.e. it has the sub-criteria in the form of checks. This approach is systematic approach for verification of the quality, since process first starts with more simple check. For example, the checks which ensures that all the elements are present, whether the relations mentioned are consistent and to verify that the company or industry standards are followed etc.

LSPCM certification covers only product quality, not the management and development process. This model is independent of software process and can be applied for verification and validation of quality of product. Complete documentation containing various checks for all the software artifacts in all life-cycle stages is available.

---

## 4. Case studies

### 4.1 Introduction

To validate our research question, we apply LSPCM and CASA to the real industrial projects undertaken by Capgemini, the development of which follows the onsite/offshore model.

The case studies chosen are the offshore and RUP based projects carried out by Capgemini. During this project, the inception phase of RUP was carried out completely by the team at an onsite location. The elaboration phase was done between the collaboration with onsite and offshore teams. This is either handled at onsite or offshore locations depending on the team preferences. The more of detailed design stages was conducted completely at the offshore end, followed by the construction phase.

### 4.2 Approach

In order to accomplish the defined objectives, the approach followed has been described:

Initially, we analyzed the design documents for the projects with emphasis on handover moments.

Secondly, application of LSPCM for the High Level and Detailed Design certification criteria on these two offshore projects undertaken by Capgemini was done. For this analysis we have chosen JAVA projects : one of which was designed using CASA and the other one using the standard RUP templates in order to analyze the ease and efficiency with CASA usage.

Next, the application of CASA templates, guidelines and checklists were analyzed with the design documents. The case study designed without using CASA was analyzed to ensure the presence of all the details which CASA captures. Whereas the case study designed using

CASA was verified with CASA guidelines to make sure that the details were captured in a correct manner in the templates and models.

To validate our research question, we apply LSPCM and CASA to the industrial cases of offshore development. Based on the assessment results, we propose the suggestions on how these quality assessment approaches can be improved. We also provide suggestions for the design documentation of these case studies based on the assessment.

The analysis was carried out in the following manner:

Using LSPCM : LSPCM is designed as a set of rules which is in the form of specific checks. LSPCM can directly be applied on the design documents. This is done by checking whether the specific checks specified by LSPCM are satisfied by the documents under assessment. Therefore the result of this assessment has been provided as a defect report depicting the checks violated and the reason for the violation. For example: "SC3.1(b) All components have unique names. This was applicable not but not satisfied as one use case was referred with the same id but with different names".

Using CASA : CASA is the Capgemini technique for analysis & design. This contains the templates for the documents, guidelines to use these templates and checklists to assist the detection of omissions from the documents under assessment. The result of this assessment is a document indicating the omissions relevant to the particular project.

Analysis : Results of the application of LSPCM on one hand and CASA on the other, will act as an input for analyzing the findings. Here, we compare the results in the following manner:

Check if some defects would have been removed if some missing details were present.

Identify the artifacts which should have captured the missing details.

The knowledge gained from the above analysis will serve as feedback to LSPCM and CASA artifacts.

### 4.3 Case study 1 : An Electronic ticket printing application

The first case study we have considered is an offshore RUP based project undertaken by Capgemini. The first case study considered for analysis is an offshore project handled by Capgemini. For this project RUP process has been followed and the following subsections contain a brief description of the case study, the assessment results using LSPCM and CASA and suggestions based on the assessment.

#### 4.3.1 Case description

This project offers its major customers as well as agencies, the possibility to order and print tickets for national transport in-house. The customer can specify the wished for product using the application (Electronic Ticket Printer System). Through an interface with the pricing server, the corresponding price can be generated. The products ordered (train tickets), can be printed on a special printer using special ticketing paper. Periodically, accumulated invoices are generated and sent to the customer. This occurs through another interface with the financial system. The target of this project is to rebuild the application, with a simplified and more robust infrastructure.

#### 4.3.2 Assessment Results

This section describes the assessment of first case study : an application for electronic printing application. In order to evaluate the quality of the design, the LSPCM criteria have been checked for high level design as well as detailed design of the case study. Based on the

assessment of the case study, inconsistencies found in the high level design and the detailed design are documented and possible solution for rectification of these inconsistencies are explained.

#### 4.3.2.1 LSPCM

This section describes the assessment results followed by suggestions for case studies using LSPCM. Disparate sections for High level and Detailed design focus on the respective major defects found during the assessment.

##### **4.3.2.1.1 High Level Design**

This subsection gives the evaluation of High Level Design of case study : an electronic ticket printing application, using LSPCM - High Level Design(HD) criteria[1](pp25-30). In the evaluation process specific checks of LSPCM – HD has been checked in the High Level Design documents. Based on the evaluation using the LSPCM criteria, inconsistencies were found and listed as follows: By resolving these inconsistencies helps to improve quality of high level design like readability and understandability. Since a well defined high level design document is the base for the detailed design and the development of a software product.

Inconsistencies found are:

The translation document from Dutch to English has not been maintained: Translation document for containing the details of class model mentioned in the Software Architecture Document has not been maintained.

The traceability matrix between requirements to the design elements (classes , components, process and packages ) has not been maintained in the available set of documents.

Reference information for certain requirements mentioned in the Software Architecture Document are not present. e.g.: System should support rotating log files based on the administrator requirements. In this requirement, reference for administrator requirements has not been mentioned in the Software Architecture Document. Since, a reference to the requirements document is mandatory in the SAD for the specific requirements.

Software Architecture has been represented using 4+1 view model[11]. In the document description of views, the description of each view elements should be described. i.e. Process view should provide name of the process, brief description of functionality of process, communication characteristic between processes. But this information for process view is not present in Software Architecture Document.

Rationale behind architectural decisions which supports the quality requirements (usability, reliability, security, efficiency, maintenance )of the system has not been mentioned in Software Architecture Document.

Description of methods and attributes mentioned in the class model is not present. In the class model mentioned method and attribute names are in Dutch. Translation document for these and brief description telling what is the functionality of method and attribute information has not been maintained.

Suggestions for Case Study - Software Architecture Document:

This subsection contains suggestions for Software Architecture Document (SAD) of Case study-1.

Translation document from Dutch to English for class model should be maintained. Since, detailed level design has been carried out in offshore site, this translated document will help the offshore team for further detailing the design and development. By this suggestion understandability and maintainability of the document can be improved.

Traceability matrix between requirements and design elements has to be maintained. During high level design phase, this information helps in checking whether all the requirements of the system have been considered. This suggestion will help for verification and maintenance of the SAD.

Reference information in the software architecture document has to be maintained. This is useful for faster understanding of software architecture of the system in cases where new members join the team when project is already in the midst of development phases.

In this case study, software architecture has been represented in the form of 4+1 view model[11]. Each view of the architecture representation has to be explained in detail as mentioned in the RUP view model. CASA - SAD guidelines explain information needed in each view. e.g.: In case of logical view of the system following information has to be maintained according to RUP process and CASA : SAD guidelines.

Overview: This subsection contains overview of important business components(or subsystems). This will give reader insight in the main functional/ business parts of the system.

Workflows: This subsection includes the most important business flows for the system, depicted in the form of flow diagram.

Layers<sup>6</sup> and packages: This section contains diagram which shows the layering and explains their responsibilities.

Rationale behind the architectural decisions has to be mentioned in the software architecture document. Software architecture is the bridge between requirements and design and development. It explains the design decisions, rationale behind the decisions before beginning the development process. This information will help the reader to understand the purpose of the design decisions. By providing this information discussion time on rationale behind the architectural decisions can be saved. This suggestion will help to improve understandability of the document.

Description of methods and attributes : Class model mentioned in the Software Architecture Document contains the overview of the system functionality in terms of classes and their main methods. This document should contain a brief description of classes and their main methods. This information will be an advantage in detailed design for further detailing each classes in terms of its method return type, attribute data type etc. In development discipline, this information will help in verification of development of all main classes. This suggestion will help in better understanding of the class diagram and system functionality.

<sup>6</sup>A layer is a set of common functions which are grouped according to the functionality it provides. E.g.: CASA explains presentation layer, service layer, process layer, business layer, resource layer.

#### **4.3.2.1.2 Detailed Design**

LSPCM is applied on the design documents of the case study and a defect list is obtained from this assessment. The defects found are categorized into three categories: minor inconsistencies, major inconsistencies and incompleteness.

##### Minor Inconsistencies

The inconsistencies which do not affect the system are referred to as minor inconsistencies. These inconsistencies do not alter the functionality of the system. The documents or artifacts could be made readable if these are done away with as removal of these inconsistencies would assure quality in terms of understandability of the intended users of this document.

The minor inconsistencies are categorized into the following 3 types:

Spelling mistakes, Grammatical issues, Inconsistencies with notation and terms.

The following minor inconsistencies have been observed with respect to the project artifacts:

- In the sample of 10 pages from a total of 120 pages in the given set of use case realization documents, on average one spelling error per page is present.
- The name of the use case in the prepared document and that used in the design models are not the same. For example : The name of the use case OR01 is Create and Print Travel Warrant in the document whereas it is mentioned as **ComposeTravelWarrent** in the design models. The notation for a single term has been specified in different forms.

## Major inconsistencies

This category summarizes the inconsistencies related to the functionalities of the system to be developed. Failing to correct them could lead to wrong interpretation of the document.

This category involves the following kinds of issues.

- Errors due to translation
- Ambiguous class attributes, function names or object instance names.
- No references mentioned

The kinds of errors involved here are as follows:

- The getter/setter method name for the attributes have been specified in Dutch, whereas the attribute name is in English.
- There is presence of anonymous objects<sup>†</sup> in the class models. As far as possible, the presence of anonymous objects should be limited in the design models. This creates an ambiguity among the developers about the data types of the objects under implementation.
- The Exception handling scenarios have not been included for each of the use case realizations. The use case realization document speaks about an “Exception details” document in each of the use case realization document, but fails to mention the reference to the same document. In such cases the developer is free to refer any similar document and this might lead to errors or incorrect functionalities being implemented.

*†anonymous object : An object of unknown class. The interface to an anonymous object is published through a protocol declaration.*

## Incompleteness

In this category, we summarize the omissions from the design related documents and glossary.

- The deployment diagram is not present in the given document depicting deployment plan.
- The use case realization documents denote the information based on the pricing server and the interfacing with it, but this system interfacing information has not been provided in the design documentation.
- In the Exception Handling section in use case realization documents, the details say that all the information is present in exception handling document. The reference to this document is missing in the use case realization document.
- The names and responsibilities of the attributes and methods is not documented
- The class details have not been maintained in the documents.
- Similarly, many project terms are not described in the glossary.

### 4.3.2.2 CASA

During assessment using CASA, the casa templates and checklists were used as the assessment tool to verify the presence of all details which CASA claims to capture in the project artifacts. The artifacts of CASA are not categorized as High Level and Detailed Design artifacts. However, based on the discussions held with the software architects and the CASA team members we came to the conclusion that for the analysis based on High level and the detailed design the CASA artifacts could be categorized as follows:

#### 4.3.2.2.1 High Level Design

##### Software Architecture Document (SAD)

This subsection describes the assessment process of Software Architecture Document of case study using CASA . CASA provides templates and guidelines for SAD. Templates provides outline information details for SAD. In the assessment process CASA - SAD template has been used as checklist to evaluate SAD of case study-1. The assessment result has been shown in the following table and details of this assessment can be found in assessment report[2]. In the following table, the reference column depicts the detailed evaluation for each section in assessment report of case study.

Result of evaluation is termed as “found”, if all the information in the section is present. If the needed information is not fully maintained in the section then the result is marked as “partially found”. The result is said to be “not found”, if the needed information has not been maintained in the section of the template.

CASA sections	Status in Casestudy -1	Reference in assessment report of case study -1.
1. Introduction	Partially found	2 (pp. 44-45)
2. Architectural goals and constraints	Partially found	2 (pp. 46-47)
3. Quality considerations and decisions	Partially found	2 (pp. 47-49)

4. Use-case view	Partially found	2(pp. 49)
5. Logical view	Found	2(pp. 50-51)
6. Process view	Not found	2(pp. 52)
7. Implementation view	Found	2(pp. 53-54)
8. Deployment view	Partially found	2(pp. 54)
9. Data view	Partially Found.	2(pp. 54-55)
10. Design constraints	Found	2(pp. 55-56)

Table 1 : Assessment using CASA for Software Architecture Document(SAD) for case study 1

Observation in Case Study : Software Architecture Document :

In case of case study -1 for designing software architecture document CASA has not been followed. From the assessment of Software Architecture Document using CASA- SAD explains the fact that the CASA-Software Architecture Document template helps for designing a more detailed Software Architecture Document. Software Architecture Document of case study-1 did not contain most of the sections as mentioned in CASA-SAD template (The defects have been mentioned in the above table). The following paragraph describes the missing information in the case study-1 : SAD, which often has impact on further stages of development (detailed design, development and maintenance) of software development.

Case study-1 SAD was more abstract. Sections mentioned in the SAD were not explained in detail.

In architecture representation views (logical view, process view, implementation view and deployment view) were not described as per the CASA-SAD guidelines. e.g.: In the SAD of case study-1, process view section has been mentioned, but details explaining the system decomposition into threads and processes\* have not been described. The details [CASA-SAD guidelines] are *name of the process, priority, mode of communication between processes, corresponding deployment units, process description : memory usage, response time, scheduling of process, availability of process*. These details will help in detailed level design for further elaboration of the design.

According to the CASA - SAD template, the Software Architecture Document should contain architectural decisions to support quality requirements of the system(usability, reliability, security, efficiency, maintenance). In case of case study-1 : SAD does not contain description which provides the information of architectural decisions to support quality requirements of the system. Architectural decisions to support quality requirements are key decisions made, that shape the architecture in order to meet the quality dimensions of the system. The description of this section includes architectural decisions, rationale behind the decision, the alternative choices of decisions and why the alternative choices are rejected. If these details are not mentioned in the SAD, issues like architectural decisions made on the system, rationale behind the decisions and impact of these decisions on overall system has to be separately discussed with stake holders like project manager, development team, end user and clients.

\* *Process: A process is an instance of a computer program that is being sequentially executed by a computer system that has the ability to run several computer programs concurrently. A process may split itself into multiple 'daughter' sub-processes, so called threads that execute in parallel, running different instructions on much of the same resources and data (or, as noted, the same instructions on logically different resources and data)(Source:Wikipedia)].*

In case of offshore development method, the issues like geographical distance between onshore and offshore teams have impact on time and budget of project. e.g.: If security is one of the quality requirements of the system, SAD should contain details of how architecture will support security requirements such as authentication, authorization, data confidentiality etc. e.g.: For authentication: Authentication involves verification of identity of the user. SAD should include for providing authentication which mechanism is considered, technology needed, related components, processes and deployment units. Some of the authentication mechanisms are role based authentication, login/log out (user name and password), smart card mechanism, retina scan, voice recognition and finger print etc. This information will help in the detailed design for further elaboration of measures to be taken for supporting quality needs of the system. E.g.: In case of security if role based authentication is considered, detailed design explains corresponding classes, methods and attributes in detail.

Suggestion for Case Study -1 Software Architecture Document:

This subsection describes the suggestions for case study-1 Software Architecture Document(SAD). Based on the observation found in the case study -1 SAD suggestions are:

Details of each architecture views has to be maintained as mentioned in the CASA-SAD guidelines. Each architecture view includes required elements and details of the elements. This will be added help in the detailed level design for elaboration of implementation of these elements. e.g.: In case of process view, processes, thread of the system, memory requirement of the process, availability, scheduling algorithm for handling threads, packages and implementation elements for processes etc. these details have to be mentioned.

Rationale behind the architecture decisions has to be maintained. As mentioned in the above paragraph, this information will help to save discussion time on architectural decisions made on the system.

*Intended users* section mentioned in CASA-SAD. This section covers list of the audiences (users, analyst, software architect, project manager, designer and development team) and relevant sections for each type of reader. This information will help the reader by serving as a reference to the related sections. This suggestion will improve readability of the document.

Definitions, acronyms and abbreviations have to be maintained. This will help the new member of the team to understand the system.

#### **Analysis Model:**

Observation in Case study-1 :

This subsection describes the assessment for analysis model in case study-1. In this case study, since CASA-design model has not been followed for creation of models of the system, there is no clear distinction of models described in CASA design model guidelines. In evaluation process, the elements in the artifacts are verified for correctness as mentioned in the format structure of the CASA- Analysis model guidelines. The result of this evaluation is that, in the model provided for case study-1, we could find class diagram. Class diagram depicts the overview of all classes in the system. As mentioned in the CASA modeling guidelines for analysis model, layer structure of the system, corresponding classes in each layer, control, boundary and entity classes of the system were not present.

Suggestion for Analysis model :

Analysis model has to be maintained in the project. Since, in analysis, requirements are structured in terms of control, boundary and entity classes. According to CASA analysis

guidelines, classes are assigned to various layers of the system, analysis package of the system are maintained. This information is input for design and implementation model.

Analysis model provides overview of the system that may be harder to get by studying the results of design and implementation model. This overview can be very valuable to new member of the team or to developers who maintain the system in general [23](pp 178).

#### **4.3.2.2.2 Detailed Design**

The CASA artifacts identified for analysis of detailed design consists of the following :

- Use case realization
- Design Model
- Implementation Model
- Deployment Model

The table below indicates to what extent the project artifacts were able to satisfy the details captured by CASA. These were analyzed against each of the templates and the checklists of the particular artifact. There can only be three possible values for status information in the case study document. Either the content of the deliverable is completely found(the section is present) or it is partially found(few details are present) or it is not found at all.

##### 1. Use case realization

The details of this assessment can be found in the assessment report [2]

CASA sections	For case study 1
Introduction	

Purpose	Found
Scope	Found
Intended Audience	Not Found
Definitions, Acronyms, and Abbreviations	Found
Issues	Not Found
References	Found
Overview	Found
View of Participating classes	
Participating Classes	Not Found
Structural diagram	Found
Flow of Events	
Flow<Use-case scenario>	Found
Brief Description	Found
Interaction Diagram	Found
Logical Interfaces	
Non-Functional Requirements	Not Found

Transaction Context	Found
Exception Handling	Found
Other non-functional	Not Found
Additional Information	
Any other details	Found

Table 2: Assessment of the Use Case Realization template for case study 1

## 2. Design Model

CASA models	In case study 1
Use Case Realization	
Actors	Found
Use cases	Found
Service Realizations	Not Found
Overview Layering	Not Found
Layers & Frames	
Presentation Layer	Not Found
Process Layer	Not Found

Business Layer	Not Found
Resource Layer	Not Found
Domain Entities	Not Found

Table 3 : Assessment of the Design Model for case study 1

### 3. Implementation Model

CASA Models	In case study 1
Implementation Elements	
Components	Not Found
Deployment Unit elements	Not Found
Deployment Units	Not Found
Implementation View	
Component Mapping Model	Not Found
Component Model	Not Found
Deployment unit Structure	Not Found

Table 4 : Assessment of the Implementation Model for case study 1

#### 4. Deployment Model

CASA Models	In case study 1
Nodes & Devices	
Nodes	Not Found
Devices	Not Found
Deployment View	Not Found
Model Building Blocks	Not Found

Table 5 : Assessment of the Deployment Model for case study 1

From the assessment results, we obtained the contents of CASA which are not present in the project documents. Hence we identified the concerns related to the design which are critical considering successful execution of the project.

Concerns	Relevance	The document which should have addressed the concern
<b>Concern1</b> : The use case realization does not depict the participating classes	The details about the classes is important for implementation of the particular use case.	Use case realization
<b>Concern 2:</b> The Non functional requirements are	The non functional requirements associated	Use case realization

<p>not documented for the use cases</p>	<p>with a use case are required to be known before implementation for a particular flow of the use case.</p>	
<p><b>Concern 3:</b> The Service realizations have not been covered in the models.</p>	<p>Service realization helps with deciding the services provided by different components. The decision has important implications on service availability, distribution, security and transaction scopes.</p>	<p>Design Model</p>
<p><b>Concern 4 :</b> The details about Overview layering is not present.</p>	<p>This describes how to separate responsibilities of components by depicting the layers and common design patterns used. This may also include the usage of presentation layer design patterns to solve common problems in the user interface.</p>	<p>Design model</p>

<p><b>Concern 5:</b> The Component model has not been designed for this project</p>		<p>Implementation model</p>
<p><b>Concern 6 :</b> The details about Deployment unit structure i.e. the deployment diagram are not designed</p>	<p>A deployment diagram models the physical deployment of artifacts on nodes. The deployment structure shows what hardware components exist and what software components run on each node. This helps the developer to understand the structure and know what files exist on which servers.</p>	<p>Deployment model</p>

Table 6: Concerns for detailed design documentation for case study 1

### 4.3.3 Remarks on LSPCM

This section summarizes the suggestions for the LSPCM model from analysis & design perspective.

#### 4.3.3.1 High Level Design

This sub section details the suggestions for LSPCM based on the assessment of case-study-1 Software Architecture Document.

**Suggestion 1:** If the software architecture has been represented in the form 4+1 view model (logical view, process view, implementation view , deployment view and use case view) following set of checks can be suggested for LSPCM –High Level Design Criteria.

**LSPCM checks for logical view :**

Logical view describes conceptual organization of the system in terms sub-systems, interfaces<sup>~</sup>, design packages and classes. This section includes significant parts of design model such as decomposition of system into sub systems and design packages. Sub systems of the system are represented in the form of component diagram, which depicts the important business component of the system, external components and interface between these components. For each of the design package describes its decomposition into classes and class utilities with class and object diagram<sup>°</sup>.

This view describes layers of the system, logical components/packages associated with each layers.

*<sup>~</sup>interfaces: a boundary across which two independent entities meet and interact or communicate with each other.*

*<sup>†</sup>Packages : Group of related components and classes.*

*<sup>°</sup>Class diagram: Class diagram contains classes, relation between classes such as association, dependencies and generalization.*

*Object diagram: Object diagram is an instance of a class diagram , showing snapshot of the detailed state of a system at a point in time*

In LSPCM we can add following checks for logical view:

- Check for component diagram. Since, sub-systems of the system depicted in the form of component diagram.
- Check for package diagram.
- Check for class and object diagram.
- Check for Interfaces. Interfaces of the system are responsible for connecting the system components to the external system components.
- Check for layers of the system, mapping of layers and corresponding logical components/ packages.

#### **LSPCM Checks for Process View:**

Process view provides basis for understanding the process organization of a system, illustrating decomposition of a system into processes and threads, and interaction between processes. The process view<sup>5</sup> takes into account some non-functional requirements, such as performance and availability.

In LSPCM we can include following checks:

- Check for name of the processes, main modes of communication between processes.
- Check for mapping of classes and subsystems onto processes and threads.
- Check for Process description. Process description contain following information of processes : CPU utilization for processes, Concurrency, priority of processes, memory usage, Scheduling, availability, scalability.

### **LSPCM Checks for implementation view:**

Implementation view represents implementation elements, such as implementation subsystems and components. This section explains:

components\* of the system, dependencies between components, dependencies on third party components/frame works, technology used to implement the components.

Component mapping model that is mapping between design packages and components. e.g.: how design packages are mapped onto projects and folders and namespaces in the development environment.

Mapping between component and deployment unit.

In LSPCM for this view following checks can be included:

1. Check for components of the system.
2. Check for component mapping model that is mapping between design packages and components.
3. Check for mapping between component and deployment unit.

\* *components : the principal computational elements and data stores that execute in a system*

### **LSPCM Checks for Deployment View :**

Deployment view describes one or more physical network-hardware configurations on which the software is deployed and runs. It explains assignment of deployment units to nodes. A deployment unit consists of a build- an executable – documents, and installation artifacts. Infrastructure of the system which consists of physical nodes like servers, storage devices, network connections, ports, firewall restrictions etc are depicted by deployment diagram. Physical nodes specifications like servers, storage device requirements, details of memory, CPU's, performance of nodes should be explained in this section.

e.g.: Application server : 4 core, 3.60 GHz per core, 12GB memory, running windows 2003, 20GB disk space minimum required.

In LSPCM we can add criteria which checks following details in this section:

- Check for deployment diagram, which is responsible for describing system infrastructure.
- Check for mapping of deployment units on execution environment.
- Check for description of physical nodes specifications like servers, storage device requirements, details of memory, CPU's , performance of nodes.

**Reason :** These information of deployment view will helpful for deciding deployment units, physical nodes of the system. It's useful information for deployment stage of software.

### **LSPCM Check for Use case view :**

Use case view describes architecturally significant use cases of the system. The purpose of this section is to make clear how the architecture works when applied in the use cases. It

describes the set of scenarios and/or use cases that represent some significant, key business functionality.

In LSPCM we can check for following points:

1. Check for description of *architectural significant use cases*<sup>✕</sup> and quality feature associated with the use case.
2. Check for description explaining how chosen architecture works for considered use case.

**Suggestion 2:** Check for if any assumption made on the system during design has recorded in High Level Design.

**Reason:** It is necessary to maintain assumptions made on the system during design, since assumptions made on the system have high impact on the system, if this information has not recorded in the document and design has been carried considering these assumption that may lead to discussion in the later stages of software development and there may be possibilities of assumptions turning in to risk for whole system. This may impact on the time and budget of software development. e.g.: For a requirement , the system must send an email to the system administrator if a server exception occurs. In this case assumption must be made, that a mail server is available in the production environment and the application will have sufficient access rights to send the email. This assumption has to be noted in the High Level Design document. This suggestion helps to improve **understandability** of the design.

<sup>✕</sup>*Architectural significant use case is business critical that is this use case has a high usage level or is particularly important to users when compared to other features. The use case intersects with both functionality and quality attributes.*

**Suggestion 3:** In LSPCM, check is required to ensure whether High Level Design documents contains technology used for system development.

**Reason:** This information has to be mentioned in the High Level Design (HD), since HD is pass to the detailed design for further decision on design such as implementation details for each components, data structures and algorithms. Therefore High Level Design should contain which technology has to be used for system development. The rationale behind the selection of particular technology for the system development.

This suggestion improves **understandability** of the system architecture.

#### 4.3.3.2 Detailed Design

When LSPCM is used as an assessment tool for the artifacts of detailed design, it is essential to look into the artifacts with implementation point of view. In most of the offshore projects the detailed design and the implementation is done offshore, but the teams responsible for design may be completely different and may be operating from a different development center than the design team. Hence it is important to keep track of minute details with respect to the design documents to avoid any ambiguous details which might lead to confusions. To avoid such circumstances within the implementation team, it is mandatory that the detailed design artifacts meet the quality demands before these are handed over to the implementation team.

**Problem :** In detailed design, algorithms for each of the use case functionality flow is an important factor.

In this case study, some of the use cases miss the important flows like the alternate flows and the exception flows.

**Solution** : LSPCM should ensure checks for presence of main, alternative and exception or error flows for each of the algorithms under consideration.

**Problem** : The ID for a use case and its name is not consistent within the document names. (i.e. two different document names used for the same use case ID).

**Solution** : A check can be added to ensure consistency with respect to naming convention followed for the documentation and the models.

#### 4.3.4 Remarks on CASA

During the interview conducted with the project manager, we were intimated that CASA was not used during the analysis & design phases of this project as CASA was not released then. Hence standard RUP templates were followed. However, CASA artifacts which are built on RUP templates.

##### 4.3.4.1 Software Architecture Document (SAD)

This subsection explains suggestions for CASA- SAD based on the assessment of case study-1- SAD.

**Suggestion 1:** In CASA - SAD glossary section should be maintained.

**Reason:** High Level Design is base for detailed design and development discipline of software development. Glossary information maintained for software architecture document will help for better understanding the software architecture of the system.

**Suggestions 2:** Translation document needs to be maintained.

**Reason:** In case of case study-1 : SAD, class models, method names and attributes names are in Dutch. In this scenario, translated version (Dutch to offshore team language (English))

of SAD has to be maintained. In CASA- SAD, if SAD has been maintained in Dutch language, a translation document needs to be maintained. It is helpful for offshore development team in better understanding of the document.

**Suggestion 3:** Traceability matrix has to be maintained for software architecture.

**Reason:** Software Architecture is a base for detailed level design and development. It is necessary to ensure that all the functionalities are considered during design. For this concern, traceability matrix between requirements and design elements (classes, packages, components, interfaces and deployment unit) in high level design is useful to verify the software architecture.

In CASA – SAD, traceability matrix between requirements and design elements has to be maintained. This suggestion will prove helpful for verification of the software architecture.

**Suggestion 4:** In CASA- SAD, architectural principles can be maintained in particular format.

**Reason:** *Architectural principles*<sup>δ</sup> has used in defining architecture and are basis for architecture decisions. Often architectural principles are confused with requirements [15]. In order to avoid confusion and for a better understanding, architectural principles can be documented in particular format[21].

The recommended format for architectural principles is :

Name: Name of Architecture Principle

Statement : State architecture principle in brief.

Rationale : Should highlight business benefits adhering this principles and how it helps for making design decisions with one instance.

Implications: Should highlight the requirements, both for the business and IT, for carrying out the principle - in terms of resources, costs, and activities/tasks.

Name	<Name of the Architectural Principle>
Statement	<Brief Introduction of Architectural Principle>
Rationale	< Highlight business benefits adhering this principle>
Implication	<Benefits or risks that may occur by this architecture principle>

Table 7 : Standard format for architectural principles

<sup>6</sup>*Architectural principles define the underlying general rules and guidelines for the use and deployment of all resources and assets across the enterprise. Architectural principles are related to business objectives and key architecture drives [21].*

## Use case realization

**Problem** : The use case realization template for CASA does not capture the flow in detail.

The section is Flow of events which might make the designers miss out the details.

**Solution** : CASA use case realization template should be made more detailed by providing concrete sections for each of the desired subsections under the Flow of events.

The subsections which needs to be included consists of :

(1)Flow chart

(2) Alternate flows

(3) Preconditions

(4) Postconditions

(5) Error conditions

[18] says that the use case realization document provides a comprehensive overview of the system, using a number of different diagrams for representing the system functions.

While technically not a part of UML, use case realization documents are related to UML use cases. A use case realization document is text that captures the detailed functionality of a use case. The document contains the following parts:

- Brief description:
- Preconditions:
- Basic Flow:
- Alternate Flows:
- Postconditions

## 4.4 Case Study 2: A Banking application

### 4.4.1 Case description

This project deals with a development of a banking sales application. For several years, Capgemini and the this client have been collaboratively working on this application. There are periodic releases planned to provide additional functionalities.

Over the last five years, several releases of this application have been delivered. This project was executed based on onsite-offshore model and handed over this delivery to Capgemini maintenance team in India, who will maintain the application for client while it is running in production. This project has been executed using an onsite offshore team. The team in the Netherlands consisted of project management, requirement specifies, a software architect and a test manager. The offshore team consisted of project management, an implementation team and a test team. The onsite team consisted of seven human resources and the offshore team had a team of twelve.

### 4.4.2 Assessment Results

This section describes the assessment of the second case study which is a banking application. For this, evaluation was done with the LSPCM certification criteria for the high level design and detailed design product areas. This was followed by the assessment with CASA artifacts. Finally based on both the assessments and the defects observed, suggestions were provided for LSPCM and CASA and also for analysis & design documentation for the case study.

#### 4.4.2.1 LSPCM

##### **4.4.2.1.1 High Level Design**

This subsection describes the assessment of High Level Design of case study -2 using LSPCM : HD criteria. Based on this evaluation inconsistencies found in the HD are listed and possible solution are provided. The detailed evaluation can be found in the assessment report of case study-2 [3].

Found inconsistencies are categorized into major inconsistencies, minor inconsistencies in the document.

**Major Inconsistencies** : This paragraph describes inconsistencies occurred due to not maintaining required information (according to LSPCM criteria) in the Software Architecture Document. These inconsistencies have impact on overall quality of the design.

Traceability matrix is not maintained.

In RUP, at the end of the elaboration phase Software architecture should be stable. This stable architecture will be base for detailed design and development. Traceability matrix between requirements and design elements (classes, packages, components, deployment units etc.) play an important role in verifying whether all the requirements of the system has been considered in the design stage of software development. This information will help for assessor and maintainer for quick review of software architecture.

Missing glossary information.

In the software architecture document glossary information has not maintained. This information will help for new member of the team as well as reader for understanding the concepts and client related terminologies used in the document.

Citation for documents are not mentioned in the document.

In the document, citation for documents are not mentioned. In this scenario reader may refer different versions of the document. E.g.: functionalities are described in the use case model. In this statement citation for use case model has to be specified, otherwise there may be a possibilities that the reader may refer a different version of document and he may miss some of the information maintained in the recent version of the document. To avoid this situation, citation for the document has to be maintained in SAD.

Translation of use case from Dutch to English: In software architecture document some of the mentions of the use case names are in Dutch. Since SAD is passed to the offshore team, translation for the use case names has to maintained in the document. E.g.: use case name: ***Uitvoerenhandeling*** is in Dutch in the document, translated information for this has to be maintained.

**Minor inconsistencies:** Minor inconsistencies are not have impact on working of system functionality. Resolving these inconsistencies makes the document much more readable and understandable. Spelling mistakes, grammar issues, abbreviations of terms missing, different notations for same term and labels are not mentioned for some of the figures in the document, these are some of the inconsistencies considered as minor inconsistencies.

In a 70 pages of software architecture document, we could find approximately 5 grammar mistakes and 10 spelling mistakes per page. In the document, for the term *Vino* term two different notations are used that is *VINO* and *Vino*. Abbreviations of some terms are not mentioned in the document .e.g.: XA, SF, RO, GAA, LDAP, CSS etc abbreviations for these terms are not mentioned in the document. Some of the figures in the SAD does not have a label.

## Suggestions for Case Study-2 Software Architecture Document

- Minor inconsistencies that is grammar issues, spelling errors, inconsistencies with notation and terms can be resolved to improve the readability of the document.
- Abbreviations, definition of all terms in SAD has to be mentioned in abbreviation section of the document. It will be helpful information for reader, who is new to the system.
- Incomplete information mentioned that is citation for reference documents could have mentioned in the SAD. Since in the project various versions of the same document has been maintained, if citation has not been mentioned in the document, there is chances of referring a different version of the document other than mentioned in the SAD.
- Traceability matrix helps to verify software architecture whether it covers all the functionalities of the system. This will be added help for assessor and maintainer of the system.
- Glossary information should be mentioned for SAD. Glossary gives a brief information for various technical and client related terms. This information helps the reader to understand the system and architecture of the system.

#### **4.4.2.1.2 Detailed Design**

Similar to the previous case, here the inconsistencies found were categorized into the following three categories : Minor inconsistencies, major inconsistencies and incompleteness.

The LSPCM checks for detailed design were used as an assessment tool to verify the analysis & design documents of the project. The following observation was noted with the project documentation.

1. In the sample of 10 pages from a total of 124 pages in the given set of Use case realization documents there is on an average maximum of 1 spelling mistake per page.
2. There are two kinds of templates followed for the Use case realization documents. i.e. CASA template and the template provided by the client. The client templates fail to cover the details captured by CASA templates.
3. The document names do not follow a standard naming convention. These exist in both English as well as Dutch
4. There is no translation document maintained for the design details in Dutch.
5. Change history is not available for any of the documents.
6. Incompleteness with respect to the content in the client templates was not elaborated in any of the documents.
7. No standard design naming convention followed for attribute/method names.
8. The attribute and method names are in Dutch and no translations documents are maintained for the information.

### Minor inconsistencies

The following minor inconsistencies were observed in the design documents.

1. In the class models there is presence of spelling errors in the class or the method names
2. The same database name has been referred in a number of ways in a single class model. For Example : Db, Data Base, db are the conventions used to refer the same database in the class models.

### Major inconsistencies

The following major inconsistencies were noted in the project design documents and the models. These defects have been elaborated in the defect report for the case study [3].

1. In the design model, there is presence of ambiguities with the attribute data type. For example : the notation used for the username/password attribute is **String password : int** and **String username : int** whereas the correct notation should have been **password : String** and **username : String**
2. There were ambiguities in the notation used for the method names in the design models. For example : **String getUsername()** which specifies the intended return type of the method as **String**. However, when this notation is used with the modeling tools the method name is considered as the **String getUsername**. The correct notation which needs to be followed is **getUsername() : String**.
3. Ambiguities are also noted between the actual attribute data types and those which are used in the getter/setter methods. For example : the method **setBirthDate(int)** which sets an **int** value to the attribute **BirthDate**. However, the getter method for

this same method is denoted as **Date getBirthDate()** which returns the data type as **Date**.

4. There are a few methods modeled which fail to specify its return type.
5. Some of the data types specified for the attributes are nonexistent in the programming language used. For example: **Result** data type is being used in the models which does not exist in Java instead of **ResultSet**.

### **Incompleteness**

1. The use case realization mentions a rule to check the level of menu group. However, no validation criteria for this functionality has been mentioned.
2. In the Participating classes section of the use case realization the Javascript file details have been provided. Such javascript validation details should be provided in the validations section of the use case realization.
3. The date pattern is missing in the validation criteria. Whether dd/mm/yy or mm/dd/yy pattern to be used.
4. The validation criteria are not specified. The document mentions the validation criteria without explaining about it.
5. The functionalities of the validations are missing, The criteria <Ctrl-5> specifies the **dateBefore** and no validation criteria for it has been specified.
6. The validation refers the term **Authentication Level** which has not been specified.

#### 4.4.2.2 CASA

It was known from the interview conducted with the software architect that CASA templates, guidelines and checklists were used during the analysis & design phases. However, the following inconsistencies were observed in each of the artifacts.

##### **4.4.2.2.1 High Level Design**

###### Software Architecture Document

This subsection explains assessment of software architecture document of case study using CASA. For evaluation of SAD the same approach has been followed as explained in the section 4.4.3.1. For software architecture document client had provided the templates and the documentation was prepared following the CASA-SAD guidelines.

The assessment results are shown in the form of table. The detailed evaluation has been presented in assessment report of case study-2 [3].

CASA sections	Status in Case study -2	Reference in assessment report of case study -2 .
1. Introduction	Found	3 (pp. 36-37)
2. Architectural goals and constraints	Partially found	3 (pp. 38-39)
3. Quality considerations and decisions	Found	3 (pp. 40)
4. Use-case view	Found	3(pp. 41)

5. Logical view	Found	3(pp. 42-43)
6. Process view	Partially found	3(pp. 43-44)
7. Implementation view	Partially found	3(pp. 45-46)
8. Deployment view	Found	3(pp. 47)
9. Data view	Not Found.	3(pp.48)
10. Design constraints	Found	3(pp. 48-51)

Table 8 : Assessment for SAD for case study 2

Observation in Software Architecture Document :

For documenting software architecture of case study-2 client provided template has been followed and CASA-SAD guidelines has used for filling sections in SAD. By assessment of SAD using CASA –SAD template, we could observe some of the information mentioned in the CASA-SAD are missing in SAD of case study-2. Following paragraph explains missing information in SAD.

Intended user information has not been maintained:

This section becomes quick help to the reader by providing the list of relevant sections for each type of reader. E.g.: The SAD document is typically intended for:

- a. Business analysts: must read Chapter 2, should read Chapter 3 and 4.
- b. Product owner: must read Chapter 2 and Chapter 3. etc.

Overview of software architecture is not present:

This section describes in brief how the Software architecture of the system has been represented in the document. It helps understand the overall picture of the Software Architecture and how it has been defined.

During the analysis, it was found that *Definition, acronyms* and *abbreviations* section was not complete: In the document abbreviations for some of the terms were not mentioned. E.g.: XA, SF, RO, GAA, LPAD etc.

In logical view, work flow information was not maintained: Work flow explains most important business flows for the system. This information helps in understanding the basic flow of system functionality. Work flow is depicted in the form of flow diagram which shows overall process of the system. By this information, the reader gets overall picture of the system functionalities.

In logical view, summary of services offered by external systems and consumed from external systems was not present : According to CASA- SAD guidelines, in logical view summary of services offered by external systems and consumed from external system information has to be maintained. This information includes name of the service, version number of service, type of service and context of service.

In process view, overview process and process description has not been maintained: According to CASA-SAD overview of process and process description has to be maintained in the document. Overview of process view includes process name, mode of communication between process, priority of process, corresponding deployment unit. Process description [ref- CASA- SAD guidelines] includes memory usage, CPU usage, scheduling mechanism etc.

In implementation view component information and implementation mapping information has not been maintained : According to CASA- SAD guidelines, Component information and

implementation mapping information is not maintained in the SAD. Component information contains component name, technology used for developing the component, communication protocol used for communication between components and brief description of component. Implementation mapping describes how design packages are mapped onto namespace, project folders.

### **Analysis Model**

This subsection describes assessment result of case study-2 analysis model using CASA-modeling guidelines for analysis model. The details of assessment has been provided in the assessment report[3].

Observation in Case study – Analysis model:

In the provided models, the template structure provided by CASA has not followed. But, we could find the all the needed information like class diagram, layer structure of the system, assignment of packages for various layers of the system etc. are maintained.

Suggestions for Case study – Analysis model:

In the class diagram mentioned in analysis model clear distinction of control, boundary and entity classes has to be maintained.

#### **4.4.2.2.2 Detailed Design**

##### **Use case realization**

For this project, CASA technique was put to use for preparation of analysis & design documents. However, clients had provided their own templates. Hence those templates were used. But on detailed analysis, we found that not all use case realization documents used these client provided templates, and these were based on CASA templates. As a result of this, there was no standard template followed for these documents. Moreover, the templates provided by the client failed to capture all the information which CASA captured. In addition to these flaws, the documents which were prepared based on CASA templates were prepared wrongly. For example: In use case realization documents, the section **Participating Classes** depicts the possible files used instead of class descriptions.

The templates provided by client did not cover the Glossary, Issues, Flow of Events, Transaction Context and Exception Handling information.

##### **Design Model**

The Design model is prepared according to CASA modeling guidelines and thus includes all the sections as per the template.

##### **Implementation Model**

The implementation model is prepared as per the guidelines laid down as per CASA. However, this model fails to cover the deployment related details.

The sections in implementation model related to deployment i.e. Deployment unit elements, Deployment units and deployment unit structure is not present within the artifacts.

### Deployment Model

The deployment model is not maintained with this project. The details related to deployment is not maintained in the artifacts.

### Concerns

Concerns	Relevance	Document which should have addressed the concern
Checks for design model	The checklist for the design model ensures the presence of mandatory details which are important with respect to project implementation.	CASA design model checklist
Deployment related details not covered	A deployment diagram models the physical deployment of artifacts on nodes.	Deployment Model
Usage of wrong data types	The data types should have been modeled correctly lest creates ambiguity to the	CASA design checklist

	developer for the correct data type	
--	-------------------------------------	--

Table 9 : Concerns for the detailed design documentation for case study 2

#### 4.4.3 Remarks on LSPCM

##### 4.4.3.1 High Level Design

This sub section explains the suggestions for LSPCM- HLD product area based on assessment of case study-2 software architecture document and analysis model.

**Suggestion 1:** In LSPCM, sub criteria can be included to check system context diagram.

**Reason:** High Level Design is the first step to analyze and consider all the requirements for a software and attempt to define structure which is able to fulfill them. In this process context diagram and its description gives overall picture of the system environment and external system associated with the system.

The objective of the system context diagram is to focus attention on external factors and events that should be considered in developing a complete set of system requirements and constraints.

This information will be helpful for further design decisions in the software architecture as well as in detailed level design.

Reader will get overall picture of the system i.e. the application, external system and interfaces between them from this picture.

This suggestion will helpful for improving **understandability** of the system.

**Suggestion 2:** In LSPCM, a check can be included to verify whether definitions, acronyms and abbreviations have been maintained in the High Level Design.

**Reason:** Definitions, acronyms and abbreviation of terms used document have to be maintained. Especially in offshore development scenario(High Level Design artifacts are created by onshore team and this document is pass it to the offshore team for detailed level design) this information will helpful for understanding different terminologies used in the document. This suggestion will prove helpful in improving the **understandability** and **readability** of High Level Design document.

**Suggestion 3:** In LSPCM, a check in needed for verification of naming convention used for interfaces and objects.

Reason: In case of case study-2, client has provided naming convention for interfaces and objects. In check can be included to verify the same naming convention is followed for all interfaces and objects in High Level Design. By this check we could check uniformity maintained throughout the document.

This suggestion will improve **uniformity** and **understand ability** of the document.

**Suggestion 4:** In LSPCM , criteria can be included whether rationale behind each architecture decisions are described in High Level Design.

**Reason:** High Level Design is responsible for capture and convey architectural decisions made on the system. It is necessary to explain rationale behind each architectural decisions (e.g.: technology, tools, components, packages etc considered for design the system). By providing this information communication time for discussing rationale behind each decision can be saved. This will be added help for offshore development scenario.

This suggestion will improve the **understandability** of software architecture decisions.

*interface: Interface is a collection of operations that are used to specify a service of a class or a component. [ ref. UML book pp. 429]*

#### 4.4.3.2 Detailed Design

**Problem** : In the detailed design it is customary to have the details of the algorithms. For example the preconditions and the post conditions pertaining to an use case are important considering the algorithm design.

**Impact** : When the details such as preconditions or the post conditions are missed in the detailed design, it poses a challenge to the developer for its inclusion in the algorithm being developed during the implementation phase, after the design documents are handed to the development team.

**Solution** : LSPCM should ensure the check for the preconditions and the post conditions for an algorithm. Such checks can be included in the detailed design criteria in section SC1.2[40]

**Problem** : The exception behavior and alternate flows have not been specified in the use case realization documents which made use of the client provided templates. **Impact** : When a detailed design is under consideration, the alternate flows and the exception behavior is an important factor for algorithm design.

**Solution** : A check is required in the LSPCM which can verify the presence of exception behavior. These checks are to be included in the SC1 Completeness criteria[40] for detailed design.

**Problem** : The use case realization documents use two different kind of templates. **Impact** : The use of two different kind of document templates for the creating of same kind of document leads to ambiguity as these both documents capture different information and some information is not documented.

**Solution** : A check consisting on the usage of single document of same kind of documents should be enforced. This check can be included in the Uniformity section SC2[40]. Since such

templates used are standard to the company, this check can be included in the section SC2.3[1] which deals with Compliance with Company Standards

**Problem** : The use case realization documents are named in Dutch as well as English. No standard naming conventions have been followed throughout the documentation.

**Impact** : The development team may not be aware of the language in which the documents are named. Hence for offshore project, a translation document needs to be maintained if the names are not in English.

**Solution** : LSPCM should check for the uniformity in the naming conventions followed. If the names are in different language, a translation document needs to be maintained.

**Problem** : The attribute/method names in models does not meet the naming standard for the programming language used.

For Example, with java as the programming language, it does not allow spaces in the method/attribute name.

**Impact** : When the design models are used to generate the code body, the presence of unaccepted method or attribute names would generate erroneous code, and thus can create a situation wherein the developer may not be in a position to change the attribute name without affecting the other attributes in the class.

**Solution** : A conformance check can be included in LSPCM which would ensure that attribute/method names conform to the programming language standards. This check can be included in the section SC3.1[1].

**Problem** : The return types for the method should be appropriate with respect to the attribute used.

**Impact :** When an attribute is of certain data type, this data type should also be the return type of the method which accesses this attribute. If this is not the case, then there are chances of logical errors in the code during the code implementation.

**Solution :** A conformance check in LSPCM for the return data types of the methods would prove helpful in such a situation. This check should be included in the section SC3.1[1]

**Problem :** More than two data types have been specified for a single attribute in the design model.

**Impact :** This is an ambiguous situation whereby during the further stages of development it might lead to a confusion during the coding phases.

**Solution :** A conformance check to ensure whether the attributes are designed with proper data types. This LSPCM check needs to be included in the section SC3.1[1].

#### 4.4.4 Remarks on CASA

##### 4.4.4.1 Software Architecture Document

This subsection provides list of possible suggestions based on the assessment of case study-2 –SAD.

**Suggestion 1:** In the CASA:SAD reading guidelines can be included for architectural decisions, deviations from architecture, open issues and risks for software architecture.

**Reason:** Reading guidelines are specific symbols followed to specify architectural decisions, deviation from architecture, open issues and risks that may occur during Software Architecture. These guidelines – symbolic representation in software architecture document will be helpful for reader to quickly understand decisions, open issues, risks of the software architecture. This kind of guidelines will helpful during review of software architecture.

This suggestion will help for improving **understandability** of the document.

**Suggestion 2:** In CASA –SAD can include separate section which defines naming convention for interfaces and objects.

Reason: In case study-2 SAD, client has provided naming convention for interfaces and objects. In CASA- SAD these naming convention specific to client or organization has to be maintained in separate section.

This suggestion will help for improving **understandability** of convention followed for interface and object naming.

**Suggestion 3:** CASA – SAD should be flexible to add additional views for the architectural representation.

**Reason:** In case study-2 software architecture has been represented using use case view, logical view, process view, implementation view, deployment view and, additional security and access control view.

In CASA- SAD software architecture representation has explained considering use case view, logical view, process view, implementation view , deployment view and additional data view.

By considering the scenario of architecture representation that it defines additional security and access control view , therefore CASA- SAD should be flexible to add additional view.

#### 4.4.4.2 Use case realization

**Problem :** In use case realization document[3], the javascript files are mentioned in section which was intended to describe the responsible classes.

**Impact :** The class details are missed out in the documents and the document fails to describe the responsibilities of the participating classes for the use case.

**Solution :** The checklist for the use case realization ensures that the business rule specified for the use case is present in the document. But there is no check to ensure the presence of the description and the responsibilities of the classes. Hence this check needs to be included.

#### 4.4.4.3 Analysis Model

**Problem:** Check list for the analysis model has not been maintained.

**Impact:** Check is needed which verifies various classes of the system- control, boundary and entity classes and these classes assigned to various layers.

**Solution:** Checklist for analysis model has to be maintained.

#### 4.4.4.4 Design Model

**Problem :** The checklist for the Design Model is not maintained.

**Impact :** The essential checks which are helpful after the design model is developed will not be carried out to ensure its completeness.

**Solution :** A checklist for design model needs to be developed.

#### 4.4.4.5 Implementation Model

**Problem** : The implementation model does not give the sub sections against which the checks are enforced.

For Example, in the implementation model, the folder Implementation Elements contains the folders Components, Deployment unit Elements and Deployment units. But all the checks in the Implementation model checklist consists of checks only against Implementation elements.

**Impact** : The details in the subfolders are not checked explicitly since the designer may not be sure about the details checked.

**Solution** : The subsection against which the check is enforced should be mentioned. This alterations in the checklist will help in making the checklists more concrete and allow the designer to carry out these checks on the respective subsection.

#### 4.4.5 Recommendations for the case studies documentation

Following are the recommendations for the case study

1. Major inconsistencies should be eliminated before the documents are handed over to the offshore team for further stages.
2. The minor inconsistencies should be removed to improve the readability of the documents.
3. There should be a change document maintained for all the changes with the document and the models.

4. A document should be maintained for the models depicting the details about the method responsibilities, attribute functionalities.
5. If the method or attribute names are in Dutch, there should be a translation document maintained for ease of usage to the offshore team.

*N.B: The recommendations for the case studies have been suggested with the consent of the respective project team.*

---

## 5. Observations

### 5.1 LSPCM

In this section we enlist the suggestions for LSPCM in terms of High Level and Detailed Design

#### 5.1.1 High Level Design

In our thesis work, suggestions are provided for LSPCM : High Level Design product area. In addition with the existing criteria, these suggestions will help LSPCM to verify and validate quality of High Level Design in case of offshore and in general.

In our thesis work, some of the suggestions are provided based on the comparison between LSPCM High Level Design certification criteria and the CASA guidelines for SAD and analysis model.

**Suggestion 1:** Check for system context diagram

**Reason :** High Level Design is the first step to analyze and consider all requirements for a software and attempt to define structure which is able to fulfill them. In this process context diagram and its description gives overall picture of the system environment and external system associated with the system.

The objective of the system context diagram is to focus attention on external factors and events that should be considered in developing a complete set of system requirements and constraints. This information will be helpful for further design decisions in the software architecture as well as in detailed level design.

Reader will get overall picture of the system that is application , external system and interfaces between them from this picture.

This suggestion will be helpful for improving **understandability** of the system. Hence this check can be suggested for completeness criteria of LSPCM.

**Suggestion 2:** Check for definitions, abbreviations and acronyms in High Level Design.

**Reason:** Definition, abbreviations and acronyms in High Level Design will help the reader in understanding the system. In offshoring, high level design is initiated at onshore end and then passed to offshore team for further detailed design and development. In such a scenario, this information will be helpful to new team members in understanding the system.

**Suggestion 3:** LSPCM is independent of software process followed for product development. In our project the case studies provided are developed by following RUP software process, based on the analysis process for case studies, following set of suggestions are provided for architectural representation used in RUP process.

In RUP, architecture of the system is represented using 4+1 view model. It contains logical view, process view, implementation view, deployment view and use case view.

The following set of checks to verify the software architecture if it has been represented in the form of 4+1 view model.

Following suggestions for LSPCM for each of views in 4+1 view model

Logical view:

1. Check for sub-systems of the system depicted in the form of component diagram.
2. Check design package of the system depicted in the form of package diagram.
3. Check for classes and class utilities depicted in the form of class and object diagram.

4. Check for interfaces of the system which connects components of the system to the external system components.

5. Check for layers of the system and mapping of layers and corresponding logical components/ packages of layers.

Process view:

1. Check whether High Level Design contains, name of the processes, main modes of communication between processes.

2. Mapping of classes and subsystems onto processes and threads.

3. Process description.

Process description contain following information of processes : CPU utilization for processes, Concurrency, priority of processes, memory usage, Scheduling, availability, scalability.

Implementation view :

1. Check for components of the system.

2. Check for component mapping model that is mapping between design packages and components.

3. Mapping between component and deployment unit.

Deployment view :

1. Check for deployment diagram describing system infrastructure.

2. Mapping of deployment units on execution environment.

3. Description of physical nodes specifications like servers, storage device requirements, details of memory, CPU's , performance of nodes.

Use case view:

1. Check for brief description of architectural significant use cases and quality feature associated with the use case.

2. Check for description of how chosen architecture works for considered use case.

**Reason:** These checks will help LSPCM in order to verify different views of the system separately.

**Suggestion 4:** Check for any assumption made on the system during design that has recorded in High Level Design.

**Reason:** It is necessary to maintain the assumptions made on the system during design, since assumptions made on the system have high impact on the system, if this information has not been recorded in the document and design has been carried considering these assumption then it may lead to discussion in the later stages of software development and there may be possibilities of assumptions turning in to risk for whole system. This may impact the time and budget of software development.

e.g.: For a requirement, the system must send an email to the system administrator if a server exception occurs.

In this case assumption must be made, that a mail server is available in the production environment and the application will have sufficient access rights to send the email.

This assumption has to be noted in the High Level Design document.

**Suggestion 5:** In LSPCM, a check whether High Level Design documents contains technology used for system development.

**Reason:** This information has to be mentioned in the HLD, since HLD is pass to the detailed design for further decision on design such as implementation details for each components, data structures and algorithms. Therefore High Level Design should contain which technology has to be used for system development. The rationale behind the selection of particular technology for the system development.

This suggestion improves **understandability** of the system architecture.

**Suggestion 6:** In LSPCM , check in needed for check naming convention used for interfaces and objects.

**Reason:** In case of case study-2, client has provided naming convention for interfaces and objects. In check can be included to verify the same naming convention is followed for all interfaces and objects in High Level Design. By this check we can ensure uniformity throughout the document.

This suggestion will improve uniformity and understandability of the document.

**Suggestion 7 :** In LSPCM, criteria can be included whether rationale behind each architecture decisions are described in High Level Design.

**Reason:** High Level Design is responsible to capture and convey architectural decisions made on the system. It is necessary to explain rationale behind each architectural decisions (e.g.: technology, tools, components, packages etc. considered for designing the system). By providing this information communication time for discussing rationale behind each decision can be saved. This will prove to be added help for offshore development scenario.

This suggestion will improve the understandability of software architecture decisions.

**Suggestion 8:** In LSPCM, a check for brief description of exception handling mechanism should be added.

**Reason:** In High Level Design, exception handling mechanism gives brief details of error that may occur in the application and a general idea of how these errors are handled by the system. The response for occurred error (pop-up window showing error and user action for error) description of the class, object or component of the system responsible for error handling should be depicted.

These information gives brief idea of exceptions of the system to designer and developer for further detailed decisions based on class, method and response message, pop-window creation for handling exceptions.

**Suggestion 9:** Check for design constraints in the High Level Design.

**Reason:** Design Constraints refers to some limitations on the conditions on which a system is developed or on the requirements of the system. Constraints could be on the technology to be used, operating system used, tools used, resource management, report and printing methods (includes mechanism used for Reporting and Printing methods – Standard Report Formats(PDF/HTML), Components used for printing(e.g.: ActiveX Component etc). mechanism for Caching (e.g.: mechanism used for client –side caching . e.g.: Browser cookies or view state) etc. Origin of constraints may be from business goals, such as time to market and budget or clients demand for particular technology, tools or a particular commercial component was chosen because of organization’s relationship with its vendor. These constraints has to be maintained in High Level Design. Since, High Level Design documents are passed to further stages of software development, this information will help for team to understand the architectural decisions.

### 5.1.2 Detailed Design

LSPCM ensures the quality of the artifacts related to detailed design in the Detailed Design product area. For offshore projects, it is inevitable to identify the artifacts based on offshore perspective. The major concern in the offshore project is communication and understanding of the documents. Here LSPCM was analyzed for the **Detailed Design** product area. Based on CASA checklists, some of the sections in LSPCM needs to be updated. The suggestions for LSPCM provided are as follows. These suggestions were verified on the documents related to the case studies.

#### **For LSPCM from CASA**

**Suggestion 1:** As far as the detailed level design is concerned, the exception handling should be given important consideration. Hence a check is required in LSPCM which ensures the presence of Exception handling scenarios in the detailed design documents.

**Reason :** During the analysis of the case studies, the analysis of use case realization documents was essential. From these documents it was noted that not all documents contained the exception handling scenarios. Hence we include this suggestion with the “Completeness” criteria.

**Suggestion :** The specific criteria “Uniformity” does not check the naming standards for the documents.

**Reason :** The document names of use case realization documents for the case studies were found in two different languages English and Dutch. Since the client and the project team did not have any specific language standards to be followed, the offshore team faced issues with these documents as it was not well acquainted with the use case names which were mentioned in Dutch.

**Suggestion 2 :** The attribute names are found in two different languages Dutch and English. With outsourcing point of view, if the attribute names are written in another language not well known by the offshore team, then it is advisable to include a translation document depicting the translated version of the attribute names.

**Reason :** The attribute and method names use Dutch as well as English. However, the usage of such words and meanings is not obvious to the offshore team. Hence a standard language known to both the onsite and offshore team needs to be used for the documentation.

#### **For CASA from LSPCM**

**Suggestion 1:** The traceability matrix needs to be maintained along with the CASA artifacts.

**Reason :** The traceability matrix is used to correlate two documents specifying the high-level and detailed design artifacts. This is helpful with implementation point of view as the coding team can assure that all the requirements are being considered during the implementation phase. Hence this can be included in the CASA artifacts.

**Suggestion 2 :** The use case realization checklist should ensure the presence of flow diagrams for the particular use case realization flow.

**Reason :** The flow diagram for a use case gives insight into the behavior about the functionality and any alternative flows which exist. Hence the presence of a flow diagram is important. Moreover it makes easier for the implementation team to ensure that all the flows are covered in the code.

## 5.2 CASA

In our project one of the goals is to provide suggestions for CASA improvements. Following suggestions are provided based on LSPCM criteria.

### High Level Design

The following suggestions for CASA : Software Architecture Document based on LSPCM – High Level Design criteria.

**Suggestion 1:** In CASA : SAD traceability matrix between requirements and design elements has to be maintained.

**Reason:** As LSPCM criteria checks for traceability matrix, this information will help for verification of the Software architecture. Before start further design or development software architecture should be stable. Traceability matrix helps in this process of verification of software architecture by checking the trace between requirements and corresponding design elements (classes, packages, methods, components, process or deployment units). This check is required to determine whether the software architecture is stable enough at the end of elaboration phase in case of CASA- RUP.

**Suggestion 2:** In CASA- Software Architecture Document glossary information has to be maintained.

**Reason:** Glossary information provides brief description of terminology used in the document. This information is necessary to ensure that each stakeholder understands the same meaning of the terminologies used within the documentation. This will help to reduce the time required for understanding the terms and avoid ambiguities.

**Suggestion 3:** Translation document has to be maintained.

**Reason:** Translation document for Software architecture Document has to be maintained, if the documentation uses another language unknown to offshore team. i.e. Dutch. This information will help offshore team to understand the system and avoid the confusion with terminologies and other design details of system.

**Suggestion 4:** In process view of CASA- Software Architecture Document, mapping between classes, subsystems and processes , threads of the system should be described.

**Reason:** The mapping between classes, subsystems and processes, threads provide a link between logical view and the process view. This information will help for further detailed design stage. The map between classes and process will give clear picture for reader regarding the system development.

---

## 6. Conclusion

### 6.1 High Level Design

High Level Design is an artifact which makes sure that the design approach will yield an acceptable system. Architecture holds the key to system development, understanding and maintenance effort. The architectural decisions captured in the high level design are carriers of system qualities such as performance, modifiability, security etc. In short, architecture is the conceptual glue that holds every phase of the project together for all its stakeholders[17]. One of the goals of our project is to provide suggestions to improve certification model LSPCM for High Level Design product area as well as CASA : technique followed in Capgemini for Analysis, Design and Software Architecture. This suggestions will inturn helpful in improving quality of High Level Design documents for offshore software development. This chapter explains main results of project. Following subsections explain in brief, the problems in High Level Design which impact the quality of design and solutions for problems. Main results of quality analysis of High Level Design have been described.

#### 6.1.1 Problems & Impact

In today's rapid advancement in technology, software projects tend to become more complex. Due to the increasing complexity, creating quality system has become a major challenge for software industry. In order to tackle this challenge in Industry there is continuous improvement in engineering practices and technologies. In spite of these solutions, software- intensive systems continue to present formidable risks and difficulties in their design, construction, deployment and, evolution. Recent attempts to address this

challenge have focused on the earliest period of design decision i.e. Software architecture[15].

As mentioned in section 1.1, expression and communication of architecture lay a foundation for quality improvement of overall software product. Issues in High Level Design which have impact on the quality of the design include incomplete and abstract documents, absence of translation documents etc.

High Level Design documents play a main role in communication of architectural decisions made on the system with various stake holders like end-users, developers, system engineers and project managers etc. High Level Design documents should be detailed i.e. it should include design decisions and rationale behind the decisions made on the system.

If high level design is incomplete and not detailed enough to convey architecture of the system, the incomplete design will be pass to further stages of software development. This in turn has impact on the overall quality of the system. Due to these issues, software architect has to spend more time on modification or rewriting high level design again with architectural decisions, rationale and impact on overall system to stakeholders. In case of offshore development due to the long distance between onshore and offshore teams, these issues may lead to the budget and time overrun of the project.

### 6.1.2 Solution

In our project, to improve documentation and validation of software architecture, first step involves analysis of LSPCM : HD and CASA : SAD and analysis model ( which is categorized under high level design). The second step involves the analysis of case studies for High Level Design artifacts. Based on this assessment, inconsistencies are found in the high level design artifacts, which has impact on its quality. In the third and final step, suggestions are provided

for CASA - SAD, analysis model and LSPCM – HD for improving the quality of high level design. By following the CASA templates, guidelines and checklists for High Level Design documents will be complete and detailed enough for understanding and maintaining the quality of the Software Architecture.

Validation of High Level Design using LSPCM will help to ensure quality of High Level Design of the system before the handover of high level design to detailed design and development.

## 6.2 Detailed Design

While requirements are meant to specify what a program or system should do, design is meant to specify how the program/system should do it. A good software design is a significant means to ensure reliability.

Details design is process of refining and expanding software architectural designs to more detailed descriptions of the processing logic, data structures and data definitions. This continues until the design is sufficiently complete to be implemented [ANSI/IEEE Std. 610.12-1990]. The architectural decisions specified in the high level design are made more concrete by providing more details about it. i.e. algorithms are made more concrete by providing the sequence diagram etc.

The detailed design acts as an important step with implementation point of view. Hence the documents and models prepared at the detailed design stage should be complete and unambiguous. In short it should satisfy the criteria for a “quality” artifact as described in Chapter 1 . If the artifacts are ambiguous at this level, it might lead to a failure of the software due to wrong implementation.

The work carried out here focuses on the quality certification for assessment of the detailed design for offshoring. Here, the existing LSPCM model and Capgemini practices for analysis & design i.e. CASA have been considered for improvement. This chapter discusses the main results of our thesis work. The problem & impact is mentioned briefly and thereby the solution.

### 6.2.1 Problems & Impact

Project delay, budget overrun and wrong implementations are the concerns faced by the software service provider companies. This research highlights the defects faced by the offshored projects with emphasis on detailed design stage of the software architecture.

Offshoring is practiced by the companies with the belief that it lowers the development costs, but the quality of the product is compromised. With detailed design under consideration we had the following problem for the implementation phases.

Problem : Quality of the detailed design artifacts is compromised.

Impact : Presence of flaws in the detailed design artifacts i.e. the presence of ambiguities with the detailed design documentation and models. For example the attribute/method names, data types, method return types etc. get carried over to the further phases of development i.e. coding/implementation. Here the development team may assume the missing details which might lead wrong implementations.

### 6.2.2 Solution

The approach which was followed in this research project was as follows.

The certification model was applied to industrial projects and the defects were listed based on the assessment. These defects were then mapped to the processes followed which

contributed toward the wrong implementations. Improvements were suggested in the industry practices followed to avert such defects in future.

Also, improvements were suggested in the certification model by including new checks which could check the previously overlooked defects and these suggestions were verified on the industrial cases considered.

### 6.3 Result

The main result of this research is the suggested improvements for the quality assessment model LSPCM and the industry practice followed CASA. The work carried out included the design documents and hence the major issues faced with the detailed design have been identified which can lead to the failure of a project. As a result, these suggestions help in averting the undesirable situation of project delay, budget overrun or wrong implementations. The work focuses on improvement of the quality of the project artifacts as well as improvement in the process artifacts i.e. CASA. Thus we conclude improved artifacts for the CASA procedure which would deliver the quality artifacts for the projects.

### 6.4 Benefits to LaQuSo (Laboratory of Quality Software)

LSPCM developed by LaQuSo, helps in ensuring the creation of quality systems. The software product certification helps the organization benefit certainty or confidence in software artifacts. The suggestions provided during this research helps in improvement of LSPCM certification criteria for High Level Design(HD) and Detailed Design(DD) project areas with focus on offshore projects. The evaluation time however depends on the size of the project. Thus it can help the outsourcing partners, the outsourcer as well as the subcontractor, to

convince the other party that the deliverables are acceptable. This research has thus provided suggestions for improvements which help improve the quality of Software architecture & Design artifacts.

## 6.5 Benefits to Capgemini

The Capgemini Project Center is the home for technology services and a repository of all the project delivery of Capgemini. The primary goals of project center are increasing the margins and revenues of the projects being undertaken by Capgemini. The CASA technique by Capgemini was shaped at the project center. The CASA project was developed with the aim of improving productivity, collaboration with distributed teams, product quality and project predictability by setting up Architecture & Design standards. In this research suggestions have been proposed for improvement of CASA based the assessment of the software architecture and Design artifacts of the projects executed by Capgemini. These suggestions will ensure the improvement of artifacts of CASA by eliminating the occurrence of defects in the project. These improvements help to ensure that the software architecture & design is devised right and thus avoid time and cost overrun.

## 6.6 Future work

The field of software quality is still a fresh discipline. In this research we have focused on the quality assessment of the software architecture and analysis & design discipline artifacts. During this research there have been numerous instances where we observed the scope for extension of this research.

In this research we have analyzed LSPCM with the industry method CASA. This can be further extended by devising a review checklist for architectural design documents. Moreover a checklist is required for the analysis & design models for the CASA artifacts. There are certain checks noted in the Product Quality Index(PQI). However, these checks need to be detailed to include checks for the attribute data types and method return types etc.

Another area of research could be to assess the quality of source code for offshoring. In our research project we focused on the elaboration discipline of the software development cycle whereby the part of the design is done onsite and the detailed design is done at offshore. Analysis of LSPCM model in comparison with implementation for offshoring can be considered. Thus comparing LSPCM model with the source code standards and guidelines available followed within the organization.

Further research could be improvement of productivity in offshore development. This result of this research can be suggestions for offshore development in order to avoid the budget and time overrun of projects with quality consideration.

---

## 7. Bibliography

- [1] LaQuSo Software Product Certification Model(LSPCM): 1- February- 2010, "<http://alexandria.tue.nl/repository/books/633706.pdf>".
- [2] Angadi,N. Kudchadker.T : Case study 1: An Electronic Ticket Printer System, 2010.
- [3] Angadi,N. Kudchadker.T : Case study 2: A Banking Application, 2010.
- [4] A Software Product Certification Model. Petra Heck, Martijn Klabbers , and Marko van Eekelen. s.l. : Springer US, 2009, Software Quality Journal.
- [5] Anwer Shamsi – Master’s thesis: Offshoring: forget cost reduction focus on quality, Open University Nederland Graduation Report, 2007.
- [6] Carmel, E. en Tjia, P., Offshoring information technology – Sourcing and outsourcing to a global workforce, 2005, Cambridge University Press, Cambridge
- [7] Rational Software. Rational Unified Process. s.l. : Rational, The software development company, 2001.
- [8] Understanding offshoring : A research framework based on disintegration, location and externalization advantages
- [9] Bass L.;Clelments P.; Kazman R. "Software Architecture in Practice", 2nd Edition Reading, MA:Addison-Wesley, 2003.
- [10] Sanders, J., Software quality, A framework for success in Software development and support. Addison-Wesley, Dublin, 1994
- [11] Kruchten.P., "The Rational Unified Process: An Introduction", 3rd Edition (2004).

- [12] Yahaya, J. Derman, A. Hamdan, A., "SCfM\_PROD: A Software Product Certification Model", ICTTA 2008, Malaysia. 2
- [13] Software Product Maturity Model : 9-June-2010  
"<http://www.stsc.hill.af.mil/crosstalk/1997/08/product.asp>".
- [14] Clements, P. Klein, M. Kazman, R.: "ATAM: Method for Architecture Evaluation", Software Engineering Institute, Pittsburg, PA, August 2000.
- [15] IEEE Recommended Practice for architectural description of software-intensive systems.
- [16] The Software Reality Check- Diagnosing Software Projects, Architech solutions, White Paper
- [17] Clements, P. Bachmann, F. Bass, L. Garlan, D. Ivers, J. Little, R. Nord, R. Stafford, J.: Documenting Software Architectures : Views and Beyond, Addison- Welsley publication, 2003.
- [18] MSDN: Software Architecture and design. Web link : 29-May-2010,"  
<http://msdn.microsoft.com/en-us/library/ee658098.aspx>".
- [19] Deutsch M., Willis R., "Software Quality Engineering : A Total Technical and Management Approach", Prentice-Hall Series in Software Engineering
- [20] Warren S. Reid, Reviving the Drowning Large-Scale IT Project. California, USA,  
"<http://www.wsrcg.com/>: WSR Consulting Group, LLC, 2007".
- [21] The Open Group Architecture Framework( TOGAF) :Architecture Principle concept, 22-July-2010, "<http://www.opengroup.org/architecture/togaf8-doc/arch/toc.html>"
- [22] IBM WebSphere Help System: <http://publib.boulder.ibm.com/infocenter>

[23] Rumbaugh,J. Booch,G. Jacobson,I,: "The Unified Software Development Process", Addison-Welsley Object Technology Series, 1999.

[24] Software Engineering Institute : Software Architecture Overview, 19-July-2010, "<http://www.sei.cmu.edu/architecture/index.cfm>".

[25] Northrop L, "The importance of Software Architecture", SEI, Carnegie Mellon University, 2002

[26] Amrita, A., Sree Kumar, A., "Practical assessment of business and IT requirements for offshoring", Eindhoven University of Technology(TU/e), 2009

---

## Appendix A. List of Supplements

### **Defect reports for the case studies**

Case Study 1 : An Electronic ticket printer system

Case Study 2 : A banking application