

Architecture Coverage: Validating Optimum Set of Viewpoints



A THESIS PRESENTED TO

DEPARTMENT OF COMPUTER SCIENCE & SOFTWARE ENGINEERING

FACULTY OF BASIC & APPLIED SCIENCES

IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE

OF

MASTER OF SCIENCES (SOFTWARE ENGINEERING)

BY

SUNIA NAEEM



Accession No. TH 11826 -

MA / MSC

004

SUA

- 1 - Computer System
- 2 - Computer Science

International Islamic University Islamabad

Faculty of Basic & Applied Sciences

Department of Computer Science & Software Engineering

FINAL APPROVAL

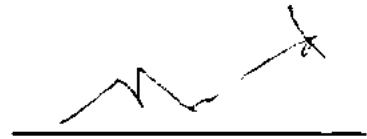
It is certified that we have read thesis entitled "Architecture Coverage: Validating Optimum Set of View Points" submitted by SuniaNaeem Reg. No. 280-FBAS/MSSE/F09. It is our judgment that this thesis is sufficient standard to warrant its acceptance by International Islamic University Islamabad for MS Degree in Software Engineering.

PROJECT EVALUATION COMMITTEE

External Examiner:

Dr. NaveedIkram

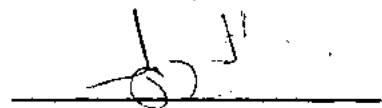
Associate Professor
Faculty of Computing
Riphah International University, Islamabad



Internal Examiner:

Dr. AyyazHussain

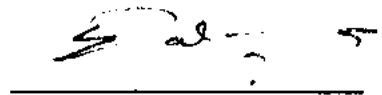
Assistant Professor
Department of Computer Sciences and Software Engineering
Faculty of Basic and Applied Sciences, IIUI



Supervisor:

Ms. Salma Imtiaz

Assistant Professor
Department of Computer Sciences and Software Engineering
Faculty of Basic and Applied Sciences, IIUI



Acknowledgement

I am very thankful to Allah who blessed me in the course of my life and especially during this thesis. It was only with Allah's help that I am able to complete this work.

I am extremely thankful to my thesis supervisor **MS. SALMA IMTIAZ** for her leadership, guidance and necessary support for completion of this research work. She always has been a great source of aspiration in looking into minute details, understanding the arguments and keeping focus in achieving the desired objectives of the research.

I am also really thankful to **DR. NAVEED IKRAM** and **SIR MUHAMMAD USMAN** for their support and guidance throughout my thesis.

I express my gratitude to my friends and family for their extraordinary support during my research work.

Last but not least my mother and father for supporting me spiritually throughout my life.

Declaration

I hereby declare and affirm that this thesis neither as a whole nor as part thereof has been copied out from any source. It is further declared that I have completed this thesis entirely on the basis of my personal effort, made under the sincere guidance of my supervisor. If any part of this report is proven to be copied or found to be a reproduction of some other, we shall stand by the consequences. No portion of the work presented in this report has been submitted in support of an application for other degree or qualification of this or any other university or institute of learning.



Sunia Naeem

280-FBAS/MSSE/F09

Dedication

I would like to dedicate our work to

ALMIGHTY ALLAH

Who has always showered His endless blessings upon us

&

To my **PARENTS**

Their words of inspiration and encouragement
In pursuit of excellence, still linger on

ABSTRACT

There are various software architecture viewpoint models but none of them provides complete coverage of software architecture domain. An optimum set of viewpoints can be selected from different software architecture viewpoint models that provide maximum coverage of software architecture domain than an individual architecture model. Software architecture coverage is the coverage of concepts that are required to effectively design and analyze software architecture. In this paper, an optimum set of viewpoints is selected by comparing five commonly used software architecture viewpoint models i.e. 4+1 view model, RM-ODP, SEI, Siemens and Rational ADS via a common comparison framework. These architecture models are compared on the evaluation criteria i.e. viewpoints, stakeholders and quality attributes. This evaluation criteria is based on IEEE Standard 1471 Recommended Practice for Software Architecture Description. Evaluation of models on a common comparison or coverage criteria allows us to combine viewpoints from these five software architecture viewpoint models. The resulting optimum set is validated in industry via multiple case studies.

Table of Contents

Chapter 1-Introduction.....	1
1.1. Motivation.....	2
1.2. Research Gap.....	4
1.3. Problem Statement.....	4
1.4. Research Aim.....	5
1.5. Research Scope.....	5
1.6. Research Question.....	5
1.7. Expected Outcome.....	5
1.8. Research Methodology.....	6
1.8.1. Case Study Objective.....	6
1.8.2. Data Collection Sources.....	6
1.8.3. Case study Design.....	7
1.8.4. Criteria for analysis of case study results.....	7
1.8.5. Criteria for Selection of Software Company.....	7
1.8.6. Type of project.....	7
1.9. Thesis Structure.....	8
Chapter 2-Literature Review.....	9
2.1. Related Work.....	10
2.2. Software Architecture Description.....	12
2.2.1. IEEE Conceptual Framework.....	13
2.3. Comparison Framework Elements.....	14
2.3.1 Software Architectural Viewpoints.....	14
2.3.2 Software Architecture Stakeholders.....	16
2.3.3. Software Quality Attributes.....	19
2.3.3.1. Run-time Qualities.....	20
2.3.3.2. Development-time Qualities.....	21
2.3.3.3. Business Quality Attributes.....	23

2.4. Comparison Framework.....	24
2.5. Mapping between Stakeholders, Viewpoints and Quality Attributes.....	25
Chapter 3-Models evaluation.....	26
3.1. Software Architecture Viewpoint Models.....	27
3.1.1. 4+1 View Model.....	27
3.1.2. SEI Viewpoint Model.....	28
3.1.3. RM-ODP Architecture Model.....	29
3.1.4. Siemens Four View Model.....	29
3.1.5 Rational ADS.....	30
3.2. Models Evaluation.....	31
3.3. Comparison Framework Coverage.....	33
Chapter 4-Optimum Set of Viewpoints.....	35
4.1 Optimum Set of Views.....	36
4.2. Mapping between Views.....	38
4.2.1. Use Case View-Conceptual View.....	38
4.2.2. Conceptual View-Module View type.....	38
4.2.3. Module View type- Component & Connector View type.....	38
4.2.4. Conceptual View- Component & Connector View type.....	38
4.2.5. Module View type-Allocation View type.....	39
4.2.6. Component & Connector View type -Allocation View type.....	39
4.2.7. Allocation View type-Test View and Module View type-Test View.....	39
4.2.8. Use Case View-Test View.....	39
4.2.9. Conceptual View-Test View.....	39
4.2.10. Test View-other views.....	40
4.3. Views Description.....	40
4.3.1. Use Case View.....	40
4.3.2. Conceptual View.....	41
4.3.3. Module View type.....	41

4.3.4. <i>Component & Connector View type</i>	41
4.3.5. <i>Allocation Viewtype</i>	42
4.3.6. <i>Test View</i>	43
4.4. Coverage of Optimum Set of Viewpoints.....	43
4.5. Implementation of Optimum Set of Viewpoints.....	45
4.5.1. <i>Case Study</i>	45
4.5.2. <i>Use Case View</i>	45
4.5.3. <i>Conceptual View</i>	46
4.5.4. <i>Module View type</i>	47
4.5.5. <i>Component & Connector Viewtype</i>	49
4.5.6. <i>Allocation Viewtype</i>	51
4.5.7. <i>Test View</i>	53
Chapter 5- Validation of Optimum Set of Viewpoints (Results & Analysis).....	54
5.1. Research Design.....	55
5.2 Data Sources.....	55
5.3. Data Analysis.....	55
5.4. Overview of Case Studies.....	56
5.4.1. <i>Project A</i>	56
5.4.1.1 Findings.....	56
5.4.2. <i>Project B</i>	57
5.4.2.1 Findings.....	57
5.4.3. <i>Project C</i>	57
5.4.3.1 Findings.....	58
5.5. Case Studies Results.....	58
5.5.1. <i>Coverage of Viewpoints</i>	58
5.5.2. <i>Coverage of Stakeholders</i>	59
5.5.3. <i>Coverage of Quality Attributes</i>	60

5.6. Discussion.....	61
5.7. Limitations:.....	64
Chapter 6-Conclusion & Future Work.....	66
6.1. Conclusion.....	67
6.2. Future Work.....	67
References.....	68
Annexure-Interview Questionnaire.....	71

List of Tables

Table 1.1.Thesis Structure	8
Table 2.1.Overview of Related Work	10
Table 2.2.Stakeholders Roles and their need of architecture documentation.....	17
Table 2.3.Elements of Comparison Framework.....	24
Table 3.1.Models Coverage of Viewpoints.....	31
Table 3.2.Models Coverage of Stakeholders.....	31
Table 3.3.Models Coverage of Quality Attributes.....	32
Table 3.4.Comparison Framework Coverage by Viewpoint Models.....	34
Table 4.1.Stakeholders Addressed by Optimum Set of Viewpoints.....	43
Table 4.2.Quality Attributes Addressed by Optimum Set of Viewpoints.....	44
Table 5.1.Overview of Case Projects.....	58

List of Figures

Figure 2.1.The ANSI/IEEE 1471 Conceptual Framework.....	13
Figure 2.2.Mapping between Stakeholders, Viewpoints and Quality Attributes.....	25
Figure 3.1.The 4+1 View Model.....	28
Figure 3.2.The SEI Viewpoint Model.....	28
Figure 3.3.The Siemens four view model.....	30
Figure 3.4.Rational ADS: Viewpoints and Views.....	30
Figure 4.1.Optimum set of Viewpoints.....	37
Figure 4.2.Use Case Diagram.....	46
Figure 4.3.Conceptual View.....	47
Figure 4.4 Layered View	48
Figure 4.5.Uses View	49
Figure 4.6.Client Server View.....	50
Figure 4.7.Communicating Processes.....	51
Figure 4.8.Deployment View.....	52
Figure 4.9.High Level Test Design and Procedures Details from Activity Diagram	53
Figure 5.1.Coverage of Viewpoints by Optimum set of Viewpoints.....	59
Figure 5.2.Coverage of Stakeholders by Optimum set of Viewpoints.....	60
Figure 5.3.Coverage of Quality Attributes by Optimum set of Viewpoints.....	61
Figure 5.4.Comparison of Coverage of Viewpoints by Optimum set of Viewpoints with individual viewpoint models' coverage.....	62
Figure 5.5.Comparison of Coverage of Stakeholders by Optimum set of Viewpoints with individual viewpoint models' coverage.....	63
Figure 5.6.Comparison of Coverage of Quality Attributes by Optimum set of Viewpoints with individual viewpoint models' coverage.....	64

Chapter 1

Introduction

1.1. Motivation

Software architecture is system's high level structure. The software architecture definition by Garlan and Shaw [9]:

“The architecture of a software system defines that system in terms of computational components and interactions between those components.”

The need for documenting software architecture is highlighted in [1], where the major reasons are its ability to communicate between stakeholders, to provide re-usable abstractions of software systems and to capture early design decisions. For describing complex architectures the most common approach was to make use of a heavily overloaded, single model that does not adequately represent the system and difficult to understand and manage [2]. Some of the disadvantages of using this approach are unreliable notations, over emphasis of one aspect, mixing of architectural styles and overlooking of individual stakeholder concerns [1].

A lot of work has been carried out to partition the architecture of the system into multiple views where each view highlights a different perspective. This approach helps in comprehension and understandability from stakeholders' point of view. Architects also come to an understanding that to develop successful software architecture we should consider a lot of different system structures simultaneously so architecture is many-faceted discipline. To capture architecture via more than one model is an appealing approach and used by many practicing architects. Software architecture research community seems to have decided that the only way to design architecture is by representing architectural designs via several related models (or views) [3].

Viewpoints are used to choose which view to produce for a particular system, and what information to represent in that view. Views and viewpoints usage has various benefits,

such as management of complexity, separation of concerns and improved communication with stakeholders. Viewpoint model [3] means a framework that describes the significant concerns

that need to be taken care of while designing software architecture. Generally software architecture models contain several viewpoints which define the models and concepts which can be used while dealing with the specific concern.

A recent research work by Nicholas May [1] surveys the different viewpoint models and highlights that existing viewpoint models need to be tailored because they do not address every concern of software architecture domain. The key purpose of this research work was to understand different software architecture models, their coverage of software architecture domain and their comparative strengths. The view point models are compared with respect to IEEE 1471-2000 Standard called the IEEE Recommended Practice for Software Architecture Description.

Along with comparison of models the research work [1] has also proposed a classification of viewpoints within a common framework that allowed to combine views from different viewpoint models and determining an optimum set of views with the purpose of providing maximum coverage to represent the architecture. Optimum set, even as do not provide complete coverage has the maximum coverage as compared to any individual viewpoint model and models' different vocabularies can be compared by common reference vocabulary.

Viewpoint models selected in this survey[1] are Kruchten's "4+1" View Model [20], Siemens Four View model [23], Software Engineering Institute (SEI) set of views [21], Rational Architecture Description Specification (ADS) [25] and ISO Reference Model of Open Distributed Processing (RM-ODP) [22].

Our research work focuses on the extension of comparison criteria described in [1]. We also evaluate viewpoint models on the extended coverage criteria and determine and validate an optimum set of views from all models as part of this research work.

1.2. Research Gap

There are a number of viewpoint models that create architecture document by means of the separation of the concerns. Each one of them defines viewpoints set and recognizes the concerns addressed. But none of them provide complete coverage of software architecture domain .The reason of selecting Nicholas work [1] is that it shows that different viewpoint models can be compared by a common comparison framework that allows to combine views of different viewpoint models and find an optimum set of viewpoints that can provide maximum coverage of software architecture domain as compared to any individual viewpoint model. Although Nicholas work is the basis for our work, but we will be identifying our own optimum set by extending the comparison framework used to evaluate different software architecture viewpoint models. We are extending comparison framework because in Nicholas work the outcomes were achieved by means of an initial arbitrary reference list and that was chosen from one of the models selected for comparison. Thus it seems better to collect an entirely independent vocabulary for the reference list.

The comparison framework will be extended to incorporate new elements and accommodate all necessary viewpoints from all the models. Comparison of models will be carried out by comparing each of their viewpoints against the comparison framework elements. We will be comparing five viewpoint models mentioned in Nicholas work because they describe architecture from multiple perspectives. Each of them recognizes the separation of concerns and state target stakeholders. Also, all of them focus on describing the software architecture structures instead of describing particular notations for each of these architecture structures.

1.3. Problem Statement

Existing viewpoint models do not provide complete coverage of software architecture domain. So an optimum set of viewpoints can be selected from different software architecture viewpoint models that provide maximum coverage of software architecture domain than any individual architecture model.

1.4. Research Aim

The aim of this research is to propose an architectural framework that will combine views from different viewpoint models and will provide more coverage of software architecture domain than any individual viewpoint model and to validate it in industry.

1.5. Research Scope

- To extend the coverage or comparison criteria mentioned in [1] which evaluates viewpoint models.
- Then evaluate viewpoint models on the extended coverage criteria.
- Determine an optimum set of views from all models.
- Validate this optimum set with the help of multiple case studies to achieve analytical generalization.

1.6. Research Question

RQ1 What is an optimum set of views for software architecture description that provides greatest coverage of software architecture domain than any of the individual software architecture viewpoint model?

1.7. Expected Outcome

Expected outcome will be optimum set of viewpoints from different models that will provide more coverage of software architecture domain than any individual viewpoint model. The resulting optimum set will be validated in industry so that it can become a basis for future documentation techniques or architectural documentation tools.

1.8. Research Methodology

At first step of this thesis, literature survey is used to find elements of comparison framework and to find coverage of software architecture concepts by already existing models. Then multiple case study design is used as research methodology to validate optimum set of views.

1.8.1. Case Study Objective

The objective of multiple case study approach is to validate that the research outcome i.e., optimum set of viewpoints provide maximum coverage of software architecture domain or coverage of the concepts that are required to design and analyze software architecture effectively. So case studies will be comparative descriptive in nature. Optimum set of viewpoints will be compared to software company baseline approach for designing software architecture of current project.

1.8.2. Data Collection Sources

Semi structured interviews will be used as a source for collecting evidence about the coverage of optimum set of views. Scripted interviews will be conducted and the questions will be prepared in advance so pre-defined questionnaire will be used and filled in print. Coverage will be measured on the basis of quality attributes, stakeholders and viewpoints modeled by the viewpoint model.

1.8.3. Case study Design

Multiple Case study design will be used to achieve analytical generalization about coverage of optimum set of views. Cases under review will be current projects in Software Companies on which optimum set of views will be applied.

Units of Analysis

- Software Architecture Documents

1.8.4. Criteria for analysis of case study results

A criterion is optimum set of views' coverage of software architecture concepts (i.e. viewpoints, stakeholders and quality attributes) that are required to design and analyze software architecture effectively. Comparison will be based on the coverage of these concepts by software company baseline approach for developing architecture and by optimum set of viewpoints.

1.8.5. Criteria for Selection of Software Company

Software Company should be CMM level 3 organization i.e. organization should have defined processes giving more consideration to documentation, standardization and also the organization should be capable of developing software product lines.

1.8.6 Type of project

Project should be at least medium sized software intensive project with requirement to build appropriate architecture and whose architecture plays critical part in the success of system. The architecture should be built by software architect or group of architects who have sound knowledge of software architecture view models.

1.9. Thesis Structure

Following table presents the overall structure of thesis.

Table 1.1.Thesis Structure

S No.	Structure Elements	Description
1	Introduction	Overall, introduction, background, research scope, research question and research methodology.
2	Literature Review	Detailed review of comparison framework elements i.e. software architecture viewpoints, stakeholders and quality attributes.
3	Models Evaluation	Evaluation of viewpoint models on a common comparison framework in order to combine views from different models.
4	Optimum Set of viewpoints	Explain optimum set of views that combine views from different viewpoint models. Describe views and mapping between views.
5	Validation of Optimum Set of Viewpoints (Results & Analysis)	Shows results of multiple case studies which are conducted to validate optimum set of viewpoints.
6	Conclusion & Future Work	Describe the conclusion drawn after validation of optimum set of viewpoints w.r.t the coverage of viewpoints. Stakeholders and quality attributes

Chapter 2

Literature Review

2.1. Related Work

In this section research works that have made an effort to compare different viewpoint models are mentioned. We have compared literature review on the basis of their focus. But our research focus is not only to do comparison of models but also use it for proposing an architectural framework.

Table 2.1.Overview of Related Work

Paper id	Paper title	Focus	Description	Models reviewed/compared	Criteria for comparison	limitations
[3]	What is included in software architecture? A case study in three software organizations	Review of software architecture models and their strengths and weaknesses are highlighted.	In this paper author surveyed some architecture models and conduct a case study on the usage of software architecture documentation practices in the Telecommunications industry.	RM-ODP, US Department of Defense frameworks TAFIM, C4ISR, 4+1 view, Zachman framework	No specific criteria	Models are reviewed from literature and their details, benefits and deficiencies are based on literature review.
[4]	Experiences Using Viewpoints for Information Systems Architecture: An Industrial Experience Report		In this paper viewpoint sets are applied to development of information systems and evaluated so weaknesses and strengths of every set of viewpoints is described and few general observations about their definition and use are presented.	4+1, RM-ODP, Siemens, Garland and Anthony	Industrial experience	Comparison is based only on observations of author. No common reference vocabulary is used for comparison.
[5]	A Comparative Analysis of Architecture Frameworks		This study provides analysis and comparison of six architecture frameworks categorized by fundamental elements such as their goals, inputs and outcomes. It	Zachman Framework, 4+1, Federal Enterprise Architecture Framework (FEAF), RMODP, Department of Defense	goals, inputs and outcomes	More focus is on classification of frameworks not on frameworks' deficiencies.

			provides classification of architecture frameworks into Software Architecture Frameworks and Enterprise Architecture Frameworks and identifies some of their deficiencies.	Architecture Framework (DoDAF), The Open Group Architecture Framework (TOGAF)		
[26]	The many faces of architectural descriptions		In this paper author provides overview of two classes of architecture frameworks Software Architecture Frameworks and Enterprise Architecture Frameworks and find some dimensions which can be helpful to understand architecture documents	Zachman Framework, the Information Framework (IFW), Integrated Architecture Framework (IAF), The Open Group Architecture Framework (TOGAF), Methodology for Architecture Description (MAD), 4+1, Siemens	No specific criteria	More focus of comparison is on the difference between two classes of architecture frameworks.
[27]	Software Architecture Evolution and Evaluation		In this paper author surveyed few architecture and compared them on the basis of methodologies and techniques they use and suggested that more architecture styles can be added to yield new architecture framework which focus on quality	Zachman Framework, 4+1, Federal Enterprise Architecture Framework (FEAF), RM-ODP, Department of Defense Architecture Framework (DoDAF), The Open Group Architecture Framework (TOGAF)	methodologies and techniques used in the frameworks	Focus of this comparison is to state only general advantages and disadvantages of architecture frameworks
[6]	Comparing the SEI Views and Beyond Approach for	Comparison of one model with other models to	This paper compares SEI with IEEE 1471 and show compliance	SEI, IEEE 1471	Requirements imposed by IEEE 1471	Only compliance of one viewpoint model is considered .Compliance of

	Documenting Software Architectures with ANSI-IEEE 1471-2000	prove its effectiveness.	of SEI with IEEE 1471.			other viewpoint models are missing.
[7]	A Rationale Focused Software Architecture Documentation method (RFSAD)		In this paper problems related to the architecture documentation are described, and to document software architecture new method (Rationale Focused Software Architecture Documentation (RFSAD)) is recommended to address those mentioned problems. And also new method is compared with available methods.	IEEE 1471, SEI, 4+1 view, RFSAD	Problems find with already existing models	Focus of this comparison is to prove effectiveness of new method, and comparison is of one model with other three models. So element of biasness is present towards new method.

2.2. Software Architecture Description

To compare the software architecture viewpoint models a common comparison framework is required. It will evaluate the models for depth of description they provide (viewpoints they address), communication needs they satisfy (stakeholders), and quality attributes they address. This framework will be based on the IEEE 1471-2000 Standard, called the IEEE Recommended Practice for Software Architecture Description [19]. This standard describes the concepts and their relationships for documenting the software architecture.

A well-known definition of architecture description from [16] is:

"An architectural description (AD) is a set of products that document architecture in a way its stakeholders can understand and demonstrate that the architecture has met their concerns."

Due to extensive importance of software architecture and emergence of software architecture documentation practices which could be standardized, IEEE 1471; the IEEE Recommended Practice for Software Architecture Description was developed. So the focus of this standard is software architecture description and it sets requirements for architecture description and addresses the contents and organization of architecture descriptions.

2.2.1. IEEE 1471 Conceptual Framework

IEEE 1471 defines concepts that are relevant for architectural descriptions. IEEE 1471 conceptual framework has some key elements and these elements are architectural stakeholders, concerns and viewpoints [19].

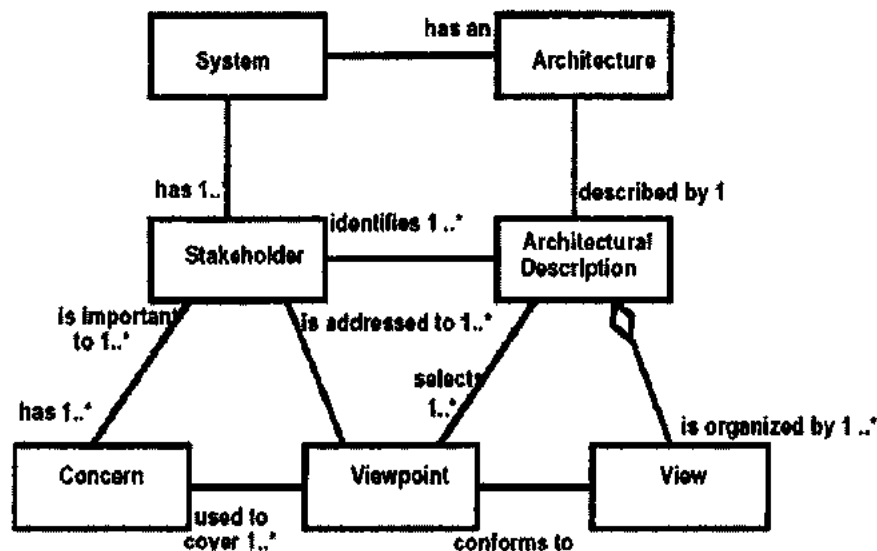


Figure 2.1. The IEEE 1471 Conceptual Framework [19]

IEEE 1471 considers stakeholders and their respective architectural concerns as essential elements in an architectural description. Architectural concern is a matter of importance to one or more stakeholders relating to the architecture. Another major element of ANSI/IEEE 1471 is that

every architecture view in an architecture description is defined relative to an architectural viewpoint as we know architecture description is planned into multiple views and each one of them denotes the system architecture with respect to a set of related architectural concerns. So an architecture viewpoint captures the rules for analyzing and constructing a particular view and acts as a view template so it can be reused across many architectural descriptions.

2.3. Comparison Framework Elements

2.3.1. Software Architectural Viewpoints

A view is a representation of a whole system from the perspective of a related set of concerns following the conventions of its viewpoint whereas software architectural viewpoints describe the resources and rules for constructing views [19] and they create purposes and audience for views. So a viewpoint specifies the models and techniques and is a pattern for constructing a view. Viewpoint is a general, reusable solution to a commonly occurring problem in software architecture within a given context. Whereas software architectural style is a specific method of construction, characterized by the features that make it notable. An architectural style defines “a family of systems in terms of a pattern of structural organization; a vocabulary of components and connectors, with constraints on how they can be combined” [9].

Viewpoints reason about quality attributes so architecture description should provide enough details or information necessary to analyze quality attributes. So we have considered the same list of viewpoints in comparison framework as mentioned in Nicholas work [1]. Only one addition is made in the list of viewpoints, i.e., conceptual viewpoint. Conceptual viewpoint [8] describes the system in form of system’s major design elements and relationship between them this viewpoint is very important because it is strongly linked with the problem domain and acts as an important means of communication when the architect interacts with domain expert. So modules can be clearly defined in module view and impact of changes in requirements can be

minimized. Viewpoints are not system specific so there are pre-defined and reusable viewpoints. Following is the detail of the most common and useful software architecture viewpoints [12].

Conceptual: This viewpoint [8] shows functional structure of the system and defines set of conceptual components linked by a set of connectors. This viewpoint is independent of implementation decisions so it is beneficial to understand interactions between entities in the problem space and planning functionality.

Decomposition: Units of this structure or viewpoint are modules that are associated with each other by "is a sub module of" relation and shows in what way larger modules are decomposed into smaller modules repeatedly until modules can be easily understood.

Uses: Units of this viewpoint are also modules which are associated by the uses relation. One module makes use of second if correctness of the first needs the presence of the second. So it defines how dependent modules are on each other.

Layered: When the uses relations are organized in a specific way, a set of layers appears, in which each layer is a logical set of related functionality. This viewpoint is frequently designed as virtual machines which hide implementation details lower from the layers above, creating portability.

Class or generalization: Elements in this viewpoint are classes and the relation is "inherits-from" or "is-an-instance-of". This viewpoint provides rationale about collections of similar behavior and also parameterized differences which are caught by sub classing. So it provides motivation about the incremental addition of functionality and re-use.

Process: this viewpoint deals with the system's dynamic aspects. Elements are processes or threads that are connected with each other by communication, synchronization etc. This viewpoint is important to plan a system's execution availability and performance.

Concurrency: This viewpoint helps to find events of parallelism and situations where resource contention may occur. Elements are logical threads. So it is used to find the requirements for issues management associated with concurrent execution.

Shared data or repository: This viewpoint contains components and connectors that access, store and create persistent data and how data is consumed and produced by runtime software elements. So it represents how data is consumed and produced by runtime elements.

Client-server: This is a good viewpoint when the system is constructed as a group clients and servers. Its constituents are clients, servers and protocols or messages which they share to carry out the system's work.

Deployment: This viewpoint provides details about how software is allocated to hardware elements. Relation in this viewpoint is "allocated-to" which shows that software elements reside on which physical units.

Implementation: This viewpoint describes how software elements are mapped to file structure in the system's development environment.

Work assignment: This viewpoint allocates responsibility to the appropriate development teams such as to implement and integrate the modules. As a part of the architecture it makes decision about who will do which work that has architectural plus management implications

2.3.2. Software Architectural Stakeholders

Stakeholder of software architecture is someone who has a vested interest in it. They implicitly or explicitly motivate the whole shape and direction of the architecture, which is developed merely aimed at their benefits and needs [16].

IEEE-1471 defines system stakeholders as:

“A stakeholder in software architecture is a person, group, or entity with an interest in or concerns about the realization of the architecture.”

Stakeholders are consumer of software architecture description and architecture description serves as mean of communicating design decisions between stakeholders so creating architecture is not enough it has to be communicated in a way so that stakeholders use it properly for their respective use [24]. There is variety of stakeholders involved in the development of software system. So architecture documentation's uses will be varied because stakeholders are well-defined by their interactions with software systems. Every stakeholder will need documentation concerning their uses. Testers for example will want to see documentation in terms of the functionality decomposition of the system. Nicholas [1] list of stakeholders is incomplete and do not provide satisfactory definitions of the stakeholders relevant for the analysis we will perform. Our analysis will be based on the stakeholders that are architecture documentation's consumers. These stakeholders will make the viewpoints analysis possible as they provide coverage of stakeholders that are addressed by the viewpoints. Nicholas list of stakeholders are also consumer of software architecture documentation but some key stakeholders and business stakeholders are missing such as product managers, business analysts and marketers. The following table shows stakeholders and reasons for their consideration

Table 2.2. Stakeholders Roles and their need of architecture documentation

Stakeholder roles	Description of Roles	Use of architecture documentation
Architect[12]	Liabile for the development of the architecture and its documentation.	Record design decisions. Providing Evidence that the architecture satisfies its requirements.
Requirements engineers[12]	Requirement Engineers gather requirements and negotiate and make tradeoffs between competing requirements.	Discuss and make tradeoffs between competing requirements.
Implementer[12,24]	Develop specific elements according to designs, requirements, and the	To understand constraints on development activities.

	architecture.	
Sub system Architect and Designer [12]	Manage sub systems.	To determine contention of resources and establish performance and other kinds of runtime resource consumption budgets
Tester[12]	Perform test and verification of the system or its elements against the formal requirements and the architecture.	Create tests based on the behavior and interaction of the software elements.
Integrator[12,24]	Take individual components and integrate them according to the architecture and design.	Creating integration plans and procedures. Finding the source of integration failures.
Maintainer[12]	Fix bugs and providing enhancements to the system throughout its life. .	To understand the consequences of a change.
Managers[24]	Accountable for the functioning of the organizational entity that owns the system, managerial/executive responsibility, responsibility for defining business processes etc.	Considering the ability of the architecture to meet business goals.
Designers of other Systems [12,24]	Manage a system with which this one must interoperate, and its interface .	To describe the operations provided and required, and the protocols for their operation.
Product Line Manager[12]	In charge for development of an entire family of products, all built using the same core assets.	Finding about a potential new member of a product family is in or out of scope.
Quality Assurance Team [12]	Analyze the architecture to make sure it meets critical quality attribute requirements.	Investigating satisfaction of quality attribute requirements of the system based on its architecture.
User[10,11,24]	Actual end users of system	Get useful insights about the system, what it does, and how it can be used effectively. How it runs on the platform. How it interacts with other software.
Customer[10,11,24]	Customer pays for the system and ensures its delivery.	Assures that required functionality and quality will be delivered. Evaluate progress. Estimate cost. Estimate what will be delivered, when, and for how much.

Project Manager[10,11,24]	Accountable for planning, sequencing, scheduling, and allocating resources to develop software components and deliver them to integration and test activities.	Set budget and schedule, evaluating progress against established budget and schedule identifying and resolving development-time resource contention.
Production Engineer[16]	Responsible to deploy, and manage software and hardware environments in which the system will be built, tested, and run.	Understanding the architectural elements that are delivered and to be installed at the customer's or end user's site, and their overall responsibility toward system function.
Supplier[16]	Supply or build the hardware, software or infrastructure on which the system will run	Knowledge of Interface and integration rules.
System administrator[16]	Run the system once it has been deployed	Knowledge of operational concepts and procedures.
Business analyst[16]	Capture and document detailed business requirements.	Analysis of benefits of the system with regard to the developing organization and its customers. Analysis of product strategy.
Product manager[13]	Accountable for software product's application domains and markets and formulates relevant requirements.	Look for support for long term company strategy.
Marketer[14,15]	Make the value judgments of the system's functionality	Knowledge of system's competitive features for selling.
Support Staff[16]	Responsible for providing support to users of the system when it is running.	Require information to solve problems with users.

2.3.3 Software Quality Attributes

An architecture description should address all stakeholders' concerns. Software architectural concerns form the basis for completeness. Software architecture description should address stakeholders' concerns otherwise it will be considered incomplete.

A well-known definition of concerns from [16] is:

"A concern about architecture is a requirement, an objective, an intention, or an aspiration a stakeholder has for that architecture."

Concerns [18] are normally driven by the need for the system to exhibit a certain quality attributes rather than to provide a particular function. So there is inherent need to consider quality attributes in each architecture view. So we are considering system quality attributes as concerns. Quality attributes can be classified into three types: Run-time, development-time and business. Nicholas list of concerns [1] is inadequate in all these three types. Below [17, 12] is the detail of all three types showing the importance of presence of these three types of system quality attributes.

2.3.3.1. Run-time Quality Attributes

Functional requirements are about what the system must do, and run-time qualities are about how well these functional requirements are satisfied so how well is judged in terms of some characteristics. So run-time quality attributes include:

Functionality: capability of the system to do the work for which it was intended. So, if components have not been allocated to correct responsibilities the system will be incapable to perform the required functionality.

Performance: refers to the responsiveness of the system it is time required to respond to an event or the number of events processed in some interval of time. As communication generally requires longer time than computation so it is every so often a function of how much interaction, and communication is between the system's components so this is visibly an architectural issue.

Availability: refers jointly to "time to failure" and "time to repair" and these two are addressed by architectural means. It is described as percentage of time the system is up and running and estimated by the measurement of time between failures plus how quickly the system resumes its operations in the event of failure

Reliability: is the system's capacity to keep operating over time. It is generally measured with "time to failure". Reliability is tied to the architecture because mean time to failure is extended

mainly by making an architecture fault tolerant which can be accomplished by replication of processing elements and connections that are critical in nature within the architecture.

Security: is ability of the system to resist unauthorized attempts at usage and denial of service although still providing its services to legitimate users.

Safety: prevent bad things from happening it can be an asset of a system so that confidence can reasonably be placed in the absence of accidents.

Usability: ease-of-use, learnability, memorability, efficiency, error avoidance, error handling, satisfaction etc.

Supportability: system's capability to provide information useful for identifying and resolving issues when system fails to work correctly.

Configurability: is a post deployment components modification or configurations such that they are capable of using a new service.

Scalability: is providing support for additional users or sites, or higher transaction volumes.

Interoperability: capability of collection of parts that constitute a system to work with another system.

2.3.3.2. Development-time Quality Attributes

Besides developing systems that satisfy users, the development organizations are also interested in artifacts' properties (design, architecture, code, etc.) of the development process. These artifacts' qualities affect the effort and costs associated with current development plus provide support for future changes or uses (enhancement, maintenance or reuse). Development-time quality requirements examples are:

Modifiability: is very closely associated to system architecture because the capability to do changes rapidly and cost effectively follows straight from the architecture. As architecture defines the components and its tasks, it too shows the situations in which each component will have to change.

Reusability: refers to how to design a system such that some of its components or its structure can be reused again in future applications.

Testability: refers to degree of ease with which test criteria can be built for the system and its components, and tests can be executed to determine if the criteria are met. It is related to several architectural issues such as separation of concerns, level of architectural documentation and the extent to which the system uses information hiding and testability is also benefited by incremental development.

Portability: is system's capability to run under different computing environments such as software, hardware or combination of the two. So platform particular concerns in architecture usually take the form of a portability layer i.e., a layer of services that provide application software an abstract interface to its environment. A system is portable to the level that all suppositions about any specific computing environment are limited to one component.

Evolvability: refers to providing support for new capabilities or capacity of system to exploit new technologies.

Localizability: capability to make adaptations due to regional differences.

Integrability: capability to enable the separately developed system's components to work together correctly that can be determined by components' external complexity, their interaction rules and the extent to which responsibilities have been clearly partitioned so all these are architectural issues.

2.3.3.3 Business Quality Attributes

Additional to quality attributes that relate directly to a system, there are several business quality attributes which repeatedly form a system's architecture. Following is the discussion of business quality attributes [12]:

Time to market: If a system or product has competitive pressure or short window of opportunity implementation time becomes important. This leads to pressure to buy or otherwise re-use existing elements. So the capability to insert or deploy a system subset can be determined by on the decomposition of the system.

Cost and benefit: The budget of development should not be exceeded. Different costs will be produced from different architectures. So highly flexible architecture will usually be more costly to build than rigid one, while it will be less costly to modify and maintain.

Projected lifetime of the system: When the system is proposed to have a long lifetime, portability, scalability, and modifiability become important.

Targeted market: system's features and the platforms on which a system runs will define size of the potential market. Therefore, functionality and portability are fundamental to market share.

Rollout schedule: Considering that product is to be declared for instance as main functionality with many features released later, customizability and flexibility of the architecture becomes important.

Integration with legacy system: When the new system has to integrate with existing systems, appropriate integration mechanisms should be defined.

2.4. Comparison Framework

Table 2.3 shows elements of our comparison framework comprising viewpoints, stakeholders and quality attributes.

Table 2.3.Elements of Comparison Framework

Viewpoints	Stakeholders	Quality Attributes
Conceptual Decomposition Uses Layered Class/Generalization Process Concurrency Shared Data Client-Server Deployment Implementation Work Assignment	Architects Requirements Engineers Sub-System Architects and Designers Implementers Testers Integrators Maintainers External System Architects and Designers Managers Product Line Managers Quality Assurance Team Users Customers Project Manager Production Engineers Suppliers System Administrators Business Analysts Product Managers Marketers Support Staff	<u>System Run-Time</u> Functionality Performance Capacity/Space Availability Reliability Security Safety Usability Supportability configurability Scalability Interoperability <u>System Development-Time</u> Modifiability Reusability Testability Portability Evolvability Localizability Integrability <u>Business</u> Time to market Cost and benefit Projected lifetime of the system Targeted market Rollout schedule Integration with legacy systems

2.5. Mapping between Stakeholders, Viewpoints and Quality Attributes

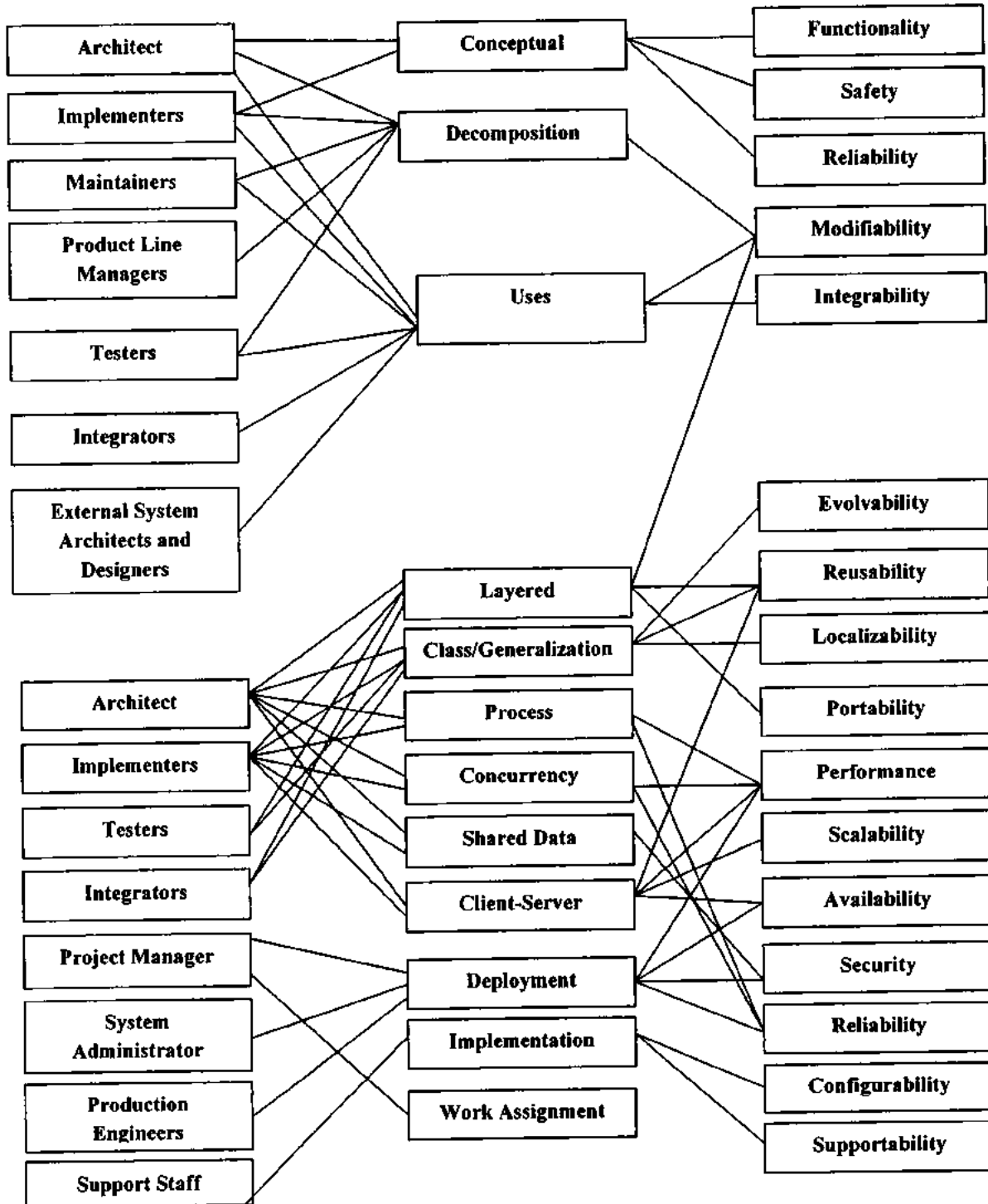


Figure 2.2. Mapping between Stakeholders, Viewpoints and Quality Attributes

Chapter 3

Models Evaluation

The models chosen for evaluation are as follow:

1. *“4+1” View Model [20].*
2. *Software Engineering Institute (SEI) set of views [21, 24].*
3. *ISO Reference Model of Open Distributed Processing (RM-ODP) [22].*
4. *Siemens Four View model [23].*
5. *Rational Architecture Description Specification (ADS) [25].*

All five models describe software architecture from multiple perspectives. Each one of them specifies stakeholders and identifies the separation of concerns. Also these models focus on describing software architecture structures instead of defining specific notations for each of these structures.

3.1. Software Architecture Viewpoint Models

3.1.1. 4+1 View Model

Kruchten’s “4+1” View Model [20] contains five interrelated views which allow different stakeholder concerns to be addressed separately. Kruchten visualized an iterative approach to design architecture and start with the description of the scenarios. In the Logical View the architect can now model the main abstractions identified from the domain. In the Development View logical elements can be mapped to modules, and to processes and tasks in the Process View. Lastly in the Physical View processes can be mapped to the hardware. Additional scenarios are modeled in each subsequent iteration following the sequence, until the architecture becomes stable. This is apparent that the 4+1 views are not independent so without first having proceeded through the scenarios and logical views it would be difficult to model other views. Thus, views depend on an iterative process to create them.

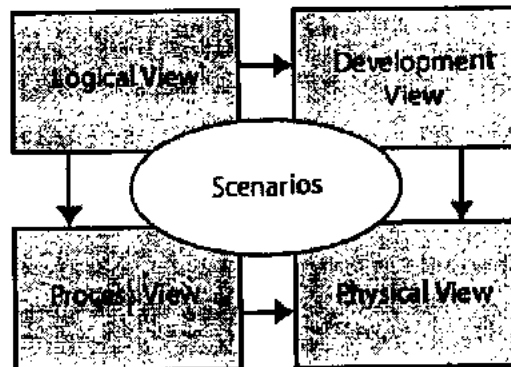


Figure 3.1. The 4+1 View Model [1]

3.1.2. SEI Viewpoint Model

SEI model [21, 24] is a practical approach for software architecture documentation. This model is centered on the notion of views, and holds that documentation consists of appropriate views and the information that relates to more than one view. Views are assembled into viewtypes, agreeing with the three ways an architect must think about a system that is as implementation units set, set of runtime elements interacting with each other to carry out the system's work, and set of elements that relate to external structures in its environment. SEI viewpoint model uses styles to decide the views that can be generated.

Viewtypes	Styles	Views
Module	Decomposition	Styles applied to particular systems
	Generalization	
	Uses	
	Layers	
Component-and-connector	Pipe-and-filters	
	Shared data	

	Communicating processes	
	Peer-to-peer	
	Client-server	
Allocation	Work assignment	
	Deployment	
	Implementation	

Figure 3.2. The SEI Viewpoint Model [21]

3.1.3. RM-ODP Architecture Model:

TFH 11826
 The Reference Model of Open Distributed Processing (RM-ODP) [22] is the ISO/IEC standard for describing open distributed processing systems. The objective of this model is to create principles for the distribution of information processing services. RM-ODP does not depend on any application domain. It has five viewpoints enterprise, information, computation, engineering, and technology viewpoints. It does not specify any links between views. Instead of notations this model uses a language for describing software architecture. Even though, it was specially designed for the distributed software domain, but structures and concerns coverage shows that RM-ODP is also applicable to software architecture in general.

3.1.4. Siemens Four View Model:

Siemens Four View Model [23] is an outcome of a study of software architecture industrial practices. After the study it is observed that software structures used to document and design architecture fit into four broad categories: *conceptual, module, execution and code* structures. Each one of these four categories provide different coverage of stakeholder concerns and also when these are separately addressed, implementation complexity is decreased and re-use and reconfiguration of software is improved. To show difference of the categories, authors

recommended that architecture of software intensive systems should be documented using four views, each of which addresses different structures categories.

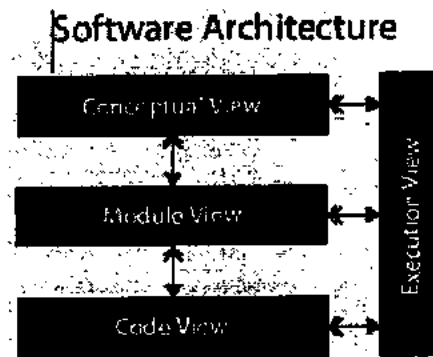


Figure 3.3. The Siemens four view model [23]

3.1.5 Rational ADS

Rational Architectural Description Specification (ADS) [25] is expansion of the “4+1” model to allow depiction of more complex architectures. In Rational ADS the “4+1” model views are partially renamed and four new views are defined. Scenarios view of “4+1” has become the Use Case view, Physical view has become the Deployment view and Development view has become the Implementation view. Nine views of Rational ADS are grouped into four viewpoints.

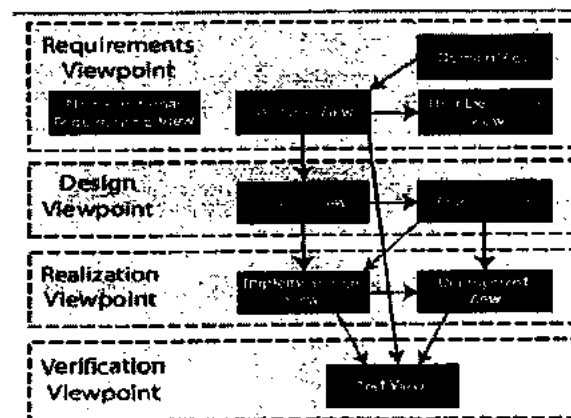


Figure 3.4. Rational ADS: Viewpoints and Views [25]

3.2. Models Evaluation

Tables 3.1, 3.2 and 3.3 show the coverage of viewpoints, stakeholders and quality attributes by the five software architecture viewpoint models. The coverage is found individually for each of the elements of comparison framework's concepts of stakeholders, quality attributes and viewpoints.

Table 3.1. Models Coverage of Viewpoints

Viewpoints	"4+1"	SEI	RM-ODP	Siemens	Rational ADS
Conceptual	Y	N	Y	Y	Y
Decomposition	Y	Y	Y	Y	Y
Uses	N	Y	N	N	N
Layered	Y	Y	Y	Y	Y
Class/Generalization	Y	Y	Y	N	N
Process	Y	Y	N	Y	Y
Concurrency	Y	Y	Y	Y	Y
Shared Data	N	Y	Y	N	N
Client-Server	Y	Y	Y	Y	N
Deployment	Y	Y	Y	N	Y
Implementation	N	Y	N	Y	Y
Work Assignment	N	Y	N	N	N

Table 3.2. Models Coverage of Stakeholders

Stakeholders	"4+1"	SEI	RM-ODP	Siemens	Rational ADS
Architects	Y	Y	N	Y	Y
Requirements Engineers	Y	N	N	Y	Y
Sub-System Architects and Designers	Y	Y	N	Y	Y
Implementers	Y	Y	Y	N	Y

Testers	Y	Y	Y	N	Y
Integrators	Y	Y	Y	Y	Y
Maintainers	N	Y	Y	N	N
External System Architects and Designers	N	Y	N	N	N
Managers	Y	Y	Y	Y	Y
Product Line Managers	Y	Y	N	N	N
Quality Assurance Team	N	N	N	N	Y
Users	Y	N	Y	N	Y
Customers	Y	N	Y	N	Y
Project Manager	N	Y	N	N	N
Production Engineers	Y	Y	Y	N	Y
Suppliers	N	Y	N	N	N
System Administrators	Y	Y	Y	N	N
Business Analyst	N	N	Y	N	Y
Product Manager	N	Y	Y	N	N
Marketer	Y	N	Y	Y	Y
Support Staff	N	Y	N	Y	Y

Table 3.3. Models Coverage of Quality Attributes

Quality Attributes	"4+1"	SEI	RM-ODP	Siemens	Rational ADS
Functionality	Y	N	Y	Y	Y
Performance	Y	Y	N	Y	N
Capacity/Space	N	Y	Y	N	Y
Availability	Y	Y	N	Y	Y
Reliability	Y	Y	N	Y	Y
Security	N	Y	Y	N	N
Safety	N	N	N	Y	N
Usability	N	N	N	N	Y
Supportability	N	Y	N	Y	Y
Configurability	N	Y	N	Y	Y
Scalability	Y	Y	N	N	Y

Modifiability	N	Y	Y	N	N
Reusability	Y	Y	N	N	Y
Testability	N	N	Y	N	Y
Portability	Y	Y	Y	Y	Y
Evolvability	Y	Y	Y	N	N
Localizability	Y	Y	N	N	N
Integrability	N	Y	N	N	N
Interoperability	Y	Y	N	Y	Y
Time to market	Y	Y	Y	Y	Y
Cost and benefit	Y	N	Y	Y	Y
Projected lifetime of the system	N	Y	Y	N	N
Targeted market	Y	N	Y	Y	Y
Rollout schedule	N	Y	Y	N	N
Integration with legacy systems	Y	N	N	N	Y

Y: provides Coverage

N: Does not provide coverage

3.3. Comparison Framework Coverage

Coverage of comparison framework elements are shown in Table 3.1, 3.2 and 3.3 respectively. Each viewpoint model provides different coverage of comparison framework elements. In case of viewpoints and quality attributes SEI provides greatest coverage. As far as stakeholders are concerned SEI and Rational ADS provide good coverage of stakeholders. Evaluation done by Nicholas [1] has many limitations. In case of stakeholders and quality attributes only those stakeholders and quality attributes are covered that are explicitly stated by viewpoint models. We

identify implicit quality attributes and stakeholders by investigating the relationship between viewpoints and quality attributes and stakeholders and quality attributes. Implicit stakeholders will be satisfied if all their concerns are addressed by viewpoints and similarly different viewpoints address different quality attributes

Table 3.4. Comparison Framework Coverage by Viewpoint Models

Models	Viewpoints	Stakeholders	Quality Attributes
4+1	8	13	14
SEI	11	15	18
RM-ODP	8	12	12
SIEMENS	7	7	12
Rational ADS	7	14	16
Total no. of Comparison Framework Elements	12	21	25

Chapter 4

Optimum Set of Viewpoints

4.1. Optimum Set of Viewpoints

When combining views from different viewpoint models the biggest obstacle is dependency between views of viewpoint models. In case of "4+1" model the views are dependent on each other i.e., being an iterative approach there is strong data flow between views. The views of the SEI and the RM-ODP model are comparatively independent. The views of Siemens model are less tightly coupled. In Rational ADS, context of lower views are provided by higher views so there is strong dependency between views.

So when combining views from different viewpoint models as we see SEI model provides good coverage of viewpoints, stakeholders and quality attributes and also its views are independent so its three views that are module, component and connector and allocation are considered for merging. As the stakeholders such as users, customers and business analysts which are not addressed by SEI can be addressed by including Use Case View from Rational ADS. As we know there is dependency between Rational ADS views as Use Case being the highest view is not dependent on any view. Also, Use Case View covers the usability concern which is not covered by SEI model. Siemens's Conceptual view is also included in optimum set as SEI model did not cover conceptual structure and its related concern functionality. Conceptual viewpoint [8] describes the system in form of system's major design elements and relationship between them this viewpoint is very important because it is strongly linked with the problem domain. So when the architect interacts with the domain expert it acts as an important means of communication and also modules can be clearly defined in module view so as a result impact of changes in requirements can be minimized.

As Rational ADS also provides good coverage of quality attributes. Rational ADS Test View is also added in optimum set of views to address testability Rational ADS's Test view addresses testability by enabling you to do test realization, preparing test cases and then forming whole test procedure also satisfying Quality Assurance Team. As we know that in Rational ADS that context of lower views are provided by higher views so we investigated and find that SEI

Allocation view type overlaps well with Rational ADS Realization viewpoint which contains Implementation and Deployment View. So context of Test View can be provided by Allocation View type of SEI model. RM-ODP views are not considered for merging because RM-ODP uses language for describing architecture and not a notation so it supports communication between different systems developers and not between other stakeholders of the same system. Figure 4.1 shows optimum set of views from different viewpoint models.

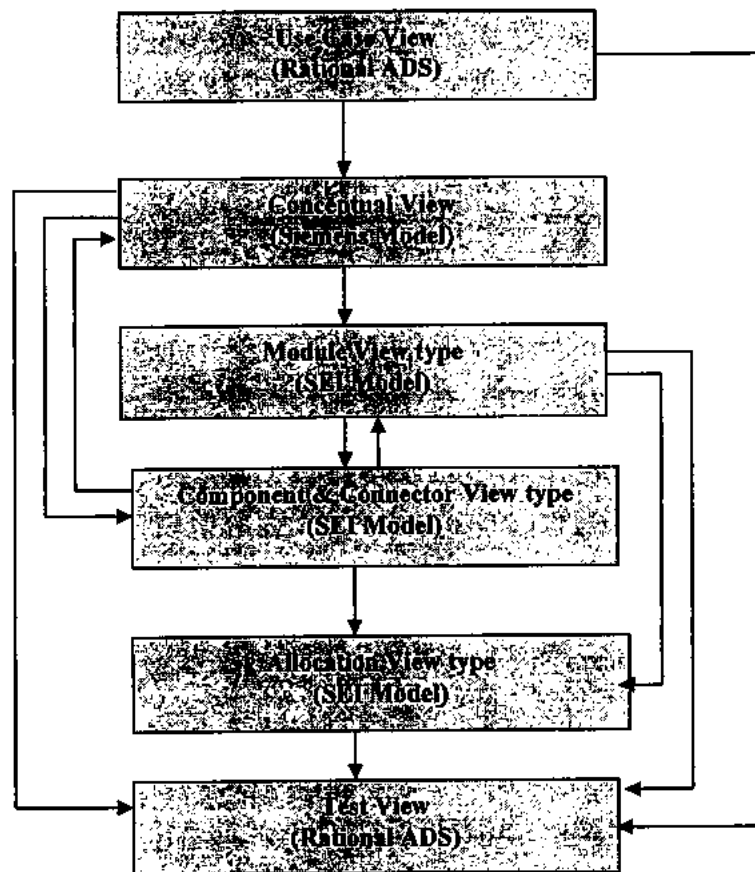


Figure 4.1. Optimum Set of Viewpoints

4.2. Mapping between Views

4.2.1. Use Case View-Conceptual View

This mapping shows how use-cases are understood by relationships between logical elements of the system.

4.2.2. Conceptual View-Module View type

This mapping shows how logical elements are realized into units of implementation. So the principal connection between the conceptual and the module views is “implemented by”. Logical elements in the conceptual view are implemented by modules in the module view type.

4.2.3. Module View type- Component & Connector View type

This mapping is dual connection between module and component & connector view type. Modules in the module view type are allocated to run time elements in component &connector view type. On the other hand, each run time elements in the component &connector view type is “implemented by” modules in the module view type.

4.2.4. Conceptual View- Component & Connector View type

This mapping is also dual connection between conceptual and component & connector viewtype. First each logical component can be related to run time element. Second which logical components will be active classes i.e, execute in their own thread of control and which components will be passive i.e., execute as part of these threads of control [25]. Also, which inter process communication is required to support the logical class interactions or constrain the implementation of logical components [25].

4.2.5. Module View type-Allocation View type

As we know allocation viewtype provides answer for questions such as: On which workstation does each software element executes? In which files or libraries is each element stored during development? And assignment of the software element to development teams? So this mapping show how each implementation component in the module view type is implemented by program units in the implementation view in allocation view type because implementation view contains directories for common library functions, include files and source code.

4.2.6 .Component & Connector View type -Allocation View type

This mapping shows how processes are assigned to run on physical nodes in the Deployment View of Allocation Viewtype.

4.2.7. Allocation View type-Test View and Module View type-Test View

Module and allocation views provide context for Test View. Allocation View type provides information in which file is each element stored and on which nodes specific tasks and processes are running during testing. Module View type provides information about implementation units' dependencies and relationships from which test cases can be generated.

4.2.8. Use Case View-Test View

There is need to use functional requirements such as a scenario from a use case to verify the important non-functional requirements such as a performance requirement. So this mapping shows how test case will need one requirement to verify another requirement [25].

4.2.9. Conceptual View-Test View

This mapping shows how test cases are mapped to associations in the design model that demonstrates any modifications required to facilitate testing [25].

4.2.10. Test View-other views

Different dimensions of testing will probably require different architectural approaches; so, the key test requirements may need to be mapped to possibly all the remaining views in order to show how the architecture is to be tested, or was changed to make it more testable [25].

4.3. Views Description

In this section there is detailed description of views taken from the relevant software architecture viewpoint models.

4.3.1. Use Case View

This view describes the important observable behavior of the system, important from architecture point of view. This behavior will influence the structure of the system. Use-Case view represents the relationships between actors and use-cases and between use-cases, where applicable and includes the description of the interactions between actors and uses cases. This view includes both static and dynamic aspect. This view includes Use-Case diagrams presenting architecturally significant use-cases, relationships between them and their external actors and also textual use-case descriptions of use case activities that show the interactions between the systems and the actors.

4.3.2. Conceptual View

Conceptual view defines the system in form of system's major design elements and the relationships between them. So conceptual view is a high-level structure of the system using design elements and relationships particular to the domain. It is independent of implementation decisions.

4.3.3. Module View type

In Views belonging to the module view type, elements are modules, which are allocated parts of functional responsibility, units of implementation and assigned to teams for implementation. Relationships between modules are is-a, is part-of and depends-on. Following styles are defined for the Module view type:

Decomposition: defines the organization of the system as modules and sub modules and demonstrates how system responsibilities are divided across them.

Generalization: to capture inheritance relationships between modules.

Uses: to capture inter module usage dependencies.

Layered: to specify how modules are arranged in layers according to their level of abstraction and each layer represents a grouping of modules that deal with interrelated set of services.

4.3.4. Component & Connector View type:

In Component and Connector view type, elements are components (principal units of computation) and connectors which facilitate communication between components. So this view

type is concerned with the system's runtime elements, their behaviors, and interactions between them. Following styles are defined for this view type and all relate to commonly occurring runtime system organizations.

Pipe-and-Filter: In this style filter alters data that it receives from one or more pipes and transfers through one or more pipes whereas pipe is a connector that carries data streams from output port of one filter to input port of another filter.

Shared-Data: Convenient for exchange of data, which has multiple accessors.

Publish-Subscribe: In this style components interact via announced events.

Client-Server: Defines system that is made up of cooperating clients and servers and connectors that are messages and protocols. It shows separation of concerns and distribution of processing elements.

Peer-to-Peer: In this style components interact with each other by exchanging services.

Communicating processes: This style describes the bundling of components into processes, which portions of the system could operate in parallel and threads of control within the system.

4.3.5. Allocation View type

Views in the allocation view type represent the relationship between software elements and the elements in one or more external environments in which the software is created and executed. Following styles are defined for this view type:

Deployment: to specify how software elements are allocated to elements of the deployment environment.

Implementation: to specify how software elements are mapped to the development environment such as their location in a codeline.

Work Assignment: to map modules to teams responsible for creating, testing, and deploying them.

4.3.6. Test View:

Test view is a representation of the Test Model [25] and purpose is to document tests requirements. Test cases can be considered to be testing requirements, so it is proposed that they must be modeled as UML use cases. These test cases will have a description and associated textual definition. Test cases will be realized by the test model, same as use cases are realized by the design model, so collaborations should be used to show this. Structural aspect of these collaborations should show any exercised interfaces and script elements used in the testing process. Class diagrams can be used showing test interfaces. Activity diagrams may be used to define high level test designs and procedures. UML interaction diagrams may be used as test scripts, showing the results of tracing the system as it runs.

4.4. Coverage of Optimum Set of Viewpoints

Tables 4.1 and 4.2 show stakeholders and quality attributes addressed by Optimum Set of Viewpoints.

Table 4.1. Stakeholders Addressed by Optimum Set of Viewpoints

Views	Stakeholders Addressed
Use Case	Users, Customers, Business Analysts
Conceptual	Architect, Implementers
Decomposition	Implementers, Maintainers, Product Line Managers, Architect, Testers

Uses	Implementers, Maintainers, Architect, Testers, Integrators, External System Architects and Designers
Generalization	Implementers, Architect, Testers, Integrators
Layered	Implementers, Architect, Testers, Integrators
Component And Connector	Implementers, Architect
Deployment	Project Manager, Testers, Integrators, Architect, System Administrator, Production Engineers
Implementation	Support Staff
Work Assignment	Project Manager
Test	Testers, Quality Assurance Team

Table 4.2. Quality Attributes Addressed by Optimum Set of Viewpoints

Views	Quality Attributes Addressed
Use Case	Usability
Conceptual	Functionality, Safety, Reliability
Decomposition	Modifiability
Uses	Modifiability, Integrability
Generalization	Evolvability, Reusability, Localizability
Layered	Portability, Modifiability, Reusability
Pipe-and-Filter	Performance
Shared-Data	Security
Client-Server	Performance, Scalability, Availability, Reusability
Peer-to-Peer	High Availability, High Scalability
Communicating Processes	Performance, Reliability

Deployment	Performance, Reliability, Availability, Security
Implementation	Configurability, Supportability
Test	Testability

4.5. Implementation of Optimum Set of Viewpoints

4.5.1. Case Study

Representation of views of optimum set is created for course registration system that support online course registration. It will allow students to register for courses and view report cards and also professors will be able to select courses to teach and record grades. Registrar will also be able to maintain information and close registration. But only course registration module is selected for modeling by optimum set of viewpoints. This module permits a student to register for courses in the current semester and student can also add, modify or delete course selections. Course Catalog System that is existing legacy system provides the course offering list for the current semester. Another legacy system i.e. Billing System supports billing of students for current semester.

4.5.2. Use Case View

Use Case view provides user centered approach for specifying system's requirements and to capture functional requirements. It is used as a communication vehicle for users, customers and business analyst. Figure 4.2 shows use case diagram of course registration system.

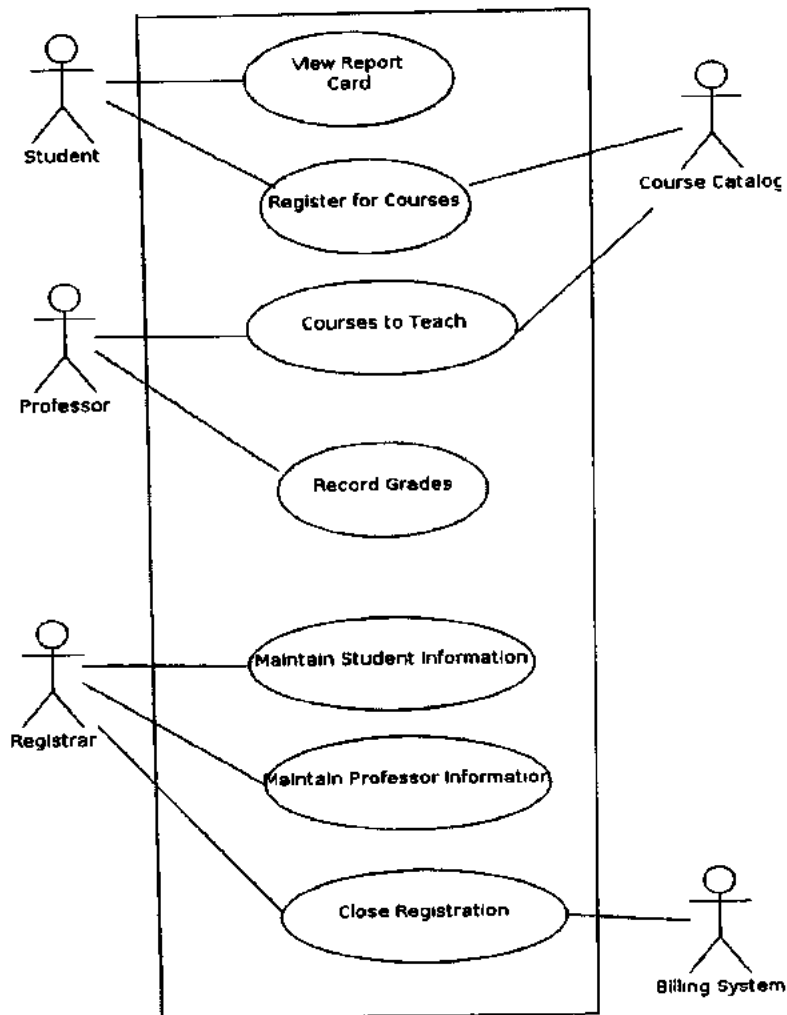
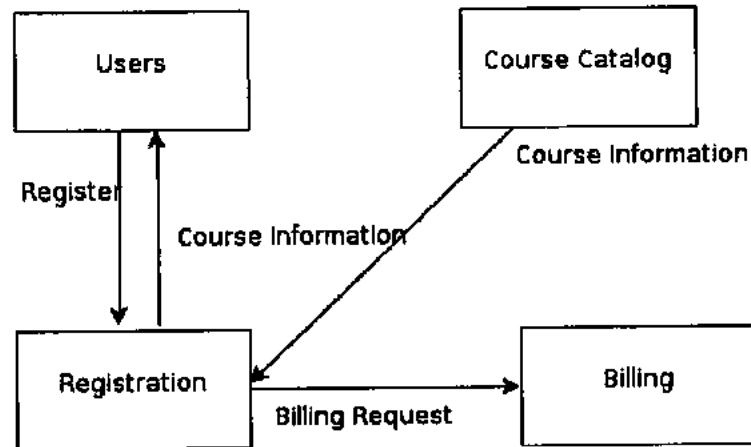


Figure 4.2. Use Case Diagram

4.5.3. Conceptual View

Conceptual View identifies conceptual components and relation between them and shows how the system fulfills functional requirements. Figure 4.3 shows conceptual view for course registration.

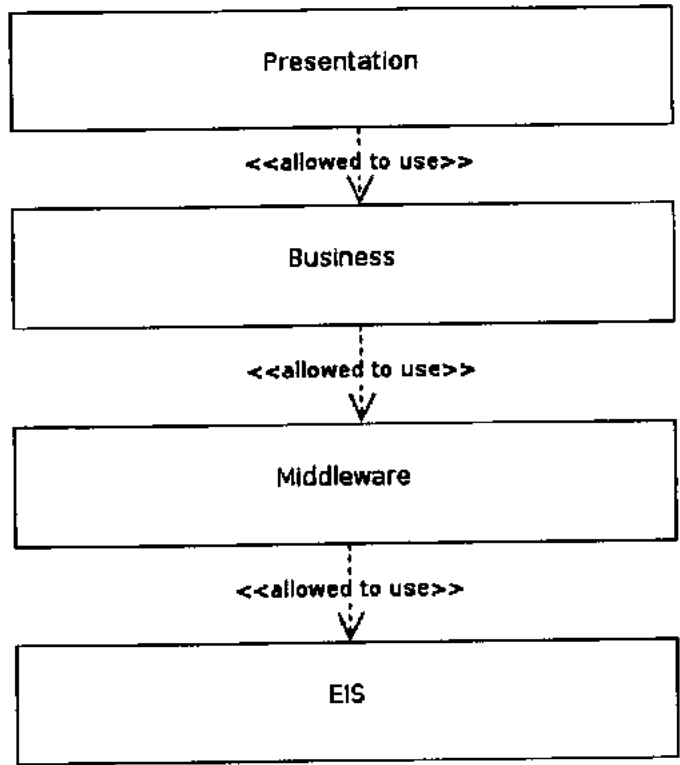


Elements	Conceptual Components and Connectors
Relations	Communication between components

Figure 4.3. Conceptual View

4.5.4. Module Viewtype

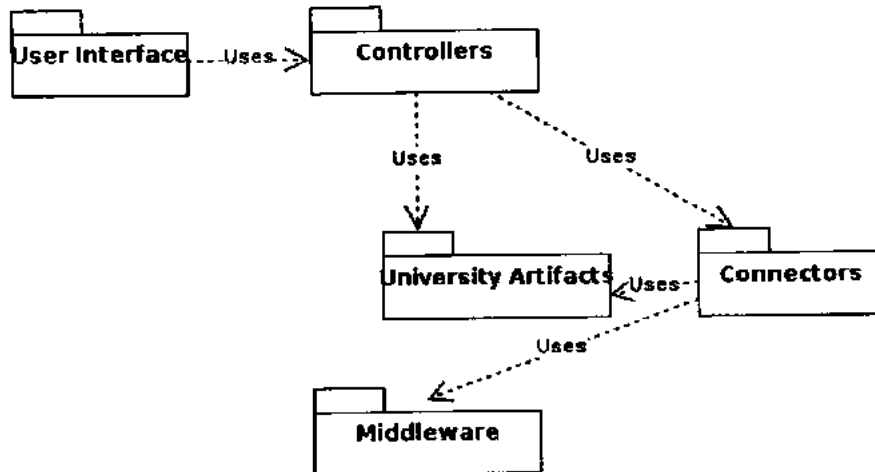
Module views help us to determine how changes in one part of a system can affect other parts and capture similar behaviors in separate modules. Different views of module viewtype such as decomposition, layered, uses and generalization provide support for various quality attributes such as modifiability, portability, reusability, evolvability, localizability etc. They also provide coverage of needs of different stakeholders such as testers, developers, product line managers, integrators etc. Figure 4.4 shows layered view for course registration.



Elements	Layers
Relations	Allowed to Use

Figure 4.4. Layered View

Figure 4.5 shows uses view for course registration.

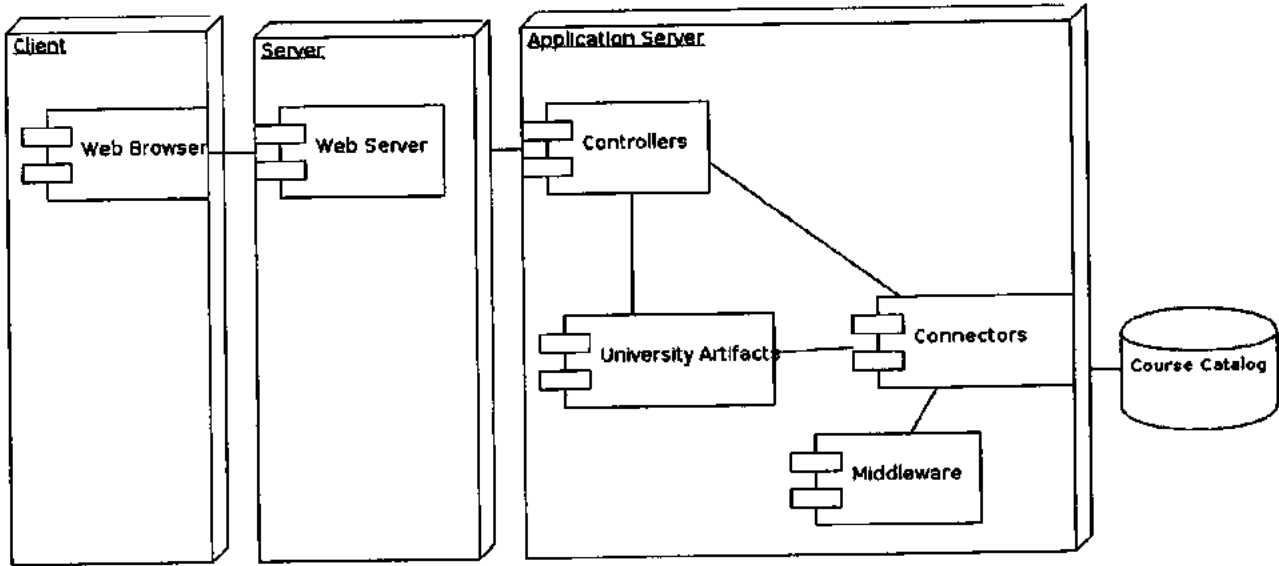


Elements	Modules
Relations	Uses: how modules depend on each other

Figure 4.5.Uses View

4.5.5. Component & Connector Viewtype

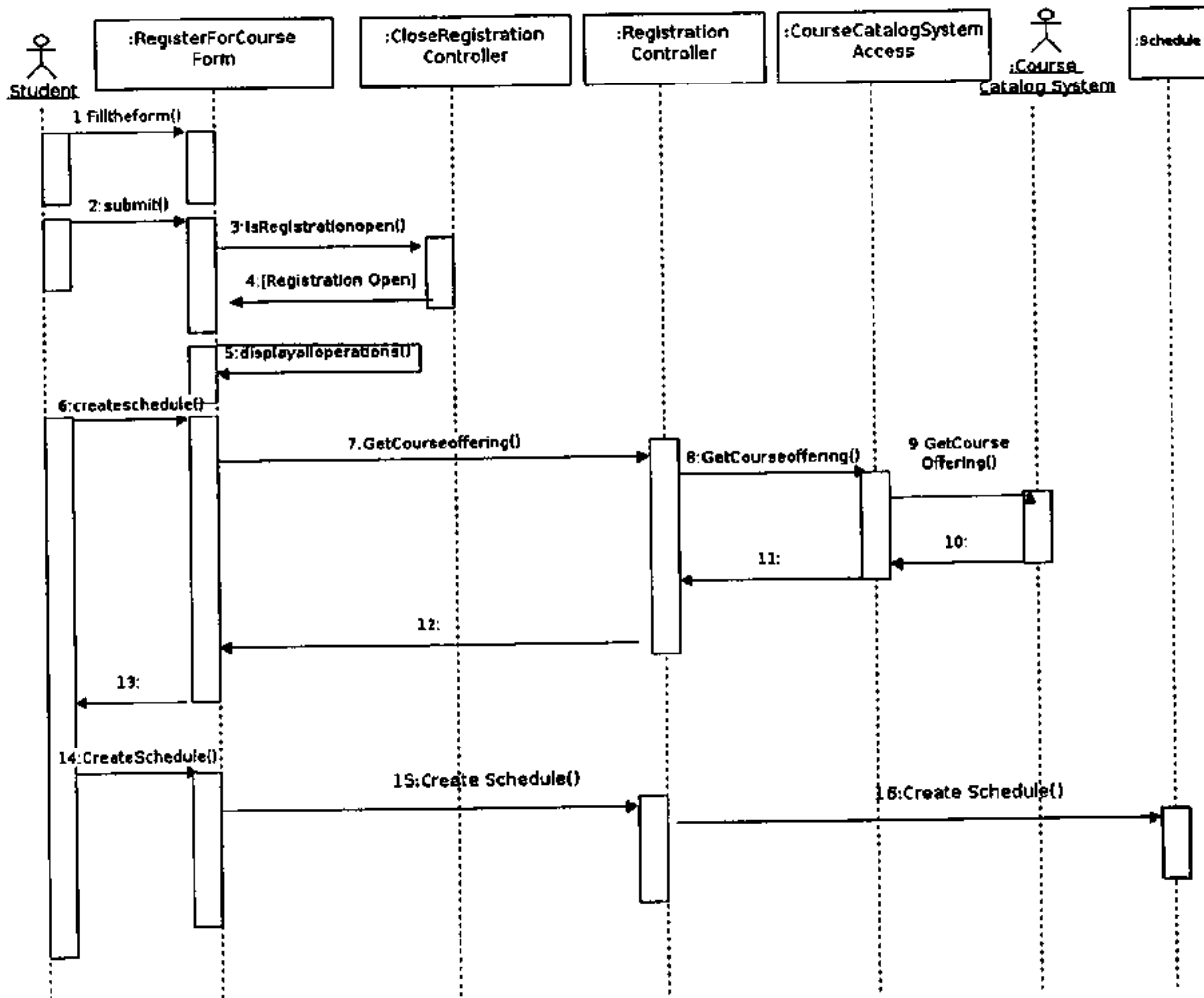
Component & Connector views show the presence of run time elements such as processes, objects, client servers, data stores etc. Different views of component and connector viewtype provide support for various quality attributes such as reliability, performance, availability, scalability, security etc. They also provide coverage of needs of different stakeholders such as architects, developers etc. Figure 4.6 shows client server view for course registration.



Elements	Clients and Servers
Relations	Request/Reply

Figure 4.6. Client Server View

Figure 4.7 shows communicating processes for course registration.



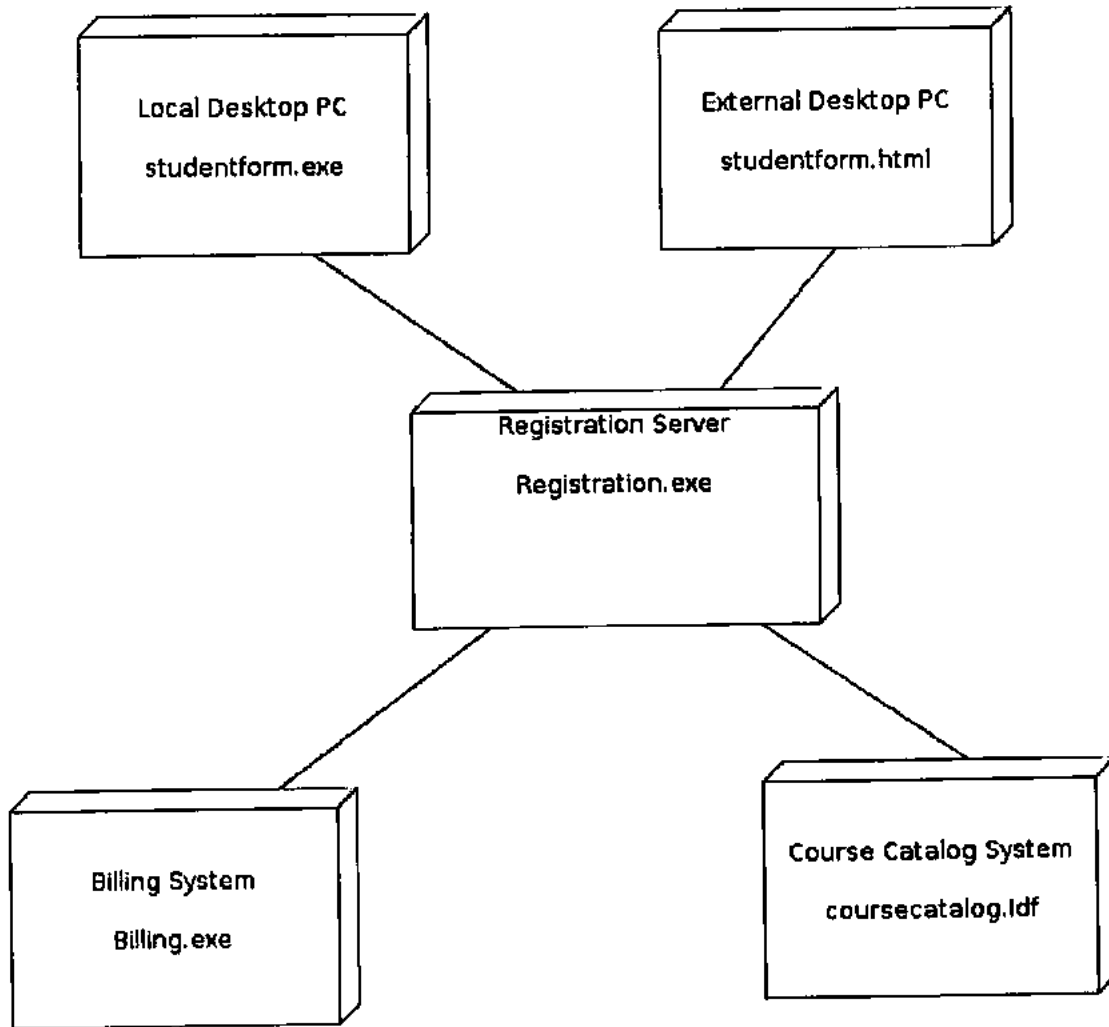
Elements	Processes or Objects
Relations	Interactions: sequences of messages

Figure 4.7. Communicating Processes

4.5.6. Allocation Viewtype

Allocation views show mapping between software elements and elements in software environment. Different views provide support for performance, reliability, security etc. They also

provide coverage of needs of different stakeholders such as testers, support staff, project managers etc. Figure 4.8 shows deployment view for course registration.



Elements	Software and Environmental Elements
Relations	Allocated to

Figure 4.8. Deployment View

4.5.7. Test View

Test View is used to verify architecture descriptions. It provides support for testability. They also provide coverage of needs of different stakeholders such as testers, quality assurance team etc. Figure 4.9 shows activity diagram for course registration which can be used to define high level test designs and procedures.

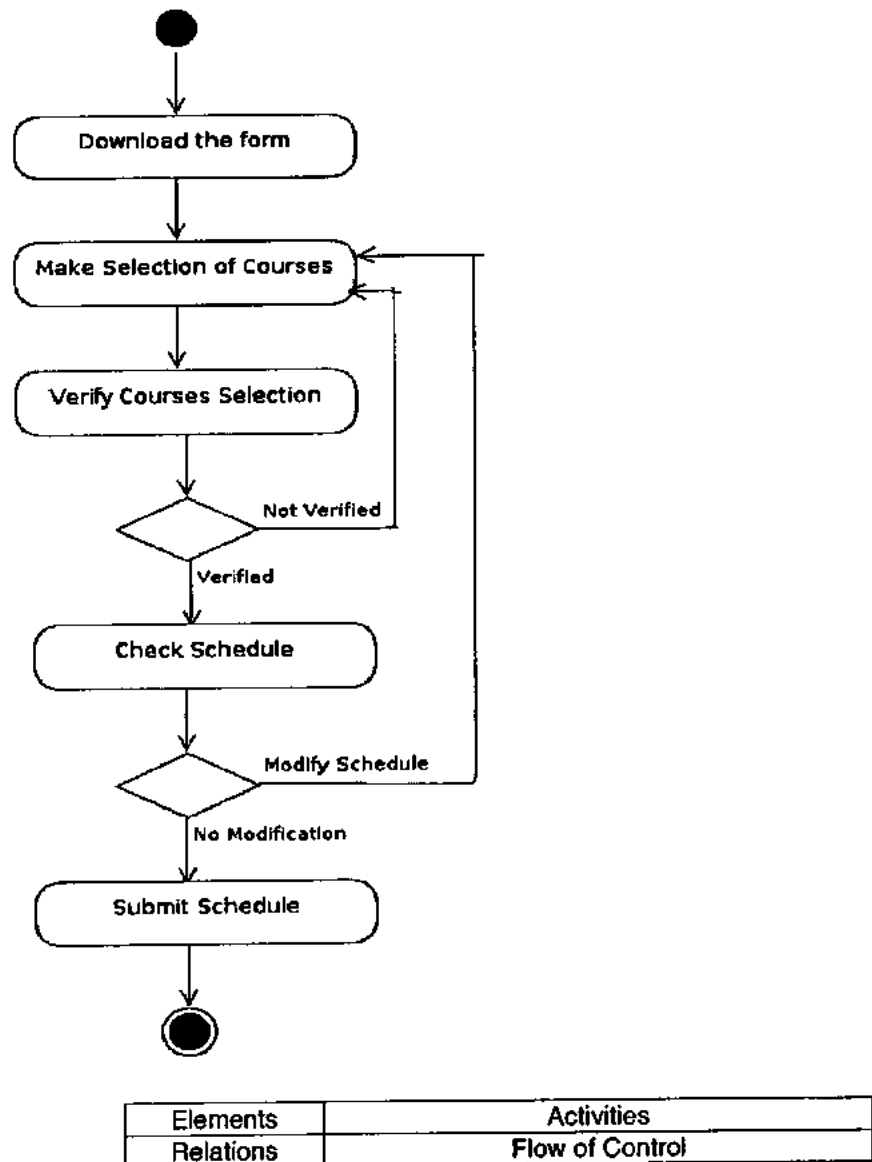


Figure 4.9. High Level Test Design and Procedures Details from Activity Diagram

Chapter 5
Validation of Optimum Set of Viewpoints
(Results & Analysis)

5.1. Research Design

In order to validate optimum set of viewpoints we conducted multiple-case study of three software intensive projects of medium to large complexity whose architectures were built using our proposed optimum set of viewpoints either by software architect or personnel who have sound knowledge of developing software architecture by using software architecture viewpoint models. We have chosen a multiple-case study approach as multiple sources of evidence provide a better validity for the findings and used purposeful sampling. We looked for projects of those software development companies that had experience in using software architecture viewpoint models and also have experienced personnel who have sound knowledge of applying views for developing architecture of applications.

5.2. Data Sources

We collected data using semi-structured scripted interviews so the questions were prepared in advance and pre-defined questionnaire were used and filled in print. We could not manage to conduct face to face interviews or interview via Skype Out calls because none of three organizations whose projects were selected agree for interview sessions due to the fact that one organization is defense related so did not allow for interview session. The other two also did not agree because of the workload and also they did not want to reveal details of their projects because of their nature.

5.3. Data Analysis

The purpose of filling the questionnaire (Annexure) was to find out optimum set of views' coverage of software architecture concepts (i.e. viewpoints, stakeholders and quality attributes) that are required to effectively design and analyze software architecture after applying it on the case projects and discuss its coverage as compared to the software architecture viewpoint model which they usually use to develop architecture of their applications. To analyze data, frequency distributions related to coverage of viewpoints, stakeholders and quality attributes by our

4.5.7. Test View

Test View is used to verify architecture descriptions. It provides support for testability. They also provide coverage of needs of different stakeholders such testers, quality assurance team etc. Figure 4.9 shows activity diagram for course registration which can be used to define high level test designs and procedures.

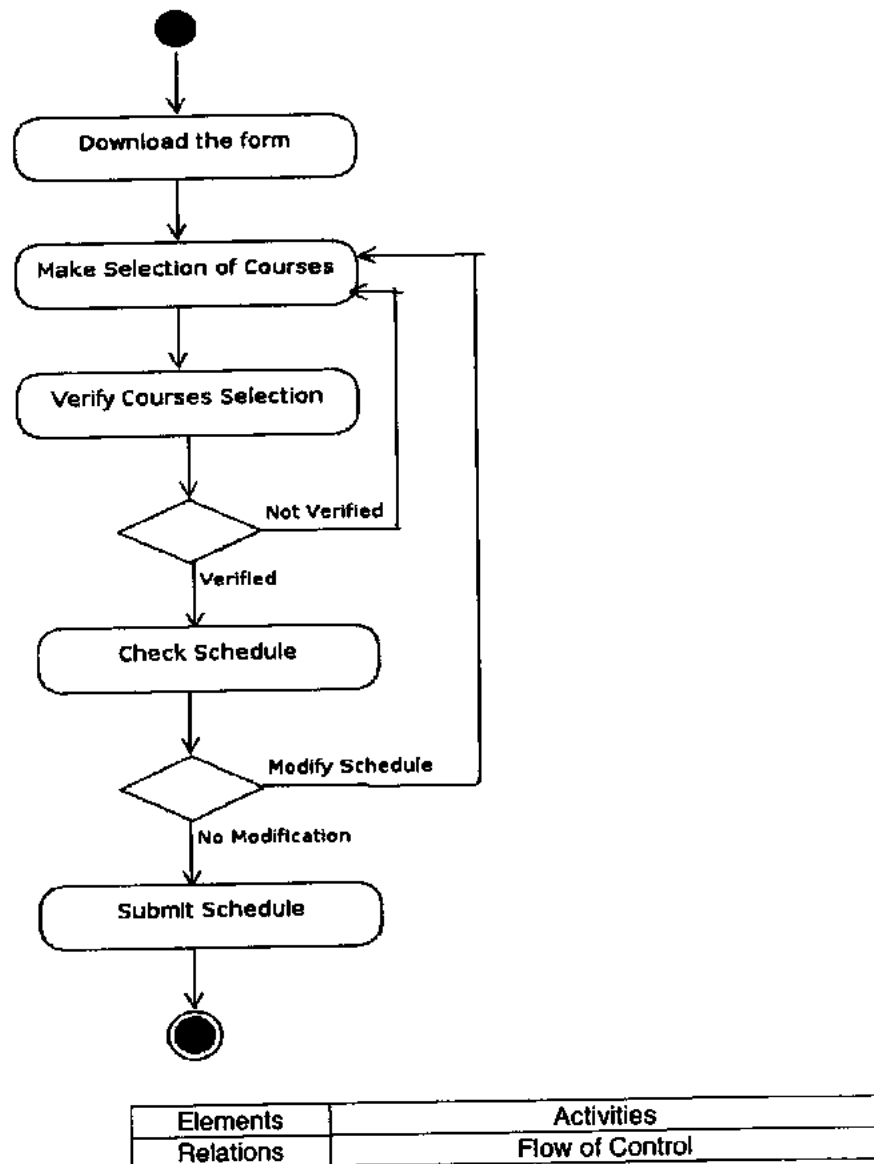


Figure 4.9.High Level Test Design and Procedures Details from Activity Diagram

Chapter 5
Validation of Optimum Set of Viewpoints
(Results & Analysis)

5.1. Research Design

In order to validate optimum set of viewpoints we conducted multiple-case study of three software intensive projects of medium to large complexity whose architectures were built using our proposed optimum set of viewpoints either by software architect or personnel who have sound knowledge of developing software architecture by using software architecture viewpoint models. We have chosen a multiple-case study approach as multiple sources of evidence provide a better validity for the findings and used purposeful sampling. We looked for projects of those software development companies that had experience in using software architecture viewpoint models and also have experienced personnel who have sound knowledge of applying views for developing architecture of applications.

5.2. Data Sources

We collected data using semi-structured scripted interviews so the questions were prepared in advance and pre-defined questionnaire were used and filled in print. We could not manage to conduct face to face interviews or interview via Skype Out calls because none of three organizations whose projects were selected agree for interview sessions due to the fact that one organization is defense related so did not allow for interview session. The other two also did not agree because of the workload and also they did not want to reveal details of their projects because of their nature.

5.3. Data Analysis

The purpose of filling the questionnaire (Annexure) was to find out optimum set of views' coverage of software architecture concepts (i.e. viewpoints, stakeholders and quality attributes) that are required to effectively design and analyze software architecture after applying it on the case projects and discuss its coverage as compared to the software architecture viewpoint model which they usually use to develop architecture of their applications. To analyze data, frequency distributions related to coverage of viewpoints, stakeholders and quality attributes by our

research outcome i.e., optimum set of viewpoints in all three cases are developed separately in the form of graphs in section 5.5. In the end chi square test is used to test null hypothesis.

5.4. Overview of Case Studies

5.4.1. Project A

Project A is software project developed by a software house (CMMI Level 3) that specializes in developing Financial, Business Management and E-government applications and project A is E-government in nature. Project A's architecture is built using optimum set of viewpoints by their software architect who has eight years experience in developing architecture of applications and after that our questionnaire is filled by him in order to find coverage of optimum set of viewpoints. Software Architect has used all views of optimum set to develop application's architecture due to project's complexity.

5.4.1.1. Findings

After analyzing data of questionnaire we found out that according to architect's views and analysis of questionnaire (Annexure) optimum set of viewpoints provide more coverage with respect to viewpoints and stakeholders' concerns as compared to the viewpoint model (i.e. Rational ADS with customization) which they usually follow for developing architecture because it ignores the internal structures of the application and hence the performance and reliability behaviors are not explicitly and individually captured so these types of problems are sufficiently covered by optimum set of views. In case of quality attributes optimum set of views provides all applicable attributes. Suggestion given by Architect is that optimum set should define how things in one view are connected and complimented in the next view such as how uses cases are linked to class and sequence diagrams and how they are connected to test cases so an overall detailed inter connectivity would needed to be defined.

5.4.2. Project B

Project B is software project developed by a software house (CMMI Level 2) that specializes in Data Management (Data warehouse, Business Intelligence, Data Mining, Document Management Application Dev., Document Management Services) in Telecom and Banking Domains and they did not give much details of project. Project B's architecture is built using optimum set of viewpoints by their Project Manager who has five years plus experience in developing architecture of applications and after that our questionnaire is filled by him in order to find coverage of optimum set of viewpoints. Project Manager has used all views of optimum set to develop application's architecture due to project's complexity.

5.4.2.1. Findings

After Analyzing data of questionnaire we found out that according to architect's views and analysis of questionnaire optimum set of viewpoints provide more coverage of business needs and maximum completeness of software architecture aspects i.e. viewpoints, stakeholders and quality attributes by customizing already available software architecture solutions. Being the SEI / CMMI certified firm they usually follow SEI's views with customization to work for implementation of data warehouse and business intelligence projects.

5.4.3. Project C

Project C is software project developed by a software house that specializes in managing the entire office automation system and providing IT support to defense organizations and project C is web based document management and filing system. Project C's architecture is built using optimum set of viewpoints by their project manager who has four years' experience in developing architecture of applications and after that our questionnaire is filled by her in order to find coverage of optimum set of viewpoints. Software Architect has used all views of optimum set except Component & Connector View type to develop application's architecture.

5.4.3.1 Findings

After Analyzing data of questionnaire we found out that according to architect's views and analysis of questionnaire (Annexure) optimum set of viewpoints provide more coverage with respect to viewpoints and quality attributes as compared to the software architecture processes or models (i.e. RUP and Rational ADS with customization) which they usually follow for developing architecture. In case of quality attributes optimum set of views provides high availability as compared to approach followed by them. Suggestion given by project manager is use case viewpoint should be added in list of viewpoints.

Table 5.1.Overview of Case Projects

Project Name	Project Type	Project Complexity	Number of employees in developing organizations	CMMI Level of Organization	Data Collection	Participants Of Case Studies	Experience Of Participants in developing Architecture
A	E-government	Medium	350+	3	Questionnaire	Software Architect	8 Years
B	Data warehouse and business intelligence	Medium	25+	2	Questionnaire	Project Manager	5+ Years
C	Web based document management and filing system	Large	80	Nil	Questionnaire	Project Manager	4 Years

5.5 Case Studies Results

5.5.1. Coverage of Viewpoints

Figure 5.1 shows coverage of software architecture viewpoints by optimum set of viewpoints after applying it on case projects. Out of 12 viewpoints optimum set of viewpoints provides

100% coverage i.e. 12 viewpoints in first case study, 92% coverage i.e. 11 viewpoints in second case study and 83% coverage i.e. 10 viewpoints in third case study.

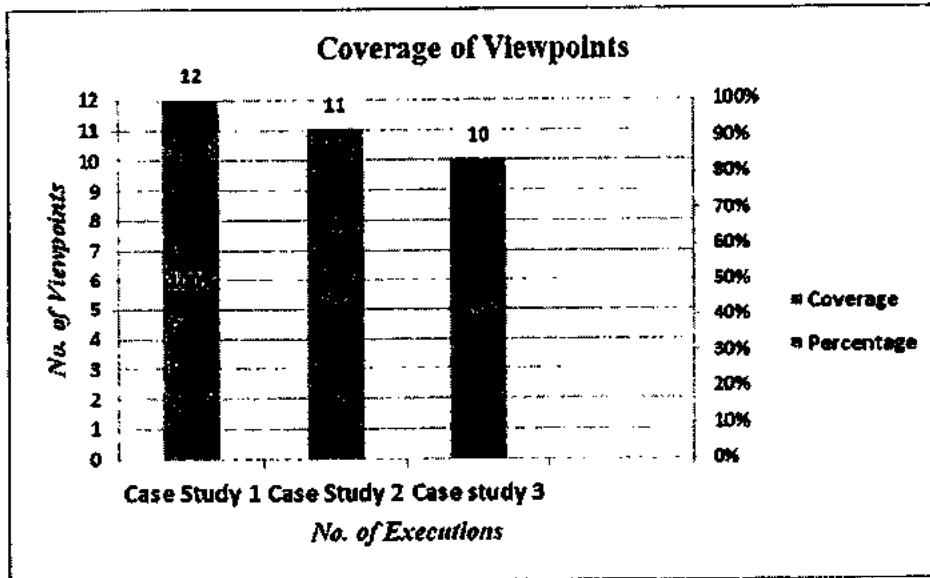


Figure 5.1. Coverage of Viewpoints by Optimum set of Viewpoints

From analysis of questionnaire results it is shown that viewpoints such as shared data, uses, generalization, implementation and work assignment which are not covered by most of models, are covered in detail by optimum set of viewpoints.

5.5.2. Coverage of Stakeholders

Figure 5.2 shows coverage of software architecture stakeholders by optimum set of viewpoints after applying it on case projects. Out of 21 stakeholders optimum set of viewpoints provides 100% coverage i.e. 21 stakeholders in first case study, 100% coverage i.e. 21 stakeholders in second case study and 76% coverage i.e. 16 stakeholders in third case study.

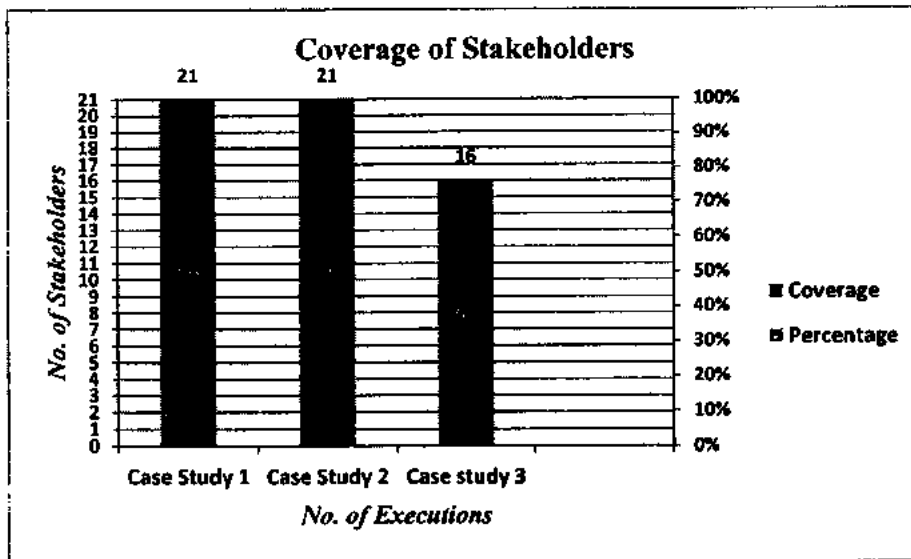


Figure 5.2. Coverage of Stakeholders by Optimum set of Viewpoints

From analysis of questionnaire results it is shown that stakeholders such as External System Architects and Designers, Product line Managers, Quality Assurance Team, Suppliers, Support Staff and Project Managers which are not covered by most of models are covered in detail by optimum set of viewpoints

5.5.3. Coverage of Quality Attributes

Figure 5.3 shows coverage of software architecture quality attributes by optimum set of viewpoints after applying it on case projects. Out of 25 quality attributes optimum set of viewpoints provides 100% coverage i.e. 25 quality attributes in first case study, 80% coverage i.e. 20 quality attributes in second case study and 96% coverage i.e. 24 quality attributes in third case study.

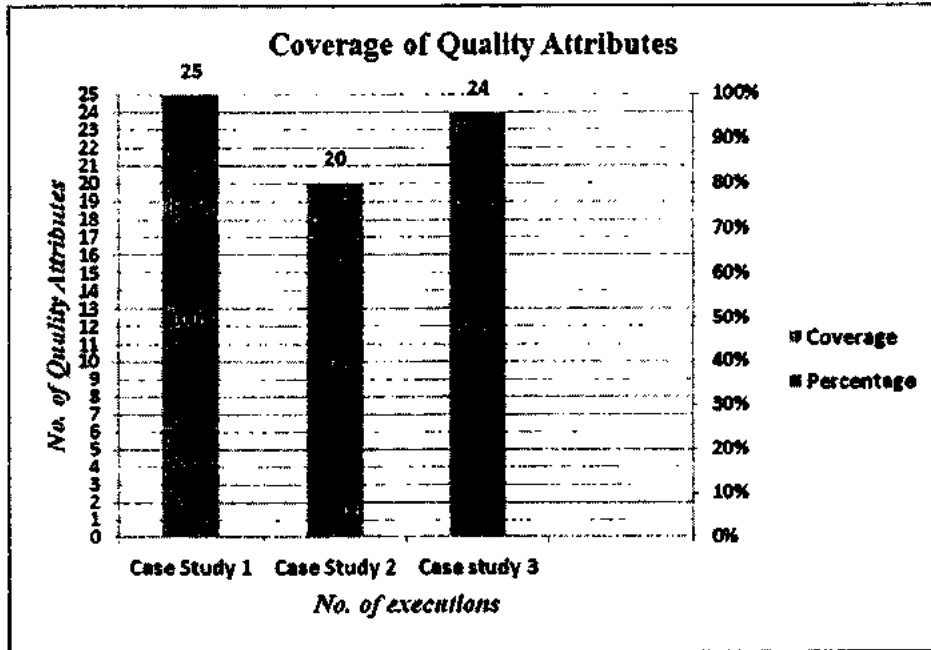


Figure 5.3. Coverage of Quality Attributes by Optimum set of Viewpoints

From analysis of questionnaire results it is shown that quality attributes such as security, modifiability, integrability, safety, supportability, projected lifetime of the system and testability which are not covered by most of models are covered in detail by optimum set of viewpoints.

5.6. Discussion

Mean coverage of concepts that are needed to effectively design and analyze software architecture i.e. viewpoints, stakeholders and quality attributes is calculated for optimum set of viewpoints and compared to coverage of viewpoints by five software architecture viewpoint models evaluated in chapter 3 and it is shown that optimum set of viewpoints provide more coverage of concepts than surveyed individual models.

Figure 5.4 shows comparison between optimum set of viewpoints and surveyed individual models with respect to coverage of viewpoints. Optimum set of viewpoints provide more coverage as compared to individual models. SEI coverage and optimum set of viewpoints

coverage is same in case of viewpoints because our comparison framework is based on IEEE 1471 Standard i.e., Recommended Practice for Architectural Description of Software-Intensive Systems and SEI model provides template for more than one representation to describe contents of view in order to comply with the IEEE 1471 and can cover all details.

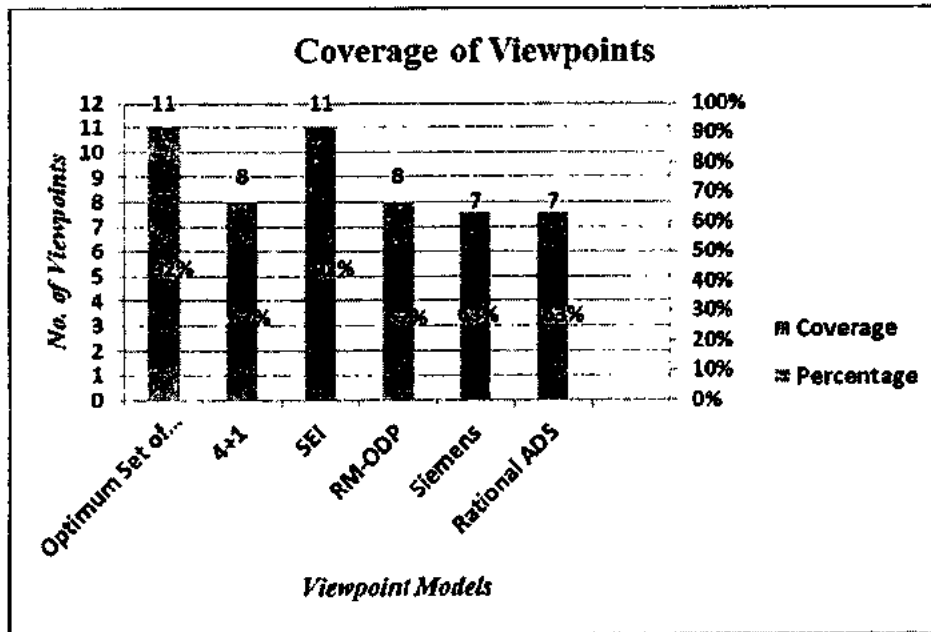


Figure 5.4. Comparison of Coverage of Viewpoints by Optimum set of Viewpoints with individual viewpoint models' coverage

Fig 5.5 shows comparison between optimum set of viewpoints and surveyed individual models with respect to coverage of stakeholders. Optimum set of viewpoints provide more coverage as compared to individual models.

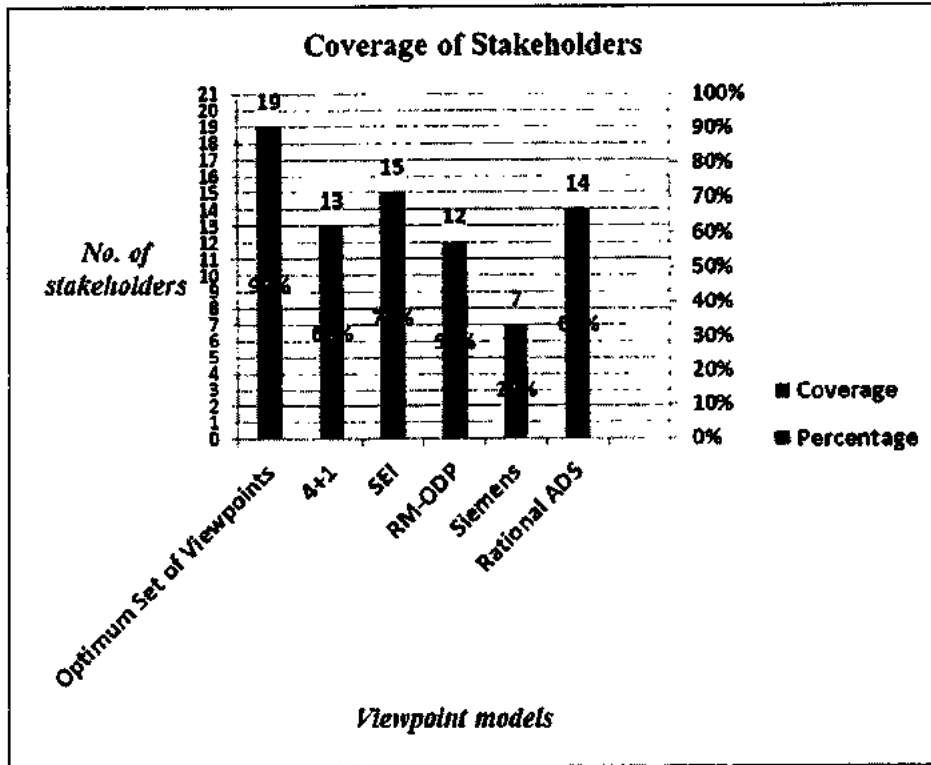


Figure 5.5. Comparison of Coverage of Stakeholders by Optimum set of Viewpoints with individual viewpoint models' coverage

Fig 5.6 shows comparison between optimum set of viewpoints and surveyed individual models with respect to coverage of quality attributes. Optimum set of viewpoints provide more coverage as compared to individual models.

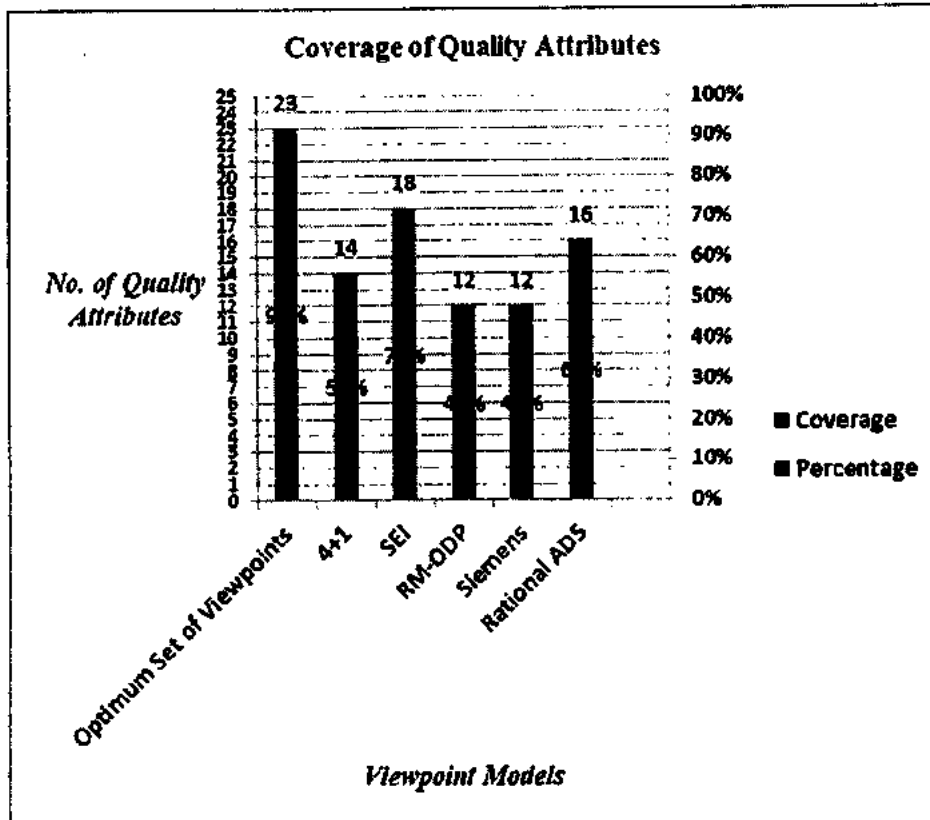


Figure 5.6. Comparison of Coverage of Quality Attributes by Optimum set of Viewpoints with individual viewpoint models' coverage

Also from case studies, it is concluded that Optimum set of views provide more coverage with respect to viewpoints, stakeholders and quality attributes of software architecture domain, than what can be achieved via individual architecture model alone.

5.7. Limitations:

Due to resource limitations and confidentiality issues, we were not able to triangulate our findings by software architectural documentation analysis and face to face interviews which can provide in depth analysis. Furthermore, close ended questions in questionnaire (Annexure) has Yes\No\Partial\Not Applicable options so while analyzing questionnaire results we assign same

scale to partial option as Yes option regarding coverage of Software architecture concepts because we have to compare coverage of optimum set of viewpoints with coverage of surveyed viewpoint models whose coverage were determined by review of literature not by software architectural documentation analysis and from review of literature partial coverage cannot be find out.

Chapter 6

Conclusion & Future Work

6.1. Conclusion:

There are a number of viewpoint models that create architecture document by means of the separation of the concerns. Each one of them describes viewpoints set and recognizes the concerns that each of them address. But none of them provide complete coverage of software architecture domain. So a set of optimum viewpoints is selected from different software architecture viewpoint models after comparing them on a common comparison framework that allows combining views from different viewpoint models.

Then we present a Multiple-case study on the application of optimum set of viewpoints to three software development projects. From the results of case studies it is concluded that Optimum set of views provide more coverage with respect to viewpoints, stakeholders and quality attributes of software architecture domain, than what can be achieved via individual architecture model alone.

6.2. Future Work

In the future, this work can be augmented by additional case projects and data can be collected and analyzed from several sources i.e., architectural documentation and face to face interviews to get a more complete understanding of coverage of software architecture concepts.

Furthermore, by modeling system from architectural documentation with five surveyed models we can get a clearer picture of their coverage of software architecture concepts and also their partial coverage of concepts can be found, which cannot be found via literature.

References:

- [1] Nicholas May, "A survey of software architecture viewpoint models", Proc. of the Sixth Australasian Workshop on Software and System Architectures, Melbourne, Australia, pp. 13-24, 2005.
- [2] Nick Rozanski, Eoin woods, "Applying Viewpoints and Views to Software Architecture", White Paper, <http://www.viewpoints-and-perspectives.info>, 2005.
- [3] K. Smolander, "What is included in software architecture? A case study in three software organizations", In Ninth Annual IEEE International Conference and Workshop on the Engineering of Computer-Based Systems, pages 131–138, Lund, Sweden, April 2002.
- [4] Eoin Woods, "Experiences using viewpoints for information systems architecture: an industrial experience report", In Proceedings of Software Architecture, First European Workshop EWSA, 182-193 (2004).
- [5] Antony Tang, Jun Han, Pin Chen, "A Comparative Analysis of Architecture Frameworks", In Proceedings of the 11th Asia-Pacific Software Engineering Conference Pages 640-647, 2004.
- [6] Paul Clements, "Comparing the SEI's Views and Beyond Approach for Documenting Software Architectures with ANSI-IEEE 1471-2000", (CMU/SEI-2005-TN-017). Software Engineering Institute, Carnegie Mellon University, 2005.
- [7] Muhammad Asad Javed, "A Rationale Focused Software Architecture Documentation method (RFSAD)", Technical Report no 2007: 115, ISSN: 1651-4769/2007, Department of Applied Information Technology, IT University of Göteborg Sweden.
- [8] Len Bass, Rick Kazman, "Architecture-Based Development", Carnegie Mellon University, Technical Report CMU/SEI-99-TR-007, ESC-TR-99-007, 1999.

[9] M. Shaw and D. Garlan, "Software Architecture: Perspectives on an Emerging Discipline", Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1st edition, 1995. ISBN 0-13-182957-2.

[10] Smolander, K. and T. Päivärinta , "Describing and Communicating Software Architecture in Practice: Observations on Stakeholders and Rationale", Proceedings of The Fourteenth International Conference on Advanced Information Systems Engineering, Toronto, Canada, 2002.

[11] Ziv Baida, "Stakeholders and Their Concerns In Software Architectures", October 2001.

[12] Len Bass, Paul Clements, Rick Kazman, Software Architecture in Practice. Addison Wesley, Boston, MA, USA, 2nd edition, 2003. ISBN 0-321-15495-9.

[13] Samuel Fricker, Tony Gorschek, and Petri Myllyperkiö, "Handshaking Between Software Projects and Stakeholders Using Implementation Proposals", REFSQ 2007, LNCS 4542, pp. 144 – 159, 2007.

[14] Wing H. Huen, "Systems Engineering of Complex Software Systems", In proceedings of 37th ASEE/IEEE Frontiers in Education Conference, October 2007.

[15] Jai Asundi, Rick Kazman and Mark Klein, "An Architectural Approach to Software Cost Modeling", SEI Interactive, March 2000.

[16] Rozanski, Nick and Eoin Woods. Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives. Boston, MA: Addison-Wesley, 2005.

[17] Ruth Malan and Dana Bredemeyer, "Defining non-functional requirements", Bredemeyer Consulting, White Paper. <http://www.bredemeyer.com/papers.htm>, 2001.

[18] "Quality Characteristics for Software Architecture", JOURNAL OF OBJECT TECHNOLOGY, Vol. 2, No. 2, 2003.

[19] IEEE. IEEE Recommended Practice for Architectural Description of Software-Intensive Systems. Institute of Electrical and Electronics Engineers, Sept. 2000. IEEE Std 1471-2000.

[20] P. Kruchten. "Architectural Blueprints - The "4+1" View Model of Software Architecture". IEEE Software, 12(6):42-50, 1995.

[21] P. Clements et al. "A practical method for documenting software architectures", <http://www-2.cs.cmu.edu/afs/cs/project/able/ftp/icse03-dsa/submitted.pdf>.

[22] ISO. Reference Model of Open Distributed Processing (RMODP). International Organization for Standardization, 1994. Technical Report 10746.

[23] Dilip Soni, Robert L. Nerd, and Christine Hofmeister, "Software architecture in industrial Applications". In Proceedings of International Conference on Software Engineering, pages 196-207, 1995.

[24] P. Clements et al, Documenting Software Architecture: Views and Beyond. Addison Wesley, Boston, MA, USA, 1st edition, 2002. ISBN 0-201-70372-6.

[25] D. Norris, "Communicating Complex Architectures with UML and the Rational ADS", In Proceedings of the IBM Rational Software Development User Conference, 2004.

[26] Danny Greefhorst, Henk Koning and Hans van Vliet, "The Many Faces of Architectural Descriptions", Information Systems Frontiers, Vol 8, Issue 2, pp 103-113, 2006.

[27] S. Roselin Mary and Paul Rodrigues, "Software Architecture- Evolution and Evaluation", (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 3, No.8, 2012

“Architecture Coverage: Validating Optimum Set of View Points”

Interview Questions

Name: _____ Qualification: _____

Role: _____

No. of years in developing software architecture: _____

Level of your organization (i.e. CMM Level): _____

Number of employees in your organization: _____

Type of applications developed in your organization: _____

This questionnaire is to validate the coverage of the optimum set of views of quality attributes, stakeholders and viewpoints.

Note: Select the Not Applicable option if the viewpoint, concern or stakeholder was not modeled.

Q: 1 Which of the following viewpoints are supported by optimum set of views?

Viewpoints	Optimum Set of Views			
	Yes	Partial	No	Not Applicable
Conceptual				
Decomposition				
Uses				
Layered				

Class/Generalization				
Process				
Concurrency				
Shared Data				
Client-Server				
Deployment				
Implementation				
Work Assignment				

Any other (if any please specify) _____

Q: 2 Which of the following stakeholders' concerns are satisfied by optimum set of views?

Stakeholders	Optimum Set of Views			
	Yes	Partial	No	Not Applicable
Architects				
Requirements Engineers				
Sub-System Architects and Designers				
Developers				
Testers				
Integrators				
Maintainers				
External System Architects and Designers				

Managers				
Product Line Managers				
Quality Assurance Team				
Users				
Customers				
Project Manager				
Production Engineers				
Suppliers				
System Administrators				
Business Analyst				
Product Managers				
Marketers				
Support Staff				

Any other (if any please specify) _____

Q: 3 Which of the following quality attributes are addressed by optimum set of views?

Quality Attributes	Optimum Set of Views			
	Yes	Partial	No	Not Applicable
Functionality				
Performance				
Capacity/Space				
Availability				
Reliability				
Security				
Safety				

Usability				
Supportability				
Configurability				
Scalability				
Interoperability				
Modifiability				
Reusability				
Testability				
Portability				
Evolvability				
Localizability				
Integrability				
Time to market				
Cost and benefit				
Projected lifetime of the system				
Targeted market				
Rollout schedule				
Integration with legacy systems				

Any other (if any please specify) _____

Q: 4 Give some information about your project (i.e. its type, duration and complexity) on which you apply the optimum set of views?

Q: 5 Please summarize the application of optimum set of views for designing architecture of your project.

Q:6 Are the terms used in the list of viewpoints, stakeholders and quality attributes familiar to you and same terms are used in industry or you use different names for them (if yes, what are they)?

Q: 7 Which model(s) you usually use for developing architecture of your projects?

Q: 8 Compare coverage of architectural approach usually follow with the optimum set of views of stakeholders, viewpoints and quality attributes (which one provide more coverage and how)?

Q: 9 Are the models i.e. SEI, Siemens and Rational ADS familiar to you and do you apply them in your projects and which one of them is commonly used and why?

Q: 10 Comment on our research outcome i.e. optimum set of views and give your suggestions?
