

# A Survey of Software Architecture Viewpoint Models\*

Nicholas May  
*nick\_may@netlink.com.au*

## Abstract

*The documentation of software architecture is carried out in many different ways. One method is to break up the description into separate perspectives that address the different concerns that stakeholders have with software architecture. These perspectives, sometimes called viewpoints, can contain multiple diagrams to describe the complete system. Various models have been proposed that detail viewpoints and specify the stakeholders and concerns that they will satisfy.*

*In this paper we survey five viewpoint models to determine the extent to which they cover the software architecture domain. We attempt to identify a set of viewpoints from different models can be combined to provide the widest possible coverage.*

*We found that no model has complete coverage, but an optimal set of viewpoints can be selected from the models. This optimal set, whilst not providing complete coverage, has a greater coverage than any of the individual viewpoint models.*

## 1 Introduction

In this section we provide an overview of the area of investigation, the motivation for this work and the results obtained.

### 1.1 What is Software Architecture and how is it documented?

Software architecture is the high level structure of a system. A simple definition of software architecture was provided by Shaw and Garlan [13, p3]:

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

---

\*This paper is an expansion of a dissertation of the same title submitted by the author in partial fulfillment of the requirements for the degree of Master of Technology (Information Technology) at RMIT University (Melbourne, Australia) in 2004.

They went on to show that designing software architectures includes determining which patterns of components and interactions to use and what constraints are placed on those patterns.

As well as the design process, an important aspect of software architecture is the production of the relevant documentation. A recent summary of the need for documenting software architecture is provided by Bass et al. [2]. They identified three motivating factors which were, to enable communication of the software architecture among stakeholders, to capture early design decisions, and to provide re-useable abstractions of software systems [2, p26].

It is a common feature of graphical documentation to use multiple, concurrent diagrams to describe the entire software architecture of a system. This overcomes the problems of crowded diagrams, inconsistent notation, mixing of architectural styles, over emphasis of one aspect and the overlooking of individual stakeholder concerns [10, p42]. However, some basis of organization of the diagrams is required. Many informal approaches are used to document software architecture, including boxes and lines and simple class diagrams [6]. In addition, at least half a dozen more formal methods for documenting software architecture have been proposed. These range from notations to systems of diagram classification. A subset of these systems includes those that seek to separate the diagrams based on the needs of the stakeholders. The separation of stakeholder concerns, or requirements, leads to a grouping of diagrams by the perspective of the software architecture that they show. These groups are variously called perspectives, views and viewtypes, but here we will refer to a perspective as a viewpoint. The classification systems that utilize these perspectives will be called viewpoint models in this paper.

### 1.2 What is the purpose of this paper?

Various models have been proposed of how to create documentation by the separation of the concerns. Each model describes a set of viewpoints and identifies the concerns that each of them address. The different models cover the same software architecture domain, including the associated organizational, business and technological environments. However, they are not compatible because each

model assigns a different significance to each of the environments [14], and the vocabulary used to describe the stakeholders and concerns varies between models.

The primary purpose of this paper is to gain an understanding of the different viewpoint models, their comparative strengths, and their relative coverage of the software architecture domain. Due to the limited scope, it is not the intention to provide an authoritative comparison framework, but to base the comparison on a standard framework and a relatively independent set of well documented elements.

In addition to a comparison of models, the classification of viewpoints within a common framework would allow to combine the viewpoints from different models and determine an optimum set of viewpoints with the greatest coverage of the software architecture domain. This could form the basis of future documentation techniques.

In this paper we survey five viewpoint models based on such a framework to determine whether we can combine viewpoints from different models to create an optimum set of with the fewest viewpoints.

The models selected for this survey are as follows:

- Kruchten's "4+1" View Model [10].
- Software Engineering Institute (SEI) set of views [5].
- ISO Reference Model of Open Distributed Processing (RM-ODP) [9].
- Siemens Four View model [16].
- Rational Architecture Description Specification (ADS) [11].

All these models focus on describing software architecture from multiple perspectives. They each specify target stakeholders and recognize the separation of concerns. In addition, all these models concentrate on describing the structures of software architecture and not on defining specific notations for each of these structures.

To be able to compare the methods of documentation, a common framework is required. This framework should assess the models for the depth of description they provide, the communication needs they satisfy, and the concerns they address. The comparison framework is based on the IEEE Standard 1471-2000 (IEEE 1471), called the IEEE Recommended Practice for Architectural Description of Software-Intensive Systems [8]. This defines the concepts and their relationships for methods of documenting software architecture. The important part of the standard, for this survey, is the relationship that viewpoints have with other elements in the framework, specifically stakeholders, concerns and models. The stakeholders and concerns relating to a particular viewpoint are usually explicitly stated, but the models are not always defined. However, models

can be differentiated by the software architecture structures that they can show. The viewpoints will be assessed for their coverage of the structures of software architecture, instead of the specific models that they can use.

To analyze the viewpoint models, we compare the coverage of each viewpoint against a reference framework, consisting of lists of elements, including stakeholders, concerns and architectural structures. The elements identified are translated into the elements, from the reference list with which they equate. The framework comprises of an initial arbitrary list for each type of element, selected from a relatively comprehensive source, and additional elements as identified in the viewpoints.

### 1.3 What were the findings?

Using the comparison framework, we found that there was considerable overlap between the viewpoint models. Then, an optimum set of viewpoints was selected that provided the greatest combined coverage. This primarily consisted of the three viewpoints from the SEI model. An additional viewpoint, from the Rational ADS, was included to complement the SEI model with viewpoints focusing on the end user and standard writers.

We found that the combined coverage of the surveyed models did not cover the entire software architecture domain and, therefore, no set could be selected with complete coverage.

## 2 Background

In this section we will answer several questions. What are the structures of software architecture? What is an architectural description? Who are the system stakeholders? What are the stakeholder concerns? The answers to these questions will provide a foundation on which to build a classification of software architecture documentation techniques.

### 2.1 What are the structures of software architecture?

Another definition of software architecture is provided by Bass et al. [2, p21]:

“The software architecture of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them.”

This definition is similar to one proposed by Perry and Wolf [12], as both define software architecture in terms of

elements, their properties and relationships. However, they suggest that the software architecture description is a consequence of early design decisions, and so the rationale is external to and not a part of the architecture itself.

The components and connectors described in the structures provide us with a classification of the basic concepts of software architecture to be documented.

In addition to the structures of software architecture, systems often share common patterns of organization, called architectural styles [13, p20]. Some of the most common that have been identified and described are the client-server, pipe and filter, object-oriented and layered styles. Each style can be defined by the type of components and connectors of which it is comprised and the constraints on how these elements can be combined. This means that a style can be described in terms of the structures that it utilizes. For instance, the object-oriented style is based on decomposition, uses and abstraction. One advantage of architectural styles is that they have known quality attributes. This allows an architect to select a style based on the system requirements, in instead of inventing an architecture from scratch [2, p25].

A list of the software architecture structures found in software systems [2, p39-40] can be found in Table 1. The architectural structures represented in this list can be identified from their associated elements and relationships.

## 2.2 What is an architectural description?

The IEEE has defined a standard for the architectural description of software-intensive systems, IEEE 1471 [8]. It includes a conceptual framework to support the description of architectures, and the required content of an architectural description. The standard was developed from a consensus of current practices. It includes the use of multiple views, reusable models within views, and the relationship of architecture to the system context.

The important definitions from this standard are as follows:

**Architectural Description** A set of views and additional architectural information.

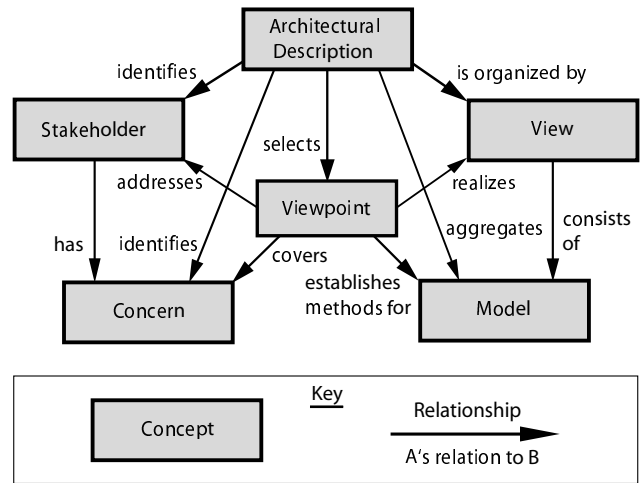
**Stakeholder** An individual, group or organization that has at least one concern relating to the system.

**Concern** A functional or non-functional requirement.

**View** A set of models representing a system from the perspective of a related set of concerns.

**Viewpoint** The conventions for creating, depicting and analyzing a view.

**Model** A particular diagram or description constructed following the method defined in a viewpoint. These pro-



**Figure 1. The concepts associated with viewpoints and their relations. Adapted from the conceptual model of software architecture description as described in IEEE Standard 1471-2000 [8].**

vide the specific description of the system, which can include identifiable sub-systems and elements.

Models are the particular representations of a system. They can be described using architectural styles, which are well known and commonly understood architectural representations [4, p18]. As a result, a property of a model will be the architectural structures that it can represent. For instance, a model that uses a layered style must be able to represent the “allowed-to-use” relationship. A measure of the models, for which a given viewpoint establishes methods, is the structures that the viewpoint can represent.

The relationships between these concepts are shown in Figure 1. This shows a subset of the framework described by the IEEE 1471. Only those concepts that are related to the viewpoint are shown because they are the concepts that will be relevant when classifying viewpoints. One exception to this is the omission of the Library Viewpoint. In this paper all viewpoints considered are project independent and, therefore, they are all library viewpoints.

The relationship between models and concerns within a viewpoint can be explicitly or implicitly defined [14]. An explicit relationship specifies the concerns and their associated models, for example as in the ISO Reference Model [9]. An implicit relationship is one from which the relevant concerns can be inferred from the diagrams and models included, for example as in UML [3]. In this paper we look only at explicit relationships, because it is relatively easier to identify the related concerns and stakeholders for each viewpoint.

### 2.3 Who are the system stakeholders?

The stakeholders of a given system are defined, for this study, as any individual, group, organization, or external system that can influence the requirements or constraints of the system. However, without a specific system we cannot identify particular stakeholders. We can define the stakeholders by their roles of interaction with software systems in general.

Several sets of stakeholder role classifications have been described. In IEEE 1471, the stakeholder roles include clients, users, architects, developers and evaluators. This is a very general classification and does not provide adequate definitions of the roles for the analysis we will perform. An alternate set is provided by Clements et al. [4], and the roles they describe are focused on the consumers of software architecture documentation. These roles will make the analysis of viewpoints possible because they cover the stakeholders that are addressed by the viewpoints.

A list of the roles of various stakeholders [4, p10-11] can be found in Table 1. As can be seen from this list, the stakeholders of any software system can be extremely varied. It is likely that their uses of any architecture related documentation will be equally varied because they are defined by their interactions with software systems. Each stakeholder will require documentation to show the relevant structures pertaining to their uses. For instance, Testers often use the documentation to determine the functional 'black boxes' and the testability of the system. They will want to see a description of the system in terms of the functionality decomposition.

Two notable omissions from this list of stakeholders are the End User and the Customer. These have probably been left out because the authors have focused on those stakeholders that need to communicate architectural information. The survey of viewpoint models may specifically identify these, or other, additional stakeholders.

### 2.4 What are the stakeholder concerns?

Software systems are implemented to meet the concerns of the stakeholders. Concerns can also be called stakeholder requirements, which can include functional and non-functional requirements.

The functional requirements are difficult to classify because the range of functionality is as varied as the range of software systems.

However, stakeholders are also concerned with the distribution of the functionality within the system, which can be described in terms of architectural structures. The quality attributes of these structures and their relationships can be mapped directly to the non-functional requirements of the system.

The non-functional requirements include all those stakeholder requirements that are not directly concerned with the services the system is to provide. Among the sets of non-functional requirements that have been described is a comprehensive classification detailed by Sommerville [15]. The author lists twelve types which are further classified into product, organizational and external requirements.

A list that includes the types of Non-Functional Requirements [15, p101] can be found in Table 1. This list provides us with a classification that should cover all the stakeholder concerns.

### 2.5 Summary

As we have seen from the IEEE 1471 conceptual framework, an architectural description consists of views and models that are generated from the selected viewpoints. The selection of viewpoints is based on the stakeholders and the concerns they have with the system to be documented. We have also seen that the stakeholders, concerns and structures of a system can be classified and, therefore, we have a method for comparing the viewpoints provided in various viewpoint models.

## 3 Related Work

Several reviews of architecture documentation have already been recorded. In this section we will look at two of them.

In the book "Documenting Software Architectures: views and beyond" the authors detail a viewpoint set for describing software architecture, the SEI set of views. The authors also examined alternate documentation models and compared them with the model they have detailed [4, p343]. This comparison was achieved by identifying the elements and relations addressed by each of the alternate models and translating these into the corresponding terminology in their viewpoint model. The models compared included; three viewpoint models, IEEE 1471, Unified Modeling Language (UML), data and control flows, and C4ISR architecture framework.

Although the translation could have been used as a basis for the comparison of viewpoints by the structures of software architecture, we decided that it would provide a better comparison to use a list of structures that was independent of any model. This list could then be expanded if additions were found in any viewpoint model.

As can be seen from the list of models, not all of them can be classified as viewpoint models. The authors also reconciled the documenting process that they detailed in terms of IEEE 1471 and mapped the models they described to the C4ISR products.

Another review of viewpoint models was provided by Smolander in the paper “What is included in Software Architecture” [14]. The author surveyed a range of documentation models and frameworks and conducted a survey of software architecture documentation practices in the Telecommunications industry. The models reviewed included; three viewpoint models, IEEE 1471, Zachman’s framework, and some additional frameworks that are not specifically viewpoint models.

Smolander then surveyed three organizations on their methods of software architecture documentation and conducted workshops with the organizations’ lead architects to analyze their approaches.

He concluded that the factors affecting viewpoint selection include stakeholders, quality attributes, the technical development environments, the work division within the organization, and the development methodology. Whilst, these are not explicitly stated in terms of the IEEE 1471 conceptual framework, each of these items can be traced to a stakeholder or a concern relating to the architectural structure. This supports the selection of the classification method chosen in this paper.

One of the models that Smolander reviews is the Information Systems Architecture (ISA) framework, originally proposed by Zachman and extended by Sowa and Zachman [17]. This provides a classification system for the documentation of software architecture concepts, developed from the building architecture metaphor.

Whilst the ISA framework provides us with a more comprehensive classification system for the documentation of a software project, the perspectives and concepts are too generalized for a comparison with the IEEE framework. In addition, the framework provides no models or structures by which to classify the cells. In essence, the ISA framework is an alternative to the IEEE 1471, but with a wider scope and at higher level.

Another framework for architecture description is the U.S. Department of Defense Architecture Framework (DoDAF) [1], which is the successor to C4ISR. Whilst this framework defines views and products, that correspond to viewpoints and models, the framework does not address the views relevant for the implementation of the system as a software architecture, and so it not been include in this survey.

## 4 Survey of Viewpoint Models

In this section we describe a framework for comparing viewpoint models and provide an overview of each model.

The analysis of the models was accomplished by comparing each of their viewpoints against the framework reference lists. By identifying the framework elements related to each viewpoint, we were able to summarize the frame-

work coverage by viewpoint model and compare them by the different elements they address.

The models selected use different terms for a viewpoint, such as views and viewtypes. Each model will be described in the language of that model, but the comparison will be made in the language of the framework.

### 4.1 A framework for comparing viewpoint models

The comparison framework used in this paper is based on the conceptual framework in IEEE 1471, as is the only standard for architectural description identified. For each of the viewpoints, the associated elements can be identified. However, these elements may not have an exact match in the framework and so each identified element must be translated into the types defined in the framework. The initial elements of the comparison framework are shown in Table 1.

It may be possible to identify new elements that cannot be translated into the framework elements. In this case, the framework will be updated to incorporate the new elements. This will allow us to expand the framework to accommodate all the viewpoints from all the models.

### 4.2 Kruchten’s “4+1” View Model

This model consists of multiple, concurrent views, that allow the different stakeholder concerns to be addressed separately. The model includes five interrelated views, see Figure 2.

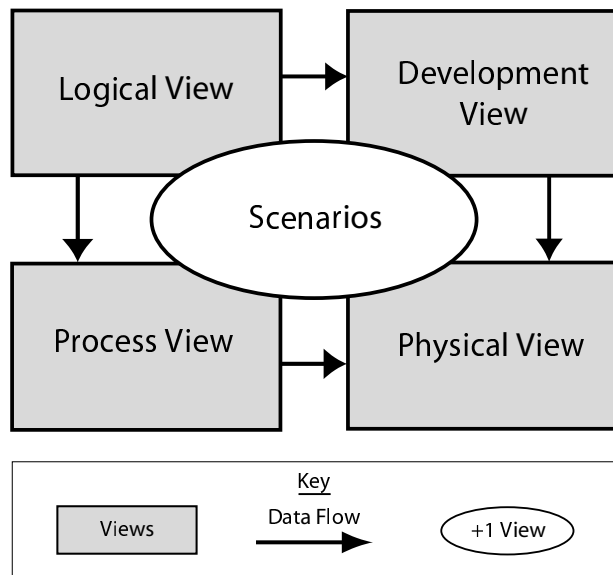


Figure 2. The “4+1” view model of Software Architecture. Adapted from “Architectural Blueprints - The “4+1” View Model of Software Architecture” [10, p43].

**Table 1. The elements of the initial comparison framework.**

<b>Structures</b>	<b>Stakeholders</b>	<b>Concerns</b>
Decomposition	Architects and Requirements Engineers	Usability
Uses	Sub-System Architects and Designers	Performance
Layered	Implementers	Space
Abstraction	Testers and Integrators	Reliability
Process	Maintainers	Portability
Concurrency	External System Architects and Designers	Delivery
Shared Data	Managers	Implementation
Client-Server	Product Line Managers	Standards
Deployment	Quality Assurance Team	Interoperability
Implementation		Ethical
Work Assignment		Privacy
		Safety

These views are each described by a blueprint and can be represented by various styles. The model is generic, so that the views can be described using alternative notations and design methods. The model also includes details of how the views are related and a process for developing the complete set of views.

Kruchten envisaged an iterative process for architecture design, starting with the description of the critical scenarios. The architect can then identify the key abstractions from the problem domain and model these in the Logical View. Logical classes can be mapped to modules and packages in the Development View, and to tasks and processes in the Process View. Finally, the processes and modules can be mapped to the hardware in the Physical View. In each subsequent iteration additional scenarios are modeled, following this sequence, until the architecture becomes stable. This usually occurs when no new key abstractions, sub-systems, processes or interfaces are discovered.

From the above process it is evident that the views are not fully independent. It would be difficult to model the views without first having proceeded through the scenarios and logical views. Furthermore, the views of the “4+1” view model conform reasonably well to the viewpoints of the IEEE 1471. However, the views of this model rely on an iterative process to create them.

### 4.3 The SEI Viewpoint Model

The SEI model of software architecture documentation has been well documented and this review is based on a paper [5], a tutorial [6], and a book [4]. In these documents the model is detailed and processes described for choosing views and documenting the architecture.

The process of documenting the architecture consists of documenting the relevant views and any additional information that corresponds to more than one view. However, some views are too complex to show in one representa-

tion, so they can be broken down into a number of view-packets. A viewpacket is the smallest piece of information a stakeholder requires, which can be represented by one or more styles. For instance, a single viewpacket could show a whole system at a high-level, and a number of additional viewpackets could be used to show the individual sub-systems in greater detail.

The SEI model includes a template for the contents of a viewpacket, so that it will comply with the IEEE 1471. This consists of a primary representation, an element catalog, a context diagram, a variability guide, architectural background, non-architectural information, and the relationship to other viewpackets.

The list of styles, and views, which can be used to represent the system is based on the structure of the system itself. This can be large in complex systems. To reduce this list, the needs of the stakeholders must be taken into consideration, and less relevant styles can be omitted. However, the specific requirements of stakeholders varies from project to project. The stakeholders associated with each of the viewtypes have been adapted from the table provided in Chapter Nine “Choosing the Views” of the source book [4, p301]. Only those stakeholders who require detailed information for the viewtype have been shown, because these are the primary target of the viewtype.

The styles of the SEI model are related to each other within their viewtypes. However, there is limited interaction between the viewtypes because of the different types of elements that they show. The exception is the allocation viewtype, which maps the modules identified in the module viewtype to elements in the external environments.

The SEI model extends IEEE 1471. The authors have developed this model as a practical method of documenting software architecture that complies with the standard. They propose that architectural styles can be used as an adaptable set of models that can change as the patterns of software architecture evolve.

As we have already seen, the particular stakeholders of a system determine which views should be documented. This is true for both the IEEE standard and the SEI model. IEEE 1471 specifies that system stakeholders have viewpoints which are used as templates to generate views, when applied to the system. The SEI model utilizes styles to determine the views of the system that can be generated. The stakeholder requirements are then used to reduce the number of views required in the document package. Each method has a different starting point, but both can result in the same “stakeholder-centric” set of views.

#### 4.4 ISO Reference Model of Open Distributed Processing

The Reference Model of Open Distributed Processing (RM-ODP) provides a framework for the standardization of open distributed processing. The aim of the framework is to develop standards for the distribution of information processing services. It is independent of any application domain.

The RM-ODP framework is based on the support for the specification of common services and infrastructure components. The system specification is divided into viewpoints to simplify the description of complex systems and separate the concerns. The five viewpoints identified by the model are the enterprise, information, computation, engineering, and technology viewpoints. A set of general concepts and rules are defined for each viewpoint, and a language to express them. They can be developed separately whilst specifying the constraints that they place on each other. There is no sequence or iteration implied in the viewpoints.

The framework includes the specification of transparencies, which factor out concerns and simplify the viewpoints. They are standard mechanisms selected to solve some requirements for a part of the system. For example, a persistence transparency removes the need to show the deactivation and reactivation of software elements.

There are no stakeholders identified for any of the views of this model. However, the RM-ODP states that it is designed to meet the needs of system developers, translated as implementers, and standard writers.

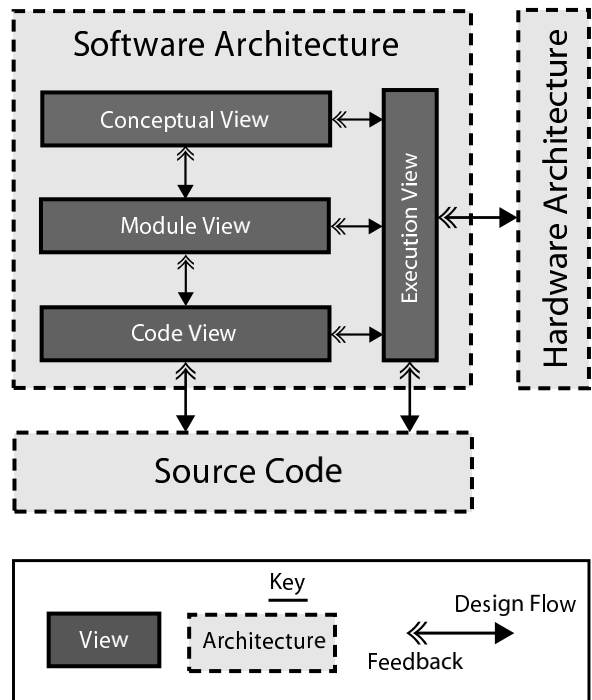
The RM-ODP does not specify any links between views. However, if a set of views show aspects of the same system, then they must have correspondences. The relationship between elements of different views should be defined by translation rules in order to provide verification of the overall software architecture.

An example of a translation rule provided by the RM-ODP is as follows [9, p32]:

“a single computational interface cannot be divided into separate engineering interfaces supported by unconnected channel structures.”

The RM-ODP provides a language for describing software architecture, and not a notation. Although, it was specifically developed for the distributed software domain, its coverage of concerns and structures shows that it is also applicable to much of software architecture in general. The emphasis on defining standard languages reflects the bias towards interoperability between domains, and not on stakeholder communication. This model is primarily to enable communication between developers of different systems, and not between other stakeholders of the same system.

#### 4.5 Siemens Four View Model



**Figure 3. The Siemens four view model of Software Architecture. Adapted from the four views of software architecture [7, p15].**

This model is a result of a study into the industrial practices of software architecture. The authors found that the structures used to design and document software architecture fall into four broad categories, which they call conceptual, module, execution and code structures. Each category addresses different stakeholder concerns. In addition, they found that when these categories are addressed separately, the implementation complexity of systems is decreased and the re-use and reconfiguration of software is improved. To separate the categories, they proposed that software architecture should be documented using four views, each of which address a different category of structures, see Figure 3.

The focus of this model is on the design approach for the software architect. As a result, the model fails to address the other stakeholders explicitly.

The four views of this model are loosely coupled. The design flow follows the information passed between views starting from the conceptual view. The feedback results from the testing of views for conformance to the non-functional requirements of the system.

Several important mappings of structures are explicitly defined in the design approach. Conceptual structures are “implemented-by” module structures, and “assigned-to” execution structures. Module structures can be “located-in” or “implemented-by” code structures. Code Structures can configure execution structures.

The Siemens four view model ignores the explicit concerns of the stakeholders, other than the software architect. However, the other stakeholders may be addressed implicitly by the concerns satisfied by each of the views. This reflects the focus of the model on the architect’s design approach and not on the documentation for communication.

#### 4.6 Rational Architectural Description Specification (ADS)

The Rational ADS is an expansion on the “4+1” model to enable the description of more complex architectures, such as enterprise, e-business, embedded systems and non-software systems. It was first defined in November 2000 and is now taught as part of Rational’s course “Principles of Architecting Software Systems (PASS)”.

It features a formal definition of requirements evolution and architecture testability, and utilizes UML notation where possible. The rationale for the architecture must still be documented separately.

The views of the “4+1” model have been partially renamed and four new views defined. The Scenarios view has become the Use Case view, the Development view has become the Implementation view, and the Physical view has become the Deployment view. The nine views have been grouped into four viewpoints, see Figure 4. The views of each viewpoint correspond to the models of the IEEE 1471 framework in the Rational ADS. The consistency of elements in different views is maintained by explicit mappings between the views. The context of lower viewpoints are provided by the mappings to higher viewpoints.

As can be seen from the explicit mappings between views shown in Figure 4, the Non-Functional Requirements View is only loosely coupled to the other views. Usually it is depicted as a list of requirements and their properties, such as priority or category. As it is not linked to any of the structural views of software architecture it has been omitted when determining the concerns addressed by this model.

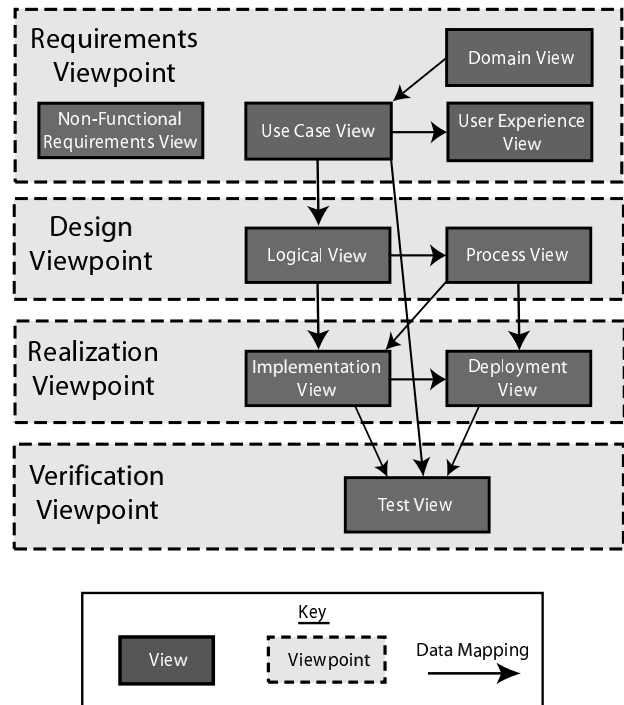


Figure 4. The Rational ADS: Viewpoints and Views. Adapted from a presentation slide [11].

## 5 Survey Findings

In this section we will look at the coverage of the framework elements by the viewpoint models, the correspondence between models, and the divergence of the models.

### 5.1 Framework coverage

Coverage is a measure of the extent to which the elements of a domain are addressed. In this case, each model was assessed by determining the coverage as the sum of all the elements of the framework across all viewpoints. Coverage was determined independently for each of the framework concepts of stakeholder, concern and architecture structure.

The coverage of the framework information space is shown in Table 2. This lists the number of framework concepts covered by the five viewpoint models surveyed. The number of viewpoints within each model that address each of the framework elements are shown in Tables 4, 5 and 6.

The greatest coverage of stakeholders is achieved by the Rational ADS. This is primarily because of the number and variety of views detailed is greater than in any of the other models. The “4+1” and SEI models also focus on satisfying the documentation needs of the stakeholders and, as a result, also provides good coverage.

**Table 2. Concept coverage of the surveyed viewpoint models.**

Model	Stakeholders	Concerns	Structures
“4+1”	7	7	11
SEI	8	9	11
RM-ODP	2	9	9
Siemens	1	7	9
Rational ADS	9	10	10
Framework	11	12	11

The RM-ODP and the Siemens model provide little coverage of the stakeholders. The focus of the RM-ODP is on defining the language of architecture and not on defining the stakeholders. The Siemens model is focused on the design approach for software architecture, and so each viewpoint is treated primarily from the architect’s perspective. Neither of these models explicitly relate viewpoints to the stakeholders, but they do implicitly satisfy the stakeholders via the concerns that the models address.

The coverage of the concerns and structures show less variation than for the stakeholders because all the models recognize that software architecture is composed of structures and constrained by concerns. As a result, each model includes enough viewpoints to describe a complete software architecture.

Additional elements were identified that were not included in the original comparison framework. These consisted of two new stakeholders which were end users, from the “4+1” view model and Rational ADS, and standards writers, from the RM-ODP.

## 5.2 Correspondence between viewpoints

Some correspondence exists between the viewpoints of the different models, most notably in the distribution of structures addressed by the viewpoints. If these are grouped into functional, dynamic and external structures, we can see that three of the models have a good correlation. Table 3 shows the corresponding viewpoints of the “4+1”, SEI and Siemens models based on a rough grouping of the structures of software architecture. The one structure not included is deployment because the models show no agreement on how it should be grouped with other structures. The “4+1” model sets it apart, the SEI model shows it as external, the Siemens model groups it with the behavioral structures, and the RM-ODP spreads deployment across several viewpoints.

The Rational ADS addresses these structures in a different way to the other models. Each of its viewpoints can represent both structural and behavioral aspects of the architecture. This results in a broader set of structures that can be represented by each viewpoint. The exception to this is the

**Table 3. The correspondence of structures between the viewpoints from different models.**

Functional	
Structures	Decomposition, Uses, Layered, Abstraction.
“4+1” model	Logical view.
SEI model	Module viewtype.
Siemens model	Module view.
Behavioral	
Structures	Process, Concurrency, Shared Data, Client-Server.
“4+1” model	Process view.
SEI model	C&C viewtype.
Siemens model	Execution view.
External	
Structures	Implementation, Work Assignment.
“4+1” model	Development view.
SEI model	Allocation viewtype.
Siemens model	Code view.
Rational ADS	Realization viewpoint.

realization viewpoint which maps closely into the external group.

The viewpoints of the RM-ODP do not correspond with these groupings. This is because it is primarily a framework for defining the languages of the viewpoints. These languages are based on a wider set of domains than the other models, which are based primarily on the software architecture domain. This blurs the boundaries because structures can appear in different groupings in different domains.

## 5.3 Divergence of models

The models vary in different foci, which reflect their primary purposes. The SEI model is biased towards communication, the “4+1” and Siemens models as biased toward design, the RM-ODP is biased toward development across domains, and the Rational ADS is biased towards requirements evolution and testability.

The SEI model consists of relatively independent viewpoints, compared to the other models. This allows a documentation package to be created independently for the different stakeholders.

The “4+1” and Siemens models include flows of information between the viewpoints. This supports the design approaches that they suggest.

The Rational ADS contains specific mappings between elements of related views. This provides an implicit design flow through the four viewpoints. This model also supports the description of the business context of the system. It is

**Table 4. The numbers of viewpoints that address the stakeholders of the comparison framework.**

Stakeholder Role	“4+1”	SEI	RM-ODP	Siemens	Rational ADS
Number of Viewpoints	5	3	5	4	4
Architects / Requirements Engineers	1	3	0	4	3
Sub-System Architects / Designers	3	2	0	0	4
Implementers	3	2	5	0	3
Testers / Integrators	1	1	0	0	1
Maintainers	0	2	0	0	1
External System Architects / Designers	0	1	0	0	0
Managers	1	1	0	0	1
Product Line Managers	1	1	0	0	2
Quality Assurance Team	0	0	0	0	0
End User	2	0	0	0	3
Standard Writers	0	0	5	0	3

the only model to address usability and quality assurance, using the user experience and test views respectively. This demonstrates the greater scope available from the number of views detailed.

The RM-ODP specifically addresses the structure of the information within the system and the business policies relevant to the system. The inclusion of these two viewpoints shows the greater scope of the description envisaged by the model. The depth of the viewpoint languages allows this model to be used as a standard dictionary for describing software architecture in detail.

In these ways the models show how they focus on different aspects of documenting software architecture.

#### 5.4 Optimal framework coverage

The five viewpoint models each provide a different coverage of the framework concepts. A useful result would be to determine a minimal set of viewpoints with the greatest coverage. This can be approached by determining which viewpoints from different models can be combined into a new set, and identifying the smallest set to provide the maximum coverage.

The biggest barrier to combining viewpoints from different models is the interrelation of viewpoints within each model. The “4+1” includes a strong data flow from one model to another, starting from the scenarios, as part of an iterative approach. There is a similar data flow through the viewpoints of the Siemens model, but these are less tightly coupled. The Rational ADS also has a strong dependency as the context for the lower viewpoints are provided by the higher viewpoints. The viewpoints of the SEI model and the RM-ODP are relatively independent, and defining any translations between the viewpoints is deferred to the level of the architectural description. As a consequence, the viewpoints from the latter two models are the best candidates for merging.

To provide a minimal set of viewpoints we must look at the coverage overlap between them. First we must select a base model to enhance and then add additional viewpoints to cover the missing concepts.

The best choice for a base model is the SEI model. This contains independent viewpoints, and overlaps well with the “4+1” and Siemens models. Whilst the SEI model provides complete coverage of the structures, it does not satisfy the stakeholders of end user or standard writer. Neither does it satisfy the concerns of standards or safety. Standards, and implicitly standard writers, and end users can be addresses by inclusion of the requirements viewpoint from the Rational ADS. This is the highest viewpoint of the model and so is not dependant on the other viewpoints. The safety concern is only explicitly addressed in the Siemens model conceptual viewpoint. However, we do not need to include this viewpoint because the component and connector viewpoints can be used to address this concern [16, p5].

The resulting optimum set of viewpoints with the greatest coverage is as follows:

- Requirements viewpoint, from Rational ADS.
- Module viewtype, from SEI model.
- C&C viewtype, from SEI model.
- Allocation viewtype, from SEI model.

The selected set does not cover the entire domain of the comparison framework. None of the models surveyed address the ethical concerns. The Rational ADS could be expanded to cover the ethical concerns using the requirements viewpoint to specify the relevant business rules and policies.

The Rational ADS verification viewpoint, which has not been included, does satisfy the quality assurance team, but it is dependant on the other viewpoints of the model to provide its context. Further investigation could determine whether this viewpoint could be adapted to satisfy this concern independently.

**Table 5. The numbers of viewpoints that address the concerns of the comparison framework.**

<b>Concern</b>	<b>“4+1”</b>	<b>SEI</b>	<b>RM-ODP</b>	<b>Siemens</b>	<b>Rational ADS</b>
Number of Viewpoints	5	3	5	4	4
Usability	0	0	0	0	2
Performance	2	1	3	2	3
Space	0	1	1	0	1
Reliability	2	1	2	1	2
Portability	3	1	1	2	2
Delivery	1	2	2	2	4
Implementation	2	3	1	2	3
Standards	0	0	2	0	2
Interoperability	1	1	3	1	3
Ethical	0	0	0	0	0
Privacy	1	1	3	0	1
Safety	0	1	0	1	0

## 5.5 Implications of results

This paper shows that the different vocabularies of the viewpoint models can be compared via a common reference vocabulary. The viewpoints from different models, when combined into an optimum set, can provide greater coverage of the software architecture domain than any of the individual viewpoint models.

The optimum viewpoint set was selected for the combined coverage that they provide and their relative independence. However, they must be connected in some aspects because they can describe the same architecture. No effort was made to determine the interrelation of this optimum set.

There are two limitations with the comparison of viewpoints by stakeholders. First, the results were obtained by using an initial arbitrary reference list. However, the list was selected from one of the models to be compared. A more rigorous method would have been to compile a completely independent vocabulary for the reference list. The second limitation arises in two of the models, the RM-ODP and the Siemens model, because they do not explicitly relate stakeholders to the viewpoints. The identification of the stakeholders satisfied by each viewpoint could have been achieved by investigating the relationship between concerns and stakeholders. The stakeholders would be implicitly satisfied if all their concerns were addressed by a viewpoint. The additional analysis required to overcome these limitations of the stakeholder comparison was beyond the scope of this paper.

## 5.6 Further research

As has been previously stated, the scope of this paper was limited. Additional effort could have been expended in identifying a more authoritative set of framework elements. In addition, the identification of the implicit con-

cerns and stakeholders addressed by each of the viewpoints would have enabled a more accurate identification of the optimum set.

A useful exercise would be to verify the selected optimum set of viewpoints. This could be achieved by modeling systems from case studies published in the source literature. If this set does cover the five models surveyed then it should be possible to model each case study with the optimum set of viewpoints.

An additional avenue of research would be to study the use of architectural transparencies, from the RM-ODP, with the selected viewpoint set. This feature could provide an effective way to simplify the documentation of software architecture.

## 6 Acknowledgements

The author would like to thank Keith Frampton and Dr. Hongyu Zhang, both of RMIT University, who proposed the area of research and provided advice for the original dissertation and on this paper. In addition, the author would like to thank Davyd Norris, of IBM Australia, for the documentation that he generously provided.

## References

- [1] Architecture Framework Working Group. DoD Architecture Framework Version 1.0, Volume I: Definitions and Guidelines. Technical report, U.S. Department of Defense, Feb. 2004. [http://www.defenselink.mil/nii/doc/DoDAF\\_v1\\_Volume\\_I.pdf](http://www.defenselink.mil/nii/doc/DoDAF_v1_Volume_I.pdf) viewed on 9th January, 2004.

Table 6. The numbers of viewpoints that address the structures of the comparison framework.

Structure	“4+1”	SEI	RM-ODP	Siemens	Rational ADS
Number of Viewpoints	5	3	5	4	4
Decomposition	2	1	4	3	3
Uses	2	1	2	2	3
Layered	1	1	2	1	2
Abstraction	2	1	2	0	2
Process	1	1	1	1	1
Concurrency	1	1	2	1	3
Shared Data	1	1	0	1	1
Client-Server	1	1	1	1	0
Deployment	1	1	2	1	1
Implementation	1	1	1	1	1
Work Assignment	1	1	0	0	1

- [2] L. Bass et al. *Software Architecture in Practice*. Addison Wesley, Boston, MA, USA, 2nd edition, 2003. ISBN 0-321-15495-9.
- [3] G. Booch et al. *The Unified Modeling Language User Guide*. Object Technology Series. Addison Wesley Professional, Boston, MA, USA, 1st edition, Sept. 1998. ISBN 0-201-57168-4.
- [4] P. Clements et al. *Documenting Software Architecture: Views and Beyond*. Addison Wesley, Boston, MA, USA, 1st edition, 2002. ISBN 0-201-70372-6.
- [5] 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> viewed on 20th September, 2004, Sept. 2002. Draft.
- [6] P. Clements et al. Tutorial F3: Documenting Software Architectures: Views and Beyond. International Conference on Software Engineering, May 2003. Portland, Oregon.
- [7] C. Hofmeister et al. *Applied Software Architecture*. Object Technology Series. Addison Wesley, Boston, MA, USA, 1st edition, 2000. ISBN 0-201-32571-3.
- [8] IEEE. *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*. Institute of Electrical and Electronics Engineers, Sept. 2000. IEEE Std 1471-2000.
- [9] ISO. *Reference Model of Open Distributed Processing (RM-ODP)*. International Organization for Standardization, 1994. Technical Report 10746.
- [10] P. Kruchten. Architectural Blueprints - The “4+1” View Model of Software Architecture. *IEEE Software*, 12(6):42–50, 1995.
- [11] D. Norris. Communicating Complex Architectures with UML and the Rational ADS. In *Proceedings of the IBM Rational Software Development User Conference*, 2004. From documents received in personal communication with the author on 12th December, 2004. Copyright 2004 IBM Australia.
- [12] D. E. Perry and A. L. Wolf. Foundations for the study of software architecture. *ACM SIGSOFT Software Engineering Notes*, 17(4):40–52, 1992.
- [13] 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.
- [14] 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. IEEE.
- [15] I. Sommerville. *Software Engineering*. Addison Wesley, Boston, MA, USA, 6th edition, aug 2000. ISBN 0-201-39815-X.
- [16] D. Soni et al. Software architecture in industrial applications. In *International Conference on Software Engineering*, pages 196–207, 1995.
- [17] J. F. Sowa and J. A. Zachman. Extending and formalizing the framework for information systems architecture. *IBM Systems Journal*, 31(3):590–616, 1992.