<Insert OrganizationName>

<Insert Project Name>  
Software Architecture Document (SAD)

CONTENT OWNER: <Insert Name>

|  |  |  |
| --- | --- | --- |
| DOCUMENT NUMBER: | RELEASE/REVISION: | RELEASE/REVISION DATE: |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |
| --- |
| **BACKGROUND**  This template is based on the Software Engineering Institute’s “View and Beyond” method for documenting software architectures, as described in Clements, et al., [*Documenting Software Architecture: Views and Beyond*](http://www.sei.cmu.edu/architecture/books.html) (Addison Wesley, 2002). The current version is available for [free download](http://www.sei.cmu.edu/architecture/arch_doc.html) from the SEI’s architecture web site.  **TIPS FOR USING THIS TEMPLATE**  To create an instance of this document:  Insert relevant information on cover sheet and in placeholders throughout.  Insert relevant information in page header: Move to a page of the body of the report, select *View > Header and Footer* from the main menu, and then replace relevant information in the header box at the top of the page.  To update the contents and page numbers in the Table of Contents, List of Figures, and List of Tables:  Position the cursor anywhere in the table to be updated.  Click the *F9* function key.  Answer “Update entire table”.  To insert a figure or table caption:  From the main menu, choose *Insert > Reference > Caption* and then either *Figure* or *Table* as needed.  Click the OK button.  Add a colon and a tab stop after the figure number in the caption itself.  The caption should use the *Caption* style.  Add a colon and a tab stop after the table/figure number in the caption itself.  **TIPS FOR MAKING YOUR DOCUMENT MORE READABLE**  A gray box containing *CONTENTS OF THIS SECTION* is provided at the beginning of most sections and subsections. After determining what specific information will be included in your document, you can remove this gray box or leave it to serve as a quick-reference section overview for your readers. In the case that text has been provided in the template, inspect it for relevance and revised as necessary.  Consider hyperlinking key words used in the document with their entries in the [Glossary](#_Glossary) or other location in which they are defined. Choose *Insert > Hyperlink*.  Don’t leave blank sections in the document. Mark them “To be determined” (ideally with a promise of a date or release number by which the information will be provided) or “Not applicable.”  Consider packaging your SAD as a multi-volume set of documentation. It is often helpful to break your documentation into more than one volume so that the document does not become unwieldy. There are many ways that this can be accomplished. The structuring of the document must support the needs of the intended audience and must be determined in the context of the project. Each document that you produce should include the date of issue and status; draft, baseline, version number, name of issuing organization; change history; and a summary. A few decomposition options are:  *A 2-Volume approach:* Separate the documentation into two volumes; one that contains the views of the software architecture and one that contains everything else. A common variant of this approach has one volume per view, and one volume for everything else.  *A 3-Volume approach:* Document organizational policies, procedures, and the directory in one volume, system specific overview material in a second, and view documentation in a third.  *A 4-Volume approach:* Create one volume for each viewtype [Clements 2002] (module, component-and-connector, allocation) that contains the documentation for the relevant [views](#view). Include all of the other information in the fourth volume.  Software interfaces are often documented in a separate volume.  In *any* case, the information should be arranged so that readers begin with the volume containing the Documentation Roadmap (Section 1 in this template). |

Table of Contents

[1 Documentation Roadmap 1](#_Toc126922689)

[1.1 Document Management and Configuration Control Information 1](#_Toc126922690)

[1.2 Purpose and Scope of the SAD 1](#_Toc126922691)

[1.3 How the SAD Is Organized 1](#_Toc126922692)

[1.4 Stakeholder Representation 1](#_Toc126922693)

[1.5 Viewpoint Definitions 1](#_Toc126922694)

[1.5.1 <Insert name of viewpoint> Viewpoint Definition 1](#_Toc126922695)

[1.5.1.1 Abstract 1](#_Toc126922696)

[1.5.1.2 Stakeholders and Their Concerns Addressed 1](#_Toc126922697)

[1.5.1.3 Elements, Relations, Properties, and Constraints 1](#_Toc126922698)

[1.5.1.4 Language(s) to Model/Represent Conforming Views 1](#_Toc126922699)

[1.5.1.5 Applicable Evaluation/Analysis Techniques and Consistency/Completeness Criteria 1](#_Toc126922700)

[1.5.1.6 Viewpoint Source 1](#_Toc126922701)

[1.6 How a View is Documented 1](#_Toc126922702)

[1.7 Relationship to Other SADs 1](#_Toc126922703)

[1.8 Process for Updating this SAD 1](#_Toc126922704)

[2 Architecture Background 1](#_Toc126922705)

[2.1 Problem Background 1](#_Toc126922706)

[2.1.1 System Overview 1](#_Toc126922707)

[2.1.2 Goals and Context 1](#_Toc126922708)

[2.1.3 Significant Driving Requirements 1](#_Toc126922709)

[2.2 Solution Background 1](#_Toc126922710)

[2.2.1 Architectural Approaches 1](#_Toc126922711)

[2.2.2 Analysis Results 1](#_Toc126922712)

[2.2.3 Requirements Coverage 1](#_Toc126922713)

[2.2.4 Summary of Background Changes Reflected in Current Version 1](#_Toc126922714)

[2.3 Product Line Reuse Considerations 1](#_Toc126922715)

[3 Views 1](#_Toc126922716)

[3.1 <Insert view name> View 1](#_Toc126922717)

[3.1.1 View Description 1](#_Toc126922718)

[3.1.2 View Packet Overview 1](#_Toc126922719)

[3.1.3 Architecture Background 1](#_Toc126922720)

[3.1.4 Variability Mechanisms 1](#_Toc126922721)

[3.1.5 View Packets 1](#_Toc126922722)

[3.1.5.1 View packet # j 1](#_Toc126922723)

[3.1.5.1.1 Primary Presentation 1](#_Toc126922724)

[3.1.5.1.2 Element Catalog 1](#_Toc126922725)

[3.1.5.1.3 Context Diagram 1](#_Toc126922726)

[3.1.5.1.4 Variability Mechanisms 1](#_Toc126922727)

[3.1.5.1.5 Architecture Background 1](#_Toc126922728)

[3.1.5.1.6 Related View Packets 1](#_Toc126922729)

[4 Relations Among Views 1](#_Toc126922730)

[4.1 General Relations Among Views 1](#_Toc126922731)

[4.2 View-to-View Relations 1](#_Toc126922732)

[5 Referenced Materials 1](#_Toc126922733)

[6 Directory 1](#_Toc126922734)

[6.1 Index 1](#_Toc126922735)

[6.2 Glossary 1](#_Toc126922736)

[6.3 Acronym List 1](#_Toc126922737)

[7 Sample Figures & Tables 1](#_Toc126922738)

List of Figures

[Figure 1: Sample Figure 1](#_Toc51125206)

List of Tables

[Table 1: Stakeholders and Relevant Viewpoints 1](#_Toc87083002)

[Table 2: Sample Table 1](#_Toc87083003)

# Documentation Roadmap

The Documentation Roadmap should be the first place a new reader of the SAD begins. But for new and returning readers, it is intended to describe how the SAD is organized so that a reader with specific interests who does not wish to read the SAD cover-to-cover can find desired information quickly and directly.

Sub-sections of Section 1 include the following.

* Section 1.1 (“Document Management and Configuration Control Information”) explains revision history. This tells you if you’re looking at the correct version of the SAD.
* Section 1.2 (“Purpose and Scope of the SAD”) explains the purpose and scope of the SAD, and indicates what information is and is not included. This tells you if the information you’re seeking is likely to be in this document.
* Section 1.3 (“How the SAD Is Organized”) explains the information that is found in each section of the SAD. This tells you what section(s) in this SAD are most likely to contain the information you seek.
* Section 1.4 (“Stakeholder Representation”) explains the stakeholders for which the SAD has been particularly aimed. This tells you how you might use the SAD to do your job.
* Section 1.5 (“Viewpoint Definitions”) explains the *viewpoints* (as defined by IEEE Standard 1471-2000) used in this SAD. For each viewpoint defined in Section 1.5, there is a corresponding view defined in Section 3 (“Views”). This tells you how the architectural information has been partitioned, and what views are most likely to contain the information you seek.
* Section 1.6 (“How a View is Documented”) explains the standard organization used to document architectural views in this SAD. This tells you what section within a view you should read in order to find the information you seek.

## Document Management and Configuration Control Information

|  |
| --- |
| **CONTENTS OF THIS SECTION**: This section identifies the version, release date, and other relevant management and configuration control information associated with the current version of the document. Optional items for this section include: change history and an overview of significant changes from version to version. |

* Revision Number: << >>
* Revision Release Date: << *>*>
* Purpose of Revision: << >>
* Scope of Revision: <<*list sections or page numbers that have been revised; provide a summary overview of the differences between this release and the previous one.>>*

## Purpose and Scope of the SAD

### Purpose

This document provides a high level overview of the evolving technical architecture for the Sales System of a retail chain using a loyalty card point system. It also provides a high-level description of the goals of the architecture, the use cases support by the system and architectural styles and components that have been selected to best achieve the use cases.

In addition to these views, this architectural description will:

* Identify the candidate patterns and tactics that will become the architecture design of the system.
* Frame the architectural design activity, begin with the given technical constraints and the structures
* Identify patterns generally suit the needs described in the product description based upon the quality attribute scenarios
* Identify tactics we apply to further refine the initial decomposition of the system and promote the necessary quality attributes.

The architecture has a set of guiding principles as well as known criteria and constraints that shape the proposed architecture. It is intended to capture and convey the significant architectural decisions which have been made on the system.

The development team can use this document to review the architecture of the system. The Architecture document will be also useful for future development teams.

### Scope

The project will aim to develop a sale system for Company A, a retail chain (hereinafter, the system) in conjunction with its launch of a point service.

The system consists of a head office server, located at the head office, and the POS terminals placed at store cashiers. The head office server and the POS terminals are connected to each other via a network. Products sold at stores have bar codes attached which indicate the product codes. These bar codes can be read with bar code readers of POS terminals.

Customer who have become point service members are issued point cards, which bear bar codes indicating their member numbers, and when they purchase products with cash, they are awarded points based on the amount of their purchase. The point award rate is fixed, and stored in the system as constant.

Points are recorded in the system, and not on the point card. The awarded points become valid for use the following day, and can be used by the member who accrued the points in place of cash when making a purchase. Points can be used in all stores.

For each product, its standard price, common to all stores, is set as a part of the product data. Each store, however, can set and use its own actual retail price instead of the standard price during the limited period specified be each store. The actual retail price must be set in advance, and it cannot be charged in the middle of the specified period.

Products are classified into product types such as food, general merchandise, etc. Not all stores carry every product type, and the range of product types carried is designated for each store.

Moreover, in addition to the sales operation, the system is also capable of performing the statistical analysis on the sales records of all stores in near real-time manner.

Both the head office server and the POS terminals are equipped with a Database, and both the head office server Database and the POS terminal Database can be accessed during the sales operation.

## How the SAD Is Organized

In order to fully document all the aspects of the architecture, the Software Architecture Document contains the following subsections.

Section 2: describes the use of each view

Section 3: describes the architectural constraints of the system

Section 4: describes the functional requirements with a significant impact on the architecture

Section 5: describes the most important use-case realization.

Section 6: describes design’s concurrency aspects

Section 7: describes how the system will be deployed.

Section 8: describes any significant persistent element.

Section 9: describes any performance issues and constraints

Section 10: describes any aspects related to the quality of service (QoS) attributes

|  |
| --- |
| **CONTENTS OF THIS SECTION**: This section provides a narrative description of the major sections of the SAD and the overall contents of each. Readers seeking specific information can use this section to help them locate it more quickly. |

This SAD is organized into the following sections:

* **Section 1 (“Documentation Roadmap”) provides information about this document and its intended audience**. It provides the roadmap and document overview. Every reader who wishes to find information relevant to the software architecture described in this document should begin by reading Section 1, which describes how the document is organized, which stakeholder viewpoints are represented, how stakeholders are expected to use it, and where information may be found. Section 1 also provides information about the views that are used by this SAD to communicate the software architecture.
* **Section 2 (“Architecture Background”) explains why the architecture is what it is.** It provides a system overview, establishing the context and goals for the development. It describes the background and rationale for the software architecture. It explains the constraints and influences that led to the current architecture, and it describes the major architectural approaches that have been utilized in the architecture. It includes information about evaluation or validation performed on the architecture to provide assurance it meets its goals.
* **Section 3 (Views”) and Section 4 (“Relations Among Views”) specify the software architecture**. Views specify elements of software and the relationships between them. A view corresponds to a viewpoint (see Section 1.5), and is a representation of one or more structures present in the software (see Section 1.2).
* **Sections 5 (“Referenced Materials”) and 6 (“Directory”) provide reference information for the reader.** Section 5 provides look-up information for documents that are cited elsewhere in this SAD. Section 6 is a *directory*, which is an index of architectural elements and relations telling where each one is defined and used in this SAD. The section also includes a glossary and acronym list.

## Stakeholder Representation

This section provides a list of the stakeholder roles considered in the development of the architecture described by this SAD. For each, the section lists the concerns that the stakeholder has that can be addressed by the information in this SAD.

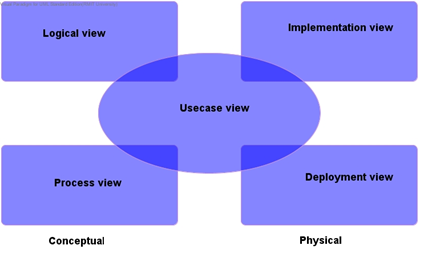
Each stakeholder of a software system—customer, user, project manager, coder, analyst, tester, and so on—is concerned with different characteristics of the system that are affected by its software architecture. For example, the user is concerned that the system is reliable and available when needed; the customer is concerned that the archi­tecture can be implemented on schedule and to budget; the manager is worried (in addition to cost and schedule) that the architecture will allow teams to work largely independently, interacting in disciplined and controlled ways. The devel­oper is worried about strategies to achieve all of those goals. The security analyst is concerned that the system will meet its information assurance requirements, and the performance analyst is similarly concerned with it satisfying real-time deadlines.

This information is represented as a matrix, where the rows list stakeholder roles, the columns list concerns, and a cell in the matrix contains an indication of how serious the concern is to a stakeholder in that role. This information is used to motivate the choice of viewpoints chosen in Section 1.5.

|  |  |  |
| --- | --- | --- |
| **CONTENTS OF THIS SECTION**: The list of stakeholders will be unique for each organization that is developing a SAD. ANSI/IEEE 1471-2000 requires that at least the following stakeholders be considered:   * Users * Acquirers * Developers * Maintainers.   You may wish to consider the following additional stakeholders. | | |
| Customer  Application software developers  Infrastructure software developers  End users  Application system engineers  Application hardware engineers | Project manager  Communications engineers  Chief Engineer/Chief Scientist  Program management  System and software integration and test engineers  Safety engineers and certifiers | External organizations  Operational system managers  Trainers  Maintainers  Auditors  Security engineers and certifiers |

## Viewpoint Definitions

The above diagram represents the overall diagram of the proposed system. Furthermore the architecture for the proposed system will be broken down using the 4 + 1 Architectural Model.



Logical view

Audience: Designers.

Area: Functional Requirements: describes the design's object model. Also describes the most important use-case realizations and business requirements of the system.

Related Artifacts: Design model

Process view

Audience: Integrators.

Area: Non-functional requirements: describes the design's concurrency and synchronization aspects.

Related Artifacts: (no specific artifact).

Implementation view

Audience: Programmers.

Area: Software components: describes the layers and subsystems of the application.

Related Artifacts: Implementation model, components

Deployment view

Audience: Deployment managers.

Area: Topology: describes the mapping of the software onto the hardware and shows the system's distributed aspects. Describes potential deployment structures, by including known and anticipated deployment scenarios in the architecture we allow the implementers to make certain assumptions on network performance, system interaction and so forth.

Related Artifacts: Deployment model.

Use Case view

Audience: all the stakeholders of the system, including the end-users.

Area: describes the set of scenarios and/or use cases that represent some significant, central functionality of the system. Describes the actors and use cases for the system, this view presents the needs of the user and is elaborated further at the design level to describe discrete flows and constraints in more detail. This domain vocabulary is independent of any processing model or representational syntax (i.e. XML).

Related Artifacts : Use-Case Model, Use-Case documents

Data view (optional)

Audience: Data specialists, Database administrators

Area: Persistence: describes the architecturally significant persistent elements in the data model

Related Artifacts: Data model.

Scenarios: The description of architecture is illustrated using a small set of use cases, or scenarios which become a fifth view. The scenarios describe sequences of interactions between objects, and between processes. They are used to identify architectural elements and to illustrate and validate the architecture design. They also serve as a starting point for tests of an architecture prototype. UML Diagram(s) used to represent scenario view include the Use case diagram

The rest of the document will elaborate on the system from an architectural point of view. It will contain diagrams showing the showing the Architecture overview, deployment view, process view, use case view, logical view, deployment view, implementation view and data view.

It will also describe the significant architectural definitions that were done during the architecture design phase. Furthermore the document will also contain use case realizations with scenarios

|  |
| --- |
| **CONTENTS OF THIS SECTION**: This section provides a short textual definition of a viewpoint and how the concept is used in this SAD. The section describes viewpoints that may be used in the SAD. The specific viewpoints will be tailored by the organization. |

The SAD employs a stakeholder-focused, multiple view approach to architecture documentation, as required by ANSI/IEEE 1471-2000, the recommended best practice for documenting the architecture of software-intensive systems [IEEE 1471].

As described in Section 1.2, a software architecture comprises more than one software structure, each of which provides an engineering handle on different system qualities. A *view* is the specification of one or more of these structures, and documenting a software architecture, then, is a matter of documenting the relevant views and then documenting information that applies to more than one view [Clements 2002].

ANSI/IEEE 1471-2000 provides guidance for choosing the best set of views to document, by bringing stakeholder interests to bear. It prescribes defining a set of viewpoints to satisfy the stakeholder community. A viewpoint identifies the set of concerns to be addressed, and identifies the modeling techniques, evaluation techniques, consistency checking techniques, etc., used by any conforming view. A view, then, is a viewpoint applied to a system. It is a representation of a set of software elements, their properties, and the relationships among them that conform to a defining viewpoint. Together, the chosen set of views show the entire architecture and all of its relevant properties. A SAD contains the viewpoints, relevant views, and information that applies to more than one view to give a holistic description of the system.

The remainder of Section 1.5 defines the viewpoints used in this SAD. The following table summarizes the stakeholders in this project and the viewpoints that have been included to address their concerns.

Table 1: Stakeholders and Relevant Viewpoints

| Stakeholder | Viewpoint(s) that apply to that class of stakeholder’s concerns |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

### <Insert name of viewpoint> Viewpoint Definition

|  |
| --- |
| There will be one of these subsections for each viewpoint defined. The subsections are as follows:   * Abstract: A brief overview of the viewpoint * Stakeholders and their concerns addressed: This section describes the stakeholders and their concerns that this viewpoint is intended to address. Listed are questions that can be answered by consulting views that conform to this viewpoint. Optionally, the section includes significant questions that cannot be answered by consulting views conforming to this viewpoint. * Elements, relations, properties, and constraints: This section defines the types of elements, the relations among them, the significant properties they exhibit, and the constraints they obey for views conforming to this viewpoint. * Language(s) to model/represent conforming views: This section lists the language or languages that will be used to model or represent views conforming to this viewpoint, and cite a definition document for each. * Applicable evaluation/analysis techniques and consistency/completeness criteria: This section describes rules for consistency and completeness that apply to views in this viewpoint, as well as any analysis of evaluation techniques that apply to the view that can be used to predict qualities of the system whose architecture is being specified. * Viewpoint source: This section provides a citation for the source of this viewpoint definition, if any.   Following is an example of a viewpoint definition.  Vie1.5.1 **Module decomposition viewpoint definition**  1.5.1.1 Abstract. Views conforming to the module decomposition viewpoint partition the system into a unique non-overlapping set of hierarchically decomposable implementation units (*modules*).  1.5.1.2 Stakeholders and Their Concerns Addressed. Stakeholders and their concerns addressed by this viewpoint include   * project managers, who must define work assignments, form teams, and formulate project plans and budgets and schedules; * COTS specialists, who need to have software elements defined as units of functionality, so they can search the marketplace and perform trade studies to find suitable COTS candidates; * testers and integrators who use the modules as their unit of work; * configuration management specialists who are in charge of maintaining current and past versions of the elements; * system build engineers who use the elements to produce a running version of the system; * maintainers, who are tasked with modifying the software elements; * implementers, who are required to implement the elements; * software architects for those software elements sufficiently large or complex enough to warrant their own software architectures; * the customer, who is concerned that projected changes to the system over its lifetime can be made economically by confining the effects of each change to a small number of elements.   1.5.1.3 Elements, Relations, Properties, and Constraints. Elements of the module decomposition viewpoint are modules, which are units of implementation that provide defined functionality. Modules are hierarchically decomposable; hence, the relation is “is-part-of.” Properties of elements include their names, the functionality assigned to them (including a statement of the quality attributes associated with that functionality), and their software-to-software interfaces. The module properties may include requirements allocation, supporting requirements traceability.  1.5.1.4 Language(s) to Model/Represent Conforming Views. Views conforming to the module decomposition viewpoint may be represented by (a) plain text using indentation or outline form [Clements 2002]; (b) UML, using subsystems or classes to represent elements and “is part of” or nesting to represent the decomposition relation.  1.5.1.5 Applicable Evaluation/Analysis Techniques and Consistency/Completeness Criteria. Completeness/consistency criteria include (a) no element has more than one parent; (b) major functionality is provided for by exactly one element; (c) the union of all elements’ functionality covers the requirements for the system; (d) every piece of source code can be mapped to an element in the module decomposition view (if not, the view is not complete); (e) the selection of module aligns with current and proposed procurement decisions. Additional consistency/completeness criteria apply to the specifications of the elements’ interfaces. Applicable evaluation/analysis techniques include (a) scenario-based evaluation techniques such as ATAM [Clements 2001] to assure that projected changes are supported economically by the decomposition; (b) disciplined and detailed mapping to requirements to assure coverage and non-overlapping functionality; (c) cost-based techniques that determine the number and composition of modules for efficient procurement.  1.5.1.6 Viewpoint Source. [Clements 2002, Section 2.1] describes the module decomposition style, which corresponds in large measure to this viewpoint. |

#### Abstract

#### Stakeholders and Their Concerns Addressed

#### Elements, Relations, Properties, and Constraints

#### Language(s) to Model/Represent Conforming Views

#### Applicable Evaluation/Analysis Techniques and Consistency/Completeness Criteria

#### Viewpoint Source

## How a View is Documented

|  |
| --- |
| **CONTENTS OF THIS SECTION**: This section describes how the documentation for a view is structured and organized. If you change the *organization* of information in Section 3, then you should also change its description in here. Otherwise, this section is all boilerplate.  If you choose to document all information in a view in a single presentation, then you will not need view packets. In that case, the template is as follows:   * Section 3.i: Name of view * Section 3.i.1: View description * Section 3.i.2: Primary presentation. This section presents the elements and the relations among them that populate this view packet, using an appropriate language, languages, notation, or tool-based representation. * Section 3.i.3: Element catalog. Whereas the primary presentation shows the important elements and relations of the view packet, this section provides additional information needed to complete the architectural picture. It consists of subsections for (respectively) elements, relations, interfaces, behavior, and constraints. * Section 3.i.4: Context diagram. This section provides a context diagram showing the context of the part of the system represented by this view packet. It also designates the view packet’s scope with a distinguished symbol, and shows interactions with external entities in the vocabulary of the view. * Section 3.i.5: Variability mechanisms. This section describes any variabilities that are available in the portion of the system shown in the view packet, along with how and when those mechanisms may be exercised. * Section 3.i.6: Architecture background. This section provides rationale for any significant design decisions whose scope is limited to this view packet. |

Section 3 of this SAD contains one view for each viewpoint listed in Section 1.5. Each view is documented as a set of view packets. A view packet is the smallest bundle of architectural documentation that might be given to an individual stakeholder.

Each view is documented as follows, where the letter *i* stands for the number of the view: 1, 2, etc.:

* Section 3.i: Name of view.
* Section 3.i.1: View description. This section describes the purpose and contents of the view. It should refer to (and match) the viewpoint description in Section 1.5 to which this view conforms.
* Section 3.i.2: View packet overview. This section shows the set of view packets in this view, and provides rationale that explains why the chosen set is complete and non-duplicative. The set of view packets may be listed textually, or shown graphically in terms of how they partition the entire architecture being shown in the view.
* Section 3.i.3: Architecture background. Whereas the architecture background of Section 2 pertains to those constraints and decisions whose scope is the entire architecture, this section provides any architecture background (including significant driving requirements, design approaches, patterns, analysis results, and requirements coverage) that applies to this view.
* Section 3.i.4: Variability mechanisms. This section describes any architectural variability mechanisms (e.g., adaptation data, compile-time parameters, variable replication, and so forth) described by this view, including a description of how and when those mechanisms may be exercised and any constraints on their use.
* Section 3.i.5: View packets. This section presents all of the view packets given for this view. Each view packet is described using the following outline, where the letter *j* stands for the number of the view packet being described: 1, 2, etc.
* Section 3.i.5.j: View packet #j.
* Section 3.i.5.j.1: Primary presentation. This section presents the elements and the relations among them that populate this view packet, using an appropriate language, languages, notation, or tool-based representation.
* Section 3.i.5.j.2: Element catalog. Whereas the primary presentation shows the important elements and relations of the view packet, this section provides additional information needed to complete the architectural picture. It consists of the following subsections:
* Section 3.i.5.j.2.1: Elements.This section describes each element shown in the primary presentation, details its responsibilities of each element, and specifies values of the elements’ relevant *properties*, which are defined in the viewpoint to which this view conforms.
* Section 3.i.5.j.2.2: Relations.This section describes any additional relations among elements shown in the primary presentation, or specializations or restrictions on the relations shown in the primary presentation.
* Section 3.i.5.j.2.3: Interfaces.This section specifies the software interfaces to any elements shown in the primary presentation that must be visible to other elements.
* Section 3.i.5.j.2.4: Behavior. This section specifies any significant behavior of elements or groups of interacting elements shown in the primary presentation.
* Section 3.i.5.j.2.5: Constraints: This section lists any constraints on elements or relations not otherwise described.
* Section 3.i.5.j.3: Context diagram. This section provides a context diagram showing the context of the part of the system represented by this view packet. It also designates the view packet’s scope with a distinguished symbol, and shows interactions with external entities in the vocabulary of the view.
* Section 3.i.5.j.4: Variability mechanisms. This section describes any variabilities that are available in the portion of the system shown in the view packet, along with how and when those mechanisms may be exercised.
* Section 3.i.5.j.5: Architecture background. This section provides rationale for any significant design decisions whose scope is limited to this view packet.
* Section 3.i.5.j.6: Relation to other view packets. This section provides references for related view packets, including the parent, children, and siblings of this view packet. Related view packets may be in the same view or in different views.

## Relationship to Other SADs

|  |
| --- |
| **CONTENTS OF THIS SECTION**: This section describes the relationship between this SAD and other architecture documents, both system and software. For example, a large project may choose to have one SAD that defines the system-of-systems architecture, and other SADs to define the architecture of systems or subsystems. An embedded system may well have a *system* architecture document, in which case this section would explain how the information in here traces to information there.  If none, say “Not applicable.” |

## Process for Updating this SAD

|  |
| --- |
| **CONTENTS OF THIS SECTION**: This section describes the process a reader should follow to report discrepancies, errors, inconsistencies, or omissions from this SAD. The section also includes necessary contact information for submitting the report. If a form is required, either a copy of the blank form that may be photocopied is included, or a reference to an online version is provided. This section also describes how error reports are handled, and how and when a submitter will be notified of the issue’s disposition. |

# Architecture Background

## Problem Background

|  |
| --- |
| **CONTENTS OF THIS SECTION**: The sub-parts of Section 2.1 explain the constraints that provided the significant influence over the architecture. |

### System Overview

|  |
| --- |
| **CONTENTS OF THIS SECTION**: This section describes the general function and purpose for the system or subsystem whose architecture is described in this SAD. |

### Goals and Context

|  |
| --- |
| **CONTENTS OF THIS SECTION**: This section describes the goals and major contextual factors for the software architecture. The section includes a description of the role software architecture plays in the life cycle, the relationship to system engineering results and artifacts, and any other relevant factors. |

### Significant Driving Requirements

|  |
| --- |
| **CONTENTS OF THIS SECTION**: This section describes behavioral and quality attribute requirements (original or derived) that shaped the software architecture. Included are any scenarios that express driving behavioral and quality attribute goals, such as those crafted during a Quality Attribute Workshop (QAW) [Barbacci 2003] or software architecture evaluation using the Architecture Tradeoff Analysis Method[[1]](#footnote-1)SM (ATAMSM) [Bass 2003]. |

## Solution Background

|  |
| --- |
| **CONTENTS OF THIS SECTION**: The sub-parts of Section 2.2 provide a description of why the architecture is the way that it is, and a convincing argument that the architecture is the right one to satisfy the behavioral and quality attribute goals levied upon it. |

### Architectural Approaches

|  |
| --- |
| **CONTENTS OF THIS SECTION**: This section provides a rationale for the major design decisions embodied by the software architecture. It describes any design approaches applied to the software architecture, including the use of architectural styles or design patterns, when the scope of those approaches transcends any single architectural view. The section also provides a rationale for the selection of those approaches. It also describes any significant alternatives that were seriously considered and why they were ultimately rejected. The section describes any relevant COTS issues, including any associated trade studies. |

### Analysis Results

|  |
| --- |
| **CONTENTS OF THIS SECTION**: This section describes the results of any quantitative or qualitative analyses that have been performed that provide evidence that the software architecture is fit for purpose. If an Architecture Tradeoff Analysis Method evaluation has been performed, it is included in the analysis sections of its final report. This section refers to the results of any other relevant trade studies, quantitative modeling, or other analysis results. |

### Requirements Coverage

|  |
| --- |
| **CONTENTS OF THIS SECTION**: This section describes the requirements (original or derived) addressed by the software architecture, with a short statement about where in the architecture each requirement is addressed. |

### Summary of Background Changes Reflected in Current Version

|  |
| --- |
| **CONTENTS OF THIS SECTION**: For versions of the SAD after the original release, this section summarizes the actions, decisions, decision drivers, analysis and trade study results that became decision drivers, requirements changes that became decision drivers, and how these decisions have caused the architecture to evolve or change. |

## Product Line Reuse Considerations

|  |
| --- |
| **CONTENTS OF THIS SECTION**: When a software product line is being developed, this section details how the software covered by this SAD is planned or expected to be reused in order to support the product line vision. In particular, this section includes a complete list of the variations that are planned to be produced and supported. "Variation" refers to a variant of the software produced through the use of pre-planned variation mechanisms made available in the software architecture. It may refer to a variant of one of the modules identified in this SAD, or a collection of modules, or the entire system or subsystem covered by this SAD. For each variation, the section identifies the increment(s) of the software build in which (a) the variation will be available; and (b) the variation will be used. Finally, this section describes any additional potential that exists to reuse one or more of the modules or their identified variations, even if this reuse is not currently planned for any increment. |

# Views

|  |
| --- |
| **CONTENTS OF THIS SECTION**: The sub-parts of Section 3 specify the views corresponding to the viewpoints listed in Section 1.5. |

This section contains the views of the software architecture. A view is a representation of a whole system from the perspective of a related set of concerns [IEEE 1471]. Concretely, a view shows a particular type of software architectural elements that occur in a system, their properties, and the relations among them. A view conforms to a defining viewpoint.

Architectural views can be divided into three groups, depending on the broad nature of the elements they show. These are:

* Module views. Here, the elements are modules, which are units of implementation. Modules represent a code-based way of considering the system. Modules are assigned areas of functional responsibility, and are assigned to teams for implementation. There is less emphasis on how the resulting software manifests itself at runtime. Module structures allow us to answer questions such as: What is the primary functional responsibility assigned to each module? What other software elements is a module allowed to use? What other software does it actually use? What modules are related to other modules by generalization or specialization (i.e., inheritance) relationships?
* Component-and-connector views. Here, the elements are runtime components (which are principal units of computation) and connectors (which are the communication vehicles among components). Component and connector structures help answer questions such as: What are the major executing components and how do they interact? What are the major shared data stores? Which parts of the system are replicated? How does data progress through the system? What parts of the system can run in parallel? How can the system’s structure change as it executes?
* Allocation views. These views show the relationship between the software elements and elements in one or more external environments in which the software is created and executed. Allocation structures answer questions such as: What processor does each software element execute on? In what files is each element stored during development, testing, and system building? What is the assignment of the software element to development teams?

These three kinds of structures correspond to the three broad kinds of decisions that architectural design involves:

* How is the system to be structured as a set of code units (modules)
* How is the system to be structured as a set of elements that have run-time behavior (components) and interactions (connectors) ?
* How is the system to relate to non-software structures in its environment (such as CPUs, file systems, networks, development teams, etc.)?

Often, a view shows information from more than one of these categories. However, unless chosen carefully, the information in such a hybrid view can be confusing and not well understood.

The views presented in this SAD are the following:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Name of view** | **Viewtype that defines this view** | **Types of elements and relations shown** | | **Is this a module view?** | **Is this a component-and-connector view?** | **Is this an allocation view?** |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

## <Insert view name> View

|  |
| --- |
| **CONTENTS OF THIS SECTION**: For each view documented in this SAD, the sub-parts of Section 3.1 specify it using the outline given in Section 1.6. This part of the template assumes you are using view packets to divide up a view into management chunks. If not, then see the note in Section 1.6 as to what outline to use for each view. |

### View Description

### View Packet Overview

This view has been divided into the following view packets for convenience of presentation:

<<list, table, or diagram>>

### Architecture Background

### Variability Mechanisms

### View Packets

|  |
| --- |
| **CONTENTS OF THIS SECTION**: For each view packet in the view, this section describes it using the outline given in Section 1.6. |

#### View packet # j

##### Primary Presentation

##### Element Catalog

###### Elements

###### Relations

###### Interfaces

###### Behavior

###### Constraints

##### Context Diagram

##### Variability Mechanisms

##### Architecture Background

##### Related View Packets

# Relations Among Views

Each of the views specified in Section 3 provides a different perspective and design handle on a system, and each is valid and useful in its own right. Although the views give different system perspectives, they are not independent. Elements of one view will be related to elements of other views, and we need to reason about these relations. For example, a module in a decomposition view may be man­ifested as one, part of one, or several components in one of the component-and-connector views, reflecting its runtime alter-ego. In general, mappings between views are many to many. Section 4 describes the relations that exist among the views given in Section 3. As required by ANSI/IEEE 1471-2000, it also describes any known inconsistencies among the views.

## General Relations Among Views

|  |
| --- |
| **CONTENTS OF THIS SECTION**: This section describes the general relationship among the views chosen to represent the architecture. Also in this section, consistency among those views is discussed and any known inconsistencies are identified. |

## View-to-View Relations

|  |
| --- |
| **CONTENTS OF THIS SECTION**: For each set of views related to each other, this section shows how the elements in one view are related to elements in another. |

# Referenced Materials

|  |
| --- |
| **CONTENTS OF THIS SECTION**: This section provides citations for each reference document. Provide enough information so that a reader of the SAD can be reasonably expected to locate the document. |

|  |  |
| --- | --- |
| Barbacci 2003 | Barbacci, M.; Ellison, R.; Lattanze, A.; Stafford, J.; Weinstock, C.; & Wood, W. *Quality Attribute Workshops (QAWs)*, Third Edition (CMU/SEI-2003-TR-016). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003. <http://www.sei.cmu.edu/publications/documents/03.reports/03tr016.html>. |
| Bass 2003 | Bass, Clements, Kazman, *Software Architecture in Practice,* second edition, Addison Wesley Longman, 2003. |
| Clements 2001 | Clements, Kazman, Klein, *Evaluating Software Architectures: Methods and Case Studies,* Addison Wesley Longman, 2001. |
| Clements 2002 | Clements, Bachmann, Bass, Garlan, Ivers, Little, Nord, Stafford, *Documenting Software Architectures: Views and Beyond*, Addison Wesley Longman, 2002. |
| IEEE 1471 | ANSI/IEEE-1471-2000, *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*, 21 September 2000. |

# Directory

## Index

|  |
| --- |
| **CONTENTS OF THIS SECTION**: This section provides an index of all element names, relation names, and property names. For each entry, the following are identified:  the location in the SAD where it was defined  each place it was used  Ideally, each entry will be a hyperlink so a reader can instantly navigate to the indicated location. |

## Glossary

|  |
| --- |
| **CONTENTS OF THIS SECTION**: This section provides a list of definitions of special terms and acronyms used in the SAD. If terms are used in the SAD that are also used in a parent SAD and the definition is different, this section explains why. |

|  |  |
| --- | --- |
| Term | Definition |
| software architecture | The structure or structures of that system, which comprise software elements, the externally visible properties of those elements, and the relationships among them [Bass 2003]. "Externally visible” properties refer to those assumptions other elements can make of an element, such as its provided services, performance characteris­tics, fault handling, shared resource usage, and so on. |
| view | A representation of a whole system from the perspective of a related set of concerns [IEEE 1471]. A representation of a particular type of software architectural elements that occur in a system, their properties, and the relations among them. A view conforms to a defining viewpoint. |
| view packet | The smallest package of architectural documentation that could usefully be given to a stakeholder. The documentation of a view is composed of one or more view packets. |
| viewpoint | A specification of the conventions for constructing and using a view; a pattern or template from which to develop individual views by establishing the purposes and audience for a view, and the techniques for its creation and analysis [IEEE 1471]. Identifies the set of concerns to be addressed, and identifies the modeling techniques, evaluation techniques, consistency checking techniques, etc., used by any conforming view. |

## Acronym List

|  |  |
| --- | --- |
| API | Application Programming Interface; Application Program Interface; Application Programmer Interface |
| ATAM | Architecture Tradeoff Analysis Method |
| CMM | Capability Maturity Model |
| CMMI | Capability Maturity Model Integration |
| CORBA | Common object request broker architecture |
| COTS | Commercial-Off-The-Shelf |
| EPIC | Evolutionary Process for Integrating COTS-Based Systems |
| IEEE | Institute of Electrical and Electronics Engineers |
| KPA | Key Process Area |
| OO | Object Oriented |
| ORB | Object Request Broker |
| OS | Operating System |
| QAW | Quality Attribute Workshop |
| RUP | Rational Unified Process |
| SAD | Software Architecture Document |
| SDE | Software Development Environment |
| SEE | Software Engineering Environment |
| SEI | Software Engineering Institute  Systems Engineering & Integration  Software End Item |
| SEPG | Software Engineering Process Group |
| SLOC | Source Lines of Code |
| SW-CMM | Capability Maturity Model for Software |
| CMMI-SW | Capability Maturity Model Integrated - includes Software Engineering |
| UML | Unified Modeling Language |

# Sample Figures & Tables



Figure 1: Sample Figure

Table 2: Sample Table

| Table Heading | Table Heading | Table Heading | Table Heading |
| --- | --- | --- | --- |
| Table Body | Table Body | Table Body | Table Body |
| Table Body | Table Body | Table Body | Table Body |
| Table Body | Table Body | Table Body | Table Body |
| Table Body | Table Body | Table Body | Table Body |

1. Appendices

|  |
| --- |
| **CONTENTS OF THIS SECTION**: Appendices may be used to provide information published separately for convenience in document maintenance (e.g., charts, classified data, API specification). As applicable, each appendix is referenced in the main body of the document where the data would normally have been provided. Appendices may be bound as separate documents for ease in handling. If your SAD has no appendices, delete this page. |

Heading 2 - Appendix

Heading 2 - Appendix

1. SM Quality Attribute Workshop and QAW and Architecture Tradeoff Analysis Method and ATAM are service marks of Carnegie Mellon University. [↑](#footnote-ref-1)