1 Stakeholders and Their Documentation Needs

The central idea of this section is that you do not select architecture views by taste or by a fixed recipe. Instead, you *systematically* choose, combine, and stage views based on:

- the **stakeholders** who must use the documentation,
- the **project constraints** (budget, schedule, skills, standards),
- and the **structures actually present** in the architecture.

The following subsections summarize the main stakeholder roles and the kinds of documentation each typically needs.

1.1 Project Managers

Concerns. Schedule, resource assignment, risk, contingency plans, the system's purpose and constraints, and how the system interacts with hardware and external systems.

Documentation they need.

- **Module views:** decomposition (modules to implement, responsibilities, complexity hints); uses and/or layered views (dependencies and likely implementation order).
- **Allocation views:** deployment (hardware, environments, external systems); work assignment (which organization or team builds what).
- **Other:** top-level context diagrams and system overview material that supports planning and communication.

Pattern. Documentation that lets them build and maintain a credible project plan and coordinate organizations.

1.2 Members of the Development Team

Concerns. What they are building, how responsibilities are partitioned, constraints, data model, interfaces, and opportunities for reuse.

Documentation they need.

- **Module views:** decomposition (where their element fits), uses/layered (dependencies and protocols), and generalization (inheritance and specialization structure).
- **C&C views:** structures involving their components and their neighbors at runtime.
- **Allocation views:** deployment (where the code runs), implementation (files, repositories, directories), and install (how the software is packaged and delivered).
- **Other:** interface documentation for their elements and those they interact with; variability guides for configurable or product-line systems; rationale and constraints that explain trade-offs.

Pattern. Enough detail to implement correctly and predict the consequences of design choices.

1.3 Testers and Integrators

Concerns. Correctness and fit of the pieces, incremental integration strategy, and where to find build artifacts, interfaces, and constraints.

Documentation they need.

- **Module views:** decomposition (components to test and integrate), uses (dependency chains that affect test order), and the data model.
- **C&C views:** all relevant runtime structures and interactions.
- **Allocation views:** deployment (where elements run), install (how to assemble a testable configuration), and implementation (where to find code and artifacts).
- **Other:** interface and behavioral specifications for modules under test and their neighbors; context diagrams focused on the test boundary.

Pattern. Similar to developers, but with extra emphasis on behavior, integration paths, and observable interfaces.

1.4 Designers of Other Systems

Concerns. Interoperability: what services are provided and required, protocols, and data models at system boundaries.

Documentation they need.

- Precise **interface specifications** of the elements they will interact with.
- The relevant **data model** of the system they integrate with.
- Top-level **context diagrams** showing external interactions.

Pattern. They do not need the entire architecture, only the parts that form the system-to-system contracts.

1.5 Maintainers

Concerns. Understanding the impact of changes, localizing modifications, and preserving original intent and rationale.

Documentation they need.

- **Module views:** decomposition, layered structure, and data model (to understand where responsibilities and state live).
- **C&C views:** runtime structures for impact analysis.
- **Allocation views:** deployment, implementation, and install views to understand operational consequences of changes.
- **Other:** rationale and constraints explaining why the system is the way it is.

Pattern. Developers' needs plus support for impact analysis and reconstruction of design intent.

1.6 Product-Line Application Builders

Concerns. Building products in a software product line by tailoring core assets, instantiating variability, and adding product-specific code.

Documentation they need.

- Everything integrators need for understanding components and deployment.
- **Variability guides** (in module and/or C&C views) that show where and how to customize or configure.

Pattern. Integrator-level information plus clear documentation of product-line variation mechanisms.

1.7 Customers

Concerns. Cost, progress, and confidence that the architecture satisfies required qualities and functionality, as well as fit with their environment.

Documentation they need.

- **C&C views** that support analysis results (performance, reliability, and other qualities).
- **Allocation views:** deployment view (fit with hardware and environment); work-assignment view in filtered form.
- High-level **context diagrams** that illustrate scope, external interfaces, and key responsibilities.

Pattern. High-level assurances and evidence of soundness rather than detailed design artifacts.

1.8 End Users

Concerns. How the system behaves from a user's perspective and whether performance and reliability will be acceptable.

Documentation they may use.

- Selected **C&C views** that emphasize flow of control and data transformation from inputs to outputs.
- **Analysis results** related to performance, reliability, and usability.
- **Deployment views** that show where functionality resides on platforms they interact with.

Pattern. Architecture diagrams are mainly helpful when they clarify behavior and usability implications.

1.9 Analysts (Quality Attribute Specialists)

Concerns. Evaluating the architecture against specific quality attributes such as performance, accuracy, modifiability, security, availability, and usability.

Documentation they need (general).

- **Module views** (especially decomposition and uses) for understanding structure and dependencies.
- **C&C views** showing processes, communication paths, and data flow.
- **Allocation views** (especially deployment) to understand nodes, networks, and redundancy.

Attribute-specific examples.

- *Performance:* communicating-processes C&C view, deployment view, and behavioral documentation with timing and concurrency.
- *Accuracy:* C&C and data-flow views that show where numerical or algorithmic errors can accumulate.
- *Modifiability:* uses and decomposition views for change-impact analysis; C&C views to expose runtime side effects.
- *Security:* deployment and context views (external connections); C&C and decomposition views highlighting where security controls are applied.
- *Availability:* communicating-processes and deployment views for identifying failure modes and redundancy.

- *Usability:* decomposition and C&C views that show how user-visible state and error handling are managed.

Pattern. Analysts treat architecture documentation as a model to feed formal or semi-formal analyses.

1.10 Infrastructure Support Personnel

Concerns. Setting up and maintaining development, build, and production infrastructure: environments, continuous integration and delivery, source control management, and configuration.

Documentation they need.

- **Module views** (decomposition and uses) describing major artifacts and dependencies.
- **C&C views** indicating what runs where and how components interact at runtime.
- **Allocation views:** deployment, install, and implementation views, which map software assets to infrastructure.
- **Other:** variability guides used for environment- or configuration-specific setups.

Pattern. An operational view of software assets and their mapping to infrastructure components.

1.11 New Stakeholders

Concerns. Onboarding: understanding the system at a high level before diving into role-specific details.

Documentation they need.

- Introductory, broad-scope material: top-level context diagrams, architectural constraints, overall rationale, and root-level views.
- The same kinds of views as their ultimate role requires, but initially at lower levels of detail.

1.12 Current and Future Architects

Concerns. Design decisions, rationale, trade-offs, and the complete architectural history of the system.

Documentation they need.

- Essentially *all* views in substantial detail.
- Rationale and constraints for all major decisions.
- Traceability between requirements, architectural choices, and quality-attribute evaluations.

Pattern. They are the most demanding readers: documentation should enable them to evolve and re-evaluate the architecture over time.

1.13 Stakeholder–View Table (Filled Version)

The qualitative mapping above can be summarized in a stakeholder–view table. Entries use the following key:

Key: d = detailed information, s = some details, o = overview information, x = anything.

	Module Views					C&C Views	Allocation Views				Other Documentation					
Stakeholder	Decomposition	Uses	Generalization	Layered	Data Model	Various	Deployment	Implementation	Install	Work Assignment	Interface Documentation	Context Diagrams	Mapping Between Views	Variability Guides	Analysis Results	Rationale and Constraints
Project managers	s	s	s	d	d	d	o	s	s	d		o				
Members of development team	d	d	d	d	d	d	s	s	s	d	d	d	d	s		d
Testers and integrators	d	d	d	d	d	s	s	s	s	d	d	s	d	s		d
Designers of other systems					d	s					d	d				
Maintainers	d	d	d	d	d	d	s	s	s	d	d	d	d	d		d
Product-line application builders	d	d	s	o	o	s	s	s	s	s	d	s	d	d		d
Customers	o	o	o	s	s	s	s			o		d			s	
End users						s	s					s			s	
Analysts	d	d	s	d	d	d	d				s	d	s		d	
Infrastructure support personnel	s	s	s	d	d	s	d	d	d					s		
New stakeholders	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Current and future architects	d	d	d	d	d	d	d	s	s	s	d	d	d	d	d	d

1.14 Stakeholder–View Table Template (Empty Version)

For use on new projects, an empty version of the same table can be used as a worksheet. Cells can be filled in after interviewing stakeholders about their information needs.

Key: d = detailed information, s = some details, o = overview information, x = anything.

	Module Views					C&C Views	Allocation Views				Other Documentation						
Stakeholder	Decomposition	Uses	Generalization	Layered	Data Model	Various	Deployment	Implementation	Install	Work Assignment	Interface Documentation	Context Diagrams	Mapping Between Views	Variability Guides	Analysis Results	Rationale and Constraints
Project managers																
Members of development team																
Testers and integrators																
Designers of other systems																
Maintainers																
Product-line application builders																
Customers																
End users																
Analysts																
Infrastructure support personnel																
New stakeholders																
Current and future architects																