Stakeholders and Their Documentation Needs

A Comprehensive Reference for Audience-Driven Architecture Documentation

Contents

1 Overview

Architecture documentation exists to serve stakeholders. The central idea of stakeholder-driven documentation 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.

Different stakeholders have different concerns, different levels of technical sophistication, different needs for detail, and different purposes for using documentation. A project manager needs different information than a developer. A security analyst needs different views than a maintainer. Understanding these differences enables architects to create documentation that serves its intended audience effectively.

This stakeholder-driven approach ensures that documentation effort is invested where it provides value. Rather than documenting everything exhaustively or following a rigid template, architects identify who will use the documentation, what they need from it, and how detailed it must be. This approach produces documentation that is used rather than documentation that sits on a shelf.

1.1 Scope and Applicability

Stakeholder analysis for documentation applies to any software project where architecture documentation is created. This includes enterprise systems with diverse stakeholders from business and technical domains; safety-critical systems where regulators and certifiers are key stakeholders; product lines where application builders need variability documentation; open-source projects where contributors and users have documentation needs; contracted development where customers and contractors have different perspectives; legacy modernization where maintainers need to understand existing systems; and agile projects where lightweight but targeted documentation serves specific needs.

The approach is particularly valuable when documentation resources are limited and must be prioritized, when stakeholders have diverse and potentially conflicting needs, when documentation must support the full software lifecycle, when projects involve multiple organizations or teams, and when documentation quality directly affects project success.

1.2 Historical Context

The stakeholder-driven approach to architecture documentation emerged from recognition that traditional documentation approaches often failed.

Early documentation practices often followed rigid templates, producing comprehensive but unused documents. Documentation was created because standards required it, not because stakeholders needed it.

The IEEE 1471 standard (2000, later ISO/IEC 42010) introduced the concept of stakeholders and concerns as fundamental to architecture description. This standard established that architecture documentation should address stakeholder concerns through appropriate views.

The "Views and Beyond" approach, developed at the Software Engineering Institute, operationalized stakeholder-driven documentation with practical guidance for selecting and combining views based on stakeholder needs.

Agile methods challenged heavyweight documentation, leading to emphasis on "just enough" documentation that serves actual needs. This reinforced the importance of understanding who needs what documentation.

Modern DevOps practices extend stakeholder analysis to operations, recognizing that deployment, monitoring, and incident response require architectural understanding.

1.3 Relationship to Architecture Documentation

Stakeholder analysis is the starting point for architecture documentation planning.

View selection depends on stakeholder needs. The views included in documentation should address stakeholder concerns. Views that no stakeholder needs should not be created.

Detail level depends on stakeholder sophistication. Some stakeholders need detailed specifications; others need high-level overviews. The same view might be documented at different levels for different audiences.

Documentation format depends on stakeholder preferences. Some stakeholders prefer graphical representations; others prefer text. Some need formal specifications; others need informal descriptions.

Documentation timing depends on stakeholder activities. Some stakeholders need documentation early for planning; others need it later for implementation or maintenance.

1.4 The Documentation Planning Process

Stakeholder-driven documentation follows a systematic process.

First, identify stakeholders—who will use the documentation and for what purposes. This includes both current stakeholders and anticipated future stakeholders.

Second, characterize stakeholder needs—what concerns each stakeholder has, what questions they need answered, what decisions they need to make.

Third, map needs to views—which architectural views address which stakeholder concerns. Multiple stakeholders may need the same view; one stakeholder may need multiple views.

Fourth, determine appropriate detail—how much detail each stakeholder needs for each view. Not all stakeholders need all details.

Fifth, plan documentation production—when to create each view, at what level of detail, and in what format.

Sixth, validate with stakeholders—confirm that planned documentation meets actual needs. Adjust based on feedback.

2 Understanding Stakeholders

Effective stakeholder-driven documentation requires deep understanding of who stakeholders are and what they need.

2.1 What Is a Stakeholder?

A stakeholder is anyone who has an interest in or concern about the system and its documentation. Stakeholders may be individuals, groups, or organizations.

2.1.1 Direct and Indirect Stakeholders

Direct stakeholders interact with the documentation themselves. They read, review, or use the documentation for their work.

Indirect stakeholders are affected by decisions made using the documentation but may not read it themselves. Their interests should still be considered.

2.1.2 Internal and External Stakeholders

Internal stakeholders are part of the development organization. They include developers, architects, managers, and testers within the team.

External stakeholders are outside the development organization. They include customers, end users, regulators, and partner organizations.

2.1.3 Current and Future Stakeholders

Current stakeholders need documentation now for ongoing activities.

Future stakeholders will need documentation later. Maintainers who will support the system years hence are future stakeholders whose needs should be anticipated.

2.2 Stakeholder Concerns

Stakeholders have concerns—interests, needs, or questions—that documentation should address.

2.2.1 Types of Concerns

Functional concerns relate to what the system does. What capabilities does it provide? How does it process data? What are its inputs and outputs?

Quality concerns relate to how well the system performs. Is it fast enough? Reliable enough? Secure enough? Modifiable enough?

Development concerns relate to building the system. How is work organized? What are the dependencies? What must be built first?

Operational concerns relate to running the system. How is it deployed? How is it monitored? How are problems diagnosed?

Business concerns relate to organizational impact. What does it cost? What value does it provide? What are the risks?

2.2.2 Concern Priorities

Not all concerns are equally important to each stakeholder. Understanding priorities helps focus documentation effort.

Primary concerns are central to the stakeholder's role and must be thoroughly addressed.

Secondary concerns are relevant but less critical. They may be addressed with less detail.

Peripheral concerns are of minor interest. They may be addressed briefly or not at all for that stakeholder.

2.3 Stakeholder Identification

Identifying stakeholders requires systematic effort.

2.3.1 Role-Based Identification

Consider standard roles: who manages the project? Who develops the code? Who tests it? Who operates it? Who uses it? Who pays for it?

The stakeholder categories in this document provide a starting checklist, but each project may have additional or different roles.

2.3.2 Organization-Based Identification

Consider organizations involved: the development team, operations, customer organizations, partner organizations, regulatory bodies, standards organizations.

Each organization may have multiple stakeholder roles within it.

2.3.3 Lifecycle-Based Identification

Consider lifecycle phases: who is involved in requirements? Design? Implementation? Testing? Deployment? Operations? Maintenance? Retirement?

Different phases involve different stakeholders.

2.3.4 Stakeholder Discovery Techniques

Interview project participants about who needs documentation.

Review project organization charts and role definitions.

Examine similar past projects for stakeholder patterns.

Consider regulatory and contractual requirements that imply stakeholders.

2.4 Stakeholder Prioritization

When resources are limited, stakeholders must be prioritized.

2.4.1 Prioritization Factors

Influence considers how much the stakeholder affects project success. High-influence stakeholders' needs take priority.

Urgency considers when the stakeholder needs documentation. Immediate needs take priority over future needs.

Legitimacy considers the stakeholder's right to documentation. Contractual or regulatory requirements establish legitimacy.

2.4.2 Prioritization Outcomes

Primary stakeholders receive full attention. Their documentation needs are thoroughly addressed.

Secondary stakeholders receive partial attention. Their needs are addressed where resources permit.

Tertiary stakeholders receive minimal attention. Their needs are noted but may not be fully addressed.

3 Stakeholder Categories

The following sections describe common stakeholder categories, their concerns, and their documentation needs. These categories represent typical roles; actual projects may have variations or additional roles.

3.1 Project Managers

Project managers are responsible for planning, coordinating, and controlling the development effort. They need documentation that supports project management activities.

3.1.1 Concerns

Project managers are concerned with schedule—when will components be complete and how do dependencies affect sequencing. They are concerned with resource assignment—who works on what and what skills are needed. They are concerned with risk—what could go wrong and what are contingency plans. They are concerned with the system's purpose and constraints—what are we building and why. They are concerned with how the system interacts with hardware and external systems—what is the deployment context.

3.1.2 Documentation Needs

Project managers need module views, particularly decomposition showing modules to implement, their responsibilities, and complexity hints that inform estimation. They need uses and layered views showing dependencies and likely implementation order for scheduling.

They need allocation views, particularly deployment showing hardware, environments, and external systems that must be coordinated. They need work assignment views showing which organization or team builds what for coordination.

They need other documentation including top-level context diagrams and system overview material that supports planning and communication with other stakeholders.

3.1.3 Documentation Pattern

Documentation that lets project managers build and maintain a credible project plan and coordinate organizations. Emphasis on structure, dependencies, and organizational mapping rather than technical detail.

3.1.4 Detail Level

Project managers typically need some details on module decomposition—enough to estimate and assign work. They need overview information on technical views that don't directly affect planning. They need detailed information on work assignment and deployment context.

3.1.5 Format Preferences

High-level diagrams that can be shared with diverse audiences. Tables mapping components to teams and schedules. Progress-trackable element lists.

3.2 Members of the Development Team

Developers implement the system. They need documentation that guides implementation and helps them understand the context of their work.

3.2.1 Concerns

Developers are concerned with what they are building—what is their assigned component's responsibility. They are concerned with how responsibilities are partitioned—what belongs in their component versus others. They are concerned with constraints—what rules must their implementation follow. They are concerned with data model—what data structures exist and how are they related. They are concerned with interfaces—how do they interact with other components. They are concerned with opportunities for reuse—what existing components can they leverage.

3.2.2 Documentation Needs

Developers need module views including decomposition showing where their element fits in the overall structure, uses and layered views showing dependencies and protocols they must follow, and generalization showing inheritance and specialization structure they must understand or implement.

They need component-and-connector views showing structures involving their components and their neighbors at runtime—how their code behaves in execution.

They need allocation views including deployment showing where the code runs, implementation showing files, repositories, and directories where code lives, and install showing how the software is packaged and delivered.

They need other documentation including interface documentation for their elements and those they interact with, variability guides for configurable or product-line systems, and rationale and constraints that explain trade-offs and design decisions.

3.2.3 Documentation Pattern

Enough detail to implement correctly and predict the consequences of design choices. Developers need to understand not just what to build but why and how it fits with the rest of the system.

3.2.4 Detail Level

Developers typically need detailed information on views relevant to their work. The level of detail may vary by experience—senior developers may need less explicit guidance than junior developers.

3.2.5 Format Preferences

Precise interface specifications. Code-level documentation in familiar formats. Diagrams showing context and interactions. Searchable reference documentation.

3.3 Testers and Integrators

Testers verify that components work correctly. Integrators combine components into working systems. Their documentation needs overlap significantly.

3.3.1 Concerns

Testers and integrators are concerned with correctness and fit of the pieces—do components work and work together. They are concerned with incremental integration strategy—what order should components be integrated. They are concerned with where to find build artifacts, interfaces, and constraints—practical information for their work.

3.3.2 Documentation Needs

They need module views including decomposition showing components to test and integrate, uses showing dependency chains that affect test order, and the data model for data-focused testing.

They need component-and-connector views showing all relevant runtime structures and interactions—what to test and how components should behave together.

They need allocation views including deployment showing where elements run for test environment setup, install showing how to assemble a testable configuration, and implementation showing where to find code and artifacts.

They need other documentation including interface and behavioral specifications for modules under test and their neighbors, and context diagrams focused on the test boundary.

3.3.3 Documentation Pattern

Similar to developers, but with extra emphasis on behavior, integration paths, and observable interfaces. Testers need to know expected behavior; integrators need to know assembly procedures.

3.3.4 Detail Level

Detailed information on interfaces and behavior. Detailed information on integration dependencies. Practical information on build and deployment procedures.

3.3.5 Format Preferences

Behavioral specifications that define expected results. Integration procedures with clear steps. Test environment configuration guides.

3.4 Designers of Other Systems

When systems must interoperate, designers of other systems need to understand the integration points.

3.4.1 Concerns

Designers of other systems are concerned with interoperability—how their system will connect with this one. They are concerned with what services are provided and required—what capabilities are available and what is expected. They are concerned with protocols—how communication occurs. They are concerned with data models at system boundaries—what data is exchanged and in what format.

3.4.2 Documentation Needs

They need precise interface specifications of the elements they will interact with—APIs, protocols, data formats, error handling.

They need the relevant data model of the system they integrate with—entities, attributes, relationships for shared data.

They need top-level context diagrams showing external interactions—where their system fits in the larger picture.

3.4.3 Documentation Pattern

They do not need the entire architecture, only the parts that form the system-to-system contracts. Internal structure is irrelevant; external interfaces are essential.

3.4.4 Detail Level

Detailed interface specifications. Overview of system context. No internal implementation details needed.

3.4.5 Format Preferences

Standard interface description formats (OpenAPI, WSDL, etc.). Protocol specifications. Data schema definitions. Example message exchanges.

3.5 Maintainers

Maintainers modify and extend the system after initial development. They may work on the system years after original developers have moved on.

3.5.1 Concerns

Maintainers are concerned with understanding the impact of changes—what will be affected by a modification. They are concerned with localizing modifications—where should changes be made. They are concerned with preserving original intent and rationale—why was the system designed this way.

3.5.2 Documentation Needs

They need module views including decomposition, layered structure, and data model to understand where responsibilities and state live.

They need component-and-connector views showing runtime structures for impact analysis—understanding how runtime behavior will change.

They need allocation views including deployment, implementation, and install views to understand operational consequences of changes.

They need other documentation including rationale and constraints explaining why the system is the way it is—essential for making changes that preserve design intent.

3.5.3 Documentation Pattern

Developers' needs plus support for impact analysis and reconstruction of design intent. Maintainers must understand not just what exists but why it exists.

3.5.4 Detail Level

Detailed information across most views. Maintainers may need to understand any part of the system, so comprehensive coverage is valuable.

3.5.5 Format Preferences

Navigable documentation that supports exploration. Cross-references between views. Rationale linked to design elements. Search capability for large systems.

3.6 Product-Line Application Builders

In software product lines, application builders create specific products by configuring and extending shared assets.

3.6.1 Concerns

Product-line application builders are concerned with building products by tailoring core assets—how to customize for specific products. They are concerned with instantiating variability—how to select among options. They are concerned with adding product-specific code—how to extend the platform.

3.6.2 Documentation Needs

They need everything integrators need for understanding components and deployment—they are building and assembling products.

They need variability guides in module and component-and-connector views that show where and how to customize or configure—the essential product-line documentation.

3.6.3 Documentation Pattern

Integrator-level information plus clear documentation of product-line variation mechanisms. The variation points and binding mechanisms are central.

3.6.4 Detail Level

Detailed information on variability mechanisms. Overview information on stable platform elements. Detailed information on extension points.

3.6.5 Format Preferences

Configuration guides. Variation point catalogs. Decision models. Product derivation procedures.

3.7 Customers

Customers procure or sponsor the system. They need confidence that their investment is sound.

3.7.1 Concerns

Customers are concerned with cost—is the project within budget. They are concerned with progress—is development on track. They are concerned with confidence that the architecture satisfies required qualities and functionality. They are concerned with fit with their environment—will the system work in their context.

3.7.2 Documentation Needs

They need component-and-connector views that support analysis results showing that performance, reliability, and other qualities will be achieved.

They need allocation views including deployment view showing fit with hardware and environment, and work-assignment view in filtered form showing organizational aspects.

They need high-level context diagrams that illustrate scope, external interfaces, and key responsibilities.

3.7.3 Documentation Pattern

High-level assurances and evidence of soundness rather than detailed design artifacts. Customers need confidence, not implementation details.

3.7.4 Detail Level

Overview information on most views. Some details on deployment and quality-relevant structures. Analysis results rather than raw architecture.

3.7.5 Format Preferences

Executive summaries. Analysis results and evaluations. High-level diagrams. Progress dashboards.

3.8 End Users

End users operate the system to accomplish their tasks. Their interaction with architecture documentation is limited but not nonexistent.

3.8.1 Concerns

End users are concerned with how the system behaves from a user's perspective—what can they do with it. They are concerned with whether performance and reliability will be acceptable—will it work well enough.

3.8.2 Documentation They May Use

Selected component-and-connector views that emphasize flow of control and data transformation from inputs to outputs—helping users understand what happens when they act.

Analysis results related to performance, reliability, and usability—evidence that the system will serve them well.

Deployment views that show where functionality resides on platforms they interact with—understanding the user-visible structure.

3.8.3 Documentation Pattern

Architecture diagrams are mainly helpful when they clarify behavior and usability implications. Most end users never see architecture documentation, but some sophisticated users benefit from understanding system structure.

3.8.4 Detail Level

Overview information focused on user-visible behavior. No implementation details.

3.8.5 Format Preferences

Simplified diagrams. User-oriented explanations. Behavior descriptions in non-technical language.

3.9 Analysts (Quality Attribute Specialists)

Analysts evaluate the architecture against specific quality attributes. Different quality attributes require different documentation.

3.9.1 Concerns

Analysts are concerned with evaluating the architecture against specific quality attributes such as performance, accuracy, modifiability, security, availability, and usability.

3.9.2 General Documentation Needs

They need module views, especially decomposition and uses, for understanding structure and dependencies.

They need component-and-connector views showing processes, communication paths, and data flow.

They need allocation views, especially deployment, to understand nodes, networks, and redundancy.

3.9.3 Attribute-Specific Needs

For performance analysis: communicating-processes C&C view, deployment view, and behavioral documentation with timing and concurrency information.

For accuracy analysis: C&C and data-flow views that show where numerical or algorithmic errors can accumulate.

For modifiability analysis: uses and decomposition views for change-impact analysis; C&C views to expose runtime side effects of changes.

For security analysis: deployment and context views showing external connections; C&C and decomposition views highlighting where security controls are applied.

For availability analysis: communicating-processes and deployment views for identifying failure modes and redundancy mechanisms.

For usability analysis: decomposition and C&C views that show how user-visible state and error handling are managed.

3.9.4 Documentation Pattern

Analysts need specific views that support their analysis methods. The required views depend on the quality attribute being analyzed.

3.9.5 Detail Level

Detailed information on views relevant to the quality attribute. Behavioral and quantitative information where relevant.

3.9.6 Format Preferences

Formal or semi-formal notations amenable to analysis. Quantitative data where available. Behavioral specifications.

3.10 Infrastructure Support Personnel

Infrastructure support personnel deploy, operate, and maintain the runtime environment.

3.10.1 Concerns

Infrastructure support personnel are concerned with deployment—how to install and configure the system. They are concerned with operations—how to monitor and manage the running system. They are concerned with troubleshooting—how to diagnose and resolve problems.

3.10.2 Documentation Needs

They need module views including decomposition and uses describing major artifacts and dependencies.

They need component-and-connector views indicating what runs where and how components interact at runtime.

They need allocation views including deployment, install, and implementation views, which map software assets to infrastructure.

They need other documentation including variability guides used for environment- or configuration-specific setups.

3.10.3 Documentation Pattern

An operational view of software assets and their mapping to infrastructure components. Focus on what runs, where it runs, and how to manage it.

3.10.4 Detail Level

Detailed information on deployment and installation. Overview of logical structure. Detailed configuration information.

3.10.5 Format Preferences

Runbooks and operational procedures. Configuration references. Deployment diagrams with infrastructure detail. Troubleshooting guides.

3.11 New Stakeholders

New stakeholders are joining the project and need to come up to speed.

3.11.1 Concerns

New stakeholders are concerned with onboarding—understanding the system at a high level before diving into role-specific details.

3.11.2 Documentation Needs

They need introductory, broad-scope material: top-level context diagrams, architectural constraints, overall rationale, and root-level views.

They need the same kinds of views as their ultimate role requires, but initially at lower levels of detail. Progressive disclosure from overview to detail supports learning.

3.11.3 Documentation Pattern

Start broad and shallow, then deepen based on role. Good onboarding documentation has clear entry points and progressive paths to detail.

3.11.4 Detail Level

Overview information initially. Increasing detail as learning progresses.

3.11.5 Format Preferences

Orientation materials. Guided tours. Layered documentation from overview to detail. Glossaries and concept explanations.

3.12 Current and Future Architects

Architects are responsible for the architecture itself. They are the most demanding documentation consumers.

3.12.1 Concerns

Architects are concerned with design decisions, rationale, trade-offs, and the complete architectural history of the system. They must understand the architecture deeply enough to evolve it.

3.12.2 Documentation Needs

They need essentially all views in substantial detail. They need rationale and constraints for all major decisions. They need traceability between requirements, architectural choices, and quality-attribute evaluations.

3.12.3 Documentation Pattern

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

3.12.4 Detail Level

Detailed information across all views. The architecture documentation is their primary working artifact.

3.12.5 Format Preferences

Comprehensive reference documentation. Version-controlled documentation that tracks evolution. Analysis tools and models. Traceability mechanisms.

4 Documentation Elements

Stakeholders' needs map to specific documentation elements. Understanding these elements helps in planning documentation.

4.1 Module Views

Module views show the static structure of code organization.

4.1.1 Decomposition View

Shows how the system is decomposed into modules. Serves stakeholders who need to understand system organization, assign work, or locate functionality.

4.1.2 Uses View

Shows functional dependencies between modules. Serves stakeholders planning development sequences, analyzing change impact, or understanding what depends on what.

4.1.3 Generalization View

Shows inheritance and specialization relationships. Serves stakeholders understanding object-oriented structures, extension mechanisms, or type hierarchies.

4.1.4 Layered View

Shows organization into layers with constrained dependencies. Serves stakeholders understanding abstraction levels, portability boundaries, or modification isolation.

4.1.5 Data Model View

Shows the structure of persistent data. Serves stakeholders understanding information architecture, database design, or data relationships.

4.2 Component-and-Connector Views

Component-and-connector views show runtime structure and behavior.

4.2.1 Process Views

Show runtime processes and their interactions. Serve stakeholders analyzing concurrency, performance, or failure behavior.

4.2.2 Communication Views

Show how components communicate. Serve stakeholders understanding protocols, data flow, or integration points.

4.2.3 Deployment-Runtime Views

Show how runtime components map to hardware. Serve stakeholders understanding performance, availability, or operational behavior.

4.3 Allocation Views

Allocation views show how software maps to non-software structures.

4.3.1 Deployment View

Shows how software maps to hardware and infrastructure. Serves stakeholders understanding operational environment, resource requirements, or infrastructure dependencies.

4.3.2 Implementation View

Shows how modules map to source artifacts. Serves stakeholders navigating the codebase, managing configuration, or understanding build structure.

4.3.3 Install View

Shows how software is packaged and installed. Serves stakeholders deploying the system, managing releases, or understanding distribution.

4.3.4 Work Assignment View

Shows how modules map to development organizations. Serves stakeholders managing development, coordinating teams, or planning resources.

4.4 Supporting Documentation

Beyond views, supporting documentation serves stakeholder needs.

4.4.1 Interface Documentation

Specifies how elements interact. Serves stakeholders implementing interactions, testing interfaces, or integrating systems.

4.4.2 Context Diagrams

Show the system in its environment. Serves stakeholders understanding scope, external dependencies, or integration context.

4.4.3 Variability Guides

Document variation mechanisms. Serve stakeholders building products, configuring systems, or understanding options.

4.4.4 Analysis Results

Document quality attribute evaluations. Serve stakeholders making decisions, assessing risks, or verifying requirements.

4.4.5 Rationale and Constraints

Document why decisions were made. Serve stakeholders maintaining the system, evolving the architecture, or understanding trade-offs.

4.4.6 Mapping Between Views

Document how elements in one view relate to elements in others. Serve stakeholders tracing across views or understanding different perspectives on the same system.

5 Tailoring Documentation to Stakeholders

Effective documentation is tailored to its audience. Several strategies support tailoring.

5.1 Layered Documentation

Organize documentation in layers from overview to detail.

5.1.1 Overview Layer

High-level descriptions for orientation. Context diagrams, system summaries, key decisions.

5.1.2 Reference Layer

Comprehensive documentation for detailed work. Full view specifications, interface details, behavioral specifications.

5.1.3 Supporting Layer

Background and rationale. Design rationale, analysis results, alternatives considered.

Different stakeholders enter at different layers and go to different depths.

5.2 Multiple Representations

Provide the same information in multiple forms.

5.2.1 Graphical Representations

Diagrams for visual understanding. Effective for showing structure, relationships, and flow.

5.2.2 Textual Representations

Prose for explanation and detail. Effective for describing behavior, rationale, and constraints.

5.2.3 Tabular Representations

Tables for systematic coverage. Effective for catalogs, properties, and mappings.

5.2.4 Formal Representations

Precise specifications for analysis. Effective for interfaces, protocols, and contracts.

Different stakeholders prefer different representations.

5.3 Filtered Views

Provide filtered versions of views for specific audiences.

5.3.1 Scope Filtering

Show only relevant portions of a view. A developer might see only their subsystem; a project manager might see only work-assignable modules.

5.3.2 Detail Filtering

Show appropriate level of detail. An executive sees high-level components; a developer sees full decomposition.

5.3.3 Attribute Filtering

Show only relevant properties. A security analyst sees security-relevant attributes; a performance analyst sees performance-relevant attributes.

5.4 Stakeholder-Specific Documents

Create documents targeted at specific stakeholders.

5.4.1 Developer Guides

Documentation organized for developer needs—interfaces, patterns, conventions.

5.4.2 Operations Guides

Documentation organized for operations needs—deployment, configuration, troubleshooting.

5.4.3 Integration Guides

Documentation organized for integration needs—APIs, protocols, examples.

5.4.4 Architecture Overviews

Documentation organized for management and customer needs—context, decisions, assurances.

5.5 Navigation Aids

Help stakeholders find what they need.

5.5.1 Tables of Contents

Organized to support stakeholder tasks.

5.5.2 Indexes

Enable finding specific elements.

5.5.3 Cross-References

Connect related information across views.

5.5.4 Stakeholder Roadmaps

Guide each stakeholder type to relevant documentation.

6 Documentation Planning Process

Systematic planning ensures documentation serves stakeholder needs.

6.1 Stakeholder Analysis

Identify and characterize stakeholders.

6.1.1 Identify Stakeholders

List all stakeholders using role-based, organization-based, and lifecycle-based identification.

6.1.2 Characterize Concerns

For each stakeholder, document their concerns, questions, and decisions.

6.1.3 Prioritize Stakeholders

Rank stakeholders by influence, urgency, and legitimacy to guide resource allocation.

6.2 View Selection

Select views based on stakeholder needs.

6.2.1 Map Concerns to Views

Identify which views address which concerns. Use the stakeholder-view table as a guide.

6.2.2 Identify Required Views

Determine which views must be created. A view is required if important stakeholders need it.

6.2.3 Identify Optional Views

Determine which views would be valuable but could be omitted if resources are limited.

6.3 Detail Planning

Determine appropriate detail for each view and stakeholder.

6.3.1 Detail Requirements

For each stakeholder-view combination, determine whether detailed, some, or overview information is needed.

6.3.2 Consolidation

Where multiple stakeholders need the same view, plan documentation that serves all (possibly with filtering for specific audiences).

6.4 Production Planning

Plan how documentation will be created.

6.4.1 Sequencing

Determine which documentation is needed when. Documentation needed for early planning comes first.

6.4.2 Responsibility

Assign responsibility for creating and maintaining each documentation element.

6.4.3 Resources

Allocate time, tools, and effort for documentation work.

6.5 Validation

Confirm that planned documentation meets needs.

6.5.1 Stakeholder Review

Review documentation plans with stakeholders to confirm coverage.

6.5.2 Adjustment

Modify plans based on feedback. Documentation planning is iterative.

7 Documentation Quality

Documentation quality affects its value to stakeholders.

7.1 Quality Attributes of Documentation

7.1.1 Accuracy

Documentation correctly describes the architecture. Inaccurate documentation is worse than no documentation.

7.1.2 Currency

Documentation reflects the current state of the architecture. Outdated documentation misleads.

7.1.3 Completeness

Documentation covers what stakeholders need. Gaps in coverage leave questions unanswered.

7.1.4 Consistency

Documentation elements agree with each other. Inconsistencies confuse and undermine confidence.

7.1.5 Clarity

Documentation is understandable by its intended audience. Unclear documentation fails its purpose.

7.1.6 Navigability

Stakeholders can find what they need. Poorly organized documentation wastes time.

7.1.7 Relevance

Documentation addresses actual stakeholder concerns. Irrelevant documentation is ignored.

7.2 Quality Assurance

7.2.1 Reviews

Stakeholder reviews verify that documentation meets needs. Technical reviews verify accuracy.

7.2.2 Validation Against Code

Verify that documentation matches implementation. Tools can help detect drift.

7.2.3 Usability Testing

Observe stakeholders using documentation. Identify where they struggle.

7.2.4 Feedback Mechanisms

Provide ways for stakeholders to report problems and suggest improvements.

8 Best Practices

Experience suggests several best practices for stakeholder-driven documentation.

8.1 Know Your Stakeholders

Invest time in understanding who will use documentation and what they need. Don't assume; ask.

8.2 Prioritize Ruthlessly

Not all documentation has equal value. Focus effort where it matters most.

8.3 Start with High-Value, High-Need Documentation

Address the most important stakeholder needs first. Documentation that enables critical decisions should come before nice-to-have details.

8.4 Iterate Based on Feedback

Documentation needs evolve. Regularly check whether documentation is meeting stakeholder needs and adjust.

8.5 Keep Documentation Current

Outdated documentation is harmful. Build documentation maintenance into development processes.

8.6 Provide Multiple Entry Points

Different stakeholders need different paths into the documentation. Provide roadmaps and guides for each audience.

8.7 Balance Comprehensiveness and Usability

Comprehensive documentation that no one can navigate fails. Usable documentation with gaps also fails. Find the appropriate balance.

8.8 Document Rationale

Stakeholders, especially maintainers and architects, need to understand why decisions were made. Document rationale alongside decisions.

8.9 Use Appropriate Tools

Select documentation tools that support stakeholder needs—collaboration, navigation, generation from models, version control.

8.10 Measure Documentation Value

Track whether documentation is used and whether it helps stakeholders. Use this information to improve.

9 Common Challenges

Stakeholder-driven documentation presents several challenges.

9.1 Stakeholder Identification Challenges

9.1.1 Missing Stakeholders

Stakeholders may be overlooked, especially future stakeholders. Systematic identification processes help.

9.1.2 Conflicting Needs

Different stakeholders may have conflicting needs. Prioritization and tailoring address conflicts.

9.1.3 Changing Stakeholders

Stakeholders change over the project lifecycle. Documentation plans must accommodate change.

9.2 Resource Challenges

9.2.1 Limited Resources

Documentation competes with other project activities. Prioritization ensures limited resources go to highest-value documentation.

9.2.2 Documentation Debt

Deferred documentation accumulates. Plan for addressing documentation debt.

9.2.3 Maintenance Burden

Keeping documentation current requires ongoing effort. Build maintenance into processes.

9.3 Quality Challenges

9.3.1 Accuracy Drift

Documentation diverges from implementation over time. Regular validation catches drift.

9.3.2 Staleness

Documentation becomes outdated as the system evolves. Trigger updates when architecture changes.

9.3.3 Inconsistency

Different documentation elements contradict each other. Cross-view validation identifies inconsistencies.

9.4 Usability Challenges

9.4.1 Navigation Difficulties

Stakeholders can't find what they need. Navigation aids and stakeholder roadmaps help.

9.4.2 Inappropriate Detail

Too much detail overwhelms; too little leaves gaps. Tailor detail to audience.

9.4.3 Format Mismatches

Documentation format doesn't match stakeholder preferences. Provide multiple representations.

10 Stakeholder–View Tables

The qualitative mapping of stakeholders to documentation needs can be summarized in stakeholder–view tables. These tables provide a quick reference for documentation planning.

10.1 Interpreting the Tables

Entries in the stakeholder–view tables use the following key:

d = detailed information needed—the stakeholder requires comprehensive coverage of this view.

s = some details needed—the stakeholder requires partial or selective coverage.

o = overview information needed—the stakeholder requires high-level coverage only.

x = anything—the stakeholder's needs vary and should be determined through discussion.

Empty cells indicate the stakeholder typically does not need that documentation element.

10.2 Using the Tables

Use the filled table as a starting point for documentation planning. Adjust based on specific project stakeholders and their particular needs.

Use the empty template to capture stakeholder needs on new projects. Fill in cells after interviewing stakeholders about their information needs.

10.3 Stakeholder–View Table (Filled Version)

Stakeholder	Module Views					C&C Views	Allocation Views				Other Documentation					
	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

Key: d = detailed information, s = some details, o = overview information, x = anything (varies by individual).

10.4 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.

	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																

Key: d = detailed information, s = some details, o = overview information, x = anything (varies by individual).

11 Conclusion

Architecture documentation exists to serve stakeholders. Effective documentation is stakeholder-driven: it addresses the concerns of identified stakeholders, provides appropriate levels of detail, and is produced in formats stakeholders can use.

The stakeholder categories and their documentation needs described in this document provide a starting framework. Each project should identify its specific stakeholders, characterize their specific needs, and plan documentation accordingly.

Key principles guide stakeholder-driven documentation: know your stakeholders and their concerns; prioritize documentation effort based on stakeholder importance; tailor documentation to audience through layering, filtering, and multiple representations; maintain documentation quality through accuracy, currency, and completeness; and iterate based on feedback to ensure documentation meets actual needs.

The stakeholder–view tables provide a quick reference for documentation planning. Use the filled table as a starting point; customize based on project-specific stakeholders and needs.

Documentation that serves its stakeholders is used, provides value, and justifies its cost. Documentation that ignores stakeholders sits unused on shelves. Stakeholder-driven documentation ensures that documentation effort produces documentation value.

References

- Clements, P., Bachmann, F., Bass, L., Garlan, D., Ivers, J., Little, R., Merson, P., Nord, R., & Stafford, J. (2010). *Documenting Software Architectures: Views and Beyond* (2nd ed.). Addison-Wesley Professional.
- Bass, L., Clements, P., & Kazman, R. (2021). *Software Architecture in Practice* (4th ed.). Addison-Wesley Professional.
- ISO/IEC/IEEE. (2011). *ISO/IEC/IEEE 42010:2011 Systems and software engineering—Architecture description.*
- Rozanski, N., & Woods, E. (2011). *Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives* (2nd ed.). Addison-Wesley Professional.
- Clements, P., & Northrop, L. (2001). *Software Product Lines: Practices and Patterns.* Addison-Wesley Professional.
- Kruchten, P. (1995). The 4+1 View Model of architecture. *IEEE Software*, 12(6), 42–50.