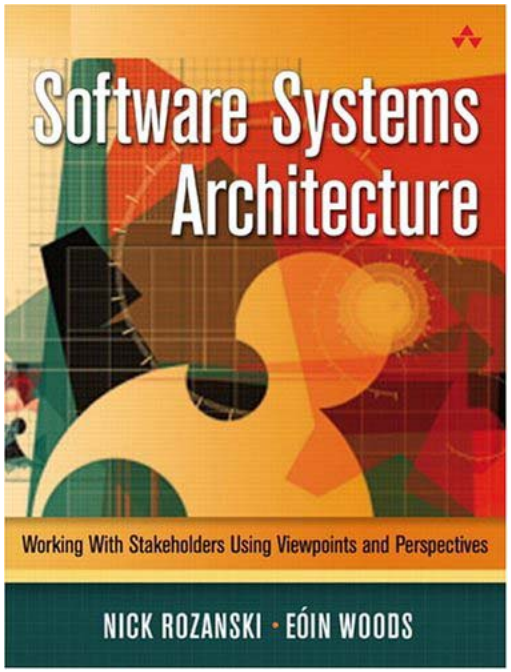
Viewpoints and Perspectives Reference Card

Nick Rozanski and Eoin Woods
www.viewpoints-and-perspectives.info



Content excerpted from *Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives* by Nick Rozanski and Eoin Woods, Addison Wesley 2005.

The book is available from Amazon.com and Amazon.co.uk and other booksellers that carry Addison-Wesley books.

Contents

Overview

Our book is based around four key concepts: ***stakeholders***, ***views***, ***viewpoints*** and ***perspectives***. The definition of each is reproduced below.

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

Definition: *A view is a representation of one or more structural aspects of an architecture that illustrates how the architecture addresses one or more concerns held by one or more of its stakeholders.*

Definition: *A viewpoint is a collection of patterns, templates, and conventions for constructing one type of view. It defines the stakeholders whose concerns are reflected in the viewpoint and the guidelines, principles, and template models for constructing its views.*

Definition: *An architectural perspective is a collection of activities, tactics, and guidelines that are used to ensure that a system exhibits a particular set of related quality properties that require consideration across a number of the system's architectural views.*

The sets of viewpoints and perspectives that we have developed for information systems architecture are illustrated by the diagram in Figure 1.

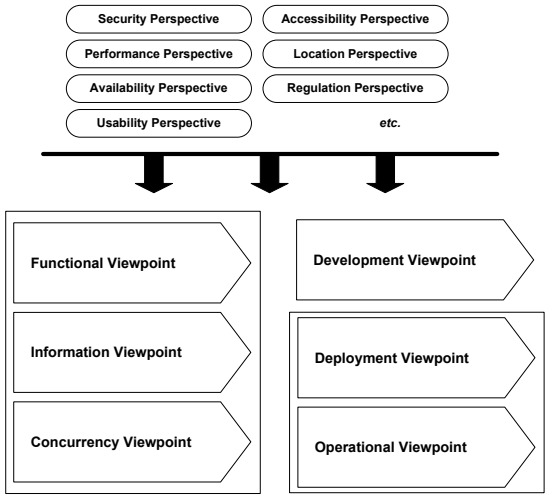


Figure 1 - Viewpoints and Perspectives

In this document, we provide a summary of each of our viewpoints and perspectives.

Viewpoint Summaries

- ***Functional***: Describes the system's functional elements, their responsibilities, interfaces, and primary interactions and drives the shape of other system structures such as the information structure, concurrency structure, deployment structure, and so on.
- ***Information***: Describes the way that the architecture stores, manipulates, manages, and distributes information. This viewpoint develops a complete but high-level view of static data structure and information flow to answer the big questions around content, structure, ownership, latency, references, and data migration.
- ***Concurrency***: Describes the concurrency structure of the system and maps functional elements to concurrency units to clearly identify the parts of the system that can execute concurrently and how this is coordinated and controlled.
- ***Development***: Describes the architecture that supports the software development process. Development views communicate the aspects of the architecture of interest to those stakeholders involved in building, testing, maintaining, and enhancing the system.
- ***Deployment***: Describes the environment into which the system will be deployed, and the dependencies the system has on its runtime environment. Deployment views capture the system's hardware environment, technical environment requirements, and the mapping of the software to hardware elements.
- ***Operational***: Describes how the system will be operated, administered, and supported when it is running in its production environment, by identifying system-wide strategies for addressing operational concerns and identifying solutions that address these.

Quality Properties Addressed by Perspectives

- ***Accessibility***: The ability of the system to be used by people with disabilities
- ***Availability and Resilience***: The ability of the system to be fully or partly operational as and when required and to effectively handle failures that could affect system availability
- ***Development Resource***: The ability of the system to be designed, built, deployed, and operated within known constraints of people, budget, time, etc.
- ***Evolution***: The ability of the system to be flexible in the face of the inevitable change that all systems experience after deployment, balanced against the costs of providing such flexibility
- ***Internationalization***: The ability of the system to be independent from any particular language, country, or cultural group
- ***Location***: The ability of the system to overcome problems brought about by the absolute location of its elements and the distances between them
- ***Performance and Scalability***: The ability of the system to predictably execute within its mandated performance profile and to handle increased processing volumes
- ***Regulation***: The ability of the system to conform to local and international laws, quasi-legal regulations, company policies, and other rules and standards
- ***Security***: The ability of the system to reliably control, monitor, and audit who can perform what actions on what resources and to detect and recover from failures in security mechanisms
- ***Usability***: The ease with which people who interact with the system can work effectively

Stakeholders

Stakeholder groups important to the development of most information systems include the following.

- ***Acquirers***: Oversee the procurement of the system or product
- ***Assessors***: Oversee the system's conformance to standards and legal regulation
- ***Communicators***: Explain the system to other stakeholders via its documentation and training materials
- ***Developers***: Construct and deploy the system from specifications (or lead the teams that do this)
- ***Maintainers***: Manage the evolution of the system once it is operational
- ***Suppliers***: Build and/or supply the hardware, software, or infrastructure on which the system will run
- ***Support staff***: Provide support to users for the product or system when it is running
- ***System administrators***: Run the system once it has been deployed
- ***Testers***: Test the system to ensure that it is suitable for use
- ***Users***: Define the system's functionality and ultimately make use of it

The characteristics of a good stakeholder can be summarised as follows.

- ***Informed***: Do your stakeholders have the information, the experience, and the understanding needed to make the right decisions?
- ***Committed***: Are your stakeholders willing and able to make themselves available to participate in the process, and are they prepared to make some possibly difficult decisions?
- ***Authorized***: Can you be sure that decisions made now by your stakeholders will not be reversed later (at potentially high cost)?
- ***Representative***: If a stakeholder is a group rather than a person, have suitable representatives been selected from the group? Do those representatives meet the above criteria for individual stakeholders?

Functional Viewpoint

The Functional view of a system defines the architectural elements that deliver the system's functionality. The view documents the system's functional structure—including the key functional elements, their responsibilities, the interfaces they expose, and the interactions between them. Taken together, this demonstrates how the system will perform the functions required of it.

Definition	Describes the system's runtime functional elements and their responsibilities, interfaces, and primary interactions
Concerns	Functional capabilities, external interfaces, internal structure, and design philosophy
Models	Functional structure model
Problems and Pitfalls	Poorly defined interfaces, poorly understood responsibilities, infrastructure modeled as functional elements, overloaded view, diagrams without element definitions, difficulty in reconciling the needs of multiple stakeholders, inappropriate level of detail, "God elements," and too many dependencies
Applicability	All systems

Stakeholders and Concerns

Acquirers	Primarily functional capabilities and external interfaces
Assessors	All concerns
Communicators	All concerns, to some extent
Developers	Primarily design philosophy and internal structure, but also functional capabilities and external interfaces
System administrators	Primarily design philosophy and internal structure
Testers	Primarily design philosophy and internal structure, but also functional capabilities and external interfaces
Users	Primarily functional capabilities and external interfaces

Checklist

- Do you have fewer than 15–20 top-level elements?
- Do all elements have a name, clear responsibilities, and clearly defined interfaces?
- Do all element interactions take place via well-defined interfaces and connectors that link the interfaces?
- Do your elements exhibit an appropriate level of cohesion and coupling?
- Have you identified the important usage scenarios and used these to validate the system's functional structure?
- Have you checked the functional coverage of your architecture to ensure it meets its functional requirements?
- Have you considered how the architecture is likely to cope with possible change scenarios in the future?
- Does the presentation of the view take into account the concerns and capabilities of all interested stakeholder groups? Will the view act as an effective communication vehicle for all of these groups?

Information Viewpoint

The ultimate purpose of any information system is to manipulate data in some form. This data may be stored persistently, as in a database management system, or it may be transiently manipulated in memory while a program executes. You use the Information view to answer questions about how your system will store, manipulate, manage, and distribute information

Definition	Describes the way that the architecture stores, manipulates, manages, and distributes information
Concerns	Information structure and content; information flow; data ownership; timeliness, latency, and age; references and mappings; transaction management and recovery; data quality; data volumes; archives and data retention; and regulation
Models	Static data structure models, information flow models, information lifecycle models, data ownership models, data quality analysis, metadata models, and volumetric models
Problems and Pitfalls	Data incompatibilities, poor data quality, unavoidable multiple updaters, key matching deficiencies, poor information latency, interface complexity, and inadequate volumetrics
Applicability	Any system that has more than trivial information management needs

Stakeholders and Concerns

Acquirers	Typically interested in data quality, archiving and data retention
Assessors	Typically focus on regulation and data quality
Communicators	May find understanding the key principles and strategies helpful
Developers	Focus on how the models will map to real databases & interfaces
System administrators	Interested in how the databases & interfaces will be managed and supported
Users	Concerned with information stored and aspects like data ownership and user-visible qualities such as timeliness & latency.

Checklist

- Do you have an appropriate level of detail in your models (no more than 20 entities)?
- Are keys clearly identified for all important entities?
- Have you defined mappings between keys, where required, and defined processes for maintaining these mappings when data items are created and removed?
- Have you defined strategies for resolving data ownership conflicts, particularly where there are multiple creators or updaters?
- Are latency requirements clearly identified, and are mechanisms in place to ensure these are achieved?
- Do you have clear strategies for transactional consistency across distributed data stores, balanced against their cost in terms of performance and complexity?
- Do you have mechanisms in place for validating migrated data and dealing appropriately with errors?
- Have you defined sufficient storage and processing capacity for archiving and restore?
- Has a data quality assessment been done? Have you created strategies for dealing with poor-quality data?

Concurrency Viewpoint

The Concurrency view is used to describe the system's concurrency and state-related structure and constraints. This involves defining the parts of the system that can run at the same time and how this is to be controlled, by defining how the system's functional elements are packaged into operating system processes and how the processes coordinate their execution.

Definition	Describes the concurrency structure of the system, mapping functional elements to concurrency units to clearly identify the parts of the system that can execute concurrently, and shows how this is coordinated and controlled
Concerns	Task structure, mapping of functional elements to tasks, interprocess communication, state management, synchronization and integrity, startup and shutdown, task failure, and reentrancy
Models	System-level concurrency models and state models
Problems and Pitfalls	Modeling of the wrong concurrency, excessive complexity, resource contention, deadlock, and race conditions
Applicability	All information systems with a number of concurrent threads of execution

Stakeholders and Concerns

Administrators	Task structure, startup and shutdown, and task failure
Communicators	Task structure, startup and shutdown, and task failure
Developers	All concerns
Testers	Task structure, mapping of functional elements to tasks, startup and shutdown, task failure, and reentrancy

Checklist

- Is there a clear system-level concurrency model?
- Are your models at the right level of abstraction? Have you focused on the architecturally significant aspects?
- Can you simplify your concurrency design?
- Do all interested parties understand the overall concurrency strategy?
- Have you mapped all functional elements to a process (and thread if necessary)?
- Do you have a state model for at least one functional element in each process and thread? If not, are you sure the processes and threads will interact safely?
- Have you defined a suitable set of interprocess communication mechanisms to support the interelement interactions defined in the Functional view?
- Are all shared resources protected from corruption?
- Have you minimized the intertask communication and synchronization required?
- Do you have any resource hot spots in your system? If so, have you estimated the likely throughput, and is it high enough? Do you know how you would reduce contention at these points if forced to later?
- Can the system possibly deadlock? If so, do you have a strategy for recognizing and dealing with this when it occurs?

Development Viewpoint

A considerable amount of planning and design of the development environment is often required to support the design and build of software for complex systems. Things to think about include code structure and dependencies, build and configuration management of deliverables, system-wide design constraints, and system-wide standards to ensure technical integrity. It is the role of the Development view to address these aspects of the system development process.

Definition	Describes the architecture that supports the software development process
Concerns	Module organization, common processing, standardization of design, standardization of testing, instrumentation, and codeline organization
Models	Module structure models, common design models, and codeline models
Problems and Pitfalls	Too much detail, overburdening the AD, uneven focus, lack of developer focus, lack of precision, and problems with the specified environment
Applicability	All systems with significant software development involved in their creation

Stakeholders and Concerns

Developers	All concerns
Testers	All concerns

Checklist

- Have you defined a clear strategy for organizing the source code modules in your system?
- Have you defined a general set of rules governing the dependencies that can exist between code modules at different abstraction levels?
- Have you identified all of the aspects of element implementation that need to be standardized across the system?
- Have you clearly defined how any standard processing should be performed?
- Have you identified any standard approaches to design that you need all element designers and implementers to follow? If so, do your software developers accept and understand these approaches?
- Will a clear set of standard third-party software elements be used across all element implementations? Have you defined the way they should be used?
- Is this view as minimal as possible?
- Is the presentation of this view in the AD appropriate?

Deployment Viewpoint

The Deployment view focuses on aspects of the system that are important after the system has been tested and is ready to go into live operation. This view defines the physical environment in which the system is intended to run, including the hardware environment your system needs (e.g., processing nodes, network interconnections, and disk storage facilities), the technical environment requirements for each node (or node type) in the system, and the mapping of your software elements to the runtime environment that will execute them.

Definition	Describes the environment into which the system will be deployed, including the dependencies the system has on its runtime environment
Concerns	Types of hardware required, specification and quantity of hardware required, third-party software requirements, technology compatibility, network requirements, network capacity required, and physical constraints
Models	Runtime platform models, network models, and technology dependency models
Problems and Pitfalls	Unclear or inaccurate dependencies, unproven technology, lack of specialist technical knowledge, and late consideration of the deployment environment
Applicability	Systems with complex or unfamiliar deployment environments

Stakeholders and Concerns

System administrators	Types, specification, and quantity of hardware required; third-party software requirements; technology compatibility; network requirements; network capacity required; and physical constraints
Developers	Types and (general) specification of hardware required, third-party software requirements, technology compatibility, and network requirements (particularly topology)
Communicators	Types and specification of hardware required, third-party software requirements, and network requirements (particularly topology)
Testers	Types, specification, and quantity of hardware required; third-party software requirements, and network requirements
Assessors	Types of hardware required, technology compatibility, and network requirements

Checklist

- Have you mapped all of the system's functional elements to a type of hardware device? Have you mapped them to specific hardware devices if appropriate?
- Is the role of each hardware element in the system fully understood? Is the specified hardware suitable for the role?
- Have you established detailed specifications for the system's hardware devices? Do you know exactly how many of each device are required?
- Have you identified all required third-party software and documented all the dependencies between system elements and third-party software?
- Is the network topology required by the system understood and documented?
- Have you estimated and validated the required network capacity? Can the proposed network topology be built to support this capacity?
- Have network specialists validated that the required network can be built?
- Have you performed compatibility testing when evaluating your architectural options to ensure that the elements of the proposed deployment environment can be combined as desired?
- Have you used enough prototypes, benchmarks, and other practical tests when evaluating your architectural options to validate the critical aspects of the proposed deployment environment?
- Can you create a realistic test environment that is representative of the proposed deployment environment?
- Are you confident that the deployment environment will work as designed? Have you obtained external review to validate this opinion?
- Are the assessors satisfied that the deployment environment meets their requirements in terms of standards, risks, and costs?
- Have you checked that the physical constraints (such as floor space, power, cooling, and so on) implied by your required deployment environment can be met?

Operational Viewpoint

The aim of the Operational viewpoint is to identify a system-wide strategy for addressing the operational concerns of the system's stakeholders and to identify solutions that address these. The Operational view focuses on concerns that help ensure that the system is a reliable and effective part of commissioning enterprise's information technology environment. For a product development project, the Operational view illustrates the types of concerns that customers of the product are likely to encounter, rather than the concerns of a specific site.

Definition	Describes how the system will be operated, administered, and supported when it is running in its production environment
Concerns	Installation and upgrade, functional migration, data migration, operational monitoring and control, configuration management, performance monitoring, support, and backup and restore
Models	Installation models, migration models, configuration management models, administration models, and support models
Problems and Pitfalls	Lack of engagement with the operational staff, lack of backout planning, lack of migration planning, insufficient migration window, missing management tools, lack of integration into the production environment, and inadequate backup models
Applicability	Any system being deployed into a complex or critical operational environment

Stakeholders and Concerns

Assessors	Functional migration, data migration, and support
Communicators	Installation and upgrade, functional migration, and operational monitoring and control
Developers	Operational monitoring and control and performance monitoring
Support staff	Functional migration, data migration, and support
System administrators	All concerns

Checklist

- Do you know what it takes to install your system?
- Do you have a plan for backing out a failed installation?
- Can you upgrade an existing version of the system (if required)?
- How will information be moved from the existing environment into the new system?
- Do you have a clear migration strategy to move workload to the new system? Can you reverse the migration if you need to? How will you deal with data synchronization?
- How will the system be backed up? Is restore possible in an acceptable time period?
- Are the administrators confident that they can monitor and control the system and do they have a clear understanding of operational procedures?
- How will performance metrics be captured for the system's elements?
- Can you manage the configuration of all of the system's elements?
- Do you know how support will be provided for the system? Is the support provided suitable for the stakeholders it is being provided for?
- Have you cross-referenced the requirements of the administration model back to the Development view to ensure that they will be implemented consistently?

Accessibility Perspective

Accessibility should take into account not only the direct users of the system—i.e., those sitting at terminals—but the indirect users as well. For example, a financial system may need to provide bank statements in Braille for blind customers. Consideration of disability aside, addressing accessibility concerns brings benefits in many cases by making systems more usable and efficient in their operation.

Desired Quality	The ability of the system to be used by people with disabilities
Applicability	Any system that may be used or operated by people with disabilities or may be subject to legislation regarding disabilities
Concerns	Types of disability, functional availability, and disability regulation
Activities	Identification of system touch points, device independence, and content equivalence
Tactics	Assistive technologies, specialist input devices, and voice recognition
Problems & Pitfalls	Ignoring these needs until too late, lack of knowledge about regulation and legislation, and lack of knowledge about suitable solutions

Applicability to Views

Functional	In theory, the functional structure should not really be affected by accessibility considerations. In practice, functional compromises may need to be made.
Information	The information structure is unlikely to be significantly affected.
Concurrency	The impact on this view is minimal.
Development	The Development view needs to raise awareness that accessibility issues are important. And, of course, you may need to accommodate disabled developers, too.
Deployment	The deployment environment is likely to be the most affected by this perspective. Special hardware may be needed to support disabled users.
Operational	The Operational view may have to take into account the needs of disabled users requiring support or the needs of disabled support staff themselves.

Checklist for Requirements Capture

- Have you identified and obtained stakeholder approval of the extent to which the system must support the needs of disabled users?
- Have you provided for the needs of indirect disabled users, such as customers who need paperwork provided in Braille format?
- Have you identified the disability legislation that affects the system and assessed the system against it?
- Have you ensured that the system meets any internal accessibility standards?
- Have you considered all points at which the system has any human interaction? For example, have you considered operational management and monitoring of the system, or printed forms that are sent to customers to be filled in?

Checklist for Architecture Definition

- How confident are you that your architectural assumptions are correct? Where you are not, are mitigating activities in place (such as a proof-of-concept)?
- Do the interactive elements of your architecture sufficiently separate presentation and content to meet the system's accessibility objectives?
- Are the interfaces between components (particularly those leading in and out of presentation devices) sufficiently generic to be able to take on board new devices without (much) rework?
- Does the architecture allow for presentation alternatives to convey meaning (e.g., text, pictures, and/or sound in a user interface)?
- Do standards for user interface design emphasize simplicity, consistency, and clarity in place? Does the architecture adhere to them?

Availability and Resilience Perspective

This perspective allows you to identify the availability and resilience needs of your system and identify solutions that take into account the costs that providing these properties incur.

Desired Quality	The ability of the system to be fully or partly operational as and when required and to effectively handle failures that could affect system availability
Applicability	Any system that has complex or extended availability requirements, complex recovery processes, or a high profile (e.g., is visible to the public)
Concerns	Classes of service, planned downtime, unplanned downtime, time to repair, and disaster recovery
Activities	Capture the availability requirements, produce the availability schedule, estimate platform availability, estimate functional availability, assess against the requirements, and rework the architecture
Tactics	Select fault-tolerant hardware, use hardware clustering and load balancing, log transactions, apply software availability solutions, select or create fault-tolerant software, and identify backup and disaster recovery solutions
Problems & Pitfalls	Single point of failure, overambitious availability requirements, ineffective error detection, overlooked global availability requirements, and incompatible technologies

Applicability to Views

Functional	Functional changes may sometimes be needed to support availability requirements, such as the ability to operate in an offline mode a network is unavailable.
Information	A key availability consideration is the set of processes and systems for backup and recovery.
Concurrency	Features such as hardware replication and failover in your system may imply changes or enhancements to your concurrency model.
Development	Your approach to achieving availability may impose design constraints on the software modules that need captured in this view.
Deployment	Availability and resilience can have a big impact on the deployment environment such as fault-tolerant hardware, disaster recovery sites, redundancy & clustering.
Operational	May need to capture processes to allow the identification and recovery of problems in the production environment and handle failure appropriately (e.g. failover & DR).

Checklist for Requirements Capture

- Are availability requirements defined, documented, and approved?
- Are availability requirements driven by business needs?
- Do availability requirements consider different classes of service, if appropriate?
- Do availability requirements strike a realistic balance between cost and need?
- Do availability requirements consider online and batch availability?
- Do availability requirements take into account variations such as period end?
- Do availability requirements take into account future changes a longer online day?
- Can availability requirements be met by the chosen hardware and software platform?
- Have you defined strategies for disaster recovery and business continuity?
- Do stakeholders have realistic expectations around unplanned downtime?

Checklist for Architecture Definition

- Does the proposed architectural solution meet the availability requirements? Can this be demonstrated, either theoretically or based on previous practical experience?
- Does the solution consider the time taken to recover from failure?
- Does the backup solution provide for the transactional integrity of restored data?
- Can online backup be achieved? If not, it is feasible to shutdown for backups?
- Has consideration been given to restoring data from corrupt or incomplete backups?
- Will the system respond gracefully to software errors, reporting them appropriately?
- Have you defined a suitable standby site in the architecture, if appropriate?
- Have you defined and tested mechanisms for switching to standby environments?
- Have you assessed the impact of availability on functionality and performance?
- Have you assessed the architecture for single points of failure and other weaknesses?
- If you developed a fault-tolerant model, does this extend to all vulnerable components?

Development Resource Perspective

All software projects are primarily constrained by time and cost. IT budgets are never unlimited, and although technology capabilities improve from year to year, so do the costs of building, deployment, and support. This perspective allows you to consider whether your architecture can be created, given development resource constraints.

Desired Quality	The ability of the system to be designed, built, deployed, and operated within known constraints related to people, budget, time, and materials
Applicability	Any system for which development time is limited, technical skills for development or operations are hard to find, or unusual or unfamiliar hardware or software is required
Concerns	Time constraints, cost constraints, required skill sets, available resources, budgets, and external dependencies
Activities	Cost estimation, development time estimation, development planning, dependency management, scoping, prototyping, and expectation management
Tactics	Incremental and iterative development, expectation management, descoping, prototyping and piloting, and fitness for purpose
Problems & Pitfalls	Overly ambitious timescales, failure to consider lead times, failure to consider physical constraints, underbudgeting, failure to provide staff training and consider familiarization needs, insufficient resource allocation for testing and rollout, insufficient time for likely rework, and overallocation of staff

Applicability to Views

Functional	Resource constraints such as short timescales or limitations on available skills often impose restrictions on functionality and on functional qualities such as generality.
Information	Complex or particularly sophisticated information models may require a large staff of specialists to implement; and so may impose restrictions on your options.
Concurrency	Concurrent architectures are often complex to implement, so you will need to consider the development skills and testing time available to you.
Development	Cost constraints may limit the number of separate development and test environments available to you.
Deployment	Again, cost constraints may limit your options for deployment, particularly where redundancy and resilience are concerned.
Operational	You need to be aware of the cost implications of your proposed operational and support architecture.

Checklist for Requirements Capture

- Have you understood the project's key constraints in terms of time and budget, as well as the room for manoeuvring if your architecture mandates extra resources?
- Have you considered physical constraints such as existing capacity and office space?
- Have you balanced the benefits of unfamiliar technologies against their costs and risks?
- Which compromises are more likely to be accepted where resource constraints necessitate this? To what extent could you limit scope, functionality, or even quality? Are you confident that savings would be realized by making such compromises?
- To what extent is there scope for deferring features until future releases of software?
- Do you understand which functional and operational principles absolutely cannot be compromised, no matter what the resource impact?

Checklist for Architecture Definition

- Is your architecture based on technologies already familiar to the developer community?
- Is your architecture based on proven technologies as opposed to innovative ones?
- Have you assessed your architecture against existing infrastructure capabilities (e.g. desktop platforms) to see whether upgrades are required?
- Have you included in plans the costs of additional infrastructures for disaster recovery, support, acceptance, and training?
- Where new or unfamiliar technologies are used, have you considered the impacts of staff training and support?
- Is your architecture simple enough to be built and supported by development/operations staff who have only recently been trained?

Evolution Perspective

The Evolution perspective addresses the concerns related to dealing with evolution during the lifetime of a system and thus is relevant to most large-scale information systems because of the amount of change that most systems need to handle.

Desired Quality	The ability of the system to be flexible in the face of the inevitable change that all systems experience after deployment, balanced against the costs of providing such flexibility
Applicability	Important for all systems to some extent; more important for longer lived and more widely used systems
Concerns	Magnitude of change, dimensions of change, likelihood of change, timescale for change, when to pay for change, development complexity, preservation of knowledge, and reliability of change
Activities	Characterize the evolution needs, assess the current ease of evolution, consider the evolution tradeoffs, and rework the architecture
Tactics	Contain change, create flexible interfaces, apply change-oriented architectural styles, build variation points into the software, use standard extension points, achieve reliable change, and preserve development environments
Problems & Pitfalls	Prioritization of the wrong dimensions, changes that never happen, impacts of evolution on critical quality properties, lost development environments, and ad hoc release management

Applicability to Views

Functional	If the evolution required is significant, the functional structure will need to reflect this.
Information	If environment or information evolution is needed, a flexible information model will be required.
Concurrency	Evolutionary needs may dictate particular element packaging or some constraints on the concurrency structure (e.g., that it must be very simple).
Development	Evolution requirements may have a significant impact on the development environment that needs to be defined (e.g., enforcing portability guidelines).
Deployment	This perspective rarely has a significant impact on the Deployment view because system evolution usually affects structures described in other views.
Operational	This perspective typically has less impact on the Operational view.

Checklist for Requirements Capture

- Have you considered which evolutionary dimensions are most important for your system?
- Are you confident that you have done enough analysis to confirm that your prioritization of evolutionary dimensions is valid?
- Have you identified specific changes that will be required and the magnitude of each?
- Have you assessed the likelihood of each of your changes actually being needed?

Checklist for Architecture Definition

- Have you performed an architectural assessment to establish whether your architecture is sufficiently flexible to meet the evolutionary needs of your system?
- Where change is likely, does your architectural design contain the change as far as possible?
- Have you considered choosing an inherently change-oriented architectural style? If so, have you assessed the costs of doing so?
- Have you traded off the costs of your support for evolution against the needs of the system as a whole? Are any critical quality properties negatively impacted by the design you have adopted?
- Have you designed the architecture to accommodate only those changes you are confident will be needed?
- Can you recreate your development and test environments reliably?
- Can you reliably and repeatedly build, test, and release your system?
- Is your chosen evolutionary approach the cheapest and least risky option of delivering the initial system and the future evolution required?

Internationalisation Perspective

The Internationalization perspective is important for any system that will have users who speak different languages or come from different countries. If systems are aimed at a specific locale with no plans to move it into a wider area, this perspective has limited relevance.

Desired Quality	The ability of the system to be independent from any particular language, country, or cultural group
Applicability	Any system that may need to be accessed by users or operational staff from different cultures or parts of the world, or in multiple languages, either now or in the future
Concerns	Character sets, text presentation and orientation, specific language needs, cultural norms, automatic translation, and cultural neutrality
Activities	Identification of system touch points, identification of regions of concern, internationalization of code, and localization of resources
Tactics	Separation of presentation and content, use of message catalogs, system-wide use of suitable character sets (e.g., Unicode), and specialized display and presentation hardware
Problems & Pitfalls	Platforms not available in required locales, initial consideration of similar languages only, internationalization performed late in the development process, incompatibilities between locales on servers

Applicability to Views

Functional	The functional structure may need to reflect how presentation is separated from content. General functionality should be independent of location.
Information	The Information view defines which stored information needs to be internationalized and how this will be achieved.
Concurrency	This perspective has minimal impact on the Concurrency view.
Development	The Development view will need to reflect the impact of these factors on the development environment. (e.g. internationalized test data or user message catalogues).
Deployment	The deployment environment may need to take into consideration such items as internationalized input and presentation devices.
Operational	The Operational view may need to consider what functionality is provided to support the maintenance and administration of localized information and services, and how support will be provided to different locations.

Checklist for Requirements Capture

- Have you agreed with stakeholders on the extent to which systems must be operable in different languages or countries, either now or in the future?
- Have you considered all points at which the system has any human interaction? For example, have you considered operational management and monitoring of the system or printed forms sent to customers to be filled in?
- Have you identified whether there is a requirement for non-Western character sets such as Kanji, which have special requirements for entry and presentation of data?
- Does your analysis consider all types of interaction—screens, keyboards, printed reports, and so on?
- If the system needs to convert between different units of measurement, have you considered how this will be done while retaining suitable data precision?

Checklist for Architecture Definition

- How confident are you that the architecture will meet all the requirements? Where you are not, are mitigating activities in place (such as a proof-of-concept)?
- Do the interactive elements of your architecture sufficiently separate presentation and content to meet the system's internationalization objectives?
- If non-Western character sets such as Kanji must be supported, do your input and output devices accommodate these?
- If standard text must be presented in multiple languages, have you designed facilities for maintaining such information?
- Does your system sizing take into consideration the extra capacity (disk storage, network bandwidth, and so on) required for multibyte character sets?

Location Perspective

The Location perspective addresses the problems that arise when systems or system elements are physically distant from one another. If all elements are located in the same place, you can usually disregard this perspective.

Desired Quality	The ability of the system to overcome problems brought about by the absolute location of its elements and the distances between them
Applicability	Any system whose elements (or other systems with which it interacts) are or may be physically far from one another
Concerns	Time zones of operation, network link characteristics, resiliency to link failures, wide-area interoperability, high-volume operations, intercountry concerns (political, commercial, and legal), and physical variations between locations
Activities	Geographical mapping, estimation of link quality, estimation of latency, benchmarking, and modeling of geographical characteristics
Tactics	Avoidance of widely distributed transactions, architectural plans for wide-area link failure, and allowance for offline operation
Problems & Pitfalls	Invalid (wide-area) network assumptions; assumption of single point administration; assumption of one primary time zone; assumption of end-to-end security; assumption of an overnight batch period; failure to consider political, commercial, or legal differences; and assumption of a standard physical environment

Applicability to Views

Functional	The Functional view is often presented independently of real-world location concerns; typically, these are modelled in the Deployment view.
Information	If data is highly distributed, the Information view should describe how information is kept synchronized, what update latencies are expected, how temporary discrepancies are handled, and how information is transformed between locations.
Concurrency	Concurrent processing across highly distributed parts of the system is likely to be problematic so the Concurrency view will need to reflect this.
Development	If system development is spread over multiple locations, the Development view needs to explain how software will be managed, integrated, and tested.
Deployment	The Deployment view must consider significant issues such as latency, lead times, and costs that are often associated with the rollout of wide-area networks.
Operational	The Operational view needs to consider how widely distributed systems are monitored, managed, and repaired.

Checklist for Requirements Capture

- Have you agreed on the physical location of each component of the architecture?
- Do you understand the requirements for throughput, response time, availability, and resilience for all connections between geographically distributed components?
- Are the performance and reliability expectations of the wide-area network realistic and achievable within the time and budget constraints?
- Have you understood how the system will accommodate multiple time zones? Does this include consideration of online and batch modes of working?
- Have the bandwidth and response time requirements of high-volume operations such as distributed backups or distributed software updates been understood and approved?
- If there is a requirement to support offline operation when wide-area connectivity is not available, are the service-level requirements for these clear and achievable?
- Do the requirements account for the legal and political situations in different countries?
- Has the wide-area network infrastructure been factored into disaster recovery plans?

Checklist for Architecture Definition

- How confident are you that the architecture will meet all the requirements? Where you are not, are mitigating activities in place (such as a proof-of-concept)?
- Have you identified all points at which network protocol translations need to take place (e.g., TCP/IP to SPX/IPX), and does the architecture allow for this?
- If there is a requirement to support remote offline operation, does the architecture incorporate suitable features to later recover and resubmit information?
- Do the disaster recovery features of the architecture extend to wide-area connectivity?

Performance and Scalability Perspective

This perspective helps you to address the two related quality properties of performance and scalability. These properties are important because, in large systems, they can cause more unexpected, complex, and expensive problems late in the system lifecycle than most of the other properties combined.

Desired Quality	The ability of the system to predictably execute within its mandated performance profile and to handle increased processing volumes
Applicability	Any system with complex, unclear, or ambitious performance requirements; systems whose architecture includes elements whose performance is unknown; and systems where future expansion is likely to be significant
Concerns	Response time, throughput, scalability, predictability, hardware resource requirements, and peak load behavior
Activities	Capture the performance requirements, create the performance models, analyze the performance models, conduct practical testing, assess against the requirements, and rework the architecture
Tactics	Optimize repeated processing, reduce contention via replication, prioritize processing, consolidate related workloads, distribute processing over time, minimize the use of shared resources, partition and parallelize, use asynchronous processing, and make design compromises
Problems & Pitfalls	Imprecise performance and scalability goals, unrealistic models, use of simple measures for complex cases, inappropriate partitioning, invalid environment and platform assumptions, too much indirection, concurrency-related contention, careless allocation of resources, and disregard for network and in-process invocation differences

Applicability to Views

Functional	May reveal the need for changes and compromises to your ideal functional structure; functional models also provide input to the creation of performance models.
Information	Allows identification of shared resources and the transactional requirements of each. May suggest possible replication or distribution approaches.
Concurrency	Problems such as contention may cause concurrency redesign. View content may provide elements of performance models and calibration metrics.
Development	May need to contain performance and scalability patterns and anti-patterns.
Deployment	Performance models and calibration metrics are derived from the Deployment view. Applying this perspective will often suggest changes to the deployment environment.
Operational	Highlights the need for performance monitoring and management capabilities.

Checklist for Requirements Capture

- Have you identified approved performance targets, at a high level at least, with key stakeholders?
- Have you considered targets for both response time and throughput?
- Do your targets distinguish between observed performance (i.e., synchronous tasks) and actual performance (i.e., taking asynchronous activity into account)?
- Have you assessed your performance targets for reasonableness?
- Have you appropriately set expectations among your stakeholders of what is feasible in your architecture?
- Have you defined all performance targets within the context of a particular load on the system?

Checklist for Architecture Definition

- Have you identified the major potential performance problems in your architecture?
- Have you performed enough testing and analysis to understand the likely performance characteristics of your system?
- Do you know what workload your system can process? Have you prioritized the different classes of work?
- Do you know how far your proposed architecture can be scaled without major changes?
- Have you identified and validated the performance-related assumptions made?
- Have you reviewed your architecture for common performance pitfalls?

Regulation Perspective

Unlike other system qualities, compliance with the law is an area where you cannot make compromises. Although you may be able to live with a system that is slow, occasionally unreliable, or potentially insecure, a system that does not comply with legal regulations may be prevented from going into production or may expose the organization to risk of prosecution.

Desired Quality	The ability of the system to conform to local and international laws, quasi-legal regulations, company policies, and other rules and standards
Applicability	Any system that may be subject to laws or regulations
Concerns	Statutory industry regulation, privacy and data protection, cross-border legal restrictions, data retention and accountability, and organizational policy compliance
Activities	Compliance auditing
Tactics	Assessment of architecture against regulatory and legislative requirements
Problems & Pitfalls	Not understanding regulations or resulting obligations, and being unaware of statutory regulations

Applicability to Views

Functional	Regulations can have a significant impact on what the system does and how it works.
Information	Especially in Europe, there is a great deal of legislation related to the retention, use, and manipulation of personal information. The impact on the Information view may include privacy, access control, retention and archive, audit, availability, and distribution.
Concurrency	This perspective has little or no impact on the Concurrency view.
Development	This perspective has little or no impact on the Development view, although if production (live) test data is to be used, there may be restrictions on this.
Deployment	This perspective has little or no impact on the Deployment view, although health and safety legislation could have an impact on the hardware deployed.
Operational	This perspective has little or no impact on the Operational view.

Checklist for Requirements Capture

- Have you identified all legislation that applies to the functionality the system supports (e.g., employment law for a human resources system, or company law for a financial system) and assessed the architecture for compliance with these?
- Have you identified the generic legislation that applies to software systems (e.g., health and safety, the environment, data protection) and assessed the architecture for compliance with these?
- Have you determined whether the system can be considered as touching on other countries in any way, and if so, what legislation it may be subject to as a result?
- Have you considered international law such as technology export restrictions?
- Have you identified the relevant internal business and technology regulations and standards? Have you assessed the architecture for compliance with these?
- If legislation requires registration with governmental agencies (e.g., the Data Protection Registrar in the United Kingdom), have you applied for this registration, or do you have plans to make this happen?
- Do your archive and retention plans conform to all applicable legislation?

Checklist for Architecture Definition

- Does your architecture accommodate any required automated interfaces to regulatory bodies (e.g., automatic upload of accounting or taxation information)? Do these interfaces conform to prescribed business and technical standards?
- Does the architecture conform to any mandated technical standards?

Security Perspective

The security perspective guides you as you consider the set of processes and technologies that allow the owners of resources in the system to reliably control who can perform what actions on particular resources.

Desired Quality	The ability of the system to reliably control, monitor, and audit who can perform what actions on these resources and the ability to detect and recover from failures in security mechanisms
Applicability	Any systems with publicly accessible interfaces, with multiple users where the identity of the user is significant, or where access to operations or information needs to be controlled
Concerns	Policies, threats, mechanisms, accountability, availability, and detection and recovery
Activities	Identify sensitive resources, define the security policy, identify threats to the system, design the security implementation, and assess the security risks
Tactics	Apply recognized security principles, authenticate the principals, authorize access, ensure information secrecy, ensure information integrity, ensure accountability, protect availability, integrate security technologies, provide security administration, and use third-party security infrastructure
Problems & Pitfalls	Complex security policies, unproven security technologies, system not designed for failure, lack of administration facilities, technology-driven approach, failure to consider time sources, overreliance on technology, no clear requirements or models, security as an afterthought, security embedded in the application code, piecemeal security, and ad hoc security technology

Applicability to Views

Functional	Reveals which functional elements need to be protected. Functional structure may be impacted by the need to implement your security policies.
Information	Reveals what data needs to be protected. Information models are often modified as a result of security design (e.g., partitioning information by sensitivity).
Concurrency	Security design may indicate the need to isolate different pieces of the system into different runtime elements, so affecting the system's concurrency structure.
Development	Captures security related development guidelines and constraints.
Deployment	May need major changes to accommodate security-oriented hardware or software, or to address security risks.
Operational	Needs to make the security assumptions and responsibilities clear, so that these aspects of the security implementation can be reflected in operational processes.

Checklist for Requirements Capture

- Have you identified the sensitive resources contained in the system?
- Have you identified the sets of principals that need access to the resources?
- Have you identified the system's needs for information integrity guarantees?
- Have you identified the system's availability needs?
- Is there a security policy, including access control and information integrity needs?
- Is the security policy as simple as possible?
- Have you worked through a formal threat model to identify security risks?
- Have you worked through example scenarios with your stakeholders so that they understand the planned security policy and the security risks the system runs?
- Have you reviewed your security requirements with external experts?

Checklist for Architecture Definition

- Have you addressed each threat identified in the threat model to the extent required?
- Have you used as much third-party security technology as possible?
- Have you produced an integrated overall design for the security solution?
- Have you considered all standard security principles when designing the infrastructure?
- Is your security infrastructure as simple as possible?
- Have you defined how to identify and recover from security breaches?
- Have you applied the results of the Security perspective to all of the affected views?
- Have external experts reviewed your security design?

Usability Perspective

Applying the Usability perspective ensures that the system allows those who interact with it to do so effectively. This perspective tends to focus on the end users of the system but should also address the concerns of any others who interact with it directly or indirectly, such as maintainers and support personnel.

Desired Quality	The ease with which people who interact with the system can work effectively
Applicability	Any system that has significant interaction with humans (users, operational staff, and so on) or that is exposed to members of the public
Concerns	User interface usability, business process flow, information quality, alignment of the human–computer interface (HCI) with working practices, alignment of the HCI with users' skills, maximization of the perceived usability, and ease of changing user interfaces
Activities	User interface design, participatory design, interface evaluation, and prototyping
Tactics	Separation of user interface from functional processing
Problems & Pitfalls	Failure to consider user capabilities, failure to use non-IT communication specialists, failure to consider how concerns from other perspectives affect usability, overly complex interfaces, assumption of a single type of user access, design based on technology rather than needs, inconsistent interfaces, disregard for organizational standards, and failure to separate interface and processing implementations

Applicability to Views

Functional	The functional structure indicates where the system's external interfaces are and thus where usability needs to be considered. It may be impacted by usability needs (e.g., the addition of interface services to support certain interaction styles) but is unlikely to be changed significantly.
Information	Information quality (the provision of accurate, relevant, consistent, and timely data) can have a large impact on usability.
Concurrency	This perspective typically has little or no impact on the Concurrency view.
Development	The results of applying the Usability perspective impact the Development view in terms of the guidelines, standards, and patterns that ensure the creation of a consistent and appropriate set of user interfaces for the system.
Deployment	This perspective has little or no impact on the Deployment view, although usability concerns could require changes to element deployment (e.g., due to response time requirements).
Operational	The Usability perspective should consider the usability needs of the system's administrators.

Checklist for Requirements Capture

- Have you identified all of the system's key touch points?
- Have you identified all of the different types of users who will interact with the system?
- Do you understand the type of usage (occasional, regular, transactional, unstructured) for each of the touch points?
- Have you taken into account the needs of support and maintenance staff and other second-line users?
- Do you understand the capabilities, experience, and expertise of the system's users? Have you correctly mapped these into requirements for presentation and support?
- Have you taken into account any corporate standards for presentation and interaction, particularly for systems exposed to the public?

Checklist for Architecture Definition

- For Web and mobile platforms, have you considered the variation in bandwidth, hardware capabilities (screen resolution), and rendering software?
- Do the interface designs align in a sensible way with the business processes they are automating?
- If your system is exposed to the general public, have you obtained any necessary approvals from your marketing department for the use of company logos and so on?