Architecture & Design

Architecture & Design

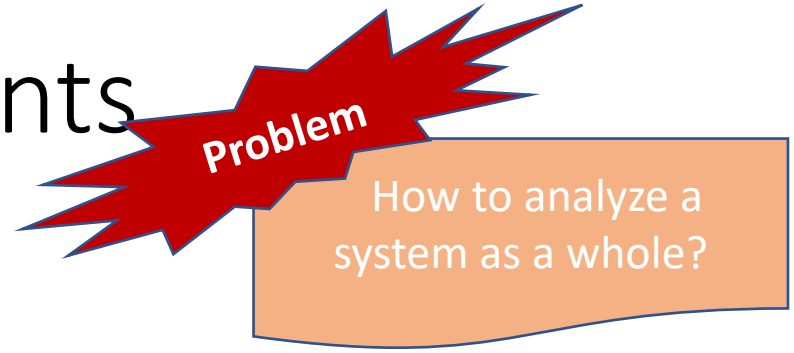
Type of systems to architect

- New Product (from scratch)
- Product in maintenance / enhancement stage
- Migration

Architectural Perspectives

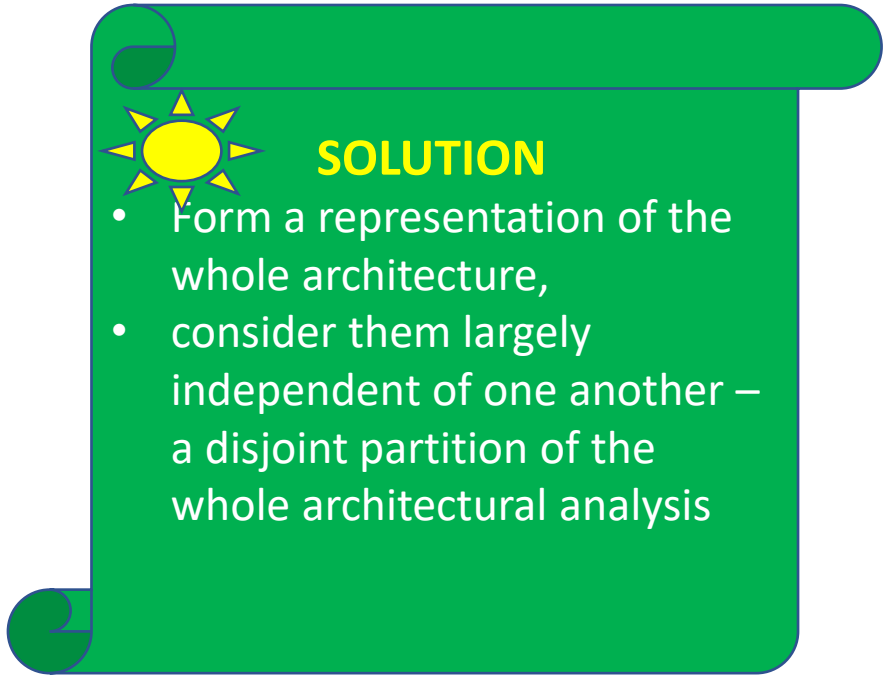
- An architectural perspective is a collection of
  - activities,
  - tactics, and
  - guidelines that are used to ensure that a system exhibits a particular set of related quality properties that require consideration across a number of the system’s architectural views.

Architectural Views, Viewpoints



- Viewpoints
- 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.
  - Eg: Functional, Informational, Deployment, security

- Views are disjoint partitions with a focus to
  - Issues
  - Concerns
  - Solutions
    - Eg: For Information View:
      - Information Structure
      - Ownership
      - Transactional Integrity
      - Data Quality
      - Timelines



View Catalog

View Name	Definition
Context	Describes the relationships, dependencies, and interactions between the system and its environment (the people, systems, and external entities with which it interacts). Many architecture descriptions focus on views that model the system's internal structures, data elements, interactions, and operation. Architects tend to assume that the "outward-facing" information — the system's runtime context, its scope and requirements, and so forth – is clearly and unambiguously defined elsewhere. However, you often need to include a definition of the system's context as part of your architectural description.
Functional	Describes the system's functional elements, their responsibilities, interfaces, and primary interactions. A Functional view is the cornerstone of most ADs and is often the first part of the description that stakeholders try to read. It drives the shape of other system structures such as the information structure, concurrency structure, deployment structure, and so on. It also has a significant impact on the system's quality properties such as its ability to change, its ability to be secured, and its runtime performance.
Information	Describes the way that the architecture stores, manipulates, manages, and distributes information. The ultimate purpose of virtually any computer system is to manipulate information in some form, and this viewpoint develops a complete but high-level view of static data structure and information flow. The objective of this analysis is to answer the big questions around content, structure, ownership, latency, references, and data migration.

View Catalog contd…

View Name	Definition
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. This entails the creation of models that show the process and thread structures that the system will use and the interprocess communication mechanisms used to coordinate their operation.
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, including capturing the dependencies the system has on its runtime environment. This view captures the hardware environment that your system needs (primarily the processing nodes, network interconnections, and disk storage facilities required), the technical environment requirements for each element, and the mapping of the software elements to the runtime environment that will execute them.
Operational	Describes how the system will be operated, administered, and supported when it is running in its production environment. For all but the simplest systems, installing, managing, and operating the system is a significant task that must be considered and planned at design time. The aim of the Operational viewpoint is to identify system-wide strategies for addressing the operational concerns of the system's stakeholders and to identify solutions that address these.

Context Viewpoint

Definition	Describes the relationships, dependencies, and interactions between the system and its environment (the people, systems, and external entities with which it interacts)
Concerns	▪ system scope and responsibilities ▪ identity of external entities and services and data used ▪ nature and characteristics of external entities ▪ identity and responsibilities of external interfaces ▪ nature and characteristics of external interfaces ▪ other external interdependencies ▪ impact of the system on its environment ▪ overall completeness, consistency, and coherence
Stakeholders	All stakeholders, but especially acquirers, users, and developers
Applicability	All systems

Back

Functional Viewpoint

Definition	Describes the system's runtime functional elements and their responsibilities, interfaces, and primary interactions
Concerns	▪ functional capabilities ▪ external interfaces ▪ internal structure ▪ functional design philosophy
Models	▪ functional structure model
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 ▪ wrong level of detail ▪ 'God' elements ▪ too many dependencies
Stakeholders	All stakeholders
Applicability	All systems

Back

Information Viewpoint

Definition	Describes the way that the system stores, manipulates, manages, and distributes information
Concerns	▪ information structure and content ▪ information purpose and usage ▪ information ownership ▪ enterprise-owned information ▪ identifiers and mappings ▪ volatility of information semantics ▪ information storage models ▪ information flow ▪ information consistency ▪ information quality ▪ timeliness, latency, and age ▪ archiving and information retention
Models	▪ static information structure models ▪ information flow models ▪ information lifecycle models ▪ information ownership models ▪ information quality analysis ▪ metadata models ▪ volumetric models
Pitfalls	▪ representation incompatibilities ▪ unavoidable multiple updaters ▪ key-matching deficiencies ▪ interface complexity ▪ overloaded central database ▪ inconsistent distributed databases ▪ poor information quality ▪ excessive information latency ▪ inadequate volumetrics
Stakeholders	Primarily users, acquirers, developers, testers, and maintainers, but most stakeholders have some level of interest
Applicability	Any system that has more than trivial information management needs

Back

Concurrency Viewpoint

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 ▪ supporting scalability ▪ startup and shutdown ▪ task failure ▪ reentrancy
Models	▪ system-level concurrency models ▪ state models
Pitfalls	▪ modeling the wrong concurrency ▪ modeling the concurrency wrongly ▪ excessive complexity ▪ resource contention ▪ deadlock ▪ race conditions
Stakeholders	Communicators, developers, testers, and some administrators
Applicability	All information systems with a number of concurrent threads of execution

Back

Development Viewpoint

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

Back

Deployment Viewpoint

Definition	Describes the environment into which the system will be deployed, including the dependencies the system has on its runtime environment
Concerns	▪ runtime platform required ▪ specification and quantity of hardware or hosting required ▪ third-party software requirements ▪ technology compatibility ▪ network requirements ▪ network capacity required ▪ physical constraints
Models	▪ runtime platform models ▪ network models ▪ technology dependency models ▪ intermodel relationships
Pitfalls	▪ unclear or inaccurate dependencies ▪ unproven technology ▪ unsuitable or missing service-level agreements ▪ lack of specialist technical knowledge ▪ late consideration of the deployment environment ▪ ignoring intersite complexities ▪ inappropriate headroom provision ▪ not specifying a disaster recovery environment
Stakeholders	System administrators, developers, testers, communicators, and assessors
Applicability	Systems with complex or unfamiliar deployment environments

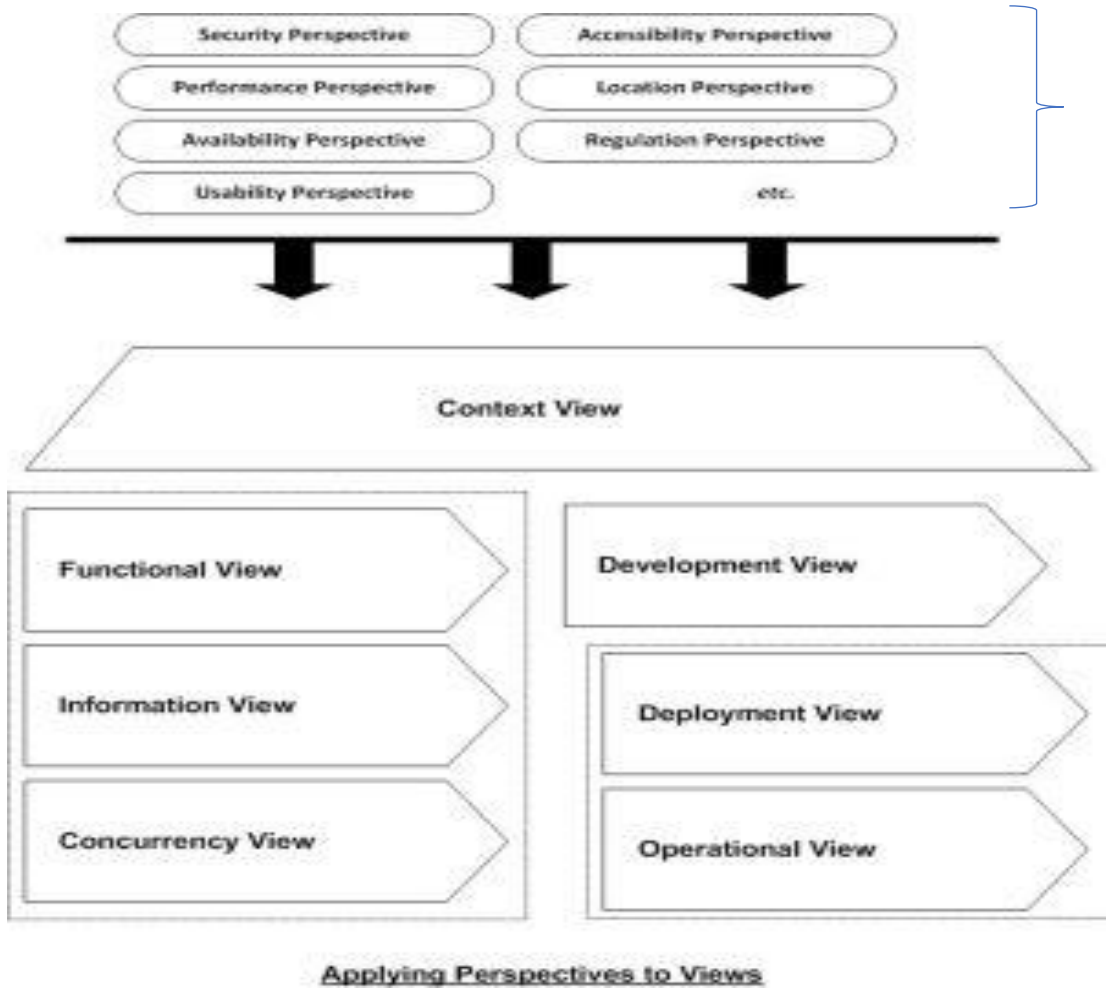
Back

Operational Viewpoint

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 ▪ alerting ▪ configuration management ▪ performance monitoring ▪ support ▪ backup and restore ▪ operation in third-party environments
Models	▪ installation models ▪ migration models ▪ configuration management models ▪ administration models ▪ support models
Pitfalls	▪ lack of engagement with the operational staff ▪ lack of backout planning ▪ lack of migration planning ▪ insufficient migration window ▪ missing management tools ▪ production environment constraints, ▪ lack of integration into the production environment ▪ inadequate backup models ▪ unsuitable alerting
Stakeholders	System administrators, production engineers, developers, testers, communicators, and assessors
Applicability	Any system being deployed into a complex or critical operational environment

Back

Architectural Perspective



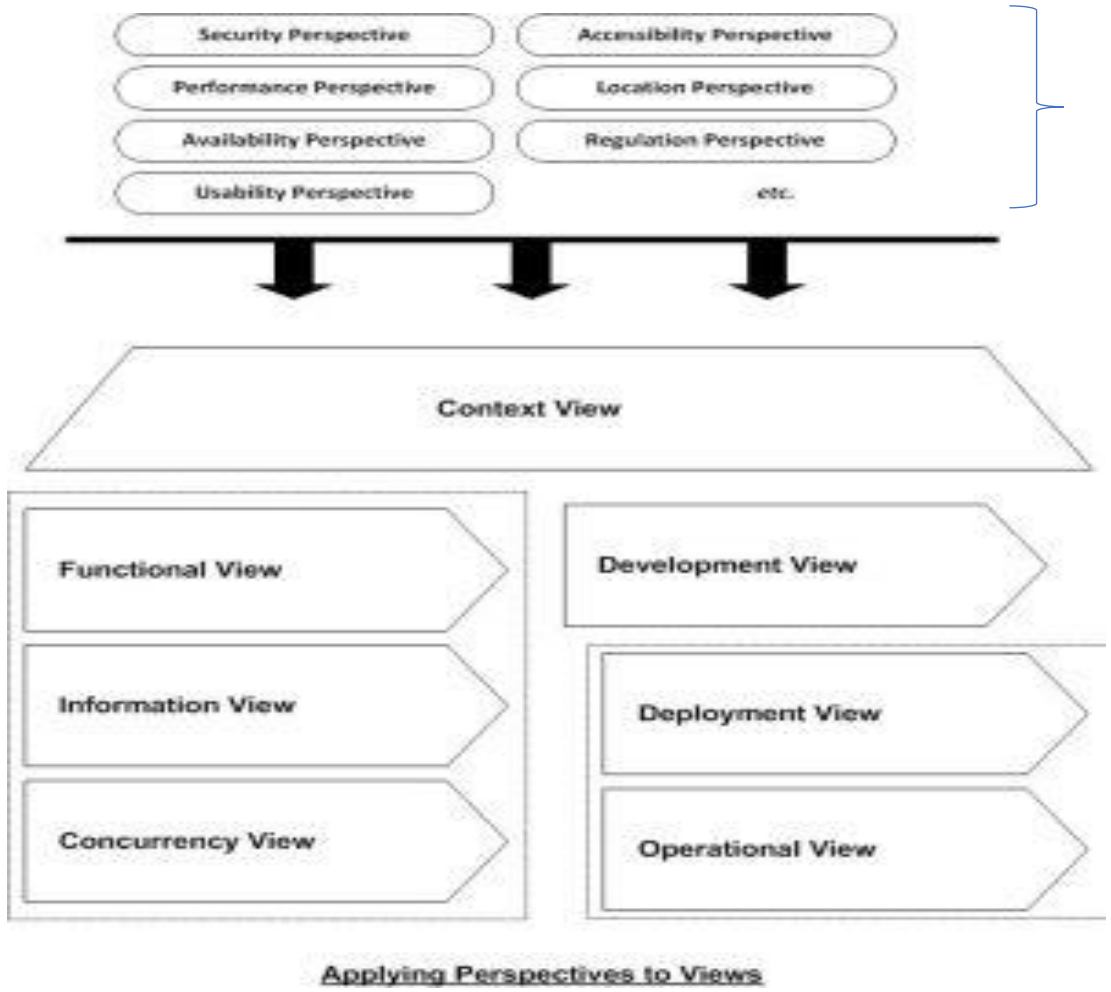
Applying Perspectives to Views

The issues addressed by perspectives are often referred to as ***nonfunctional requirements*** (NFRs) of the architecture

How do Perspectives help?

1. A perspective provides a framework to guide and formalize this process
2. You never work with perspectives in isolation but instead use them with each view to analyze and validate the qualities of your architecture and to drive further architectural decision making.
3. We describe this as ***applying the perspective to the view***.

Architectural Perspective



Applying Perspectives to Views

The issues addressed by perspectives are often referred to as ***nonfunctional requirements*** (NFRs) of the architecture

For a good Architect, use perspectives to

1. Understand the quality properties that are required
2. Assess and review the architectural models to ensure that the architecture exhibits the required properties
3. Identify, prototype, test, and select architectural tactics to address cases when the architecture is lacking

Architecture Perspective Catalog

Perspective Name	Definition
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 around people, budget, time, and materials
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

Accessibility Perspective

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

Back

Availability & Resilience Perspective

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 ▪ disaster recovery
Activities	▪ capture the availability requirements ▪ produce the availability schedule ▪ estimate platform availability ▪ estimate functional availability ▪ assess against the requirements ▪ rework the architecture
Tactics	▪ select fault-tolerant hardware ▪ use high-availability clustering and load balancing ▪ log transactions ▪ apply software availability solutions ▪ select or create fault-tolerant software ▪ design for failure ▪ allow for component replication ▪ relax transactional consistency ▪ identify backup and disaster recovery solutions
Pitfalls	▪ single point of failure ▪ cascading failure ▪ unavailability through overload ▪ overambitious availability requirements ▪ ineffective error detection ▪ overestimation of component resilience ▪ overlooked global availability requirements ▪ incompatible technologies

Back

Development Resource Perspective

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 ▪ external dependencies
Activities	▪ cost estimation ▪ development time estimation ▪ development planning ▪ dependency management ▪ scoping ▪ prototyping ▪ expectation management
Tactics	▪ incremental and iterative development ▪ expectation management ▪ descoping ▪ prototyping and piloting ▪ fitness for purpose
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 ▪ overallocation of staff ▪ difficulty getting access to knowledgeable business stakeholders

Back

Evolution Perspective

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	▪ product management ▪ magnitude of change ▪ dimensions of change ▪ likelihood of change ▪ timescale for change ▪ when to pay for change ▪ changes driven by external factors ▪ development complexity ▪ preservation of knowledge ▪ reliability of change
Activities	▪ characterize the evolution needs ▪ assess the current ease of evolution ▪ consider the evolution tradeoffs ▪ rework the architecture
Tactics	▪ contain change ▪ create extensible interfaces ▪ apply design techniques that facilitate change ▪ apply metamodel-based architectural styles ▪ build variation points into the software ▪ use standard extension points ▪ achieve reliable change ▪ preserve development environments
Pitfalls	▪ prioritization of the wrong dimensions ▪ changes that never happen ▪ impacts of evolution on critical quality properties ▪ overreliance on specific hardware or software ▪ lost development environments ▪ ad hoc release management

Back

Internationalization Perspective

Desired Quality	The ability of the system to be independent from any particular language, country, or cultural group
Applicability	Any system which 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 ▪ currency conversions and exchange rates ▪ cultural neutrality
Activities	▪ identification of system touch points ▪ identification of regions of concern ▪ internationalization of code ▪ localization of resources
Tactics	▪ separation of presentation and content ▪ use of message catalogs ▪ system-wide use of suitable character sets (eg Unicode) ▪ specialized display and presentation hardware ▪ currency conversion mechanisms
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 ▪ insufficient consideration of currency exchange

Back

Location 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) ▪ use of the public Internet ▪ physical variations between locations
Activities	▪ geographical mapping ▪ estimation of link quality ▪ estimation of latency ▪ benchmarking ▪ modeling of geographical characteristics
Tactics	▪ avoidance of widely distributed transactions ▪ architectural plans for wide-area link failure ▪ allowance for offline operation
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 ▪ assumption that public networks are high-bandwidth, low-latency, and highly available ▪ assumption of a standard physical environment

Performance & Scalability Perspective

Back

Desired Quality	The ability of the system to predictably execute within its mandated performance profile and to handle increased processing volumes in the future if required
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 ▪ peak load behavior
Activities	▪ capture the performance requirements ▪ create the performance models ▪ analyze the performance models ▪ conduct practical testing ▪ assess against the requirements ▪ rework the architecture
Tactics	▪ optimize repeated processing ▪ reduce contention via replication ▪ prioritize processing ▪ consolidate related workload ▪ distribute processing over time ▪ minimize the use of shared resources ▪ reuse resources and results ▪ partition and parallelize ▪ scale up or scale out ▪ degrade gracefully ▪ use asynchronous processing ▪ relax transactional consistency ▪ make design compromises
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 ▪ database contention ▪ transaction overhead ▪ careless allocation of resources ▪ disregard for network and in-process invocation differences

Regulation Perspective

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 which may be subject to laws or regulation
Concerns	▪ statutory industry regulation ▪ privacy and data protection ▪ cross-border legal restrictions ▪ data retention and accountability ▪ organizational policy compliance
Activities	▪ compliance auditing
Tactics	▪ assessment of architecture against regulatory and legislative requirements
Pitfalls	▪ not understanding regulations or resulting obligations ▪ being unaware of statutory regulations

Back

Security Perspective

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	▪ resources ▪ principals ▪ policies ▪ threats ▪ confidentiality ▪ integrity ▪ availability ▪ accountability ▪ detection and recovery ▪ security mechanisms
Activities	▪ identify sensitive resources ▪ define the security policy ▪ identify threats to the system ▪ design the security implementation ▪ assess the security risks

Back

Security Perspective contd…

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 ▪ use third-party security infrastructure
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 ▪ ignoring the insider threat ▪ assuming the client is secure ▪ security embedded in the application code ▪ piecemeal security ▪ ad hoc security technology

Back

Usability Perspective

Desired Quality	The ease with which people who interact with the system can work effectively
Applicability	Any system which has significant interaction with humans (users, operational staff etc) or which is exposed to members of the public
Concerns	▪ user interface usability ▪ business process flow ▪ information quality ▪ alignment of HCI (human-computer interface) with working practices ▪ alignment of HCI with users' skills ▪ maximization of the perceived usability ▪ ease of changing user interfaces
Activities	▪ user interface design ▪ participatory design ▪ interface evaluation ▪ prototyping
Tactics	▪ separation of user interface from functional processing
Pitfalls	▪ failure to consider user capabilities ▪ failure to use human–computer interface 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 ▪ failure to separate interface and processing implementations

Back