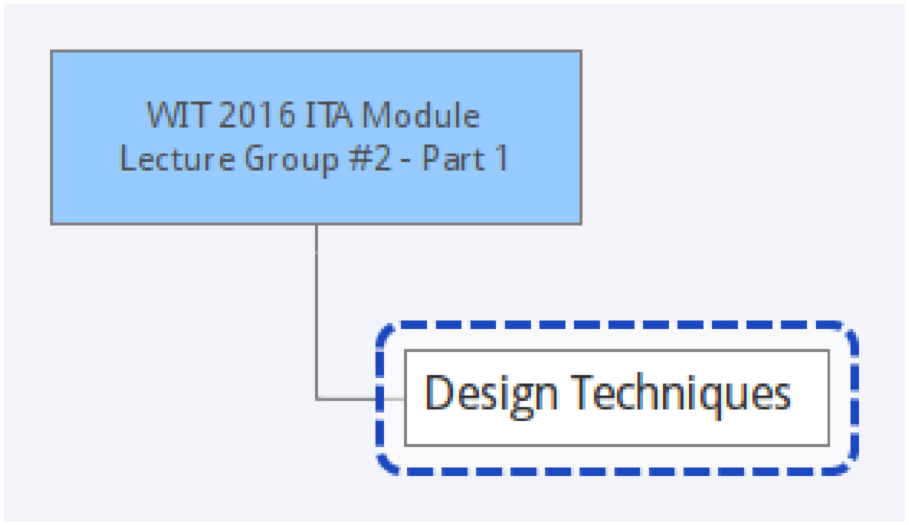
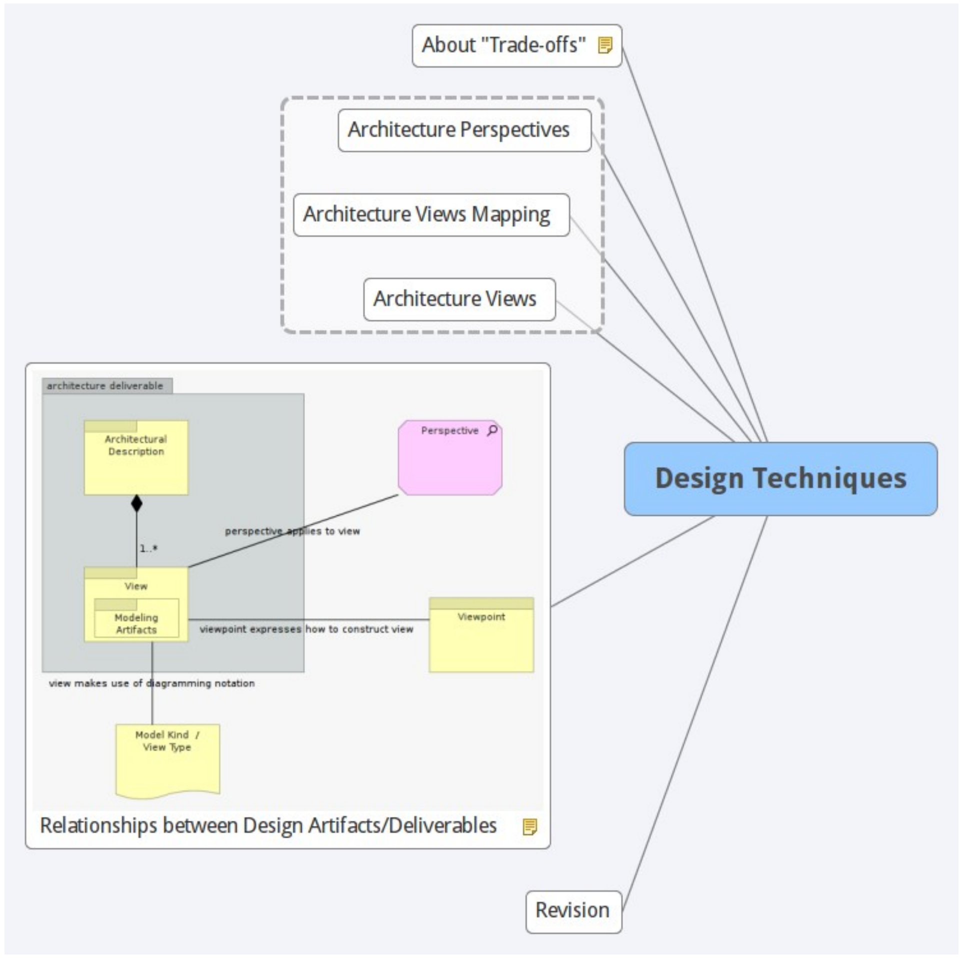
WIT 2016 ITA Module

Lecture Group #2 - Part 1b
Architecture Perspectives

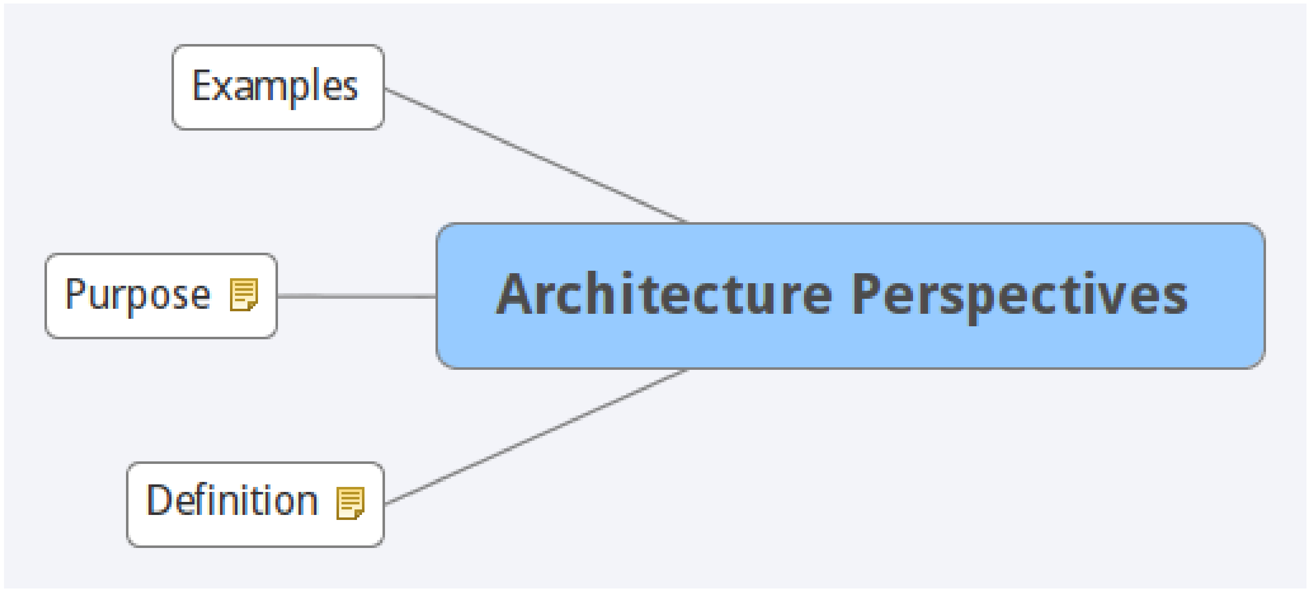
Lecture Group #2 - Part 1



Design Techniques


About "Trade-offs"
Architecture Perspectives
Architecture Views Mapping
Architecture Views
architecture deliverable
Architectural Description
Perspective
perspective applies to view
1..*
View
Modeling Artifacts
viewpoint expresses how to construct view
Viewpoint
view makes use of diagramming notation
Model Kind / View Type
Relationships between Design Artifacts/Deliverables
Design Techniques
Revision

Architecture Perspectives



Definition

Architectural Perspectives (also referred as 'abilities') ensure that a system exhibits a particular set of related quality properties that require consideration ACROSS the architectural views of a solution design.

The aim is to express the "externally-visible" quality properties expressed in the solution recommendation.

An architectural perspective is a record of field-experiences, typically a list of problems to consider, answering tactics, and guidelines.

Perspectives provide a framework to guide architects to surface gaps/risks, and challenge the design in some aspect.

Purpose

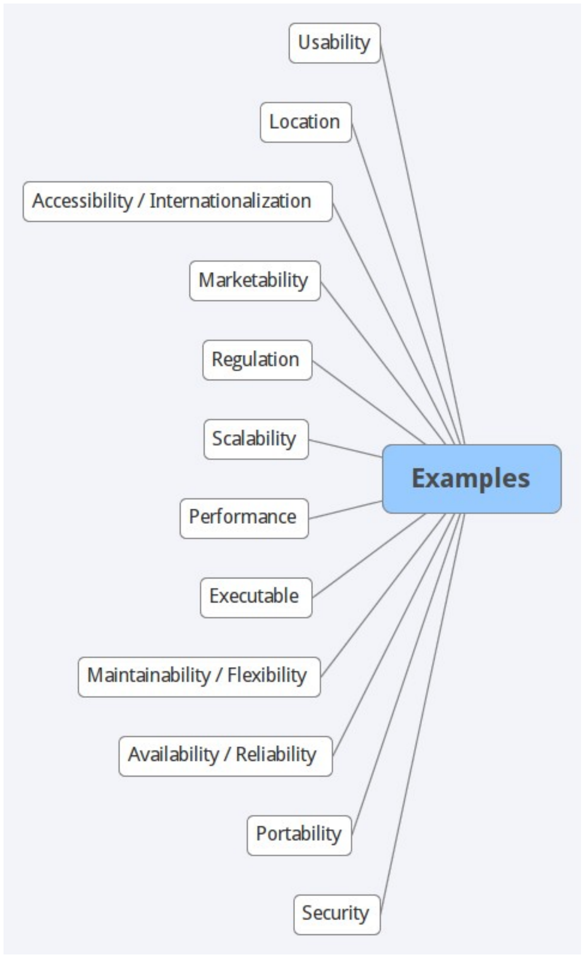
The purpose of architecture perspectives is to:

- demonstrate the FITNESS FOR PURPOSE of the solution architecture,

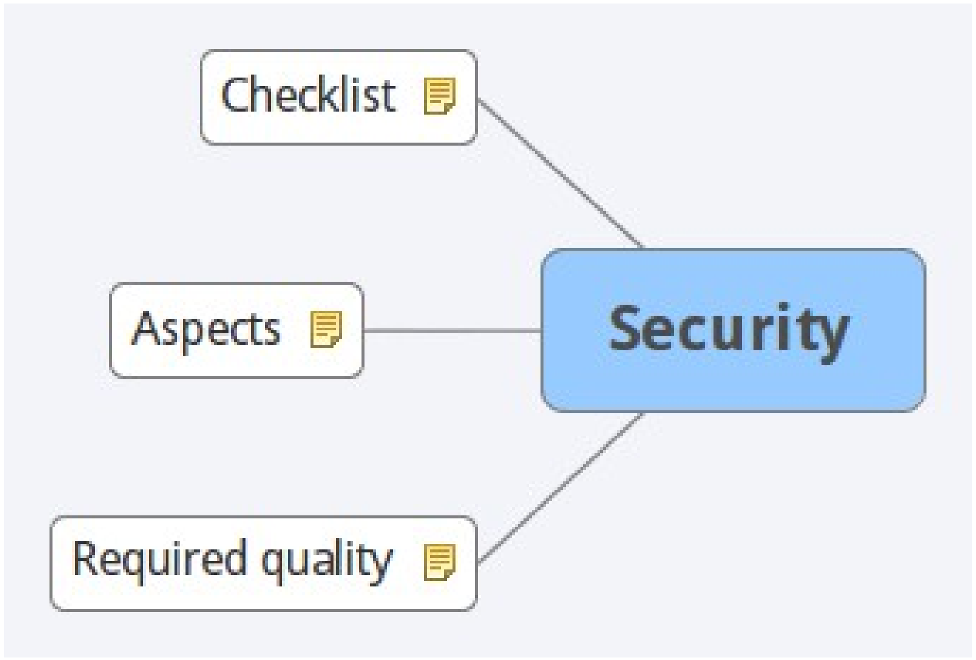
- assess and review the architectural models to ensure that the architecture exhibits the required properties,

- surface architectural gaps, design aspect overlooked by an architect.

Examples


Usability
Location
Accessibility / Internationalization
Marketability
Regulation
Scalability
Examples
Performance
Executable
Maintainability / Flexibility
Availability / Reliability
Portability
Security

Security



Required quality

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.

Aspects

Concerns: Policies, threats, mechanisms, accountability, availability, and detection and recovery.

Tactics: Threat identification, threat assessment, vulnerability analysis, application of security technology.

Pitfalls: Complex security policies, unproven security technologies, system not designed for failure, lack of administration facilities, technology-driven approach, failure to consider time sources, over reliance on technology, no clear requirements or models, security as an afterthought, security embedded in the application code, piecemeal security, and ad hoc security technology.

Checklist

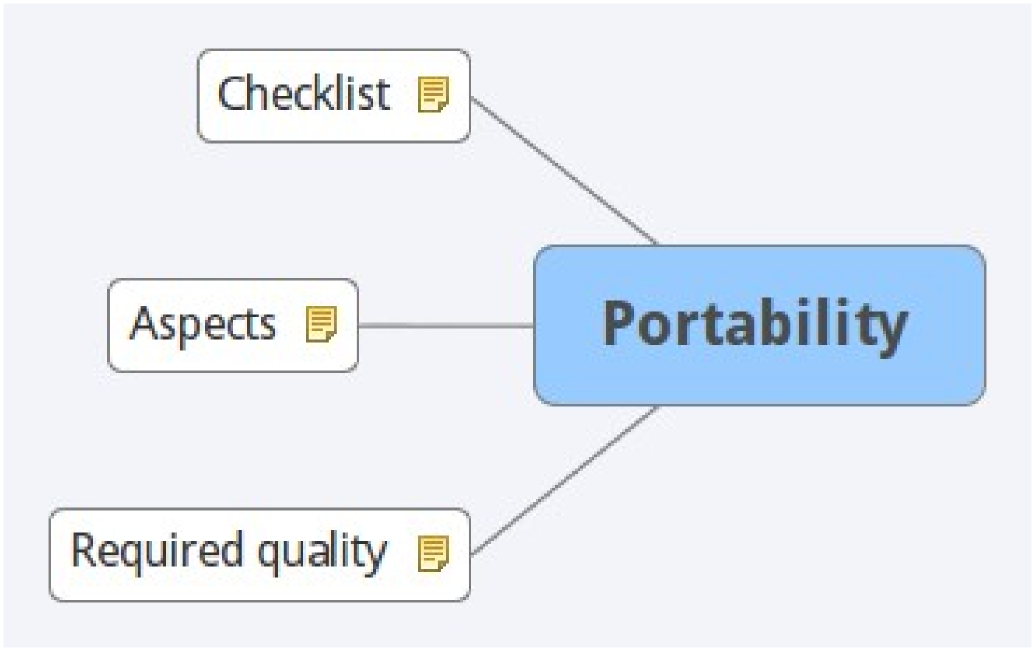
What is the authentication policy of your architecture?

What is the authorization policy of your architecture?

What provides the token to use for authentication when accessing services or systems?

What trusted subsystems are not requiring any further identification other than application credentials?

Portability



Required quality

The ability of the architecture swap from a vendor platform to another, similar vendor platform.

Aspects

Concerns: Vendor lock-in.

Common Tactics: Reliance on Open middleware Standards, Platform independent Technologies, Loose coupling.

Common Pitfalls: Over-reliance on Proprietary Software & Hardware.

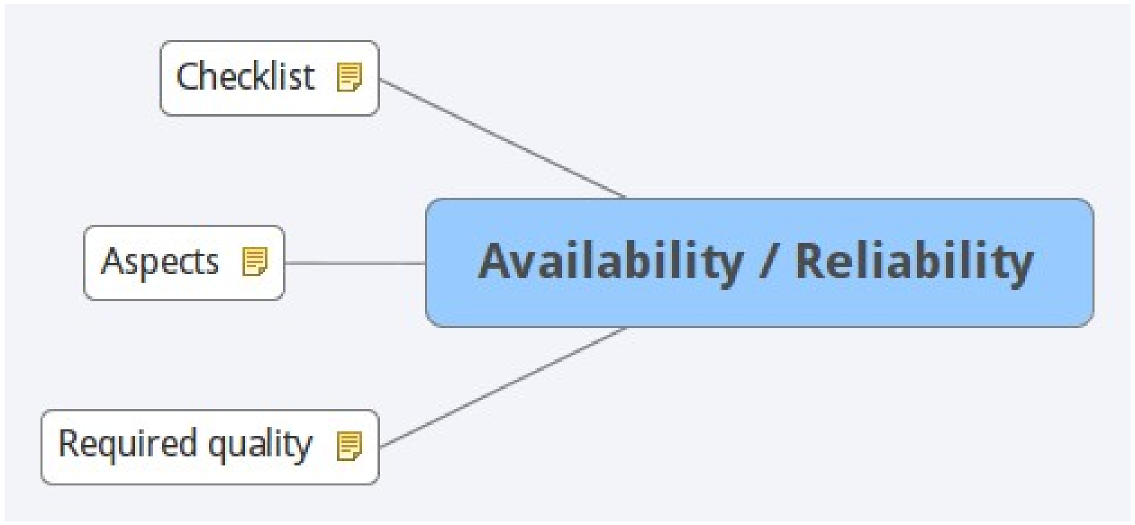
Checklist

What is the vendor platform goes out of business tomorrow?

What is the risk of the vendor going out of business tomorrow?

Can your architecture function with a vendor providing similar features (e.g. set of APIs, platform)?

Availability / Reliability



Required 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.

Aspects

Concerns: Planned / unplanned downtime, mean time between failures, mean time to repair, disaster recovery, redundancy, clustering, fail-over, fault-tolerance, transaction completion & recovery.

Common Tactics: Monitoring/Notification mechanisms, Availability Models, Queuing Mechanisms, Clustering, Fail-over mechanisms.

Common Pitfalls: No manual/automated recovery planning (even simple reboot). Over-reliance on costly Distributed Transactions, Concurrency and Resource Contention.

Checklist

What is the Service-Level Agreement between IT and users of the application?

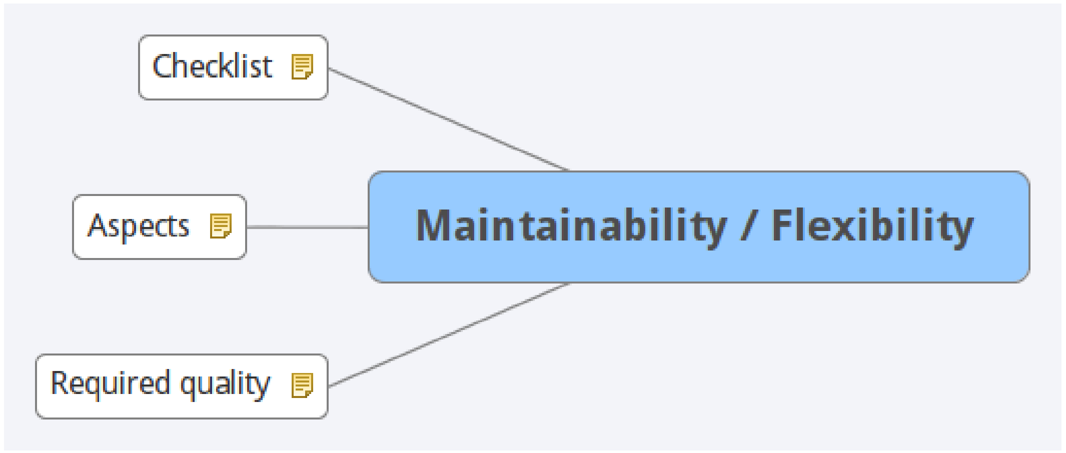
What is the recovery time?

What sub-systems the architecture requires to be available for the overall solution to remain up-and-running?

Is the architecture providing means to monitor the availability of the application?

What is the sequence of steps to follow to re-start/re-boot the deployed architecture solution?

Maintainability / Flexibility



Required 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.

Aspects

Concerns: flexibility, extensibility, functional evolution, deployment evolution, integration evolution.

Techniques: design for change, architectural assessment, configuration management, automated testing, build and release management.

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.

Checklist

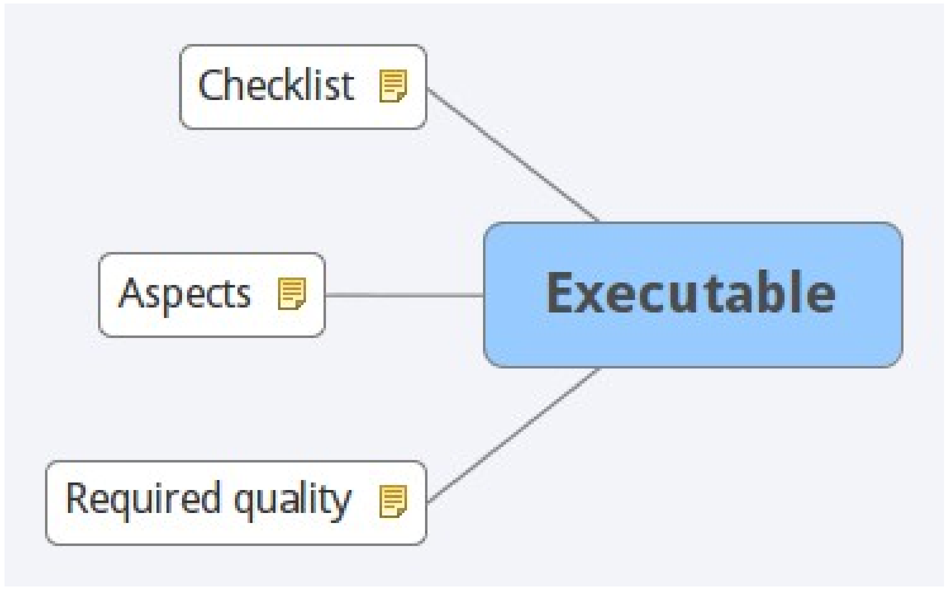
Does the architecture solution require a complete/planned re-deployment to add new features?

Has the application reached its critical mass?

Could the architecture be broken down in autonomous sub-systems / independent components?

Does the solution architecture provide for extensions?

Executable



Required quality

The ability of the system to be designed, built, deployed, and operated within known constraints around people, budget, time, and materials.

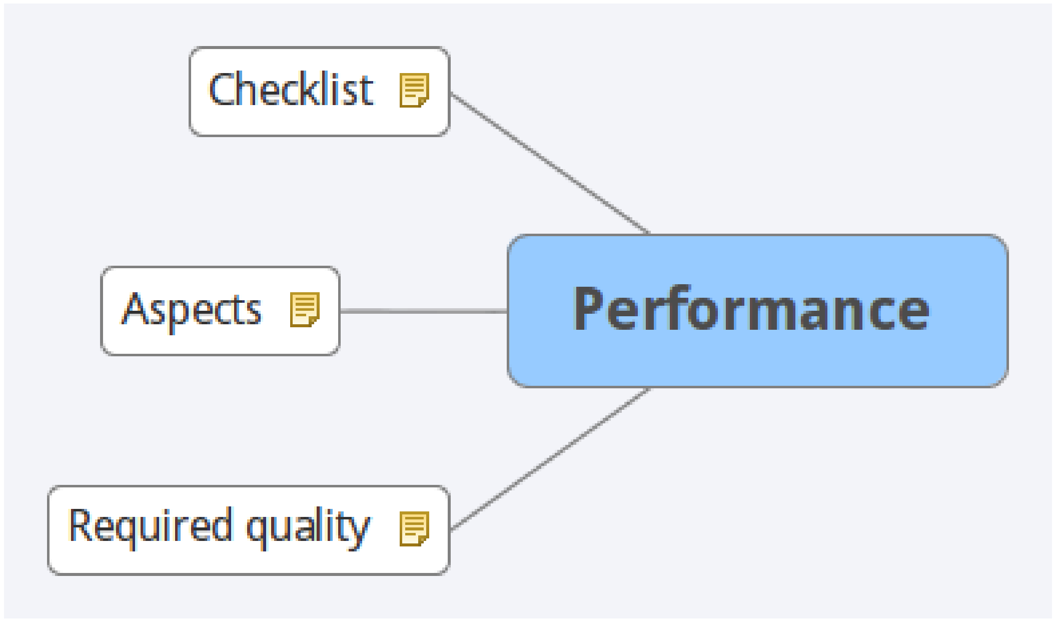
Aspects

Concerns: Time, budget, skills, people availability (internal and external).

Checklist

Can the system be built within people, time, budget constraints?

Performance



Required quality

The ability of the system to predictably execute within its mandated performance profile and to handle increased processing volumes.

Aspects

Concerns: processing volume, response time, responsiveness, throughput, predictability.

Common Tactics: Optimize repeated processing for efficiency, 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, caching, database archiving.

Common Pitfalls: Imprecise 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.

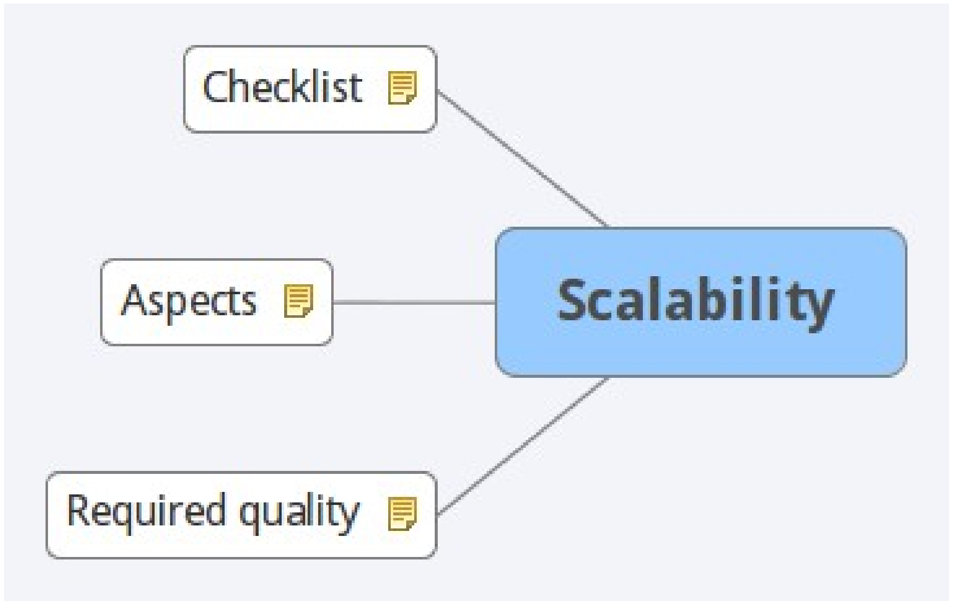
Checklist

What is the expected throughput, responsiveness of the solution architecture?

What is the correlation between throughput and response time?

Is the perception of responsiveness from a user standpoint more or less important than components response time/throughput?

Scalability



Required quality

The ability of the architecture to vertically scale, referred as scaling-up: means adding more power to a single server, such as more memory.

The ability of the architecture to horizontally scale, referred as scaling-out: means adding more servers.

Aspects

Concerns: processing volume, response time, responsiveness, throughput, predictability.

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.

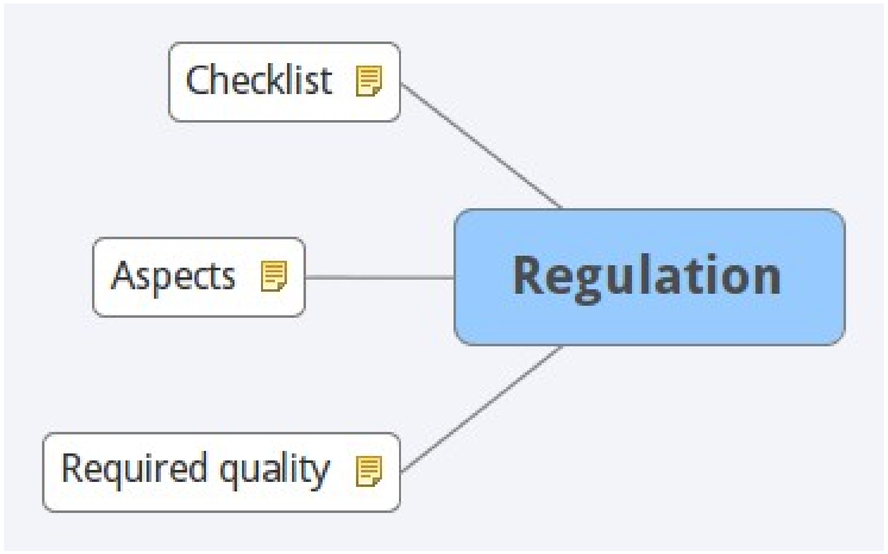
Pitfalls: Imprecise goals, unrealistic models, use of simple measures for complex cases, inappropriate partitioning, invalid environment and platform assumptions, too much indirection, concurrency-related contention, careless allocation of resources.

Checklist

What is the nature of the infrastructure / data center hosting the architecture?

Which components of the architecture be distributed?

Regulation



Required quality

The ability of the system to conform to local and international laws, quasi-legal regulations, company policies, and other rules and standards

Aspects

Concerns: Compliance with local (ex. U.S states)/international laws and regulations.

Common Tactics: Maintainability, Instrumentation, Logging, Archiving, Records Management.

Common Pitfalls: Design of short-term & punctual solutions.

Checklist

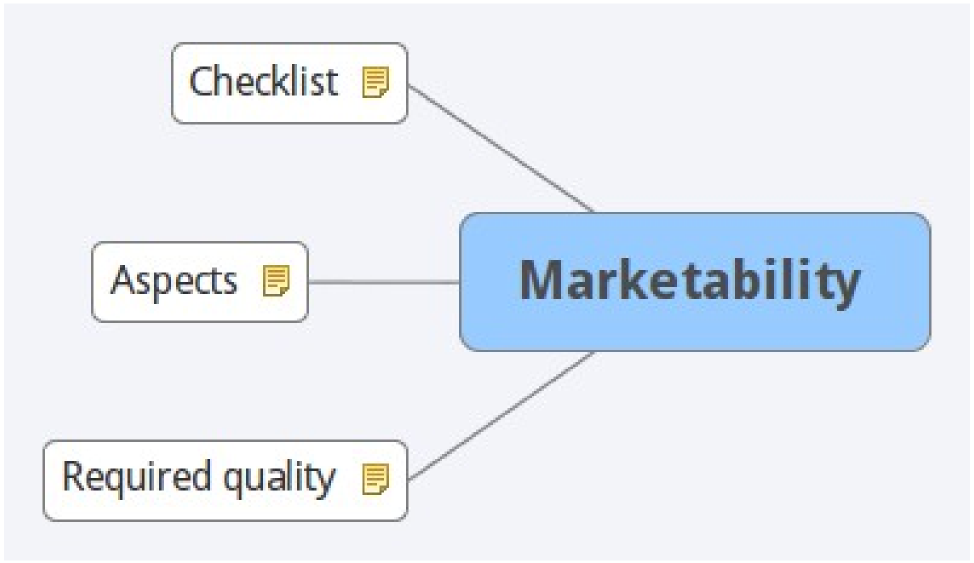
Does the architecture conform with data privacy laws?

Does the architecture meet required regulatory constraints?

Is the behavior of the solution auditable by external parties?

Should outputs of the solution architecture (ex. document contracts) be customized for geographies?

Marketability



Required quality

The ability to synchronise the delivery of features with weekly/monthly/yearly business cycles.

Aspects

Concerns: Right-time/Right-place, Speed-to-Market, Speed-to-Delivery.

Common Tactics: Architectural Scope, Architectural Increments, Lean or Agile community Processes.

Common Pitfalls: Deviation from road maps, Agile/Lean teams in non-Agile/non-Lean organizations.

Checklist

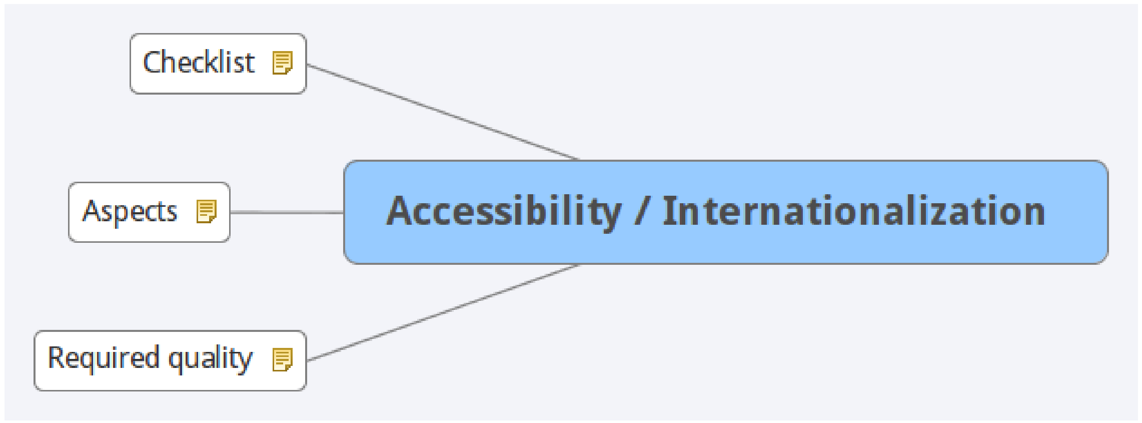
What will be the impact of the solution architecture in relation to time-to-market aspects?

What is right/sensible size of the solution to be architected given business life-cycle?

Should a smaller/bigger solution architecture be designed?

What are the consequence of not delivering the solution in time?

Accessibility / Internationalization



Required quality

The ability of the system to be used by people with disabilities.

Aspects

Concerns: Ease of use, Audience reach, Discrimination.

Common Tactics: Classification of Audience, Conformance to Accessibility standards, Close collaboration with Usability.

Common Pitfalls: No clear Accessibility Requirements, Accessibility as an afterthought, Internationalization as an afterthought.

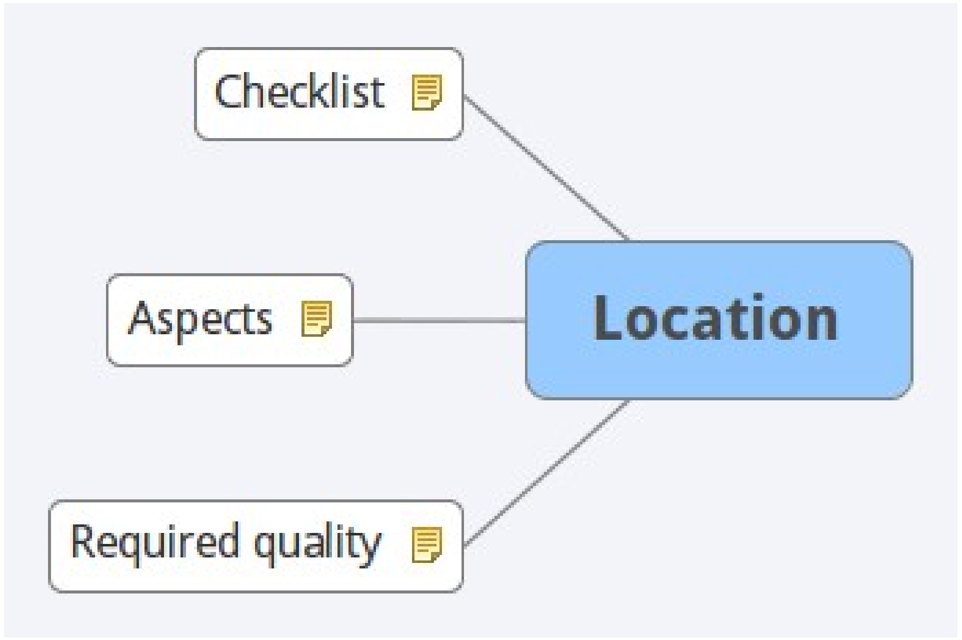
Checklist

Is the system independent of language, country and culture?

Should the architecture rely on a content management system?

What is the user audience for the solution architecture?

Location



Required quality

The ability of the system to overcome problems brought about by the absolute location of its elements and the distances between them.

Aspects

Concerns: Time zones of operation, network link characteristics, intercountry concerns (political, commercial, and legal), network coverage variations between locations.

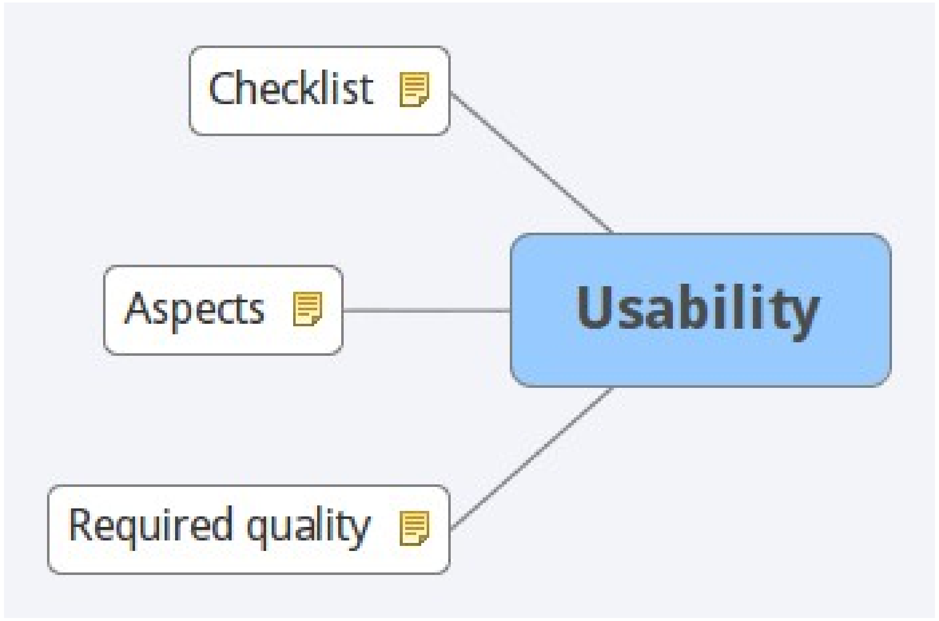
Common Tactics: Geographical mapping, estimation of link quality, estimation of latency, benchmarking, offline operation/features of application.

Common Pitfalls: Assumption of one primary time zone, assumption of uniform network performance, assumption that public networks are high-bandwidth, low-latency, and highly available.

Checklist

Will the system work, given its required geographical constraints?

Usability



Required quality

The ease with which people who interact with the solution can work effectively.

Aspects

Concerns: Ease of use, Audience reach, User Experience, Customer Journey.

Common Tactics: Usability as Requirement, Panel testing with frequent feedback loops, short implementation cycles.

Common Pitfalls: Usability as common-sense, Usability as a Technical responsibility, Usability as Graphical Design only.

Checklist

Can people use the system effectively?

About "Trade-offs"

Architecture trade-offs are typically described as the conscious design choice to:

- prefer a quality property of the architecture over another

- leave a functionality, non-functional gap present in the end solution design

Trade-offs are architectural decisions.

Architects are forced to make choices:

- constrained by scope, time, money

- because risk is assessed as low

Trade-offs must be recorded in the architecture description.

Reason for trade-offs (i.e choices) must be described in detail for future reference.