Requirements Engineering

- Elicitation (obtain raw requirements)
- Analysis (formalization, modeling)
- Specification (refine, organize, document)
- Validation (review)
- Management (change control, tracing)

Functional Correctness

- Provide required functional relationship between inputs and outputs
- Abstracts from timing, etc.
- Orthogonal to other quality attributes
- Typically not an architectural concern
- Almost any architecture can be made to function correctly (at a price).

How to Do Design?

- Top-down approach (not the only way)
- Need: User Requirements and Software Requirements (incl. conceptual models)
- Requirements are (*partly*) given in advance
- Architectural Design also validates, refines, and elicits requirements

Requirements and Architecture

- Architecture and Implementation (incl. deployment) together determine qualities of final product
- How much arch. and impl. contribute varies per quality
- Architecturally Significant Requirement (sometimes abbreviated as ASR)
- Functional vs. non-functional requirements

ISO 9126-1 Quality Model

Attribute	Sub-characteristic
Functionality	Accuracy, suitability, interoperability, compliance and security
Reliability	Maturity, fault tolerance and recoverability
Usability	Understandability, learnability and operability
Efficiency	Time behaviour, resource and utilization
Maintainability	Analysability, changeability, stability and testability
Portability	Adaptability, installability, conformance and replaceability

Business Qualities

- Time to market
- Cost and benefit
- Projected lifetime
- Roll-out schedule (of multiple features)
- Integration with legacy systems

Key Quality Attributes

- Performance (timely response vs task size)
- Availability (deliver service when expected)
- Usability (can user accomplish tasks easily)
- Scalability (accommodate more “usage”, while maintaining quality)
- Security (prevent unauthorized use)
- Modifiability (allow for reasonable changes)
- Verifiability (can conformance be checked)

ISO 9126 Quality Metrics

- *Internal* quality metrics measure the *sytem-design+code*
- *External* quality metrics measure the *system-in-operation*
- The standards define the metrics, their purpose, measurement formulae, interpretation, etc.

Performance depends on

- Distribution of functionality, nature of interfaces and protocols (architectural)
- Amount of communication (architectural)
- Allocation of shared resources (arch.)
- Choice of algorithms (non-architectural)
- Coding (non-architectural)

Usability depends on

- Choice and layout of UI widgets (non-arch.)
- Consistent style (could be architectural, if various components have their own UI)
- Responsiveness, cancel, undo, help, error handling, internationalization facilities (most likely architectural)

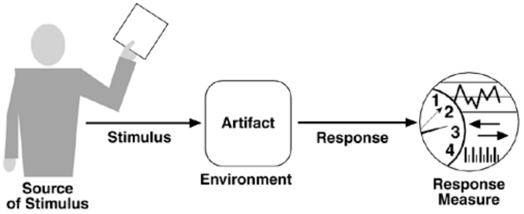
Modifiability depends on

- Distribution of functionality (coherence, coupling: architectural)
- Coding techniques (non-architectural)

Specify Requirements

- Requirements must be verifiable, the earlier the better
- Quality attributes are notoriously hard to specify and verify (compared to functional requirements)
- Quality attribute communities use their own terminology; there is overlap
- Quality attributes are hard to determine before design, so do it during design
- Quality attribute scenarios

Quality Attribute Scenario

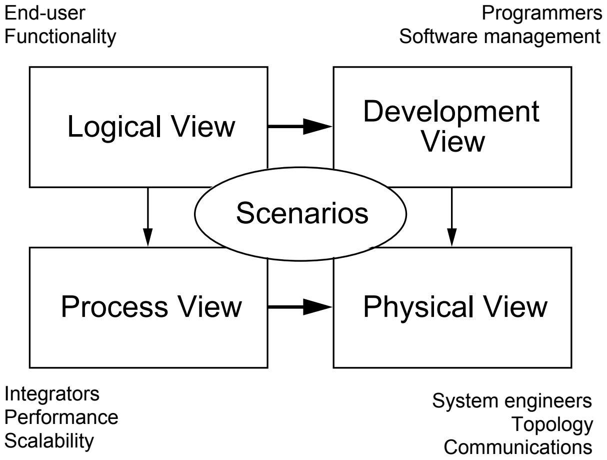


- Source of stimulus: generator of stimulus
- Stimulus: action to consider
- Environment: state/condition of context
- Artifact: thing being stimulated
- Response (by artifact on stimulus)
- Response measure (quantitative judgment)

General vs Concrete

- General scenarios: system independent (can be formulated in advance)
- Concrete scenarios: specific to a particular system (can often be obtained by specialization of general scenarios)
- Typically use collections of scenarios

Kruchten’s 4+1 Views



End-user
Functionality

Programmers
Software management

Logical View

Development View

Scenarios

Process View

Physical View

Integrators
Performance
Scalability

System engineers
Topology
Communications

Example: Performance of web order system

- Source: the user
- Stimulus: web form submission
- Environment: normal working conditions
- Artifact: the system
- Response: load & display confirmation page
- Response measure: page is displayed in less than 5 seconds 99% of the time

Performance Notes

- Throughput: transactions/messages/events/requests processed per second; average vs peak; input characteristics/mix
- Response Time, Latency: distribution constraints if not a fixed amount
- Real-Time Deadlines: hard, soft, time scale
- Capacity: number of records; temporary, persistent; access characteristics
- Accuracy: numerical
- Overhead: error protection, crypto, logging

Availability in General

- Source: internal/external to system
- Stimulus: fault (no response, crash, early/late response, wrong format/value)
- Environment: normal/degraded operation
- Artifact: processors, communication channels, persistent storage, processes
- Response: log, notify, corrective action, degrade
- Response measure: time interval/percentage (must-be available, allowed degraded), mean-time between failure, mean-time to repair

Performance in General

- Source: one or more, possibly internal
- Stimulus: individual/periodic/sporadic/ stochastic events
- Artifact: (sub)system
- Environment: normal/overload mode
- Response: handle stimulus, change service level
- Response measure: latency, deadline, throughput, jitter, miss rate, data loss

Availability Concerns

- How system failure is detected.
- How frequently system failure may occur.
- What happens when a failure occurs.
- How long a system is allowed to be out of operation.
- When failures may occur safely.
- How failures can be prevented.
- What kinds of notifications are required when a failure occurs.

Modifiability in General

- Source: end user, developer, administrator
- Stimulus: change request to add/delete/…
- Artifact: component, platform
- Environment: at run/build/design time
- Response: Localize entities to be modified, realize/verify/deploy modifications
- Response measure: number of elements changed, cost, effort, side-effects

Usability in General

- Source: end user
- Stimulus: minimize impact of errors
- Artifact: the system
- Environment: at runtime
- Response: provide undo/cancel operation
- Response measure: user satisfaction

Security in General

- Engineering discipline in itself
- Doing this well requires a major effort
  - Confidentiality (protected against unauthorized access)
  - Integrity (protected against unauthorized change)
  - Nonrepudiation (transaction cannot be denied)
  - Assurance (signature)
  - Availability (no denial of service)
  - Auditing (preserve historic trail of activities)

Usability Concerns

- Learning system features
- Using a system effectively
- Minimizing the impact of user errors
- Adapting the system to user needs
- Increasing confidence and satisfaction