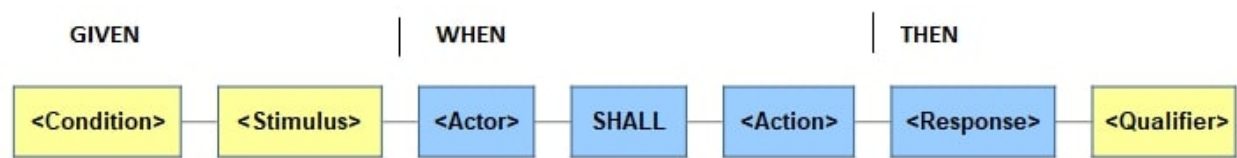


REQUIREMENTS ENGINEERING

Sentence Template for System Requirements

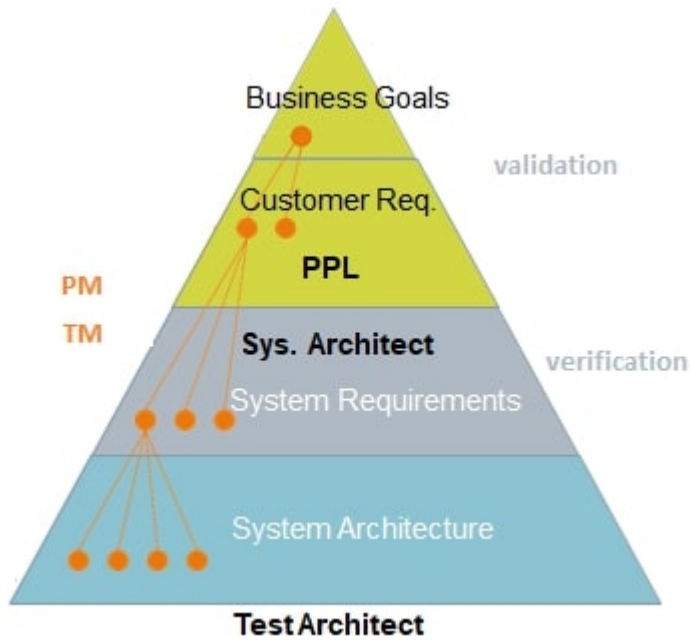
Given When Then



Characteristics of Good Requirements

| Quality Criteria | Business Goals | Customer Requirements | System Requirements |
|-------------------------|----------------|-----------------------|---------------------|
| Individual Requirements | | | |
| Necessary | Mandatory | Mandatory | Mandatory |
| Implementation free | Mandatory | Mandatory | Mandatory |
| Unambiguous | Optional | Mandatory | Mandatory |
| Consistent | Mandatory | Mandatory | Mandatory |
| Complete | Optional | Optional | Mandatory |
| Singular | Mandatory | Optional | Mandatory |
| Feasible | Optional | Optional | Mandatory |
| Verifiable | Optional | Mandatory | Mandatory |
| Traceable | Mandatory | Mandatory | Mandatory |
| Set of Requirements | | | |
| Complete | Optional | Optional | Mandatory |
| Consistent | Mandatory | Mandatory | Mandatory |
| Affordable | Optional | Optional | Mandatory |
| Bounded | Optional | Optional | Mandatory |

Requirements Traceability



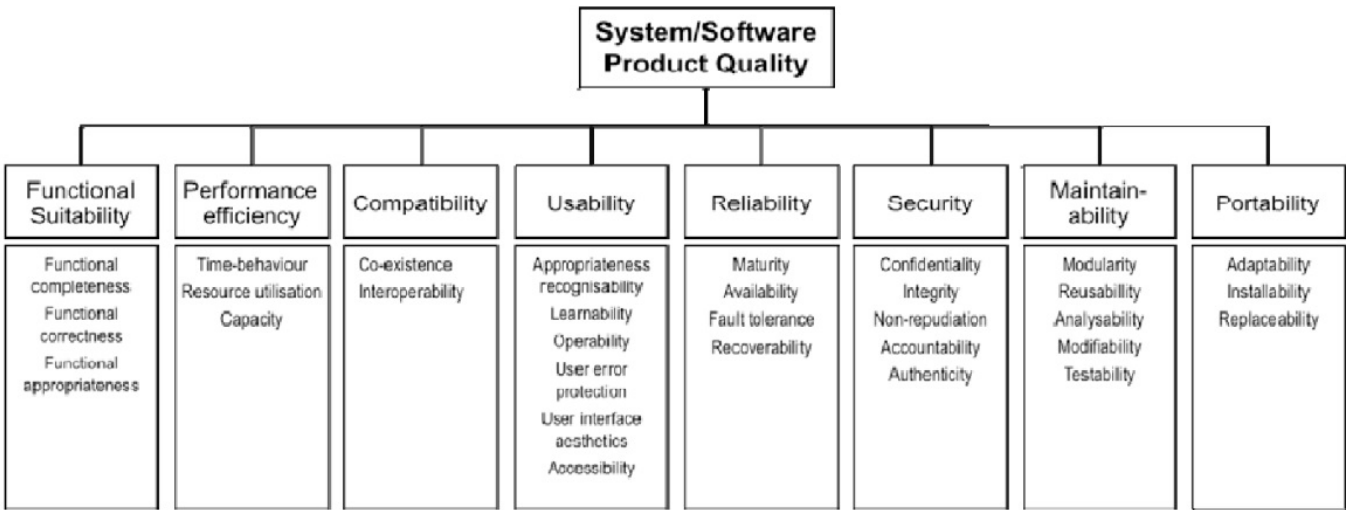
Tracing Usage:

| Analysis | Description | Process |
|------------|--------------------------|-----------------------|
| Derivation | Why is this here? | cost-benefit analysis |
| Impact | What if this changed? | change management |
| Coverage | Have I covered all reqs? | management report |

NFRs / Quality Attributes

[Software Quality Characteristics pdf](#)

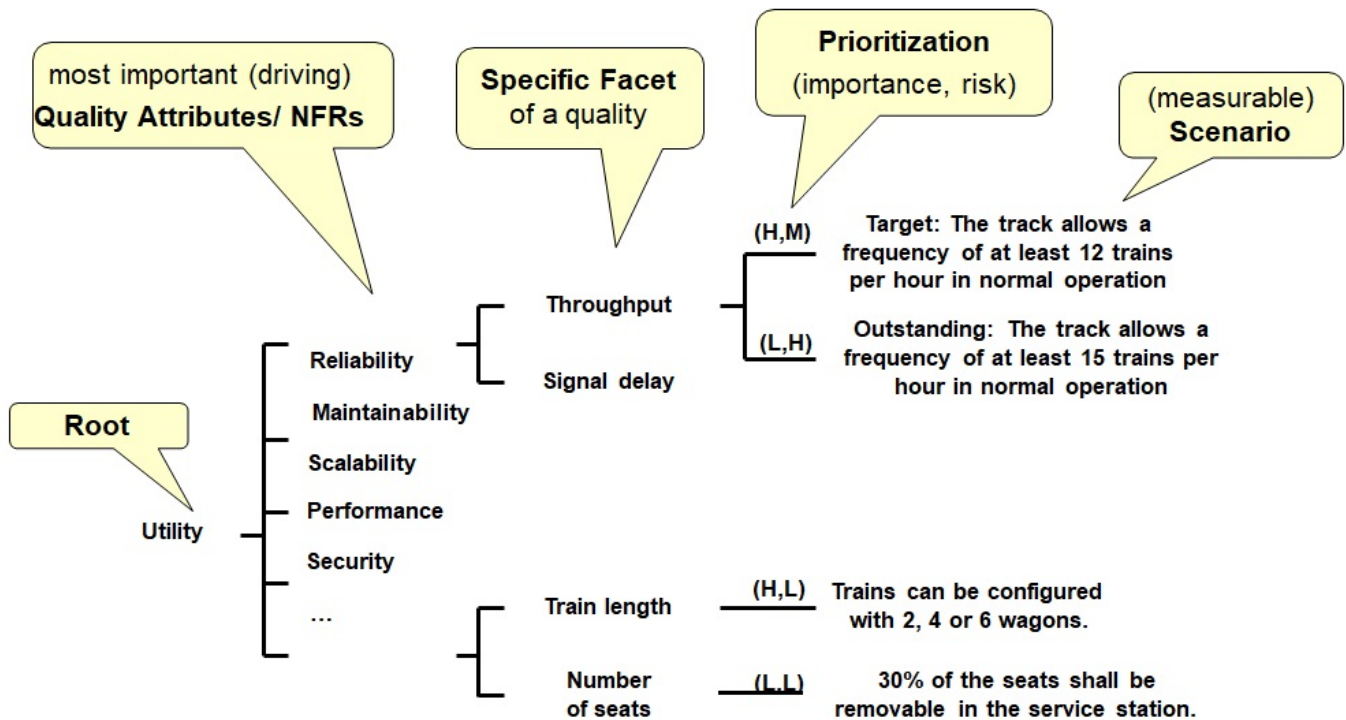
New ISO/IEC 25010 Product quality model



Step 1: Utility Tree

Utility Tree

Ingenuity for life



Step 2: Scenario Description Template / Workflows

To reduce ambiguity and increase testability, each measurable scenario in the Utility Tree gets described with this template.

Scenario description Template

SIEMENS
Ingenuity for life



- Source of stimulus** Who/what initiates the scenario.
- Stimulus** Which periodic, stochastic or sporadic event initiates the scenario.
- Artifact** What is the relevant unit; e.g. a (part of a) system or a feature.
- Environment** What is the environmental condition for this scenario; e.g. normal, startup / shut down, maintenance, emergency, overload, etc.
- Response** How does the artifact react to the event in the given environment. This may cause an environment change (e.g. from normal to shutdown mode).
- Response measure** How can the response be measured, using indicators like:
- the time it takes to process the event (latency or deadline); or the variation in time (jitter)
 - the amount of data, material or energy that can be processed in a particular time interval (throughput)
 - or a characterization of the events that cannot be processed (e.g. miss rate, data / energy / material loss)

Step 3: Refine Scenarios

Scenario refinement table

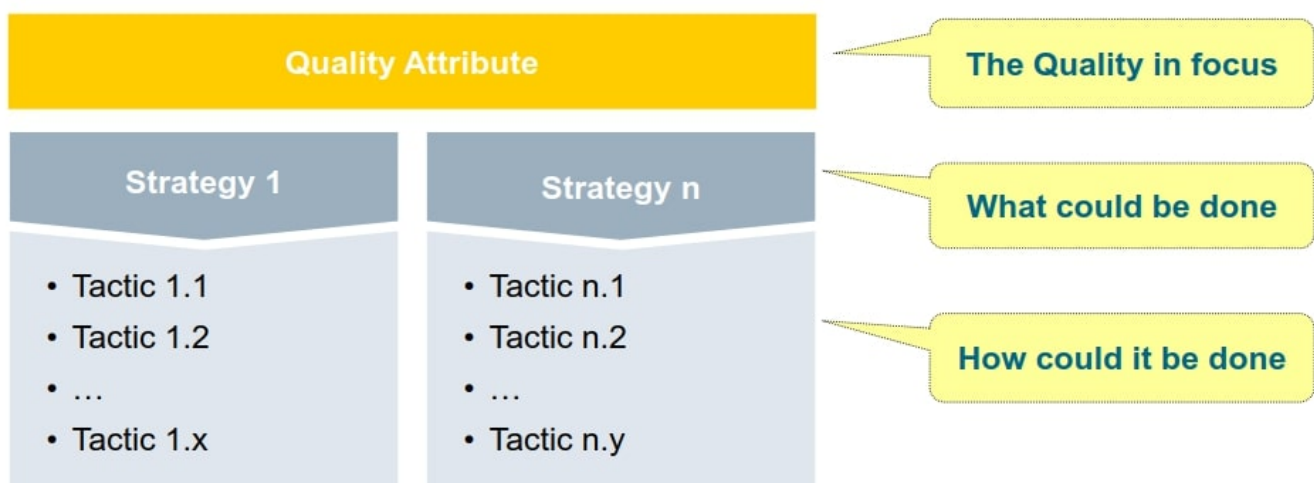
Example from SEI Quality Attribute Workshop (QAW)

| Scenario Refinement for Scenario N | | |
|-------------------------------------|-----------------------------|--|
| Scenario(s): | | When a garage door opener senses an object in the door's path, it stops the door in less than one millisecond. |
| Business Goals: | | safest system; feature-rich product |
| Relevant Quality Attributes: | | safety, performance |
| Scenario Components | Stimulus: | An object is in the path of a garage door. |
| | Stimulus Source: | object external to system, such as a bicycle |
| | Environment: | The garage door is in the process of closing. |
| | Artifact (If Known): | system's motion sensor, motion-control software component |
| | Response: | The garage door stops moving. |
| | Response Measure: | one millisecond |
| Questions: | | How large must an object be before it is detected by the system's sensor? |
| Issues: | | May need to train installers to prevent malfunctions and avoid potential legal issues. |

Step 4: Implement Design Strategies to support the NFRs

.. and use Design patterns when needed...

Design Strategy Template for Quality Attributes



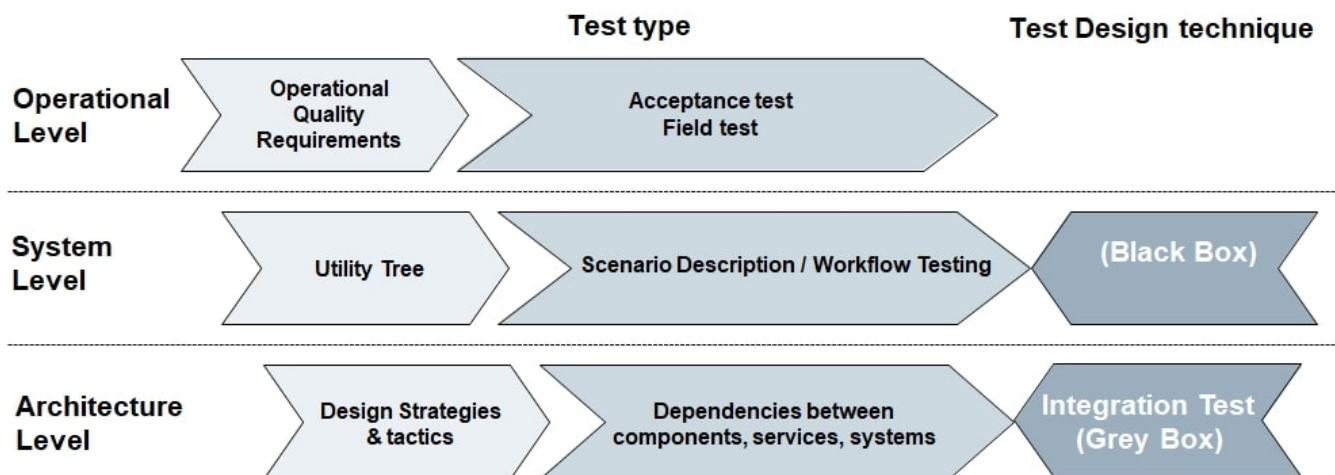
| Performance of IT Systems | | | Increase Testability (NFR/ Quality Attribute) | | |
|--|---|---|--|---|--|
| Resource demand | Resource management | Resource arbitration | make it simple | make it observable | make it controllable |
| <ul style="list-style-type: none">• Increase computation efficiency• Reduce computational overhead• Manage event rate• Control frequency of events / raw data | <ul style="list-style-type: none">• Introduce concurrency• Maintain multiple copies of data/services• Adapt HW resources (processor, memory, network) | <ul style="list-style-type: none">• Scheduling policy• Distribution• Localization | <ul style="list-style-type: none">• Interface driven• No code dupe• Low <u>cyelo.complex</u>.• Low coupling | <ul style="list-style-type: none">• Log• Trace• Testable interfaces | <ul style="list-style-type: none">• Dependency inj.• Law of Demeter• <u>Config</u> & data files• Object IDs (auto.) |

Design Pattern example
"Performance of Image Processing"

| Performance of Image Processing | | | Caching | Eager Acquisition |
|--|--|---|---|--|
| Resource demand | Resource management | Resource arbitration | <ul style="list-style-type: none">• <u>Context</u>: Systems that repeatedly access the same set of resources and need to optimize performance.• <u>Solution</u>: Resources stored temporarily in fast-access buffer (cache); subsequent access through cache instead of resource provider. | <ul style="list-style-type: none">• <u>Context</u>: Systems that must satisfy high predictability and performance in resource acquisition time.• <u>Solution</u>: Resources eagerly acquired before their actual use by a provider proxy; resources then kept in an efficient container; requests intercepted by the proxy. |
| <ul style="list-style-type: none">• Use advanced image processing algorithms (if possible GPU based)• Introduce caching strategies• Either allow upper bound of clients or scale-out mechanisms depending on resources | <ul style="list-style-type: none">• Introduce a pool of worker threads for background loading, storing.• Don't copy mass data but use meta files and refs for image processing/copying• Use thumbnail images for browsing• Only use full resolution when images are• Apply eage. | <ul style="list-style-type: none">• Schedule resources preferably for processing visible images | <p><u>Pooling</u></p> <ul style="list-style-type: none">• <u>Context</u>: Systems that continuously acquire and release resources of same/similar type.• <u>Solution</u>: Multiple instances of one type of resource managed in a pool; released resources are put back into the pool. | <p><u>Resource Lifecycle Manager</u></p> <ul style="list-style-type: none">• <u>Context</u>: Systems that require simplified management of the lifecycle of their resources.• <u>Solution</u>: Resource usage separated from resource management through introduction of a Resource Lifecycle Manager (RLM). |

Testing NFRs

Testing NFRs



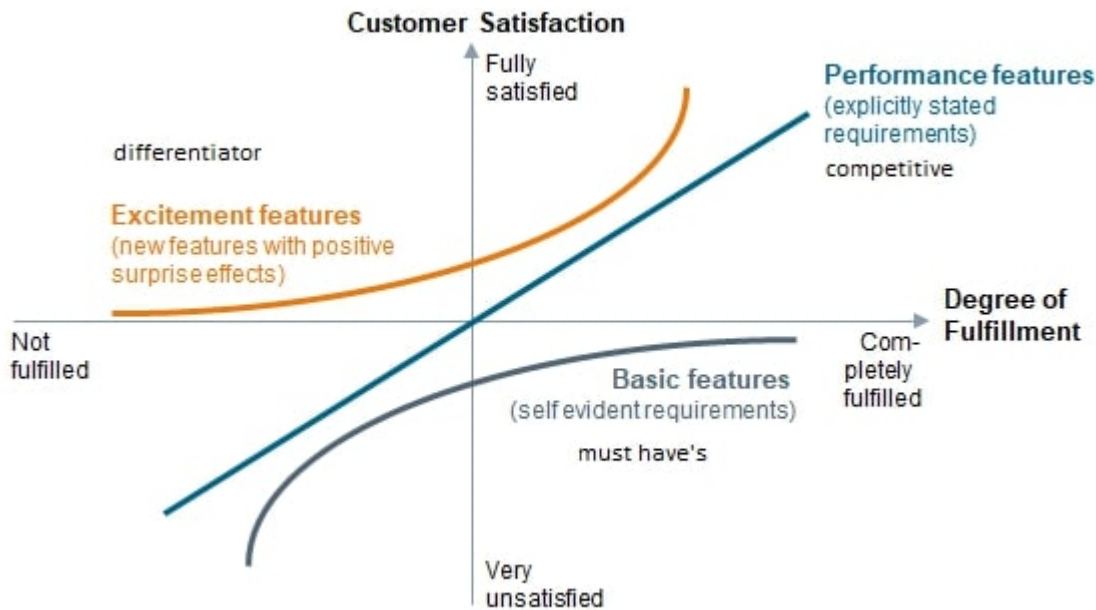
Some NFR testing approaches

| | | |
|---|--|---|
| Security <ul style="list-style-type: none"> Fuzz Testing Vulnerability assessment Vulnerability testing Penetration test Security audit Security review Network scanning | Performance <ul style="list-style-type: none"> Load testing Stress testing Spike testing Endurance/Soak testing Benchmarking | Conformance <ul style="list-style-type: none"> Protocol tests Radiated immunity testing Radiated emissions testing Conformity assessment |
| Safety <ul style="list-style-type: none"> Hazard and operability analysis/study (HAZOP) Defect history tracking Statistical testing Probabilistic risk assessment (PRA) Failure mode and effect analyses (FMEA) Fault tree Analysis (FTA) Reliability Testing | Usability <ul style="list-style-type: none"> Hallway testing Remote usability testing (Automated) Expert review | Others <ul style="list-style-type: none"> Recovery testing Volume testing |

Additionally – on Architecture Level many test design techniques can be used, too

KANO Model

For the Product Manager KANO is a tool to find the right mix between the different feature types. For the Test Architect KANO helps to identify risks.

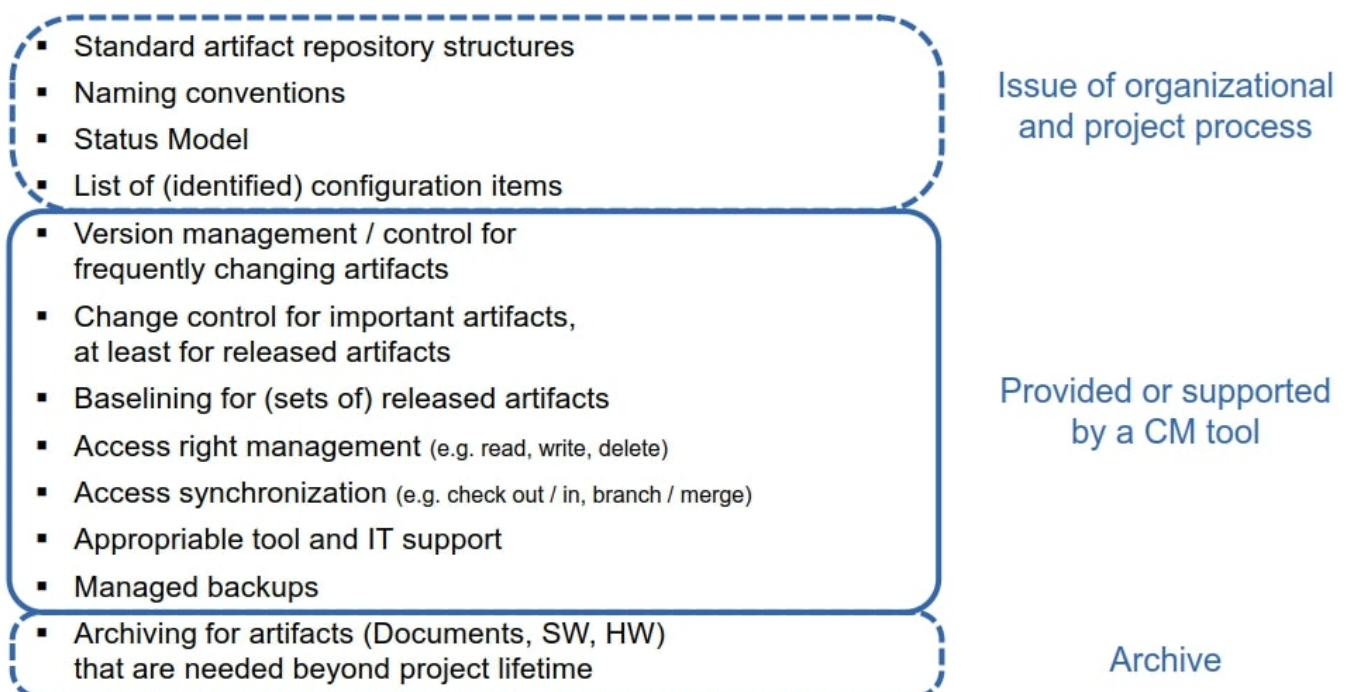


Configuration Management

Source Control, configurations, environment...

Devops: continuous integration, deployment, testing...

Attributes of Professional Configuration Management



Change Request Management

Change Request: any issue that cannot go in a patch or point rev, has to go into the product asap.

Change Management: because requirements are incomplete, erroneous, ambiguous.

Defect Management: because tests reveal defects that need correction.

Tool -> Feature -> trace & Test

CCB:

**Change Request Management:
Evaluate and Decide on Change**



SIEMENS
Ingenuity for life

Understand impact on the system and all disciplines or sub-systems.

Decision by Change Control Board (CCB)

A group of persons with assigned responsibility and authority to make decisions on the change.

Typically includes

- Product Manager
- Project Manager
- (System) Architect

Impact Analysis

- Rough initial evaluation by the CCB to decide on further action
- Deep analysis by experts
 - on architecture, cost, schedule
 - all affected disciplines
 - all relevant sub-systems
- Propose further proceeding (accept / defer / reject)