UNIVERSITÄT DES SAARLANDES

System Quality (Non-Functional Requirements)

Architectural Thinking for Intelligent Systems

Winter 2020/2021

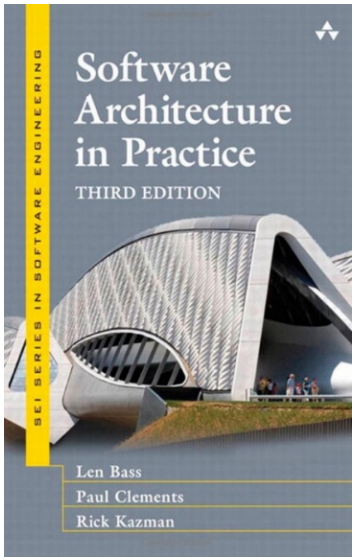
Prof. Dr. habil. Jana Koehler

Agenda

- Importance of non-functional requirements
  - Specification of system quality attributes
- Making qualities measurable with scenarios
- Quality Characteristics in ISO/IEC 25010:2011

Suggested Reading

- Chapter 1.4 (pages 19-21)
- Chapter 2.1 (pages 25-27)
- Chapter 4.1- 4.4 (pages 63-70)
- Chapter 4.6 (pages 72-77)



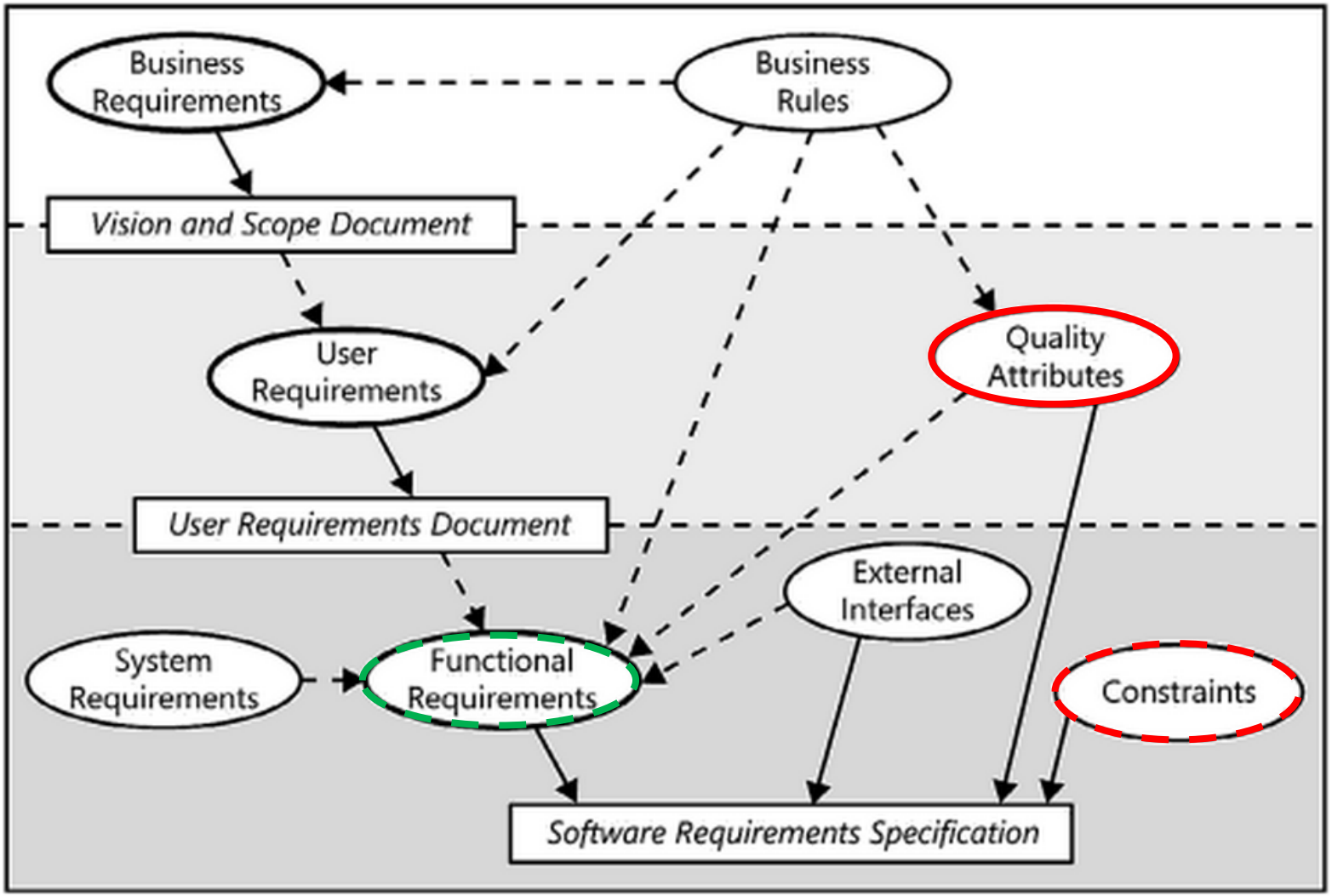
- Optional reading
  - Chapters 5-12 of the book discuss each quality attribute in detail
  - Read the two chapters that cover the quality criteria that you selected for your work on the assignment
  - ISO 25010 (page 10-16) explains the quality criteria used in the standard

Tutorial Assignment 5:

- We review the quality attributes of the ISO 25010 standard and interpret them in the context of the semester project.

- We decide for 2 non-functional requirements that will guide the architecture for our system.

- We write scenarios to make these requirements measurable.



Wiegers/Beatty: Software Requirements, 3rd edition 2013

System Qualities and Non-functional Requirements

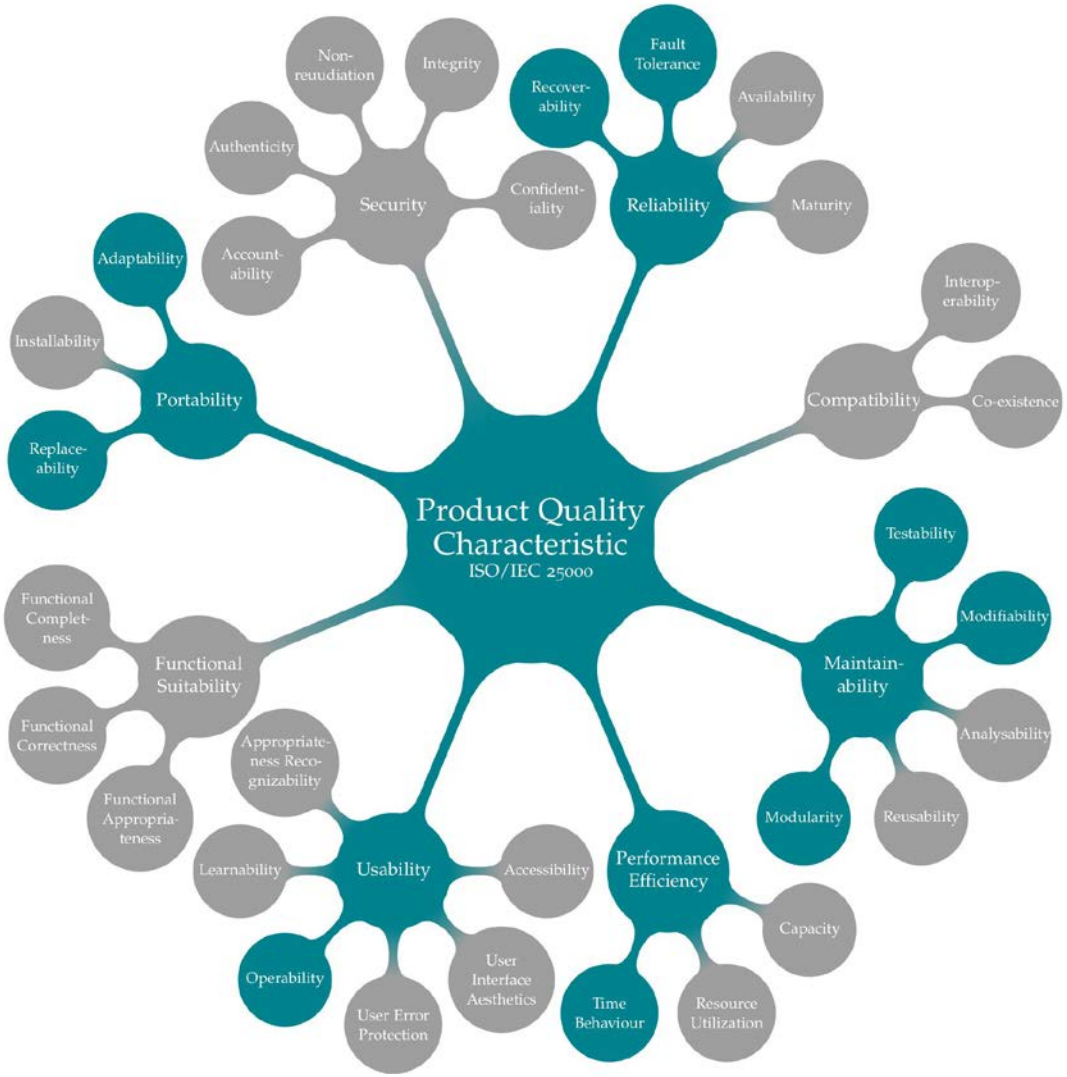
- Functional requirements
  - System functionality = system capabilities, services, responsibilities & behavior
  - *«What does it offer to me, what can I do with it?»*

- Non-functional requirements
  - How system functionalities are fulfilled
  - *«How well does the system meet my needs?»*
  - *«How well does it offer its services to me»*
  - What decisions have already been taken (constraints)

Systems undergo change because maintenance or portability is difficult, or because performance, scalability, security, usability ... are insufficient.

Functional enhancements come in 2nd place.

System Qualities determine System Architecture



Quality Attributes

- Measurable & testable!!!

- Quality attributes are fulfilled by an architecture by anchoring specific structures with specific interactions and a certain behavior in the architecture

- Functional requirements are met by transferring responsibilities to specific elements in the architecture

  ➢ **Architecture is not determined by functionality, but only by system qualities!**

2 Relevant Groups of Quality Attributes

- Qualities of the system at runtime = observable behavior
  - Availability
  - Performance
  - Usability
  - Interoperability
  - Security
- Qualities of the system at development time
  - Modifiability
  - Testability
  - Reusability

Quality Attributes are not Independent of each Other

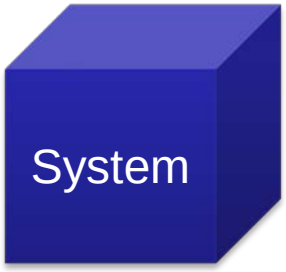
- Quality can only be measured indirectly, not absolutely
- Quality is relative and different for different stakeholders
- Quality of architecture <> Quality of software
- A complete implementation of the functional requirements does not allow statements to be made about the quality achieved

- Quality attributes are at the heart of architectural thinking
  - Need to make compromises in decisions
  - "Trade-offs"

How can we describe Quality Attribute Requirements?

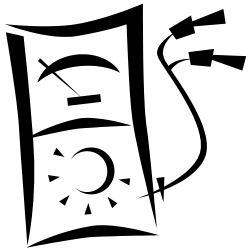
Stimulus Source

Stimulus



Environment Artifact

Response



Response Measure

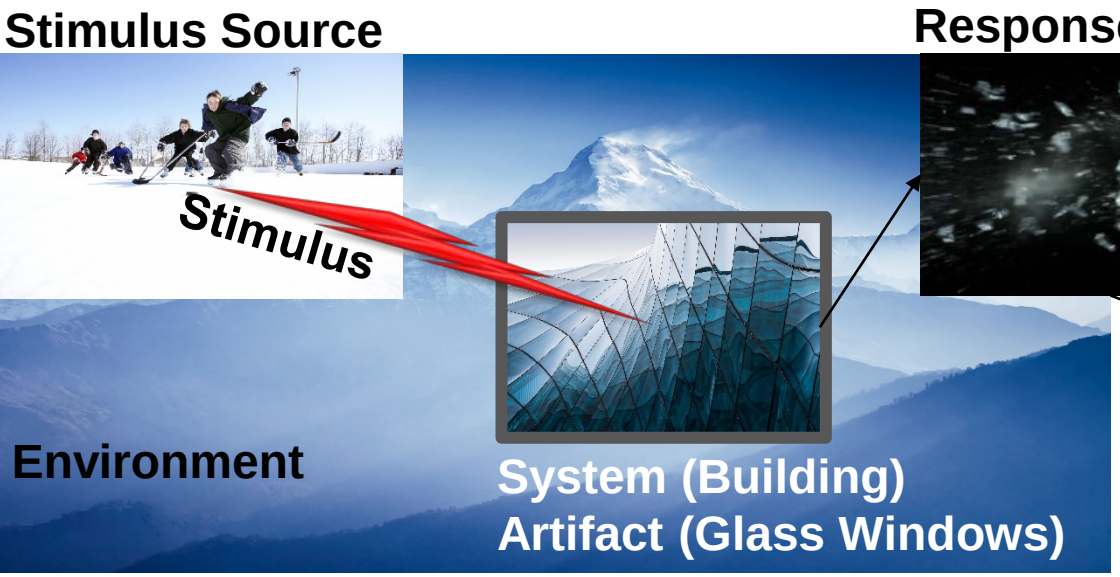
- **"Scenario"**
  - **What happens when a stimulus affects a system in a certain situation?**

Specification of Scenarios

1. Stimulus source
2. Stimulus
3. Environment
4. Artifact
5. Response
6. Response measure

- Scenarios make non-functional requirements (the quality attributes) SMART
- It often requires more than one scenario per non-functional requirement

Intuitive Scenario Example: Breaking Glass



- Environment: in which context, under which circumstances and boundary conditions does a stimulus occur?
- Artifact: part of the system that is affected
- Stimulus: how did the external event affect/influence the system?
- Response: what exactly happens in/with the artifact?
- Response measure: how can an external observer measure and quantify the impact of the stimulus and the amount of response?

Selected Quality Attributes

Usability

- How easy is it for the user to use the system?
- What support does the user receive?

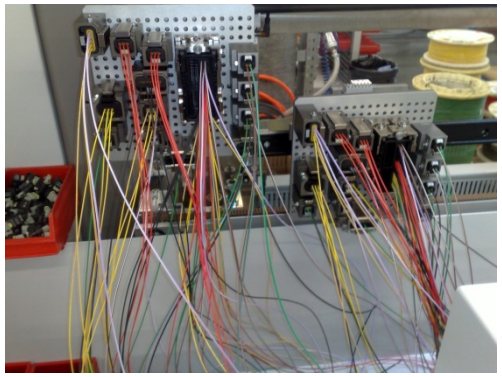
- Learning to use a new system, «self-explaining»
- Adequate information visualization
- Effects of erroneous user behavior
  - Adaptability of the system to the user
  - Confidence and satisfaction of the user

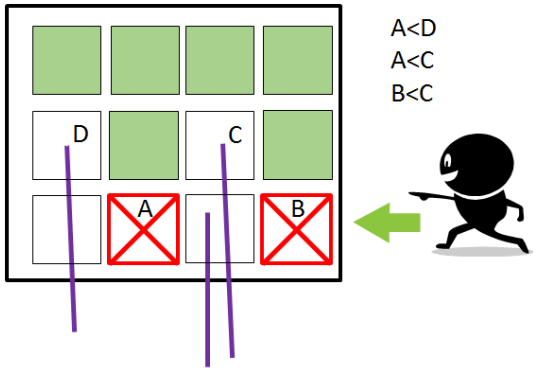
- «App x can be used to <x> within 1 minute after download»

Scenario Pattern for Usability

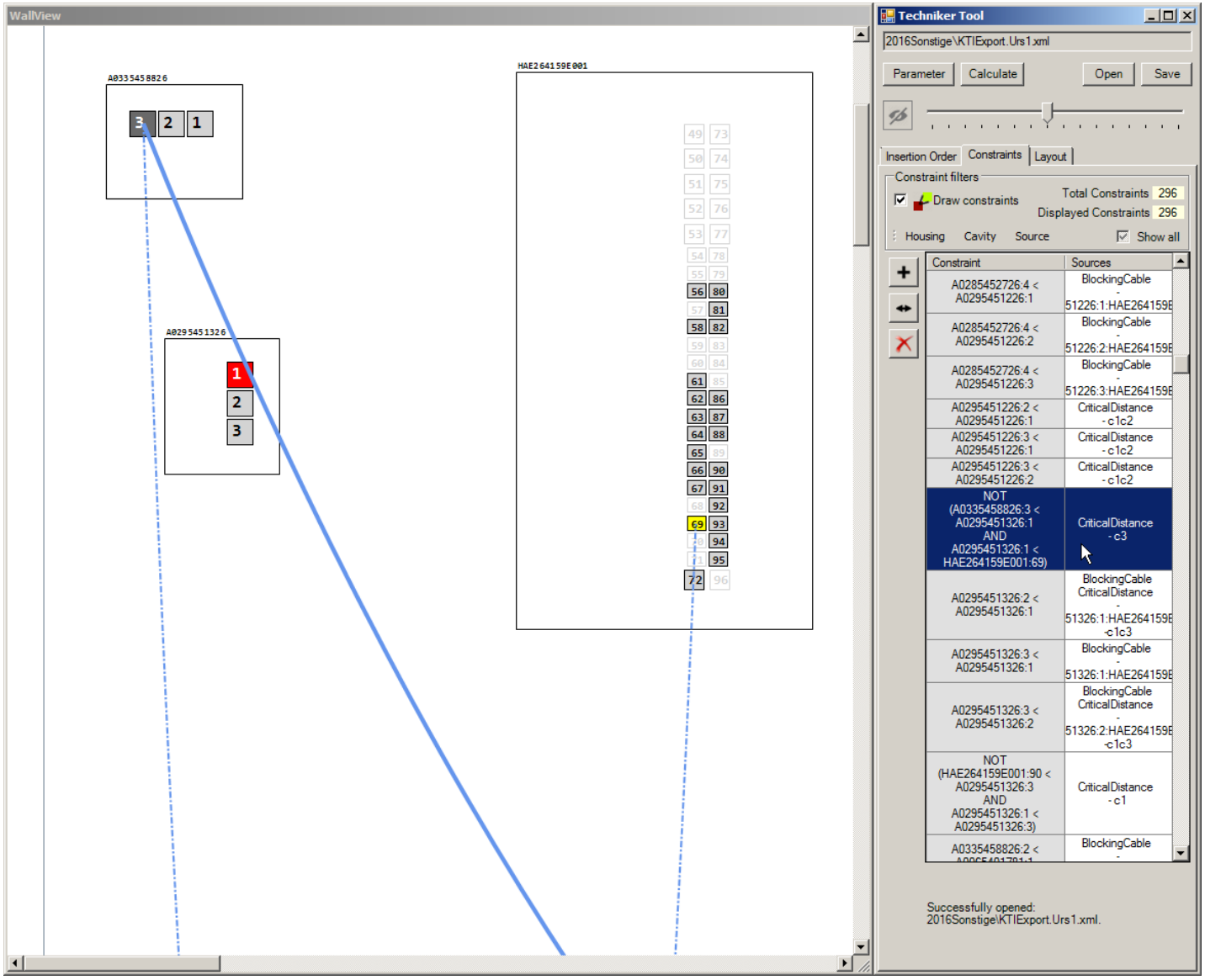
Part of the Scenario	Description
Stimulus source	Stakeholder (user, architect, developer,...)
Stimulus	Stimulus source tries to: - learn how to use the system efficiently and correctly - understand the operating or usage concepts of the system - minimize the effects of operating errors - adapt system to own needs (configure) - understand system and software architecture
Environment	Normal operation, runtime, installation or configuration time
Artifact	Complete system (including user interface, control flow, architecture documentation)
Response	- System supports by examples or explanations - Incorrect operation has local/overlapping effects - System (part) is not configurable - System and software architecture with underlying concepts is not documented
Response measure	- Time spent, number of errors, number of user goals/tasks achieved, user satisfaction, increase in knowledge, ratio of successful uses to total uses, proportion of errors occurring - Extent of damaged data, loss of time, abortion of interactions

Example

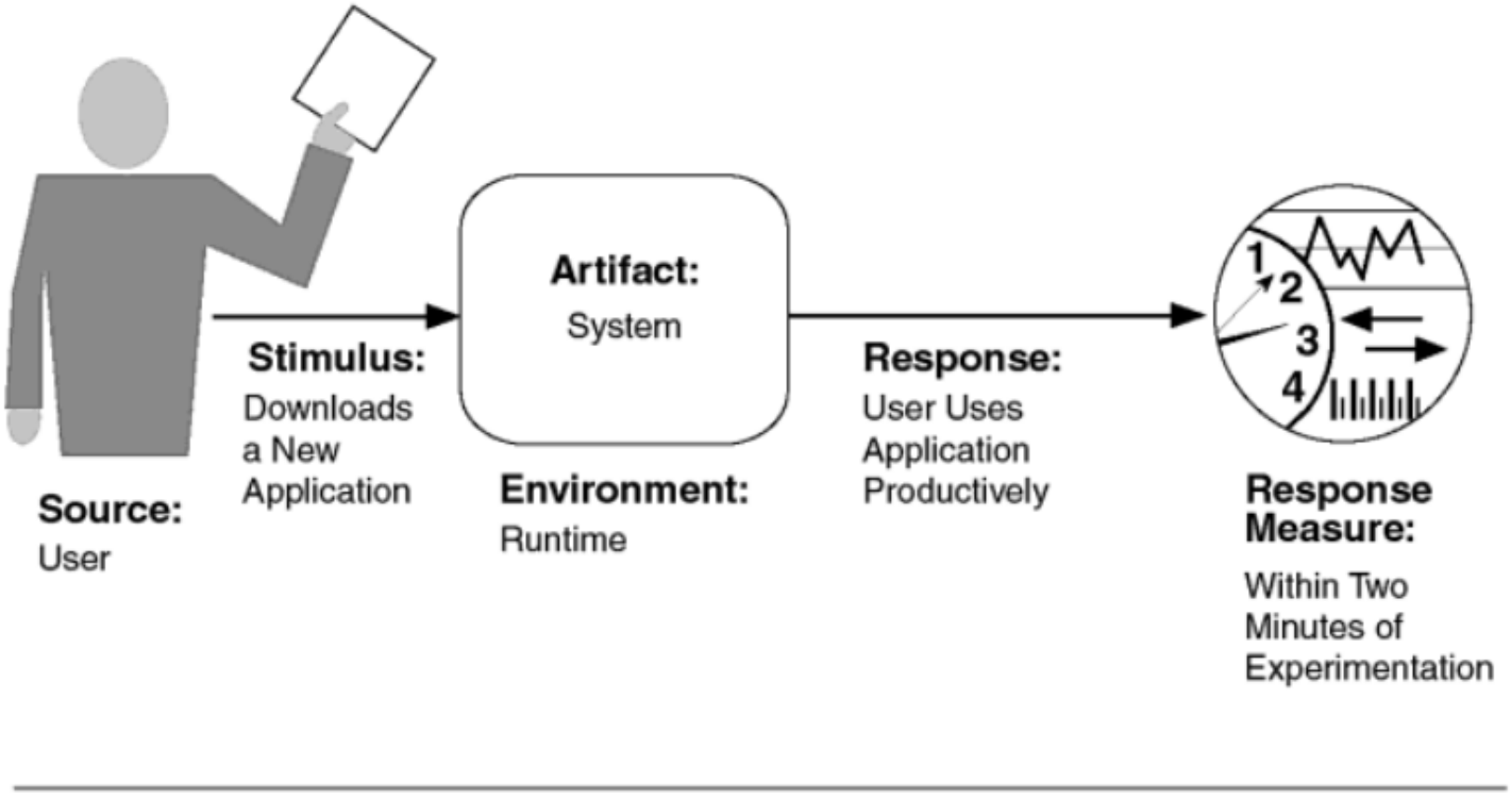




Part of the scenario	Description
Stimulus source	Komax technician
Stimulus	Desire for visualization of constraints between 2 cavities
Environment	Topwin Software
Artifact	AI System GUI
Response	Display of all constraints
Response metrics	With only 2 clicks in 2 scrollable windows


WallView
A0335458826
3
2
1
HAE264159E001
49 73
50 74
51 75
52 76
53 77
54 78
55 79
56 80
57 81
58 82
59 83
60 84
61 85
62 86
63 87
64 88
65 89
66 90
67 91
68 92
69 93
70 94
71 95
72 96
A0295451326
1
2
3
Techniker Tool
2016Sonstige\KTIExport.Urs1.xml
Parameter
Calculate
Open
Save
Insertion Order
Constraints
Layout
Constraint filters
Draw constraints
Total Constraints 296
Displayed Constraints 296
Housing
Cavity
Source
Show all
Constraint
Sources
A0285452726:4 < A0295451226:1
BlockingCable - 51226:1:HAE264159E
A0285452726:4 < A0295451226:2
BlockingCable - 51226:2:HAE264159E
A0285452726:4 < A0295451226:3
BlockingCable - 51226:3:HAE264159E
A0295451226:2 < A0295451226:1
CriticalDistance - c1c2
A0295451226:3 < A0295451226:1
CriticalDistance - c1c2
A0295451226:3 < A0295451226:2
CriticalDistance - c1c2
NOT (A0335458826:3 < A0295451326:1 AND A0295451326:1 < HAE264159E001:69)
CriticalDistance - c3
A0295451326:2 < A0295451326:1
BlockingCable CriticalDistance - 51326:1:HAE264159E -c1c3
A0295451326:3 < A0295451326:1
BlockingCable - 51326:1:HAE264159E
A0295451326:3 < A0295451326:2
BlockingCable CriticalDistance - 51326:2:HAE264159E -c1c3
NOT (HAE264159E001:90 < A0295451326:3 AND A0295451326:1 < A0295451326:3)
CriticalDistance - c1
A0335458826:2 <
BlockingCable -
Successfully opened: 2016Sonstige\KTIExport.Urs1.xml.

Example



Sample concrete usability scenario

Availability (availability/reliability)

"Ninety percent of life is just showing up." Woody Allen

- Functionalities are available and usable when needed

Stability
Reliability

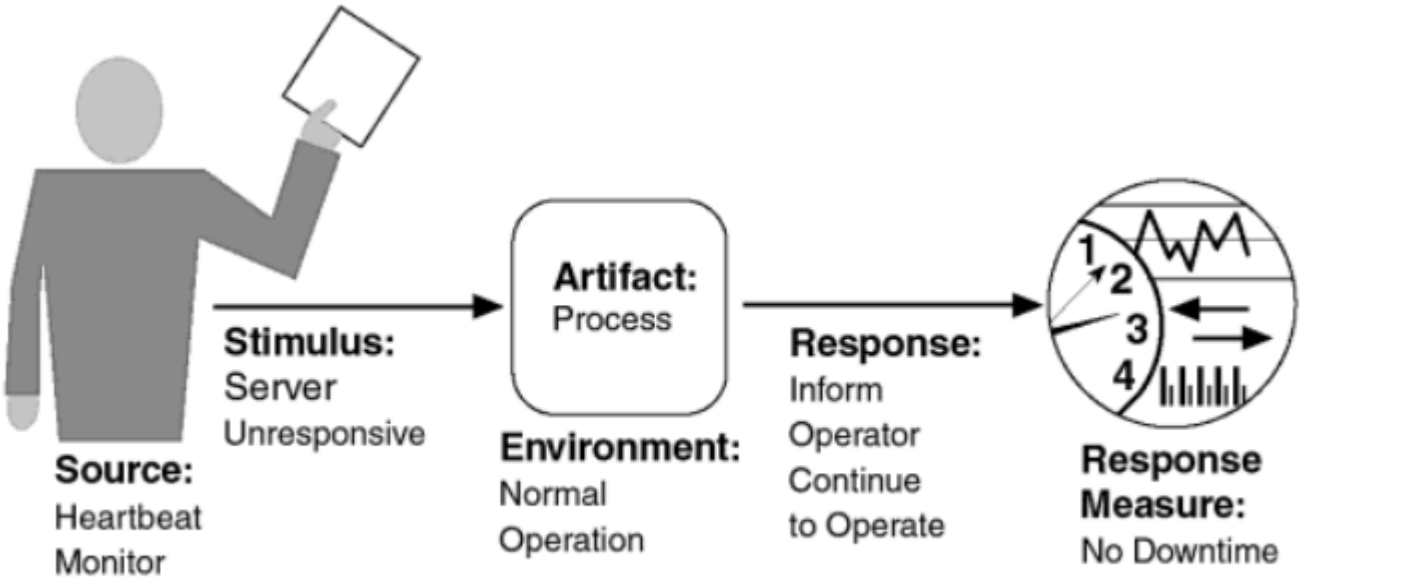
+ Ability to avoid failure or to restore functionality after failure within a specified period of time

- Which errors occur (internal, external)? How often? What response do they trigger? How are they prevented? How are they noticed, communicated, and corrected? How long is the system or artifact "down"?

Scenario Pattern for Availability

Part of the scenario	Value
Source	Internal/external: people, hardware, software, physical infrastructure, physical environment
Stimulus	Fault: omission, crash, incorrect timing, incorrect response
Environment	Normal operation, startup, shutdown, repair mode, degraded operation, overloaded operation
Artifact	Processors, communication channels, persistent storage, processes
Response	- Prevent the fault from becoming a failure (breakdown of system parts) - Detect the fault: logging, notifications (people, system) - Recover from the fault: disable source of events causing the fault, be temporarily unavailable while repair is being effected, fix or mask the fault/failure or contain the damage it causes, operate in a degraded mode while repair is being effected
Response measure	- Time or time interval when system must be available, availability percentage - Time to detect/repair the fault, time in degraded mode … - Number or percentage of faults the system can detect, prevent, handle without failing

Example



Sample concrete availability scenario

Types of Errors Preventing Availability (possible Stimuli)

- Omission
  - no response to a stimulus
- Crash
  - Omissions that occur repeatedly
- Timing
  - Response occurs at the wrong time (too early, too late)
- Response
  - Incorrect response of the system

Possible Responses to Keep Availability Up

- Prevention
  - Add redundancy, safety functions, load balancing to system architecture
- Detection & Isolation
  - Logging of the error
- Recovery
  - Notify users and other systems
  - Start actions to limit potential damage
  - Limit availability or functionality of the affected system (parts)

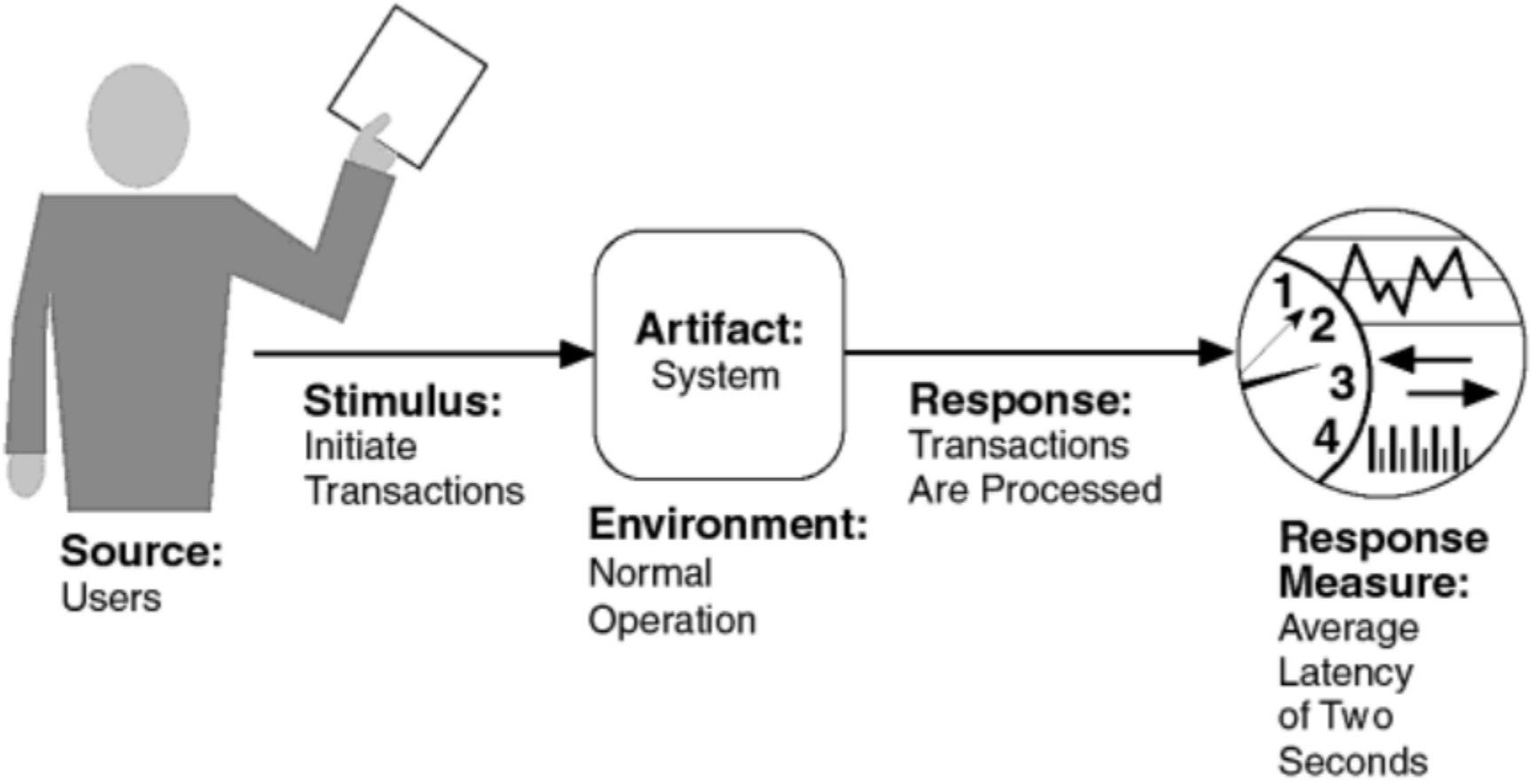
Performance

- System reaction in a certain time to a certain event
  - often associated with scalability requirements
- Events occur periodically, stochastically, sporadically
- Measurement of Response
  - latency: time between event occurrence and system reaction
  - deadlines, throughput (transactions/s), jitter (latency variance, number of unprocessed events)

Scenario Pattern for Performance

Part of the scenario	Description
Stimulus source	internal or external
Stimulus	Occurrence of the stimulus (periodic, stochastic, random)
Environment	Normal operation, high load, overload, emergency operation
Artifact	Complete system (or certain parts: servers, databases)
Response	- Processing the event (execution behavior) - Change in execution behavior (complete vs. partial processing) - Restricted use of functions, data, etc. - Change in runtime behavior/resource usage by the system
Response measure	- Latency, response time, throughput - Error rates, amount of lost data or no longer available functions - Fluctuations in loads and response times - Use specific quantities and durations/time points!

Exampleof a Performance Scenario



Sample performance scenario

Modifiability / Extensibility

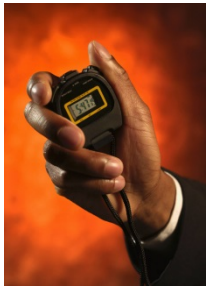
- Degree to which a stakeholder is able to make changes in the system
  - What can change?
  - How likely is a change?
  - When does a change happen and by whom?
  - What will it cost? What is the risk?

User

GUI change

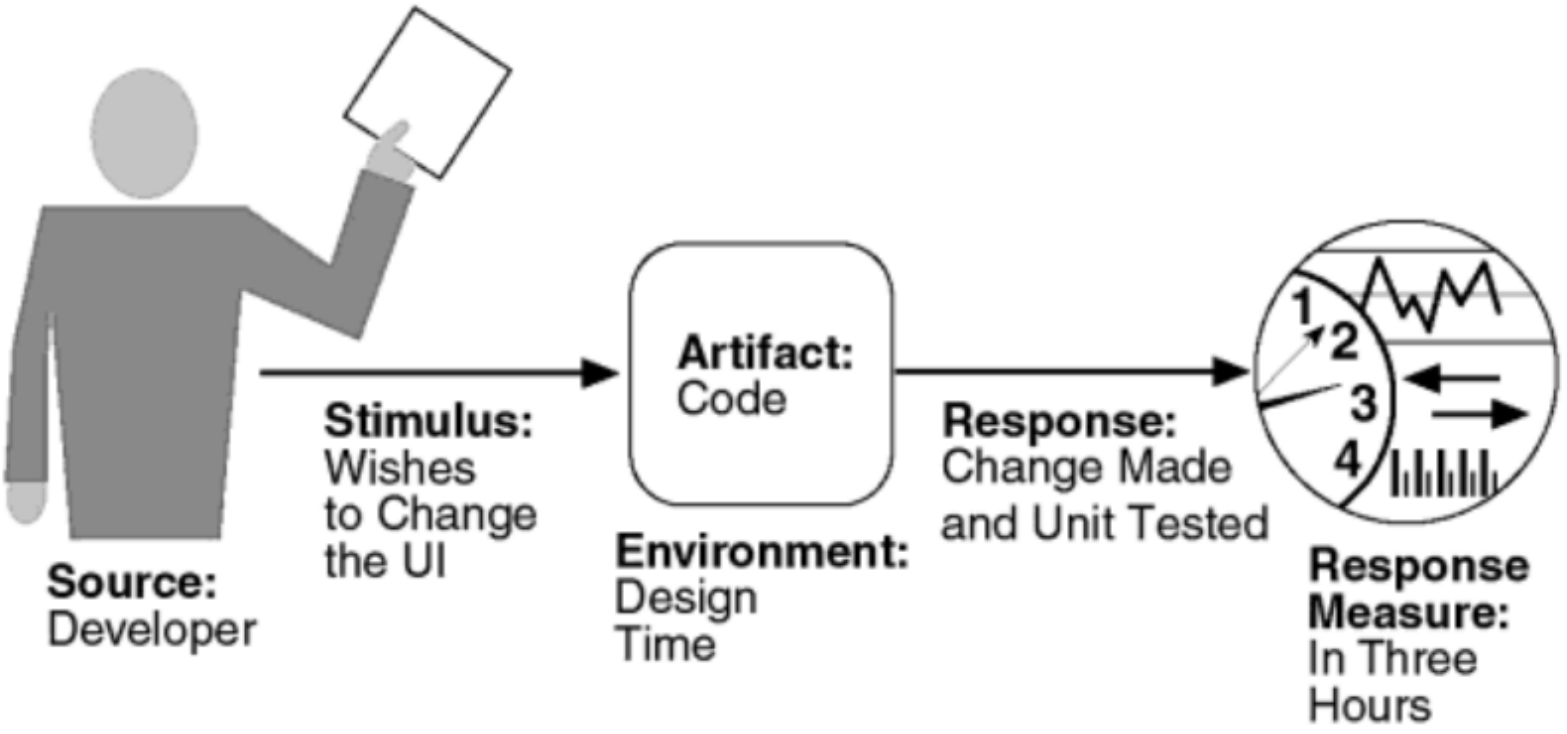
" tile surface"

Delete tile or add new one



< 1 second

Example of a Modifiability Scenario



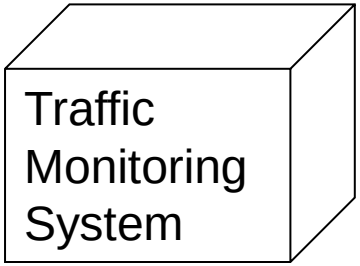
Sample concrete modifiability scenario

Interoperability

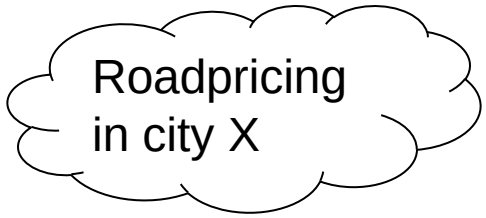
- Degree to which two or more systems can exchange information in a meaningful way
  - syntactic data exchange (interfaces)
  - semantic data interpretation
- With whom? With what? Under what conditions?

My car

Entrance to city & time

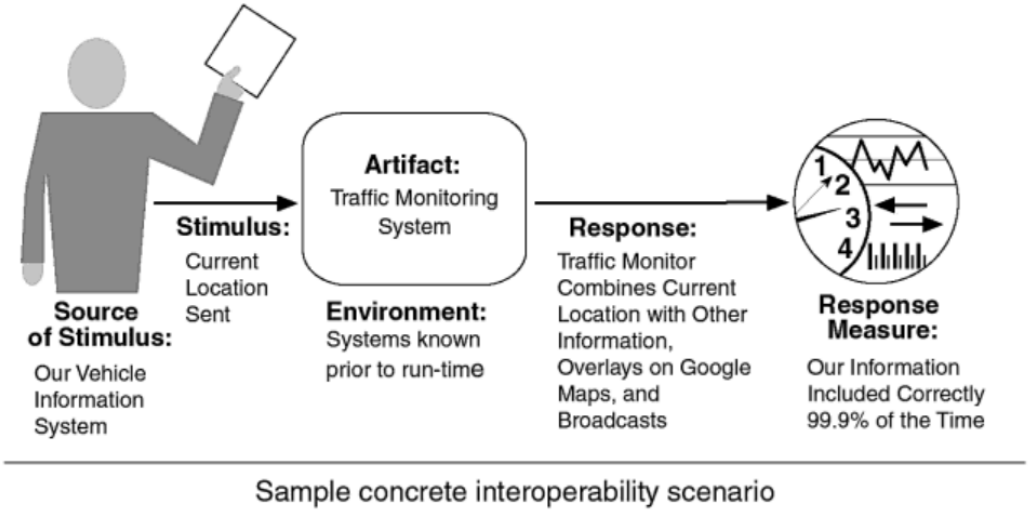


Env: system known



99.99% of all invoices are correct

Example



Sample concrete interoperability scenario

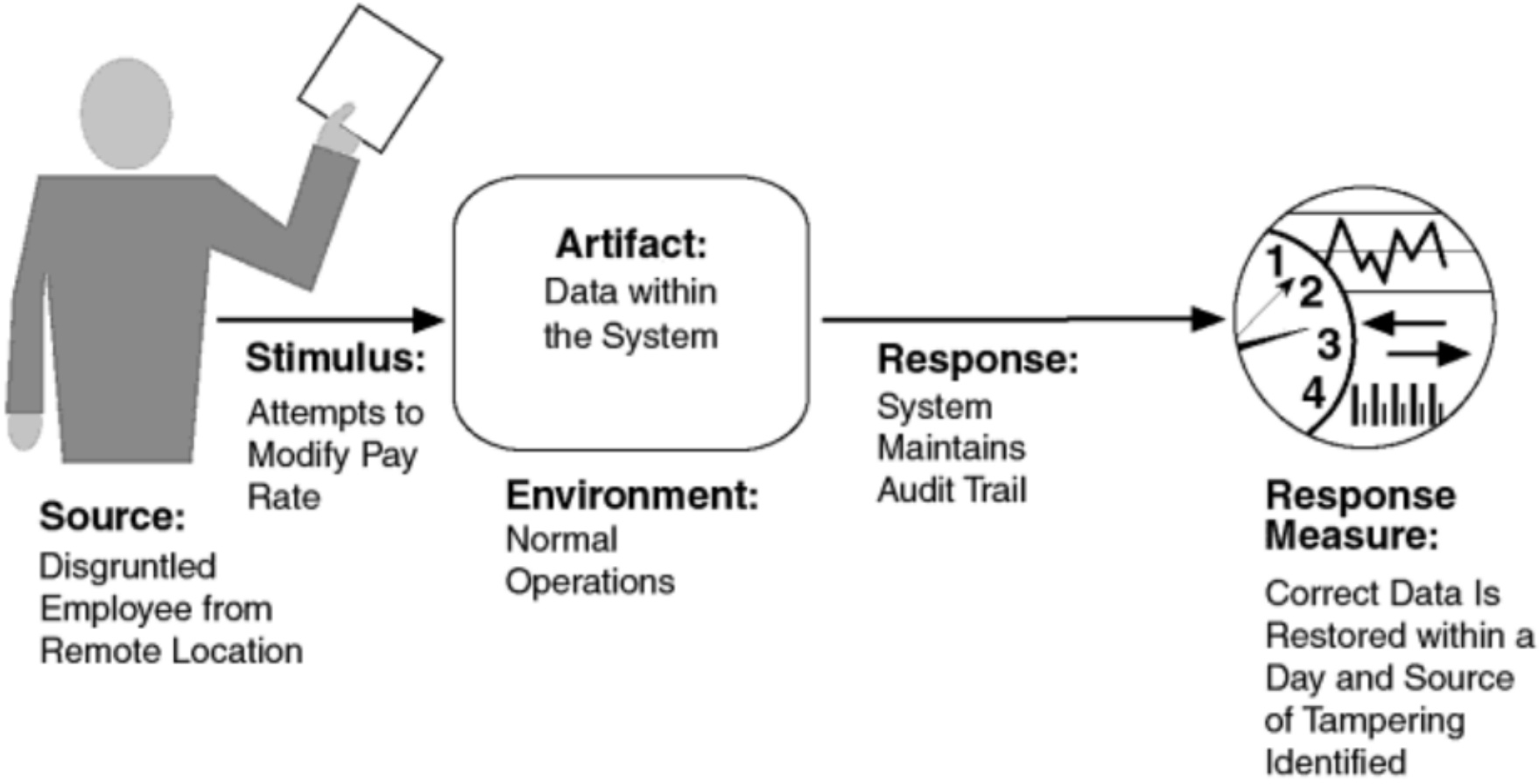
Part of the Scenario	Description
Stimulus source	Car information system
Stimulus	Message regarding the current position
Environment	Traffic Monitoring system (+ located vehicles)
Artifact	Message receiver of the traffic monitoring system
Response	Current position transmitted from receiver to traffic monitoring server
Response measure	99.99% of all captured positions reach the server

Security

- Ability of a system
  - to protect data and information from unauthorized access, but allow authorized access by users (and other systems)
  - to ensure information integrity (protect against unauthorized manipulation)
  - to be available for legitimate use
  - to recognize, resist, respond to and recover from attacks

- Example: "Denial of service" attack does not prevent that books can be ordered in an online shop

Example of a Security Scenario

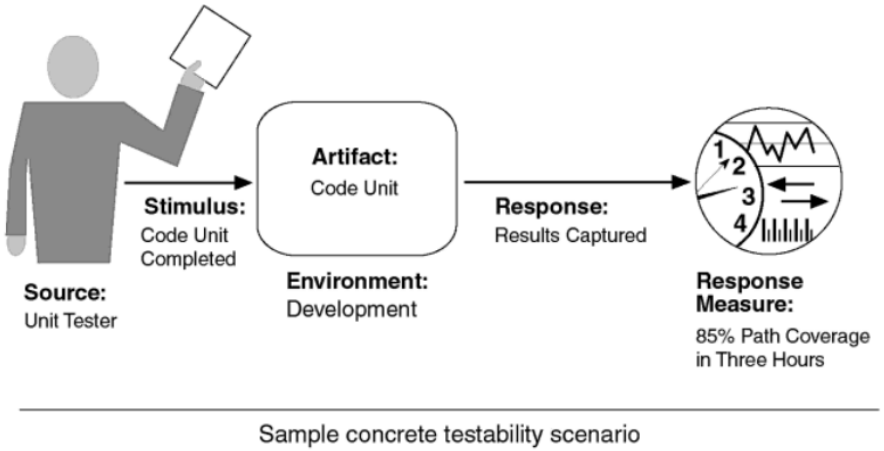


Sample concrete security scenario

Testability

- How easy (or difficult) is it to detect (and locate) errors in the system?
- «*A system is testable, if it gives up its faults easily*»

- Source: unit tester
- Stimulus: code unit completed
- Artifact: code unit
- Environment: development
- response: written & executed tests with results
- measure: 85% path coverage in 3 hours



Sample concrete testability scenario

Example Qualities of AI Systems

- Quality attributes in destination control
  - Usability: what happens to a passenger when transportation capacity is exceeded?
  - Portability: how to adapt to different buildings and users?
  - Performance: what to optimize to scale the control to huge traffic density?

- Quality attributes in cable tree wiring control software
  - Usability: ease of applying the software
  - Performance: how many cables in an instance can the CP-SAT solver handle to find optimal solutions in 5 minutes?

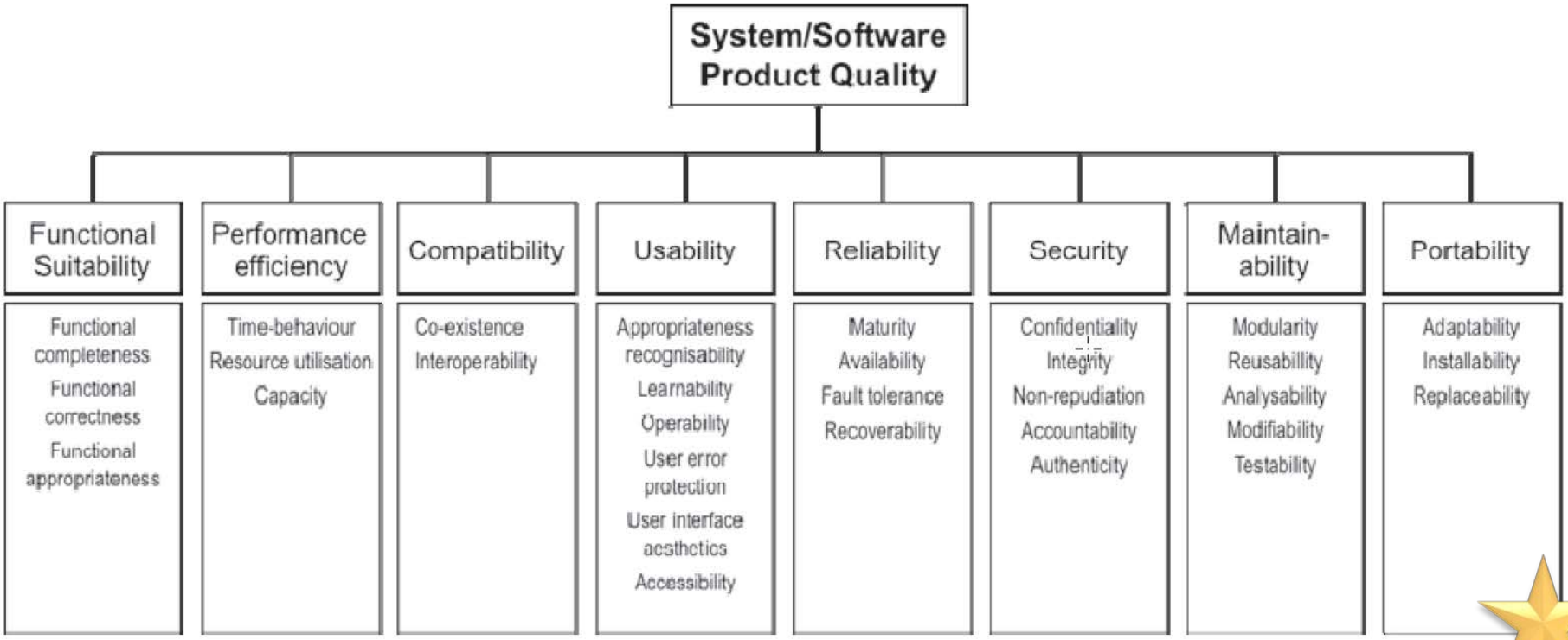
Writing Good Quality Attributes

1. Consult the sources of a functional request and further break down how the function should work – «how will an external user (human or system) perceive the quality?
2. Analyze collected stimuli, responses, measures – what is the most important aspect of the quality?
3. Think about how a required response can be achieved
   - What exactly will happen as output of the system?
   - What can cause a system fail to produce the response and how will this affect the attribute? Which system structures can prevent the failure?
4. Focus on measurability! By what method will you measure the response? How much effort will it be?

Quality Attributes in ISO/IEC 25010:2011 (Reviewed and confirmed in 2017)

Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE)

ISO 25010 (page 10-16) explains the quality criteria shown below



Summary

- Non-functional requirements (system qualities) define the architecture!
- Put the 2 most important quality attributes in the focus of architectural thinking
- Use scenarios to make qualities measurable
- Writing good scenarios takes time and requires a detailed analysis to understand stimulus source, stimulus, environment, artifact, response and measure
- Scenario patterns provide helpful guidance
- Write more than one (many!) scenarios for each quality attribute

Working Questions

1. What role do non-functional requirements and quality attributes play in architectural thinking?
2. Give examples of quality attributes and explain what quality aspect of a system they describe.
3. How and why do we specify a quality attribute using scenarios?
4. What is the relationship between a quality attribute and a use case? How would you add information about a quality attribute in a use case description?
5. The ISO/IEC 25010 standard lists functional suitability as a quality attribute. Explain why.