AARHUS UNIVERSITET

Software Architecture in Practice

Architectural Requirements:

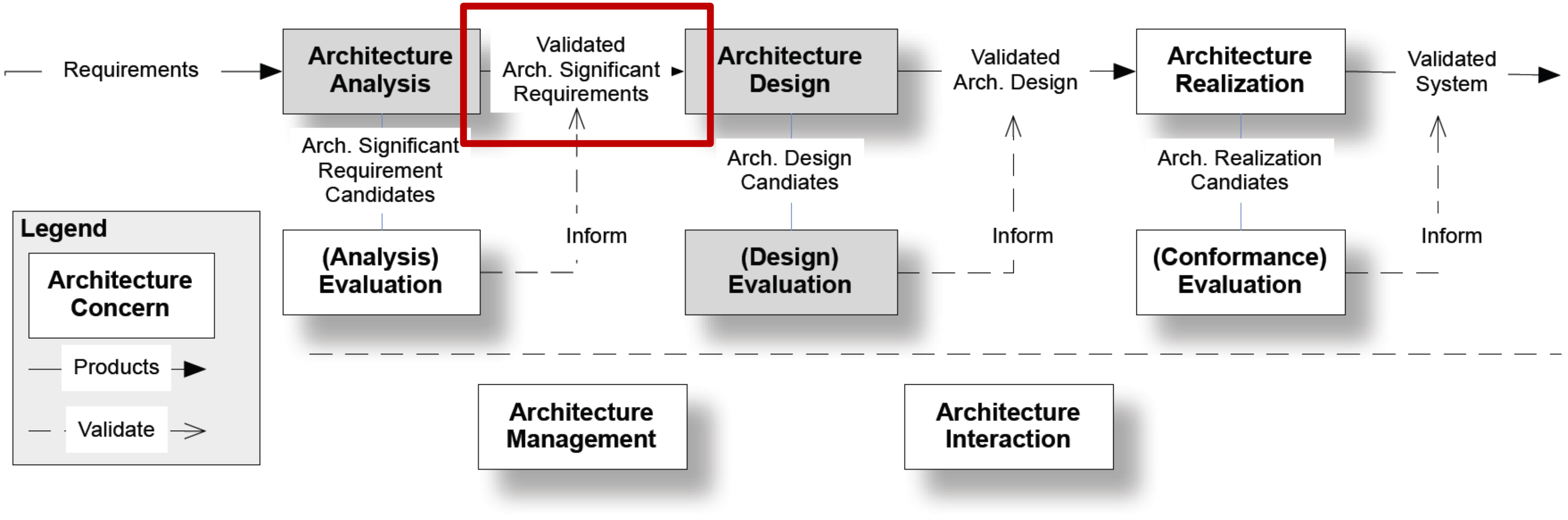
Quality Attributes

And

Quality Attribute Scenarios (QAS)

Introduction

- To design – we need *requirements*
  - And to design architecture – we need architectural requirements



AARHUS UNIVERSITET

But – let us start with an exercise

AARHUS UNIVERSITET

Good or bad architecture?

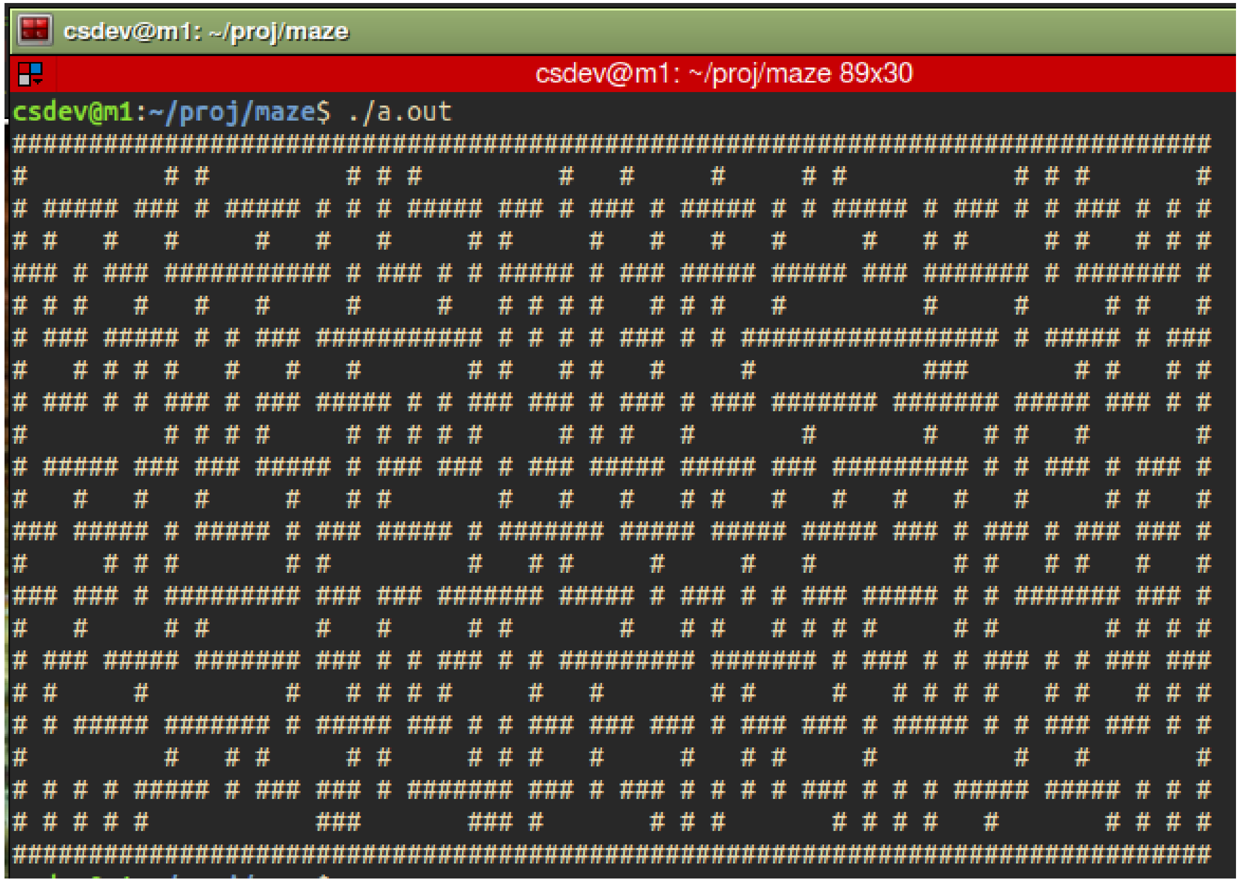
- Question: Is this little C program an example of *good* or *bad* software?

```
int a[1817];main(z,p,q,r){for(p=80;q+p-80;p-=2*a[p])for(z=9;z--
;)q=3&(r=time(0) +r*57)/7,q=q?q-1?q-2?1-p%79?-1:0:p%79-
77?1:0:p<1659?79:0:p>158?-79:0,q?!a[p+q*2
]?a[p+=a[p+=q]=q]=q:0:0;for(;q++-1817;)printf(q%79?"%c":"%c\n","
#"[!a[q-1]]);}
```

- Exercise 1: Argue that this is a good program!
- Exercise 2: Argue that this is a bad program !

(What does the C program do?)

- Process
  - Paste string into file ‘m.c’
  - Install ‘gcc’
  - gcc m.c
  - ./a.out
  - Done…



```
csdev@m1:~/proj/maze$ ./a.out
###############################################################################
#         # #         # # #         #   #     #     # #           # # #       #
# ##### ### # ##### # # # ##### ### # ### # ##### # # ##### # ### # # ### # # #
# #   #   #     #   #   #     # #     #   #   #   #     #   # #     # #   # # #
### # ### ########### # ### # # ##### # ### ##### ##### ### ####### # ####### #
# # #   #   #   #     #     #   # # # #   # # #   #         #     #     # #   #
# ### ##### # # ### ########### # # # # ### # # ################# # ##### # ###
#   # # # #   #   #   #       # #   # #   #     #           ###       # #   # #
# ### # # ### # ### ##### # # ### ### # ### # ### ####### ####### ##### ### # #
#         # # # #     # # # # #     # # #   #       #       #   # #   #       #
# ##### ### ### ##### # ### ### # ### ##### ##### ### ######### # # ### # ### #
#   #   #   #     #   # #       #   #   #   # #   #   #   #   #   #     # #   #
### ##### # ##### # ### ##### # ####### ##### ##### ##### ### # ### # ### ### #
#     # # #       # #         #   # #     #     #   #         # #   # #   #   #
### ### # ######### ### ### ####### ##### # ### # # ### ##### # # ####### ### #
#   #     # #       #   #     # #       #   # #   # # # #     # #       # # # #
# ### ##### ####### ### # # ### # # ######### ####### # ### # # ### # # ### ###
# #     #         #   # # # #     #   #       # #     #   # # # #   # #   # # #
# # ##### ####### # ##### ### # # ### ### ### # ### ### # ##### # # ### ### # #
#         #   # #     # #     # # #   #     #   # #     #         #   #       #
# # # # ##### # ### ### # ####### ### # ### # # # # ### # # # ##### ##### # # #
# # # # #           ###       ### #       # # #       # # # #   #       # # # #
###############################################################################
```

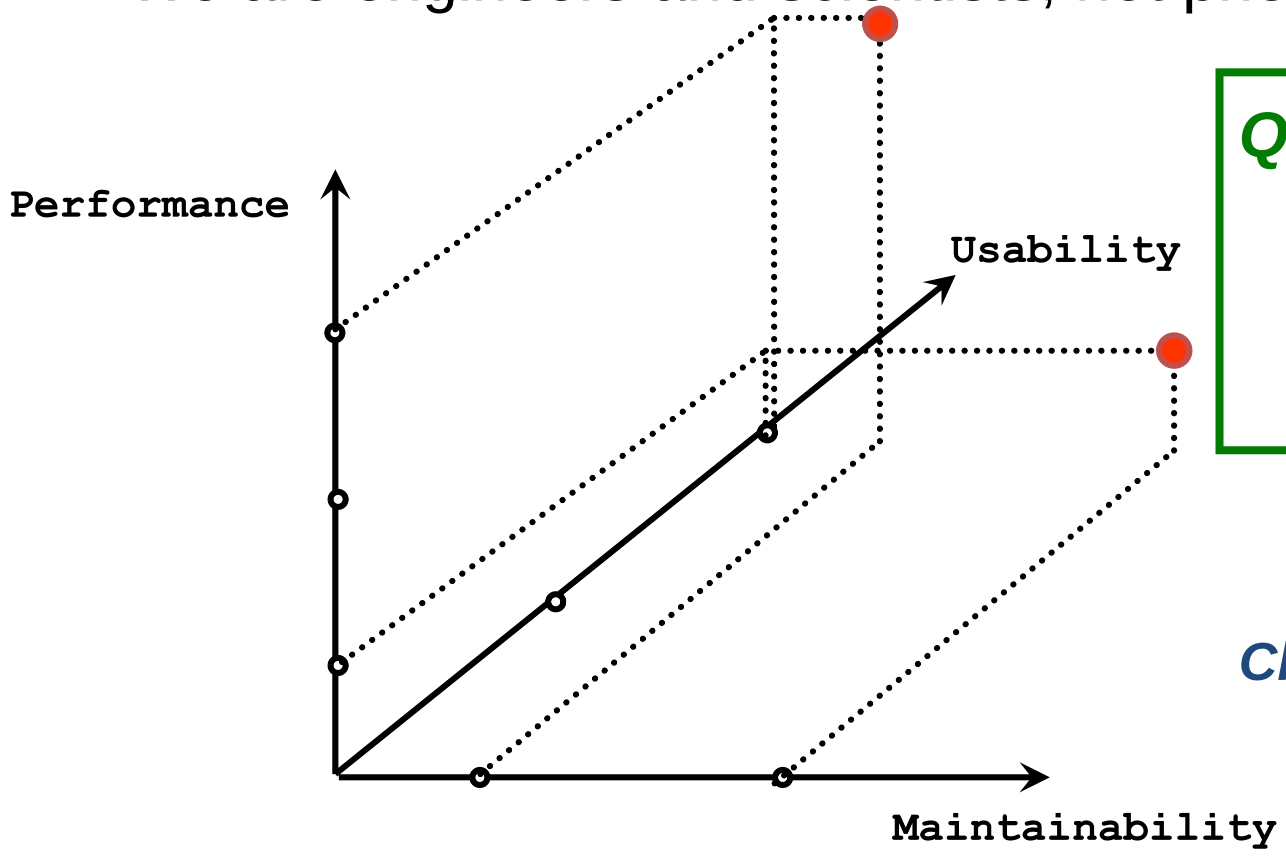
Quality Attributes

- The problem about ”good” or ”bad” is that they are subjective measures...

- We need to *measure* our software. This requires
  - that we define the aspects/**qualities** we measure
  - that we agree on some kind of scale: a **metric**

No such thing as *good* or *bad*

- We are engineers and scientists, not priests ☺



Quality Framework

Quality Attribute

Metric

Measurement

Choose alternatives

Definition

- Bass et al. §3

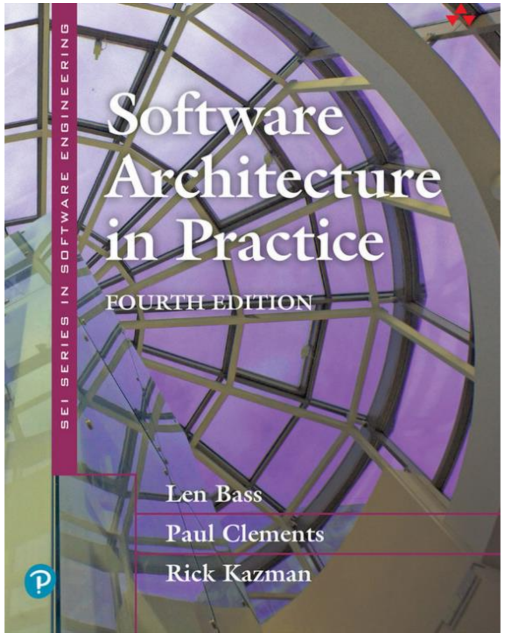
  – *A quality attribute (QA) is a **measurable or testable property** of a system that is used to indicate **how well the system satisfies the needs** of its stakeholders beyond the basic function of the system.*

- Often called 'non-functional requirements'
  – Which is a *non-sense term*… If the server is not available, will you then argue that the system is still functional?

Contribution of Bass et al.

- Proposes uniform measurement template
  - **Quality Attribute Scenarios**
  - Key point: *Same* template for radically different qualities, like *performance* or *security*

- Anchors quality in specific *context*
  - Quality Attribute **Scenarios**
  - No quality is globally achievable


SEI SERIES IN SOFTWARE ENGINEERING
Software Architecture in Practice
FOURTH EDITION
Len Bass
Paul Clements
Rick Kazman

Quality framework (Bass et al.)

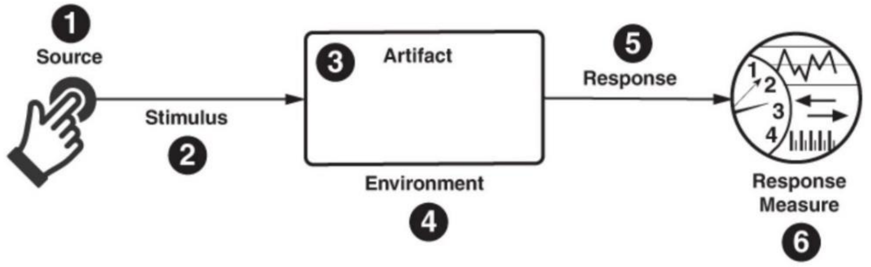
- Quality attributes
  - **Availability**
  - Deployability
  - Energy efficiency
  - Integrability
  - **Modifiability**
  - **Performance**
  - Safety
  - **Security**
  - **Testability**
  - **Usability**

- Other Quality attributes
  - Buildability
  - Conceptual integrity

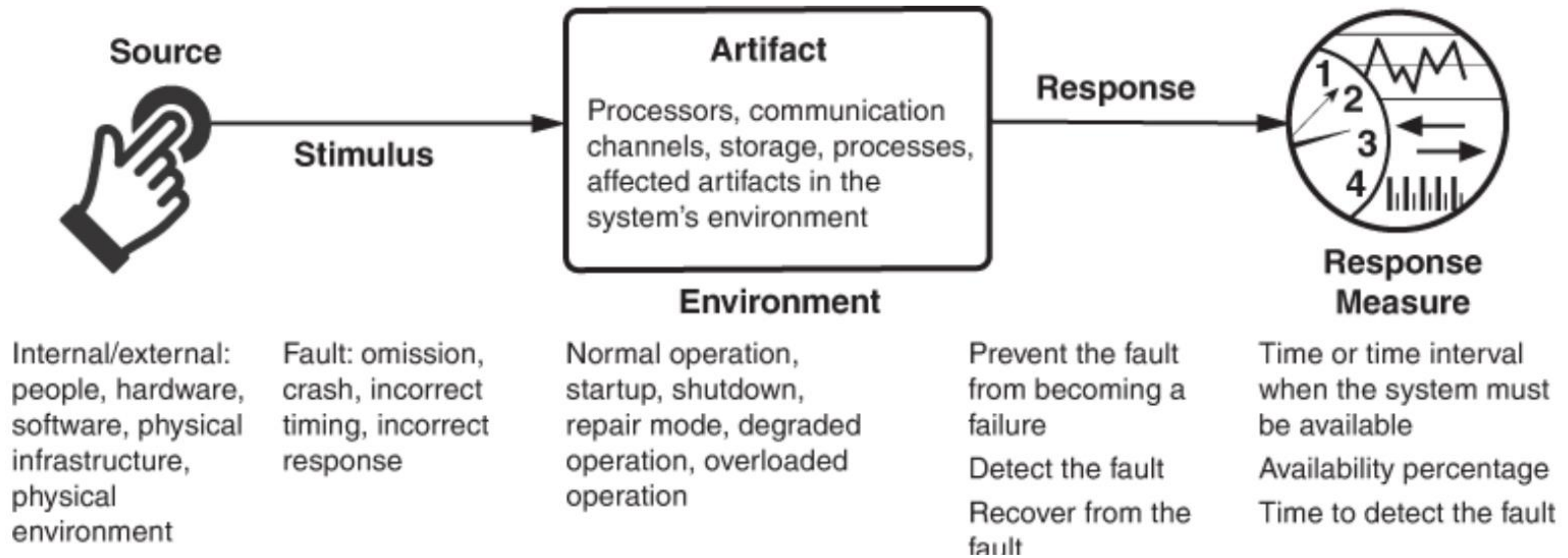
- *Historical note*
  - 4edition names 10 QA
    - 1ed only named 6 QA, those in **bold**

A writing template

- ❶ ***Source of stimulus***. This is some entity (a human, a computer system, or any other actuator) that generated the stimulus.
- ❷ ***Stimulus***. The stimulus is an event that arrives at a system.
- ❸***Environment***. The stimulus occurs within certain conditions. The system may be in an overload condition or may be running when the stimulus occurs, or some other condition may be true.
- ❹ ***Artifact***. Some artifact is stimulated. This may be the whole system or some pieces of it.
- ❺ ***Response***. The response is the activity undertaken after the arrival of the stimulus.
- ❻ ***Response measure***. When the response occurs, it should be measurable in some fashion so that the requirement can be tested.



Example: Availabilty



Processors, communication channels, storage, processes, affected artifacts in the system's environment

Internal/external: people, hardware, software, physical infrastructure, physical environment

Fault: omission, crash, incorrect timing, incorrect response

Normal operation, startup, shutdown, repair mode, degraded operation, overloaded operation

Prevent the fault from becoming a failure

Detect the fault

Recover from the fault

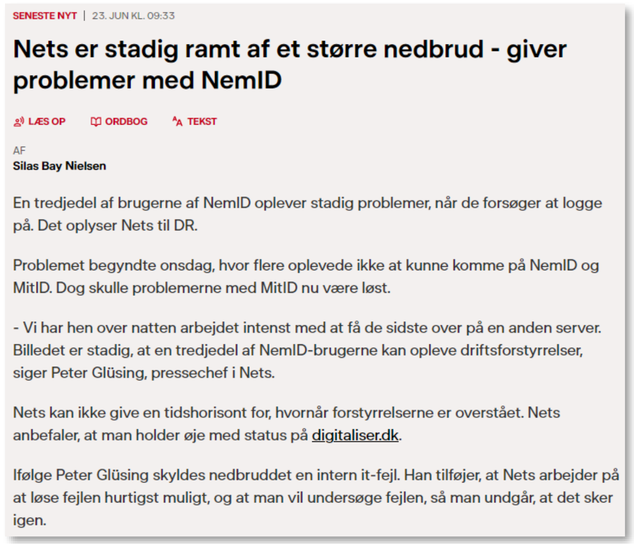
Time or time interval when the system must be available

Availability percentage

Time to detect the fault

Example: NemID

- Informal requirement on availability (“oppetid”)
  - “NemID should always be available”
- QaS formulated requirement
  - Source: Internal hardware error
  - Stimulus: Server crash
  - Artifact: Server/server program
  - Environment: Normal operation
  - Response: ① Fault logged, ② Recover
  - Response time:
    - Fault detected within 30 secs,
    - Recovery completed within 5 minutes



SENESTE NYT | 23. JUN KL. 09:33

Nets er stadig ramt af et større nedbrud - giver problemer med NemID

LÆS OP ORDBOG TEKST

AF
Silas Bay Nielsen

En tredjedel af brugerne af NemID oplever stadig problemer, når de forsøger at logge på. Det oplyser Nets til DR.

Problemet begyndte onsdag, hvor flere oplevede ikke at kunne komme på NemID og MitID. Dog skulle problemerne med MitID nu være løst.

- Vi har hen over natten arbejdet intenst med at få de sidste over på en anden server. Billedet er stadig, at en tredjedel af NemID-brugerne kan opleve driftsforstyrrelser, siger Peter Glüsing, pressechef i Nets.

Nets kan ikke give en tidshorisont for, hvornår forstyrrelserne er overstået. Nets anbefaler, at man holder øje med status på digitaliser.dk.

Ifølge Peter Glüsing skyldes nedbruddet en intern it-fejl. Han tilføjer, at Nets arbejder på at løse fejlen hurtigst muligt, og at man vil undersøge fejlen, så man undgår, at det sker igen.

Key point

- The key point of the template is

- *Some **source** generates some events (**stimuli**) that arrives at some **artifact** under some conditions (**environment**) and must be dealt with (**response**) in a satisfactory way (**response measure** = the architectural requirement)*

AARHUS UNIVERSITET

Generators

- Chapter 4-13 lists *generators* for the contents of these QAS.
  - **General Scenarios**

- Consider them a *good first attempt*

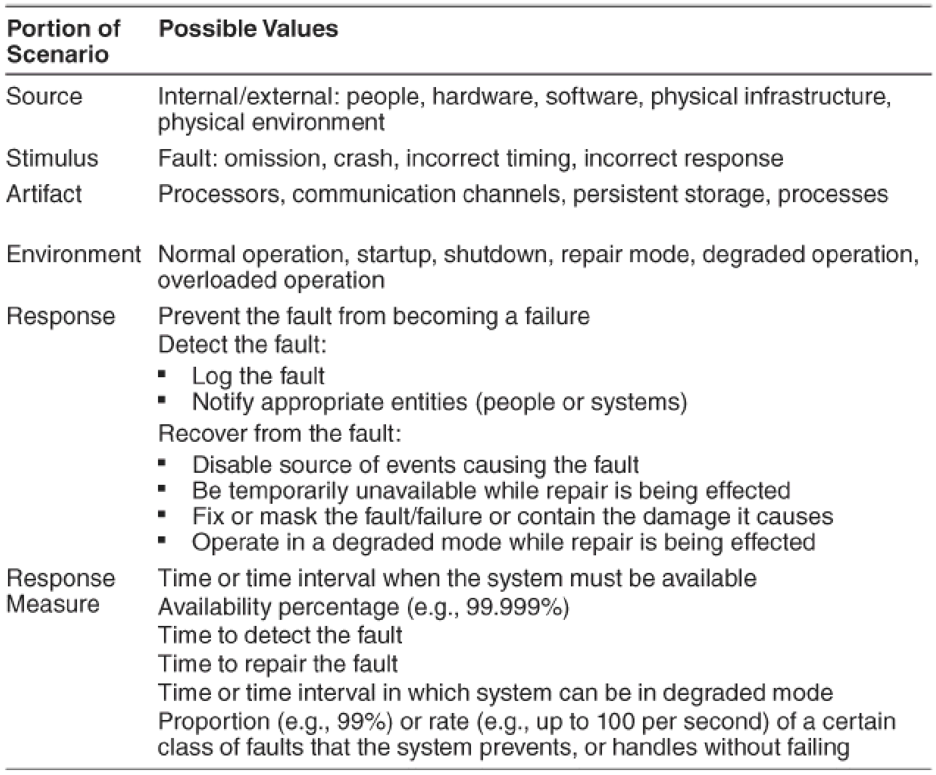
- Feel free to elaborate and refine
  - As long as you stay true to the given quality attribute and the key point of the template ☺
  - *And do not invent new, confusing, terms for those already in the generator!*

'Sløvhed' as new QA?

Example: Availability

- Concerned with the probability that the system will be operational when needed

Table 5.3. Availability General Scenario



Portion of Scenario	Possible Values
Source	Internal/external: people, hardware, software, physical infrastructure, physical environment
Stimulus	Fault: omission, crash, incorrect timing, incorrect response
Artifact	Processors, communication channels, persistent storage, processes
Environment	Normal operation, startup, shutdown, repair mode, degraded operation, overloaded operation
Response	Prevent the fault from becoming a failure Detect the fault: ▪ Log the fault ▪ Notify appropriate entities (people or systems) 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 the system must be available Availability percentage (e.g., 99.999%) Time to detect the fault Time to repair the fault Time or time interval in which system can be in degraded mode Proportion (e.g., 99%) or rate (e.g., up to 100 per second) of a certain class of faults that the system prevents, or handles without failing

Note: Table is from 3rd edition, but same contents…

Example: Availability

- An *internal crash* occurs in the *inventory database process* during *normal operation.* The response is *logging of the fault* and recovery by *repair and return to normal operation (inventory db up and running) within 5 seconds*

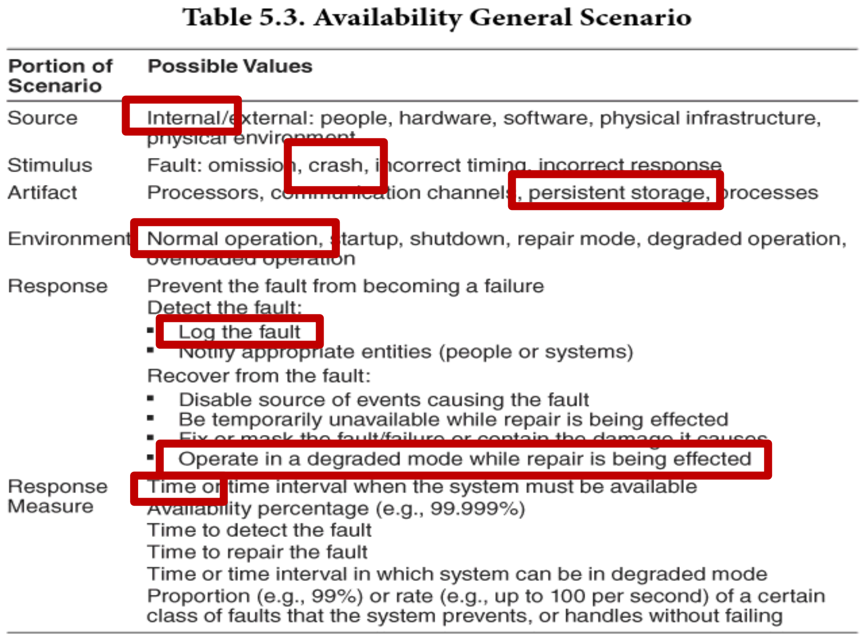


Table 5.3. Availability General Scenario

Portion of Scenario	Possible Values
Source	Internal/external: people, hardware, software, physical infrastructure, physical environment
Stimulus	Fault: omission, crash, incorrect timing, incorrect response
Artifact	Processors, communication channels, persistent storage, processes
Environment	Normal operation, startup, shutdown, repair mode, degraded operation, overloaded operation
Response	Prevent the fault from becoming a failure Detect the fault: ▪ Log the fault ▪ Notify appropriate entities (people or systems) 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 the system must be available Availability percentage (e.g., 99.999%) Time to detect the fault Time to repair the fault Time or time interval in which system can be in degraded mode Proportion (e.g., 99%) or rate (e.g., up to 100 per second) of a certain class of faults that the system prevents, or handles without failing

Tactics

- *So – what do we do then?*

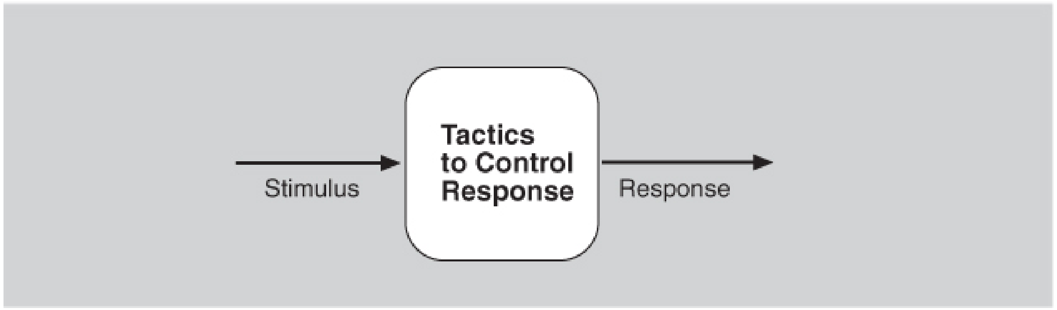


Figure 4.3. Tactics are intended to control responses to stimuli.

Definition: **Tactic** is a design decision that influences the achievement of a quality attribute response

Major Focus next seminar!

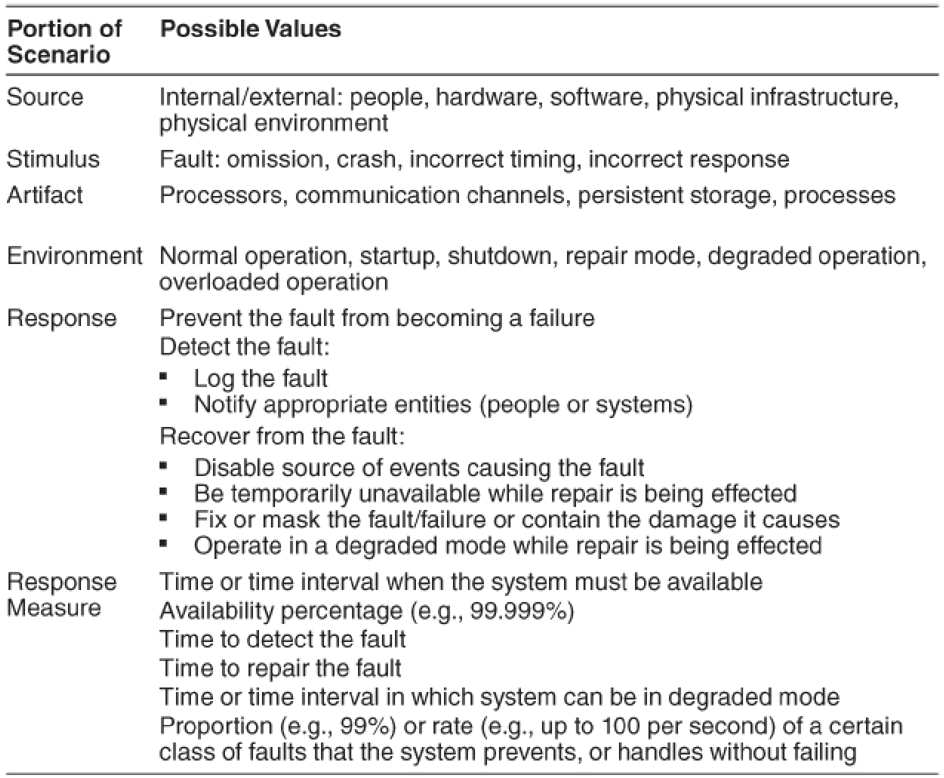
AARHUS UNIVERSITET

The QAS General Scenarios

Availability

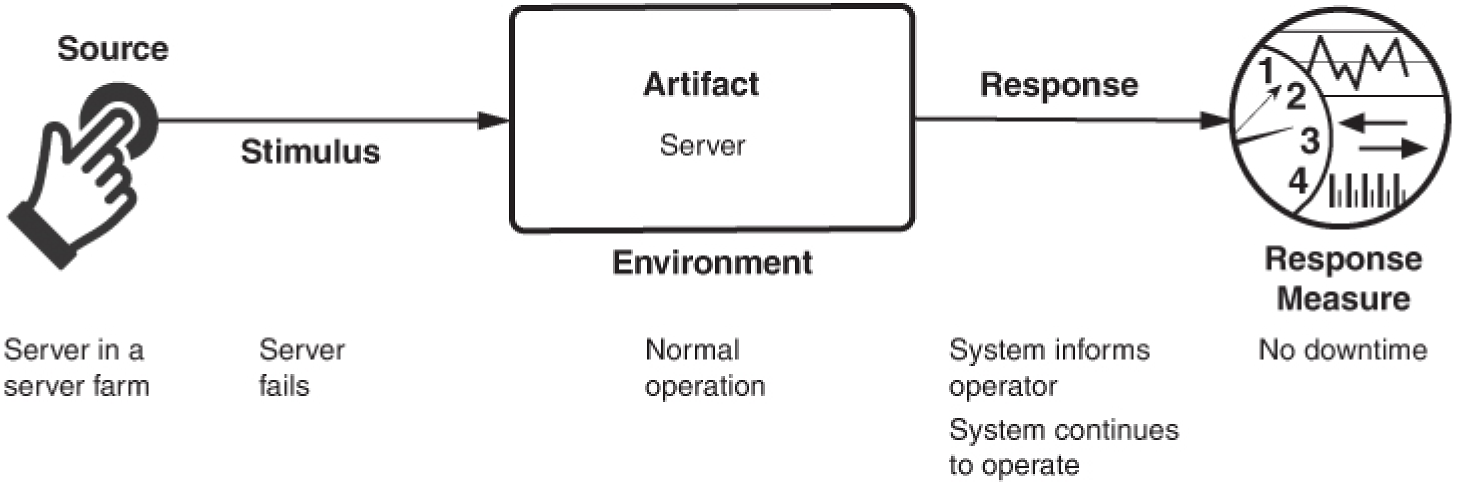
- Concerned with the *probability that the system will be operational when needed*

Table 5.3. Availability General Scenario



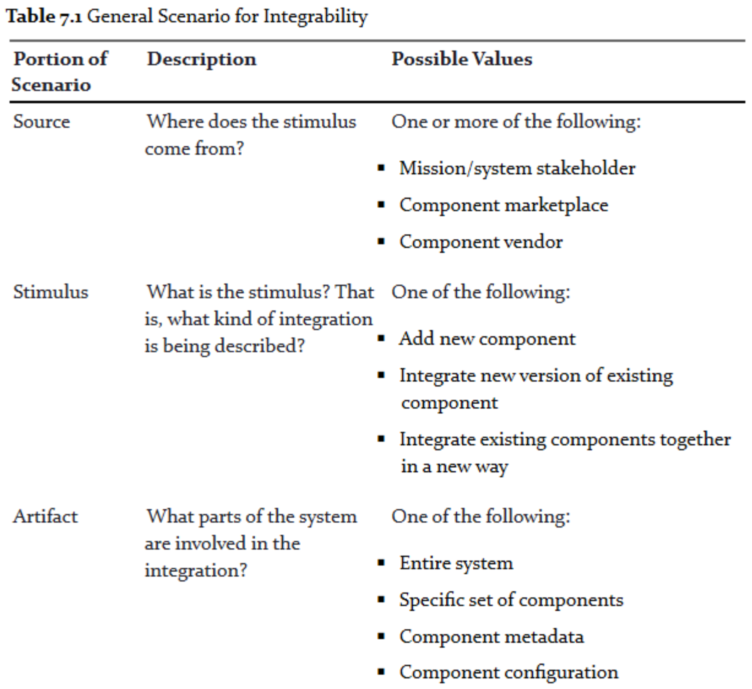
Portion of Scenario	Possible Values
Source	Internal/external: people, hardware, software, physical infrastructure, physical environment
Stimulus	Fault: omission, crash, incorrect timing, incorrect response
Artifact	Processors, communication channels, persistent storage, processes
Environment	Normal operation, startup, shutdown, repair mode, degraded operation, overloaded operation
Response	Prevent the fault from becoming a failure Detect the fault: ▪ Log the fault ▪ Notify appropriate entities (people or systems) 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 the system must be available Availability percentage (e.g., 99.999%) Time to detect the fault Time to repair the fault Time or time interval in which system can be in degraded mode Proportion (e.g., 99%) or rate (e.g., up to 100 per second) of a certain class of faults that the system prevents, or handles without failing

Example


Source
Stimulus
Artifact
Server
Response
Environment
Response Measure
1
2
3
4
Server in a server farm
Server fails
Normal operation
System informs operator
System continues to operate
No downtime

Integrability

- Concerned with the *costs and risks of integrating separately developed components (so the resulting system behaves correctly)*
  - *Especially in the face of externally developed components*



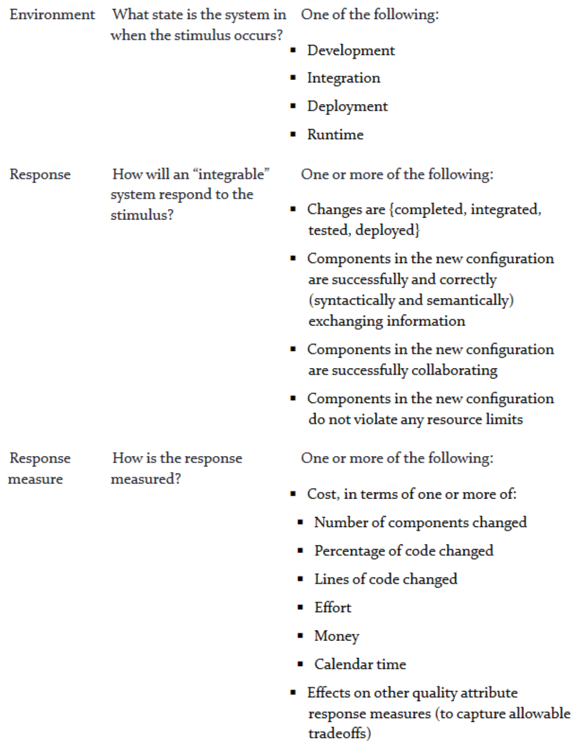
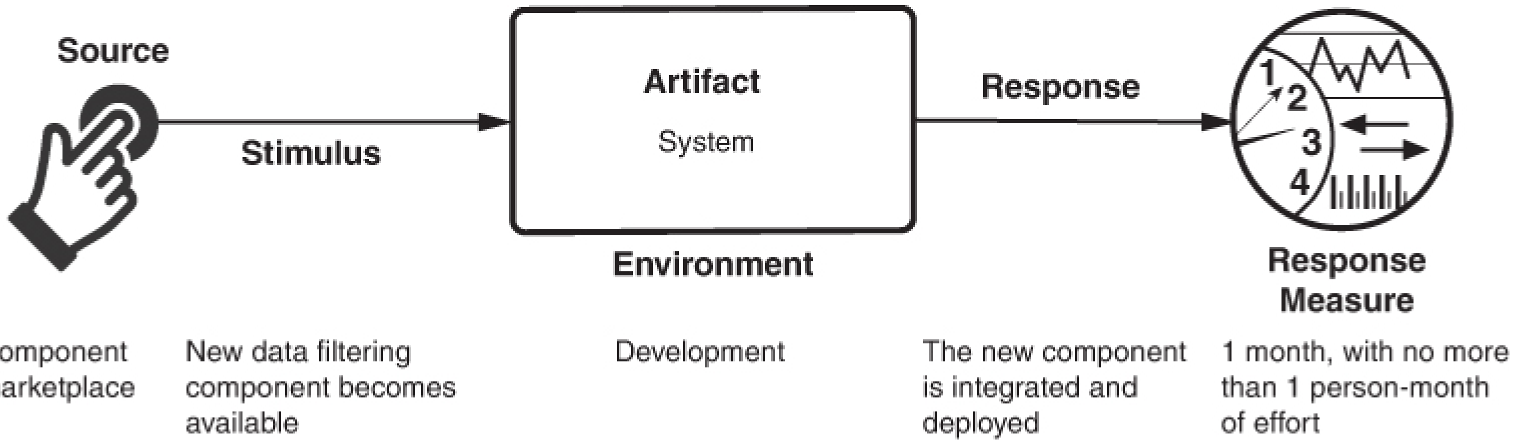


Table 7.1 General Scenario for Integrability

Portion of Scenario	Description	Possible Values
Source	Where does the stimulus come from?	One or more of the following: ▪ Mission/system stakeholder ▪ Component marketplace ▪ Component vendor
Stimulus	What is the stimulus? That is, what kind of integration is being described?	One of the following: ▪ Add new component ▪ Integrate new version of existing component ▪ Integrate existing components together in a new way
Artifact	What parts of the system are involved in the integration?	One of the following: ▪ Entire system ▪ Specific set of components ▪ Component metadata ▪ Component configuration
Environment	What state is the system in when the stimulus occurs?	One of the following: ▪ Development ▪ Integration ▪ Deployment ▪ Runtime
Response	How will an "integrable" system respond to the stimulus?	One or more of the following: ▪ Changes are {completed, integrated, tested, deployed} ▪ Components in the new configuration are successfully and correctly (syntactically and semantically) exchanging information ▪ Components in the new configuration are successfully collaborating ▪ Components in the new configuration do not violate any resource limits
Response measure	How is the response measured?	One or more of the following: ▪ Cost, in terms of one or more of: ▪ Number of components changed ▪ Percentage of code changed ▪ Lines of code changed ▪ Effort ▪ Money ▪ Calendar time ▪ Effects on other quality attribute response measures (to capture allowable tradeoffs)

Example

- (Bit weird this one?)



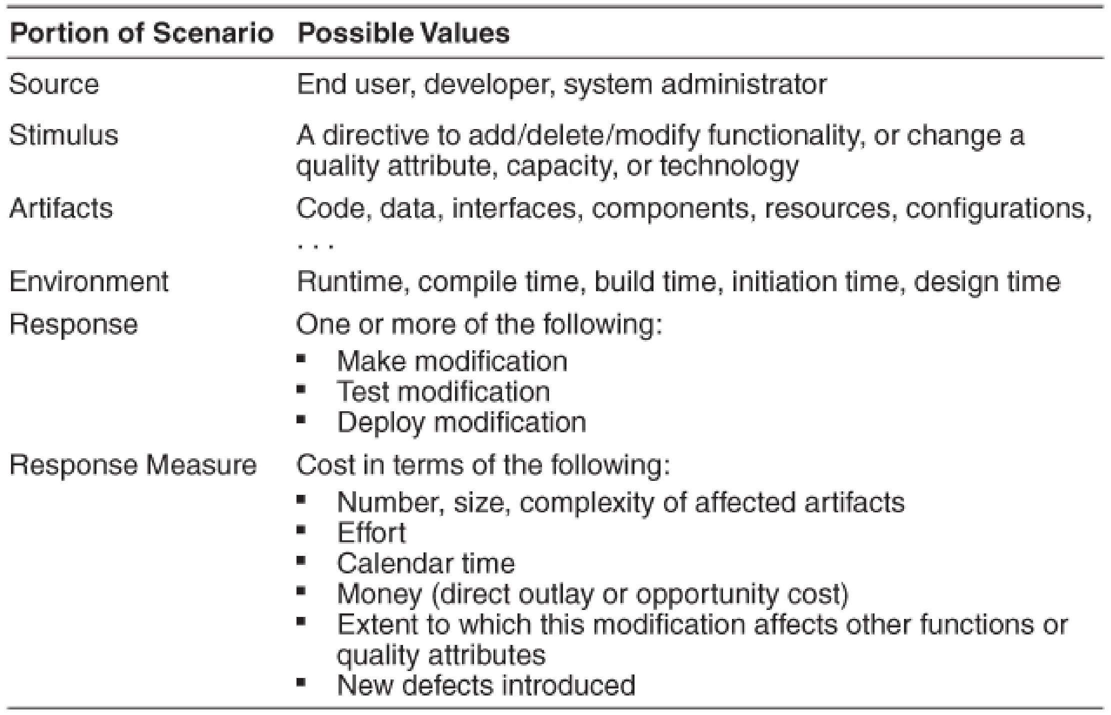
Source	Stimulus	Environment	Response	Response Measure
Component marketplace	New data filtering component becomes available	Development	The new component is integrated and deployed	1 month, with no more than 1 person-month of effort

Figure 7.1 Sample integrability scenario

Modifiability

- Concerned with *the ease with which the system supports change*

Table 7.1. Modifiability General Scenario

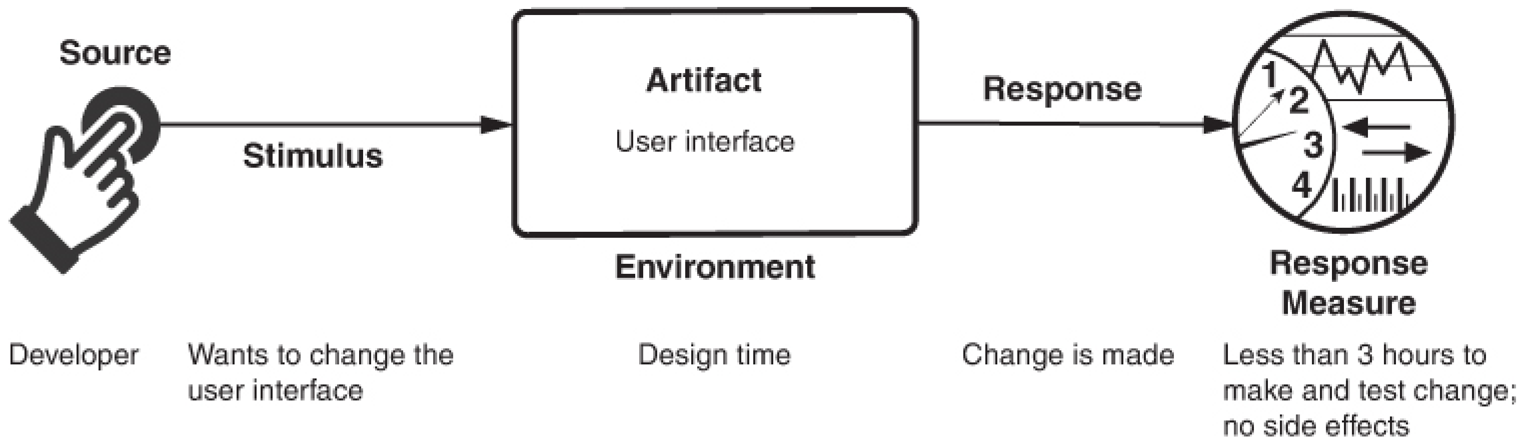


Portion of Scenario	Possible Values
Source	End user, developer, system administrator
Stimulus	A directive to add/delete/modify functionality, or change a quality attribute, capacity, or technology
Artifacts	Code, data, interfaces, components, resources, configurations, . . .
Environment	Runtime, compile time, build time, initiation time, design time
Response	One or more of the following: ▪ Make modification ▪ Test modification ▪ Deploy modification
Response Measure	Cost in terms of the following: ▪ Number, size, complexity of affected artifacts ▪ Effort ▪ Calendar time ▪ Money (direct outlay or opportunity cost) ▪ Extent to which this modification affects other functions or quality attributes ▪ New defects introduced

AARHUS UNIVERSITET

Example

- (A bad example – 'what UI change?')



AARHUS UNIVERSITET

Performance

- Concerned with *ability to meet timing requirements*

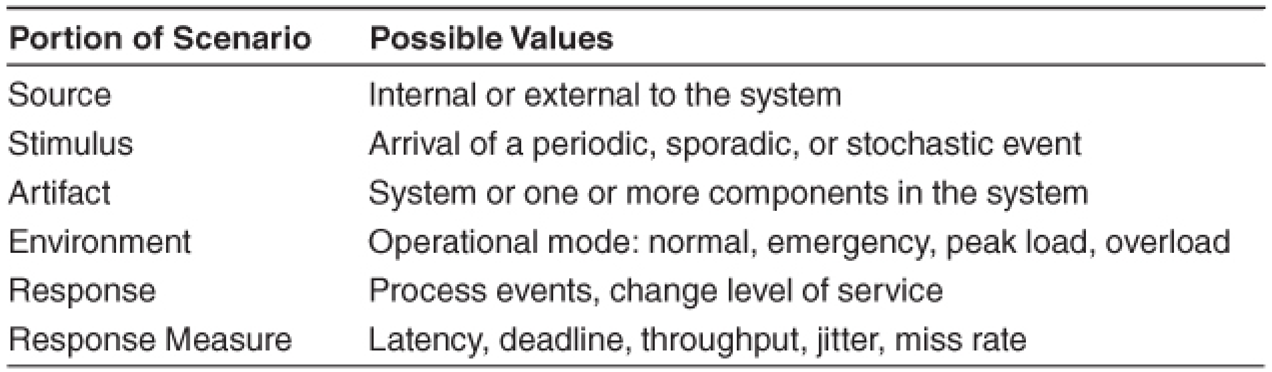
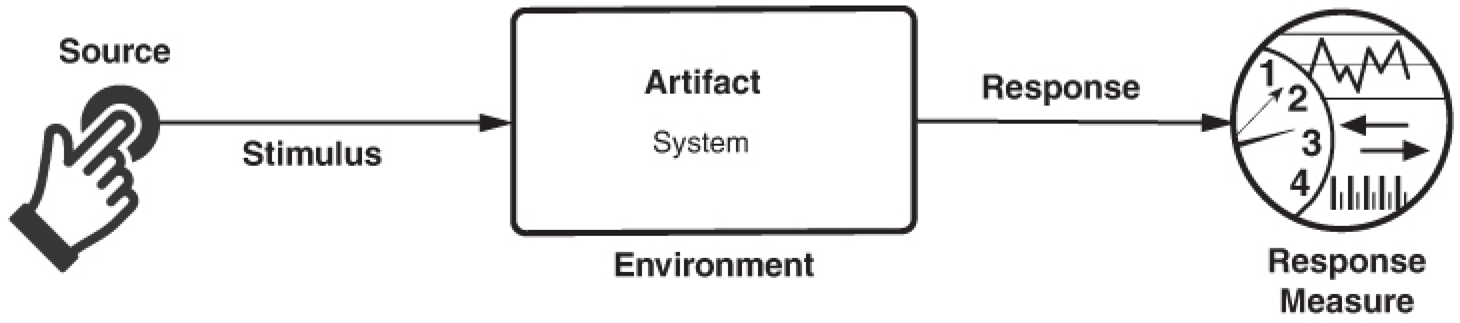


Table 8.1. Performance General Scenario

Portion of Scenario	Possible Values
Source	Internal or external to the system
Stimulus	Arrival of a periodic, sporadic, or stochastic event
Artifact	System or one or more components in the system
Environment	Operational mode: normal, emergency, peak load, overload
Response	Process events, change level of service
Response Measure	Latency, deadline, throughput, jitter, miss rate

Note: 4ed's Table a bit more elaborate…

Example



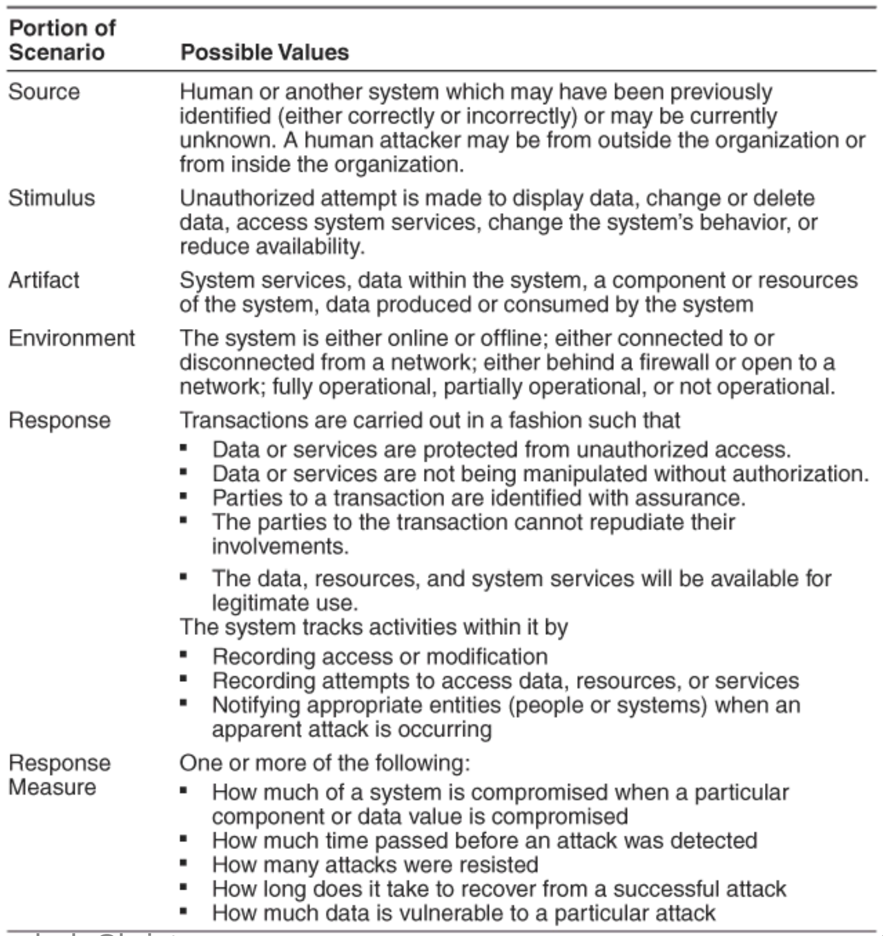
500 users	Initiate 2,000 requests in a 30-second interval	Normal operations	Processes all requests	Average latency of 2 seconds

AARHUS UNIVERSITET

Security

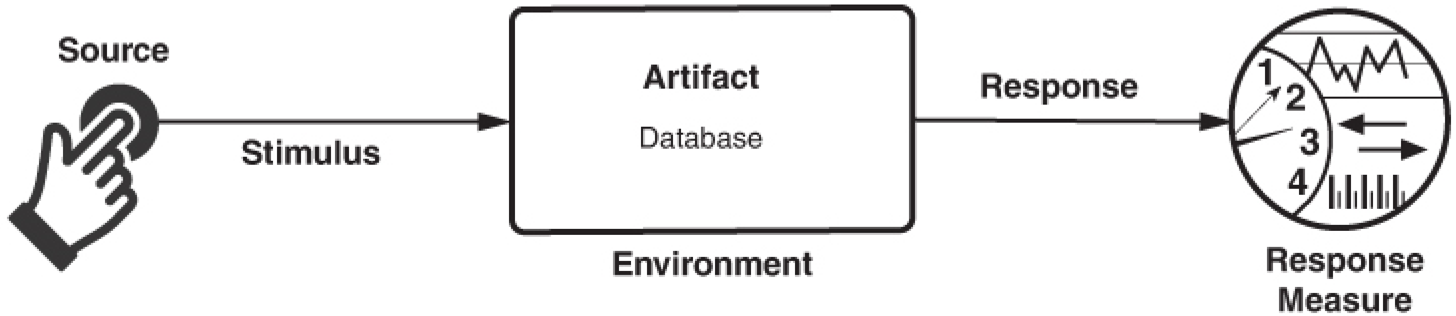
- Concerned with *ability to protect data and information from unauthorized access while still providing access to people/systems that are authorized*

Table 9.1. Security General Scenario



Portion of Scenario	Possible Values
Source	Human or another system which may have been previously identified (either correctly or incorrectly) or may be currently unknown. A human attacker may be from outside the organization or from inside the organization.
Stimulus	Unauthorized attempt is made to display data, change or delete data, access system services, change the system's behavior, or reduce availability.
Artifact	System services, data within the system, a component or resources of the system, data produced or consumed by the system
Environment	The system is either online or offline; either connected to or disconnected from a network; either behind a firewall or open to a network; fully operational, partially operational, or not operational.
Response	Transactions are carried out in a fashion such that ▪ Data or services are protected from unauthorized access. ▪ Data or services are not being manipulated without authorization. ▪ Parties to a transaction are identified with assurance. ▪ The parties to the transaction cannot repudiate their involvements. ▪ The data, resources, and system services will be available for legitimate use. The system tracks activities within it by ▪ Recording access or modification ▪ Recording attempts to access data, resources, or services ▪ Notifying appropriate entities (people or systems) when an apparent attack is occurring
Response Measure	One or more of the following: ▪ How much of a system is compromised when a particular component or data value is compromised ▪ How much time passed before an attack was detected ▪ How many attacks were resisted ▪ How long does it take to recover from a successful attack ▪ How much data is vulnerable to a particular attack

Example

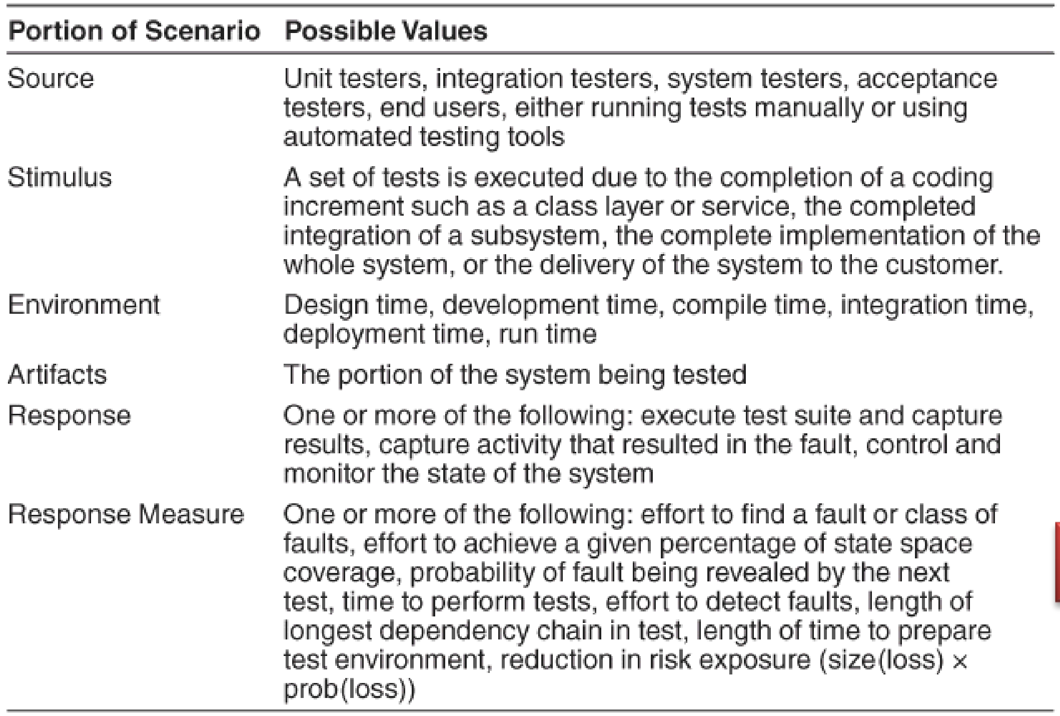


Source	Stimulus	Environment	Response	Response Measure
A disgruntled employee at a remote location	Attempts to improperly modify the pay rate table	Normal operations	The unauthorized access is detected and the system maintains an audit trail	Correct data is restored within one day

Testability

- Concerned with the *ease with which the software can be made to demonstrate its faults*

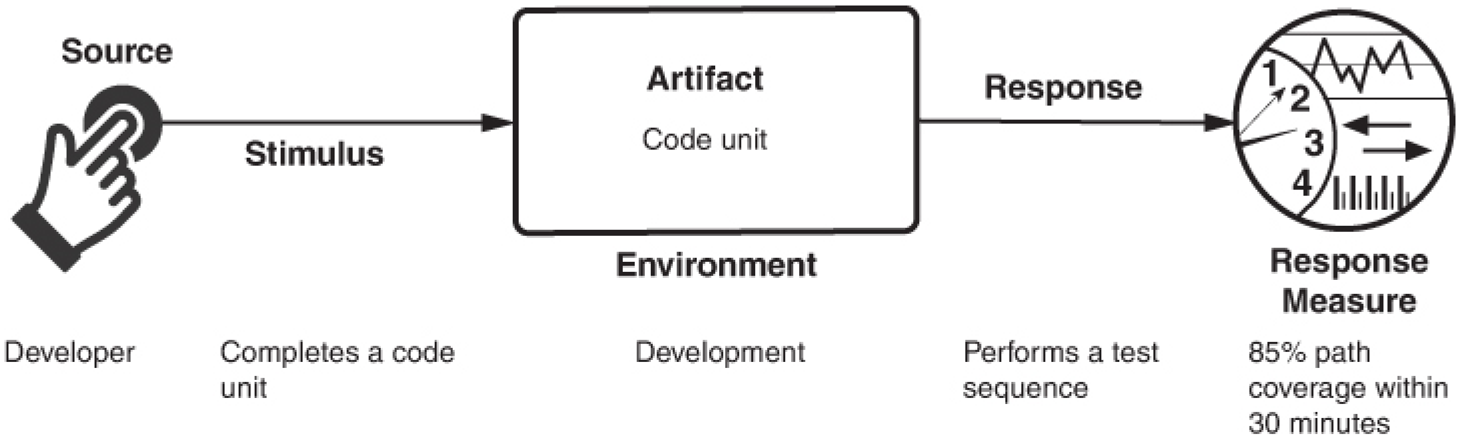
Table 10.1. Testability General Scenario



Portion of Scenario	Possible Values
Source	Unit testers, integration testers, system testers, acceptance testers, end users, either running tests manually or using automated testing tools
Stimulus	A set of tests is executed due to the completion of a coding increment such as a class layer or service, the completed integration of a subsystem, the complete implementation of the whole system, or the delivery of the system to the customer.
Environment	Design time, development time, compile time, integration time, deployment time, run time
Artifacts	The portion of the system being tested
Response	One or more of the following: execute test suite and capture results, capture activity that resulted in the fault, control and monitor the state of the system
Response Measure	One or more of the following: effort to find a fault or class of faults, effort to achieve a given percentage of state space coverage, probability of fault being revealed by the next test, time to perform tests, effort to detect faults, length of longest dependency chain in test, length of time to prepare test environment, reduction in risk exposure (size(loss) × prob(loss))

*)

Example


Source
Stimulus
Artifact
Code unit
Environment
Response
1
2
3
4
Response Measure
Developer
Completes a code unit
Development
Performs a test sequence
85% path coverage within 30 minutes

Comment

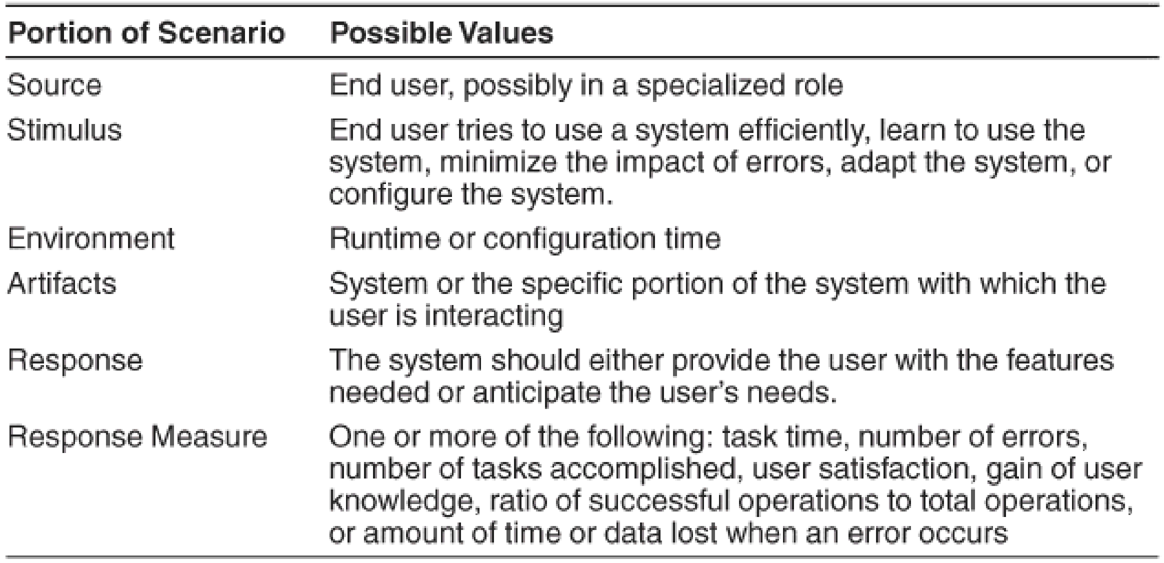
- *) I find an important Response Measure is missing in the generator

- Namely: **Time to express/execute test case**
- To me, a *testable architecture is one that allows me to express an automated test case easily (= in short time)*
  - I.e. having the ‘local method call IPC’ in Broker allows me to express full client-server roundtrip tests easily
    - Compared to if I had to spin up a server every time…

Usability

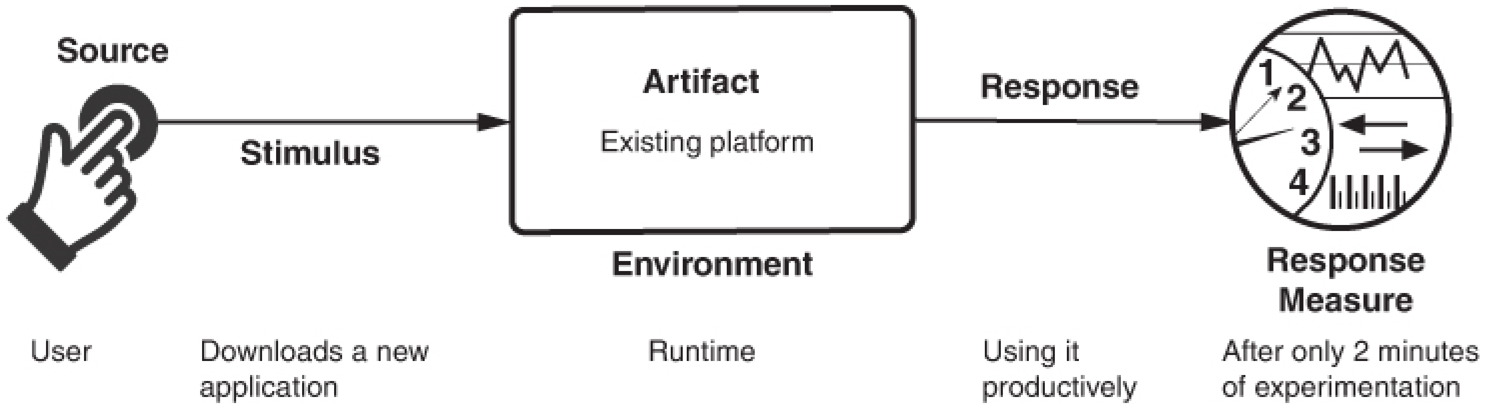
- Concerned with how *easy it is for the user to accomplish a desired task and the kind of user support the system provides*

Table 11.1. Usability General Scenario



Portion of Scenario	Possible Values
Source	End user, possibly in a specialized role
Stimulus	End user tries to use a system efficiently, learn to use the system, minimize the impact of errors, adapt the system, or configure the system.
Environment	Runtime or configuration time
Artifacts	System or the specific portion of the system with which the user is interacting
Response	The system should either provide the user with the features needed or anticipate the user's needs.
Response Measure	One or more of the following: task time, number of errors, number of tasks accomplished, user satisfaction, gain of user knowledge, ratio of successful operations to total operations, or amount of time or data lost when an error occurs

Example


Source
Stimulus
Artifact
Existing platform
Environment
Response
Response Measure
User
Downloads a new application
Runtime
Using it productively
After only 2 minutes of experimentation

Deployability

- Concerned with *the time and effort for software to be allocated to an environment for execution*

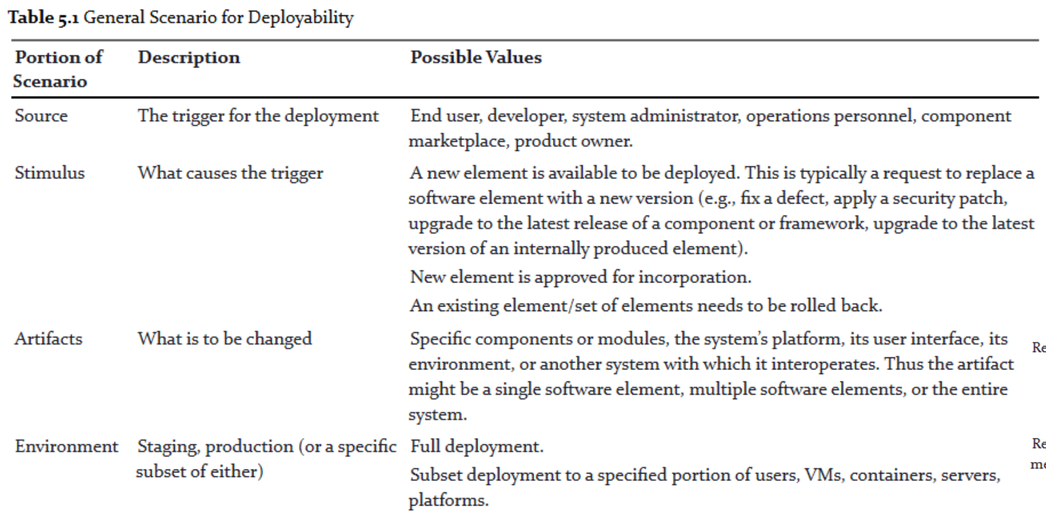
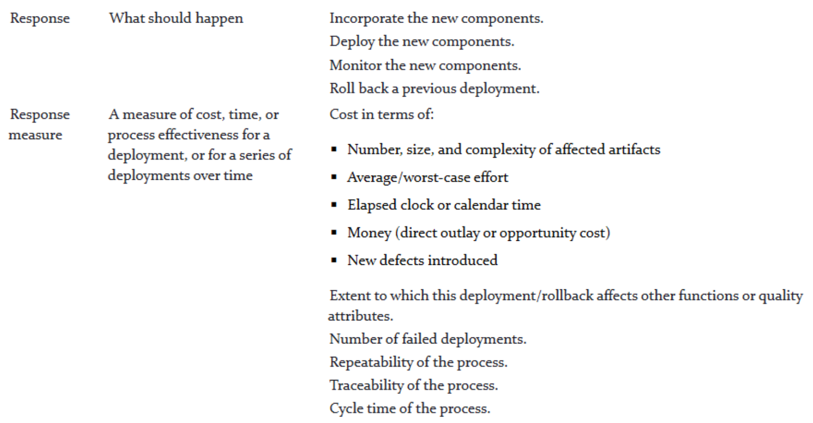
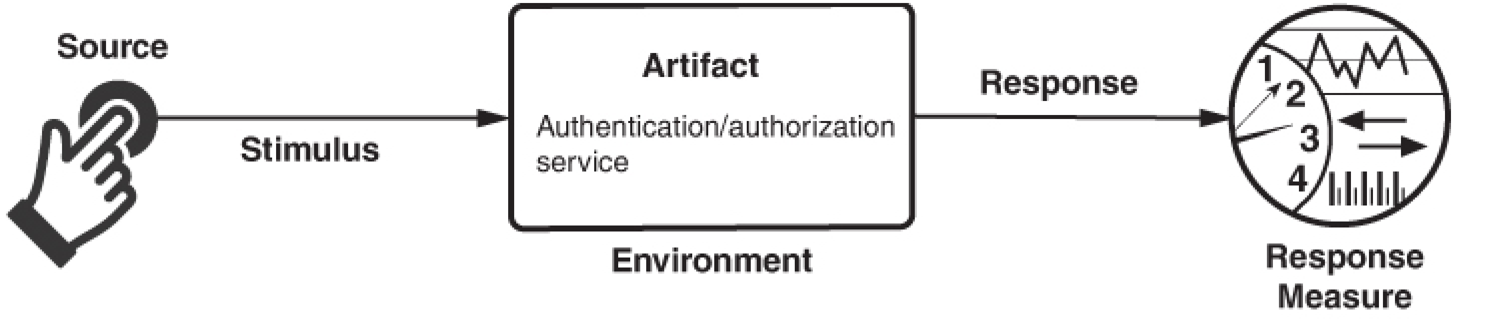



Table 5.1 General Scenario for Deployability

Portion of Scenario	Description	Possible Values
Source	The trigger for the deployment	End user, developer, system administrator, operations personnel, component marketplace, product owner.
Stimulus	What causes the trigger	A new element is available to be deployed. This is typically a request to replace a software element with a new version (e.g., fix a defect, apply a security patch, upgrade to the latest release of a component or framework, upgrade to the latest version of an internally produced element). New element is approved for incorporation. An existing element/set of elements needs to be rolled back.
Artifacts	What is to be changed	Specific components or modules, the system's platform, its user interface, its environment, or another system with which it interoperates. Thus the artifact might be a single software element, multiple software elements, or the entire system.
Environment	Staging, production (or a specific subset of either)	Full deployment. Subset deployment to a specified portion of users, VMs, containers, servers, platforms.
Response	What should happen	Incorporate the new components. Deploy the new components. Monitor the new components. Roll back a previous deployment.
Response measure	A measure of cost, time, or process effectiveness for a deployment, or for a series of deployments over time	Cost in terms of: ▪ Number, size, and complexity of affected artifacts ▪ Average/worst-case effort ▪ Elapsed clock or calendar time ▪ Money (direct outlay or opportunity cost) ▪ New defects introduced Extent to which this deployment/rollback affects other functions or quality attributes. Number of failed deployments. Repeatability of the process. Traceability of the process. Cycle time of the process.

Example

- (A bit weird example?)



Component marketplace	New release of the authentication/authorization service is made available and the product owner decides to incorporate it	Production	The new service is tested in-house and deployed to production servers	Within 40 hours and no more than 120 person-hours of effort No defects introduced; no SLA violated

Deployability is highly important for modern software systems. However, it is a major focus of my 'MSDO' fagpakke, so in this course – I will not address it further here…

AARHUS UNIVERSITET

The Last Two

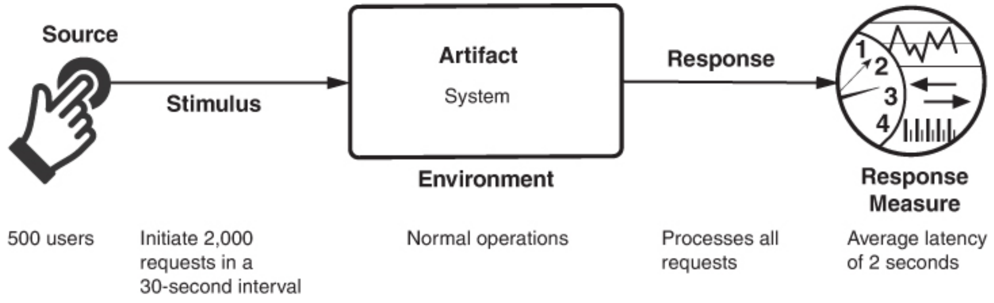
- Safety
  - Concerned with *the system's ability to avoid straying into states that cause or lead to damage, injury, or loss of life to actors in its environment.*
    - We will not cover safety in this course…

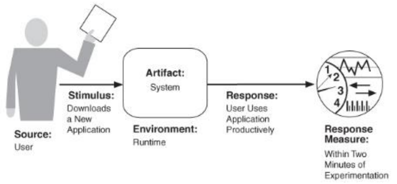
- Energy Efficiency
  - Concerned with *the system's ability to conserve/minimize power consumption while providing it's services*
    - We will cover that in much more detail in the second course…

Summary

QAS

- QAS capture *architectual quality attribute requirements* in a common format
  - *Some **source** generates some events (**stimuli**) that arrives at some **artefact** under some conditions (**environment**) and must be dealt with (response) in a satisfactory way (**response measure** = the architectural requirement)*



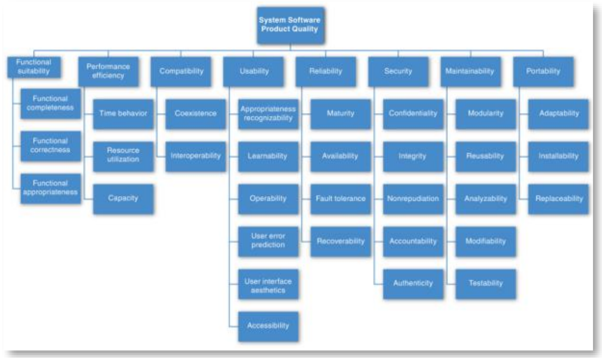

Source:
User
Stimulus:
Downloads a New Application
Artifact:
System
Environment:
Runtime
Response:
User Uses Application Productively
1
2
3
4
Response Measure:
Within Two Minutes of Experimentation

- *The **response measure** is central – measurable!*

AARHUS UNIVERSITET

Discussion

- There are other frameworks, of course…
  - ISO 25010 / SQuaRE



- Richards & Ford
  - *Architecture Characteristics*

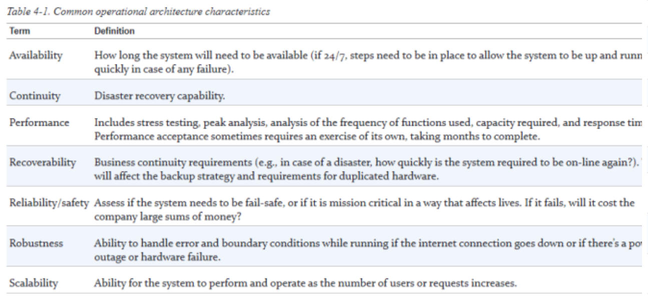


Table 4-1. Common operational architecture characteristics

Term	Definition
Availability	How long the system will need to be available (if 24/7, steps need to be in place to allow the system to be up and runn quickly in case of any failure).
Continuity	Disaster recovery capability.
Performance	Includes stress testing, peak analysis, analysis of the frequency of functions used, capacity required, and response tim Performance acceptance sometimes requires an exercise of its own, taking months to complete.
Recoverability	Business continuity requirements (e.g., in case of a disaster, how quickly is the system required to be on-line again?). will affect the backup strategy and requirements for duplicated hardware.
Reliability/safety	Assess if the system needs to be fail-safe, or if it is mission critical in a way that affects lives. If it fails, will it cost the company large sums of money?
Robustness	Ability to handle error and boundary conditions while running if the internet connection goes down or if there's a po outage or hardware failure.
Scalability	Ability for the system to perform and operate as the number of users or requests increases.

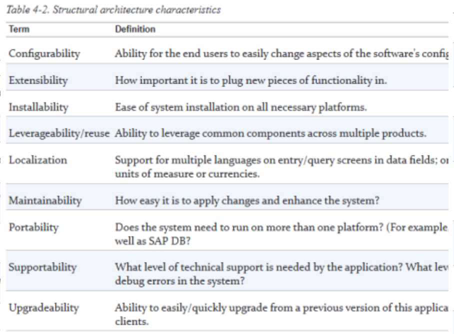


Table 4-2. Structural architecture characteristics

Term	Definition
Configurability	Ability for the end users to easily change aspects of the software's config
Extensibility	How important it is to plug new pieces of functionality in.
Installability	Ease of system installation on all necessary platforms.
Leverageability/reuse	Ability to leverage common components across multiple products.
Localization	Support for multiple languages on entry/query screens in data fields; o units of measure or currencies.
Maintainability	How easy it is to apply changes and enhance the system?
Portability	Does the system need to run on more than one platform? (For example well as SAP DB?
Supportability	What level of technical support is needed by the application? What lev debug errors in the system?
Upgradeability	Ability to easily/quickly upgrade from a previous version of this applica clients.

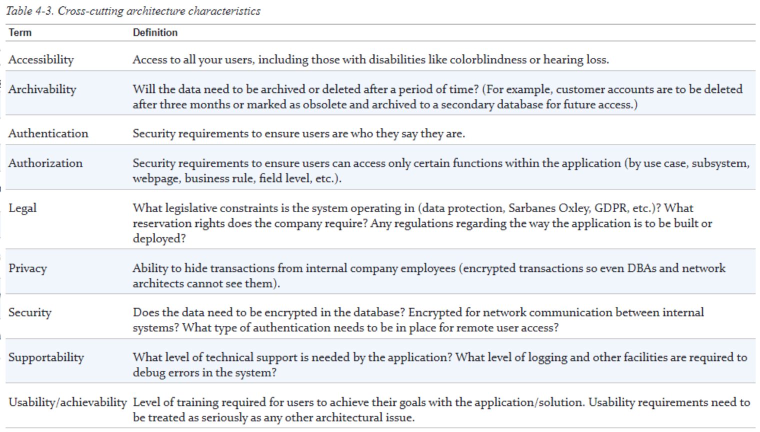


Table 4-3. Cross-cutting architecture characteristics

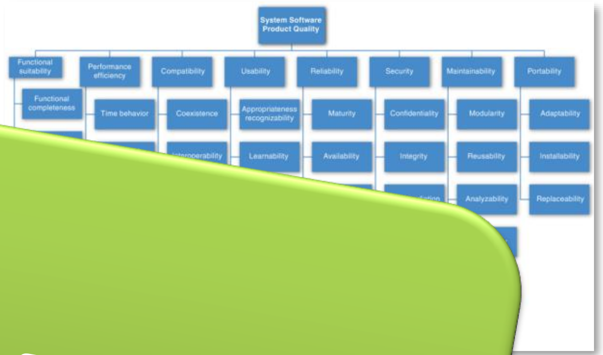
Term	Definition
Accessibility	Access to all your users, including those with disabilities like colorblindness or hearing loss.
Archivability	Will the data need to be archived or deleted after a period of time? (For example, customer accounts are to be deleted after three months or marked as obsolete and archived to a secondary database for future access.)
Authentication	Security requirements to ensure users are who they say they are.
Authorization	Security requirements to ensure users can access only certain functions within the application (by use case, subsystem, webpage, business rule, field level, etc.).
Legal	What legislative constraints is the system operating in (data protection, Sarbanes Oxley, GDPR, etc.)? What reservation rights does the company require? Any regulations regarding the way the application is to be built or deployed?
Privacy	Ability to hide transactions from internal company employees (encrypted transactions so even DBAs and network architects cannot see them).
Security	Does the data need to be encrypted in the database? Encrypted for network communication between internal systems? What type of authentication needs to be in place for remote user access?
Supportability	What level of technical support is needed by the application? What level of logging and other facilities are required to debug errors in the system?
Usability/achievability	Level of training required for users to achieve their goals with the application/solution. Usability requirements need to be treated as seriously as any other architectural issue.

AARHUS UNIVERSITET

Discussion

- There are other frameworks, of course…
  - ISO 25010 / SQuaRE
- Richards
  - *Archit*

Less is More…



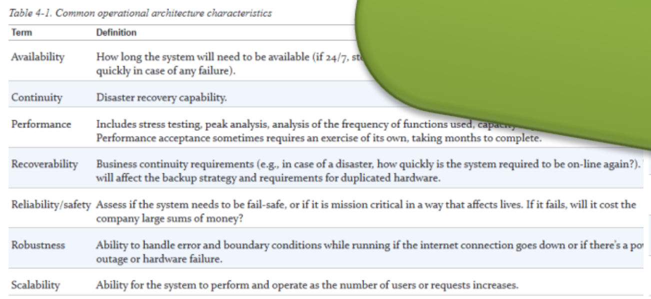
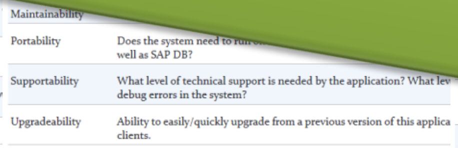
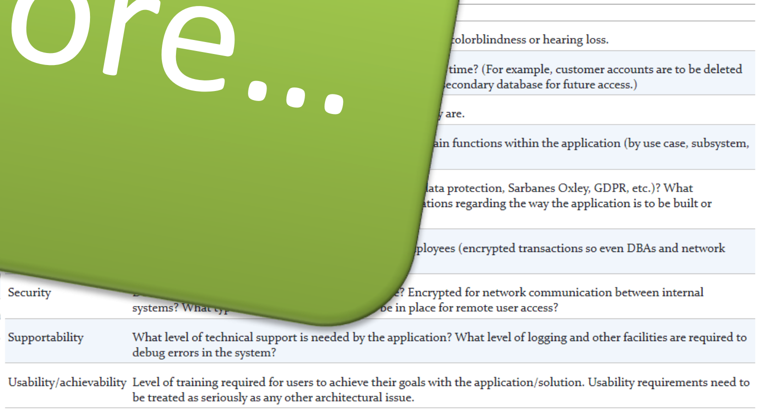


Table 4-1. Common operational architecture characteristics

Term	Definition
Availability	How long the system will need to be available (if 24/7, st quickly in case of any failure).
Continuity	Disaster recovery capability.
Performance	Includes stress testing, peak analysis, analysis of the frequency of functions used, capac Performance acceptance sometimes requires an exercise of its own, taking months to complete.
Recoverability	Business continuity requirements (e.g., in case of a disaster, how quickly is the system required to be on-line again?). will affect the backup strategy and requirements for duplicated hardware.
Reliability/safety	Assess if the system needs to be fail-safe, or if it is mission critical in a way that affects lives. If it fails, will it cost the company large sums of money?
Robustness	Ability to handle error and boundary conditions while running if the internet connection goes down or if there's a po outage or hardware failure.
Scalability	Ability for the system to perform and operate as the number of users or requests increases.



Maintainability	
Portability	Does the system need to run well as SAP DB?
Supportability	What level of technical support is needed by the application? What lev debug errors in the system?
Upgradeability	Ability to easily/quickly upgrade from a previous version of this applica clients.



	colorblindness or hearing loss.
	time? (For example, customer accounts are to be deleted econdary database for future access.)
	are.
	ain functions within the application (by use case, subsystem,
	ata protection, Sarbanes Oxley, GDPR, etc.)? What ations regarding the way the application is to be built or
	ployees (encrypted transactions so even DBAs and network
Security	e? Encrypted for network communication between internal systems? What typ be in place for remote user access?
Supportability	What level of technical support is needed by the application? What level of logging and other facilities are required to debug errors in the system?
Usability/achievability	Level of training required for users to achieve their goals with the application/solution. Usability requirements need to be treated as seriously as any other architectural issue.