AARHUS UNIVERSITET

Software Architecture in Practice

Quality attributes

Good or Bad?

Measurable criterions required...

What does “good” mean?

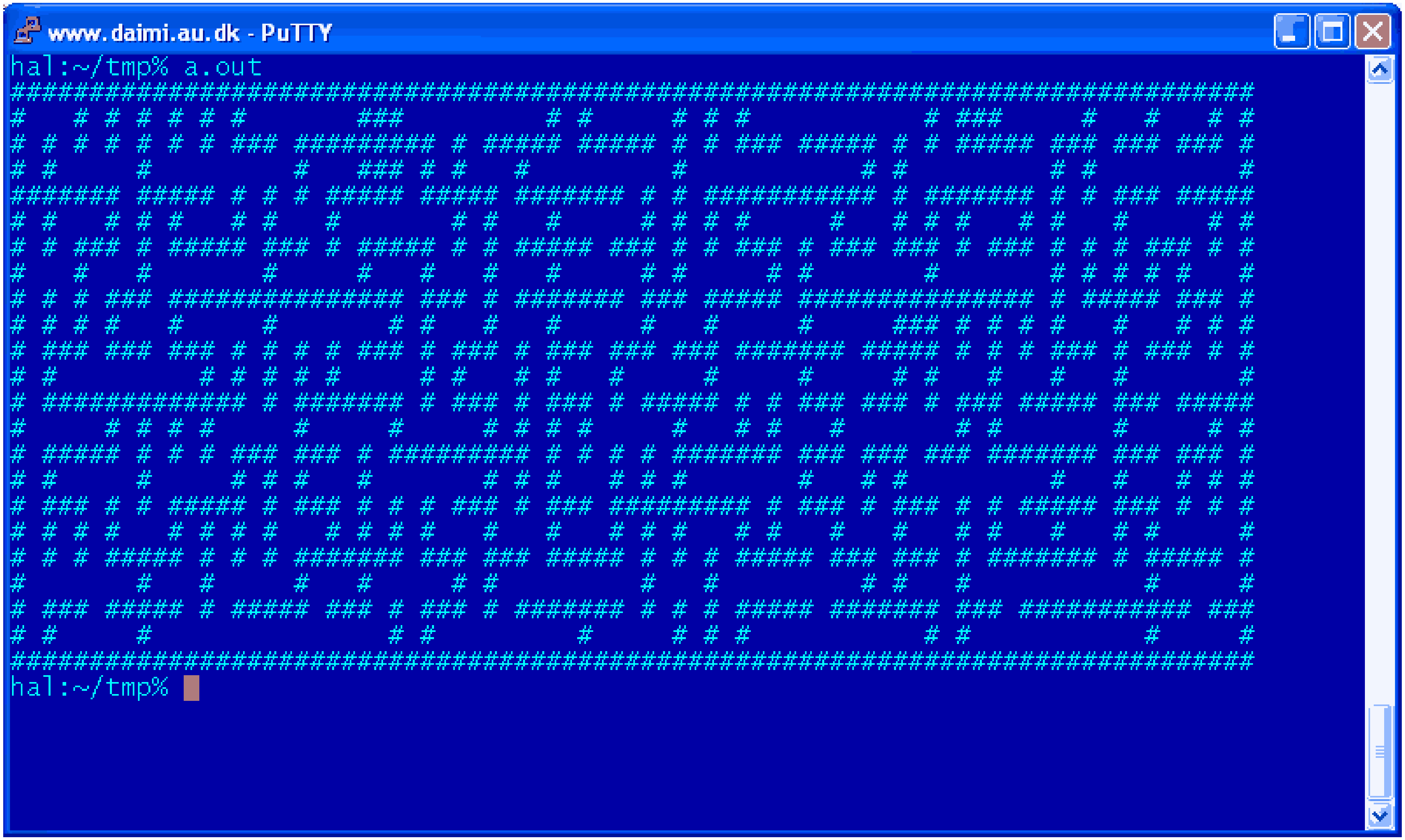
Question: Is this little C program an example of good or bad software?

```
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 did the C program do?)



```
hal:~/tmp% a.out
###############################################################################
#   # # # # # #       ###         # #     # # #           # ###     #   #   # #
# # # # # # # ### ######### # ##### ##### # # ### ##### # # ##### ### ### ### #
# #     #         #   ### # #   #         #           # #         # #         #
####### ##### # # # ##### ##### ####### # # ########### # ####### # # ### #####
# #   # # #   # #   #       # #   #     # # # #     #   # # #   # #   #     # #
# # ### # ##### ### # ##### # # ##### ### # # ### # ### ### # ### # # # ### # #
#   #   #       #     #   #   #   #     # #     # #       #       # # # # #   #
# # # ### ############### ### # ####### ### ##### ############### # ##### ### #
# # # #   #     #       # #   #   #     #   #     #     ### # # # #   #   # # #
# ### ### ### # # # # ### # ### # ### ### ### ####### ##### # # # ### # ### # #
# #         # # # # #     # #   # #   #     #     #     # #   #   #   #       #
# ############# # ####### # ### # ### # ##### # # ### ### # ### ##### ### #####
#     # # # #     #     #     # # # #     #   # #   #       # #       #     # #
# ##### # # # ### ### # ######### # # # # ####### ### ### ### ####### ### ### #
# #     #     # # #   #       # # #   # # #       #   # #         #   #   # # #
# ### # # ##### # ### # # # ### # ### ######### # ### # ### # # ##### ### # # #
# # # #   # # # #   # # # #   #   #   # # #   # #   #   #   # #   #   # #     #
# # # ##### # # # ####### ### ### ##### # # # ##### ### ### # ####### # ##### #
#       #   #     #   #     # #         #   #         # #   #           #     #
# ### ##### # ##### ### # ### # ####### # # # ##### ####### ### ########### ###
# #     #               # #         #     # # #           # #           #     #
###############################################################################
hal:~/tmp% 
```

The need for measuring

The server should be highly available...

***My** software is really reusable!*

Our high performance server will...

These are simply claims ☺

Actual measurements on well defined scale is better...

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**

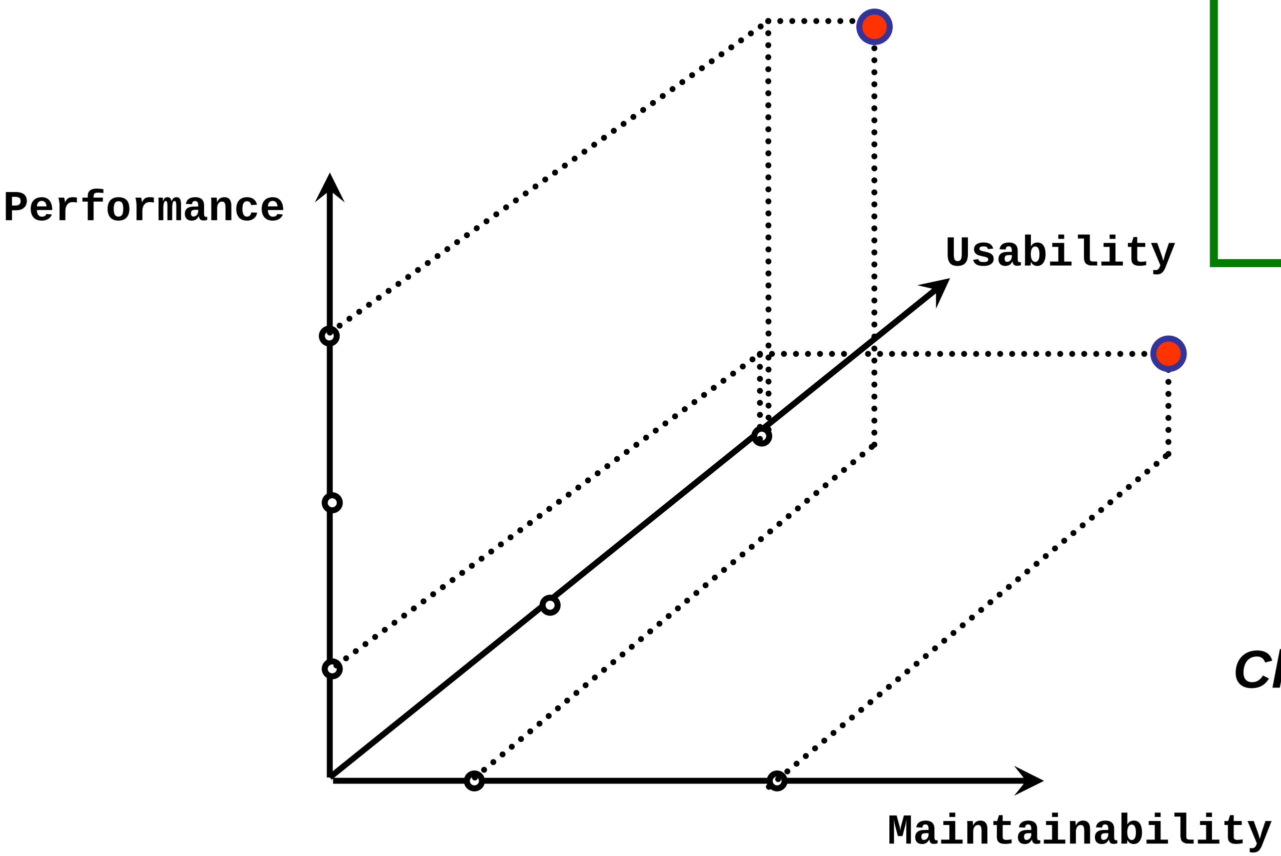
Measuring quality

Quality Framework

Quality Attribute

Metric

Measurement



Choose alternatives

’Quality communities’

One aspects of qualities is that most of them have dedicated research communities associated:

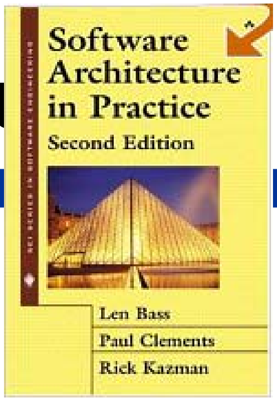
- performance freaks (algorithm people, database, ...)
- usability freaks (HCI – human computer interface)
- security freaks
- cost freaks (managers ☺)
- reusability freaks (pattern community ☺)

...which has lead to lack of common vocabulary…

- user input, attack, event, failures, are all *stimulus*

SAiP has done a great job to provide common ground

SAiP’s contrib



Aim:

- Characterization of architectural quality attributes and how this can be used to express quality requirements for a system

Quality definition:

- None ☺
  - point to historical background in various research communities that has lead to diverse vocabularies ☹.

Evaluation basis:

- Scenario-based –
  - **Quality attribute scenarios** define yardsticks of quality.
  - before the fact – applicable at the “back-of-a-napkin” phase

Quality framework (Bass et al.)

System quality attributes

- Availability
- Modifiability
- Performance
- Security
- Testability
- Usability

Business qualities

- Time to market
- Cost
- Projected lifetime
- Targeted market
- Roll-out schedule
- Integration with legacy sys.

Architectural qualities

- Conceptual integrity
- Correctness and completeness
- Buildability

Qualities in context

What is the context for a given quality?

- – is performance globally important? Modifiability?
- – a QA (usually) only makes sense in a given context:
  - e.g. response-time is important during normal operation but slow initialization may be OK.
  - modifiability is important in components/subsystems that are estimated to be changed often.

Quality Attribute ***Scenarios***

- – ground quality in specific “use situation”

Quality attribute scenarios (QAS)

***Source of stimulus**.* This is some entity (a human, a computer system, or any other actuator) that generated the stimulus.

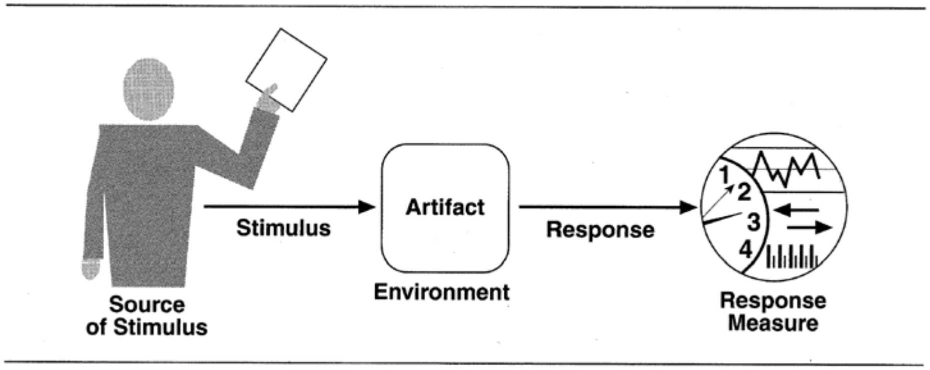
***Stimulus**.* The stimulus is a condition that needs to be considered when it arrives at a system.

***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.

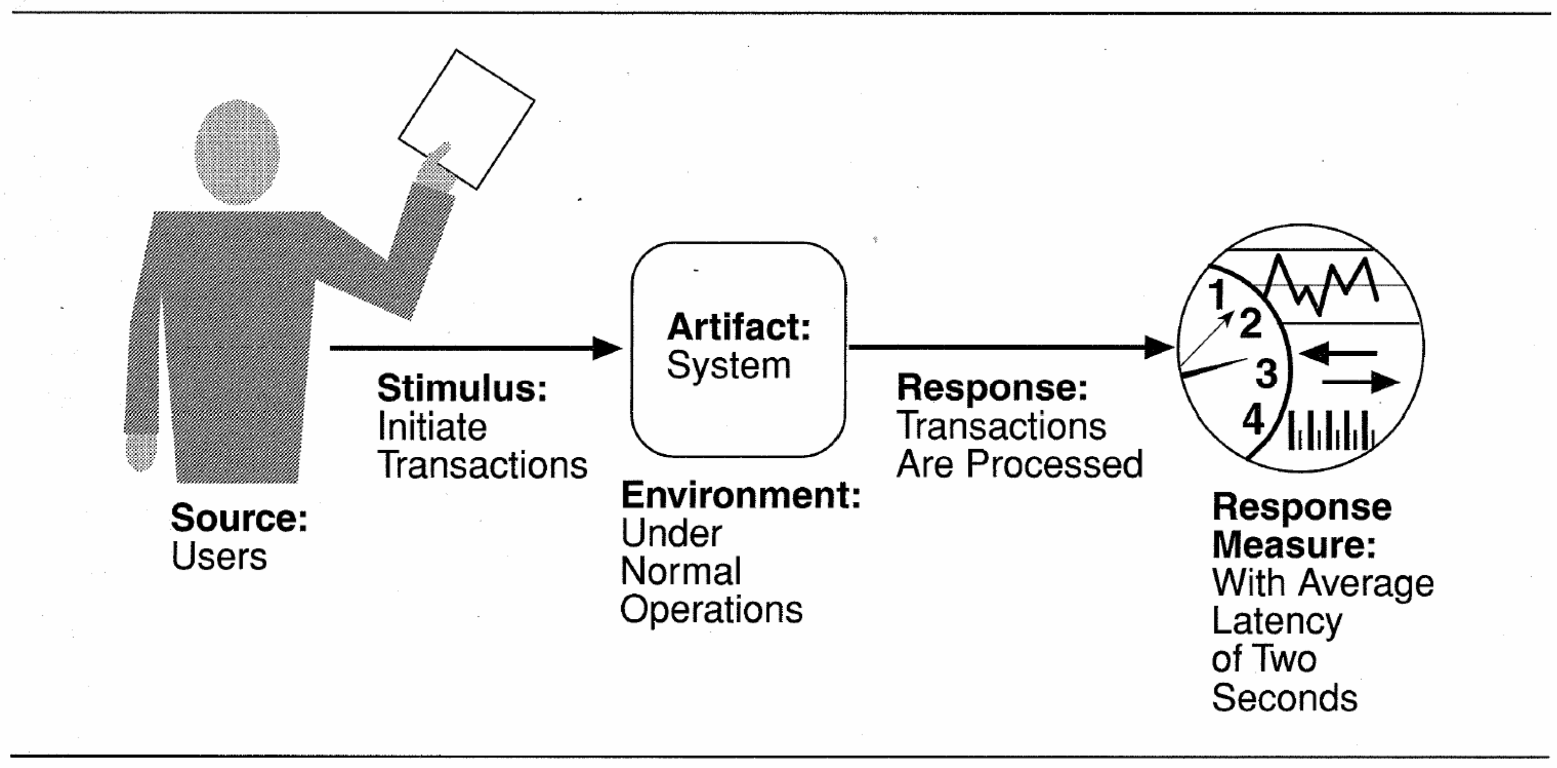
Artifact. Some artifact is stimulated. This may be the whole system or some pieces of it.

***Response**.* The response is the activity undertaken after the arrival of the stimulus.

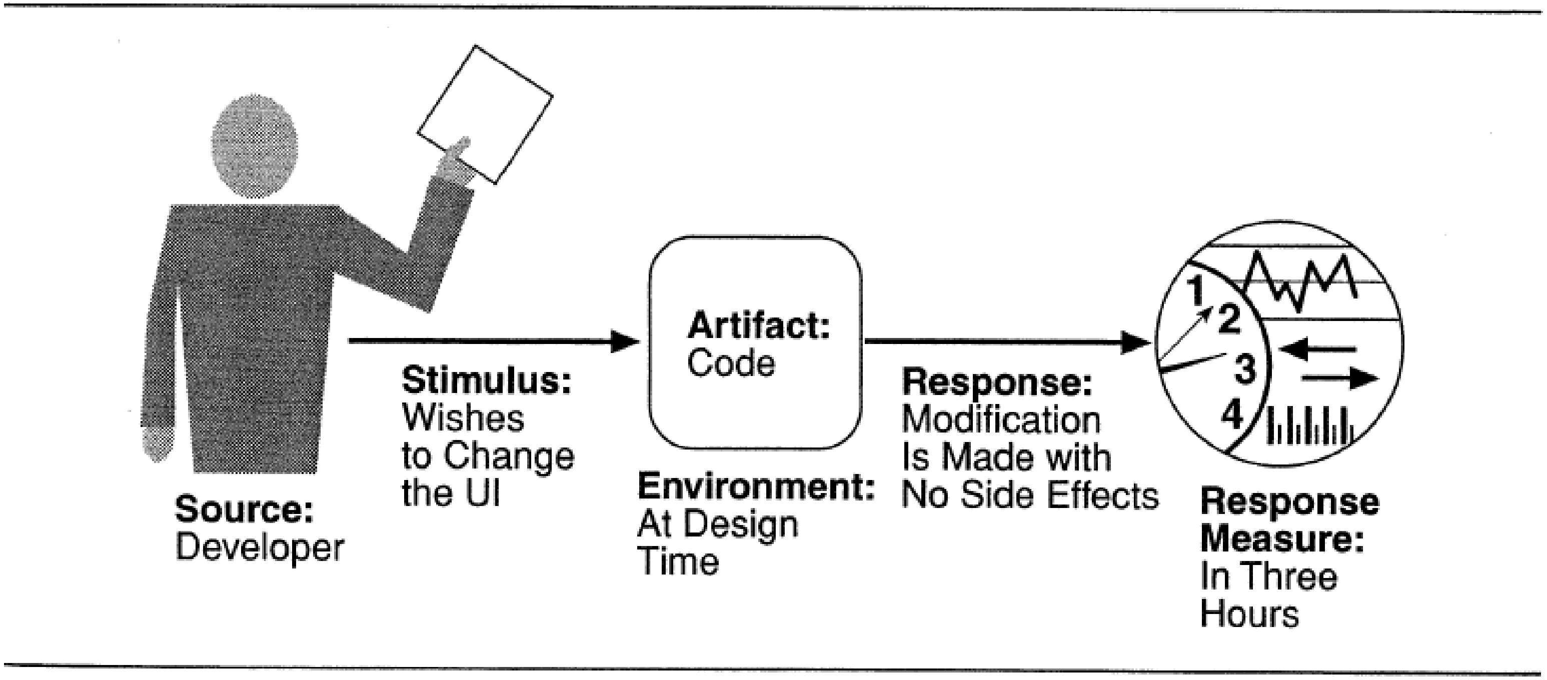
***Response measure**.* When the response occurs, it should be measurable in some fashion so that the requirement can be tested.



Example: Performance



Example: Modifiability



Source: Developer

Stimulus: Wishes to Change the UI

Artifact: Code

Environment: At Design Time

Response: Modification Is Made with No Side Effects

Response Measure: In Three Hours

Benefits

Operational definition

- Given architecture and wanted quality – we can say “yes” or “no” !!!

Scenario aspect

- avoid those “global statements”
  - we want high performance”
    - yes – but which *part* should perform???

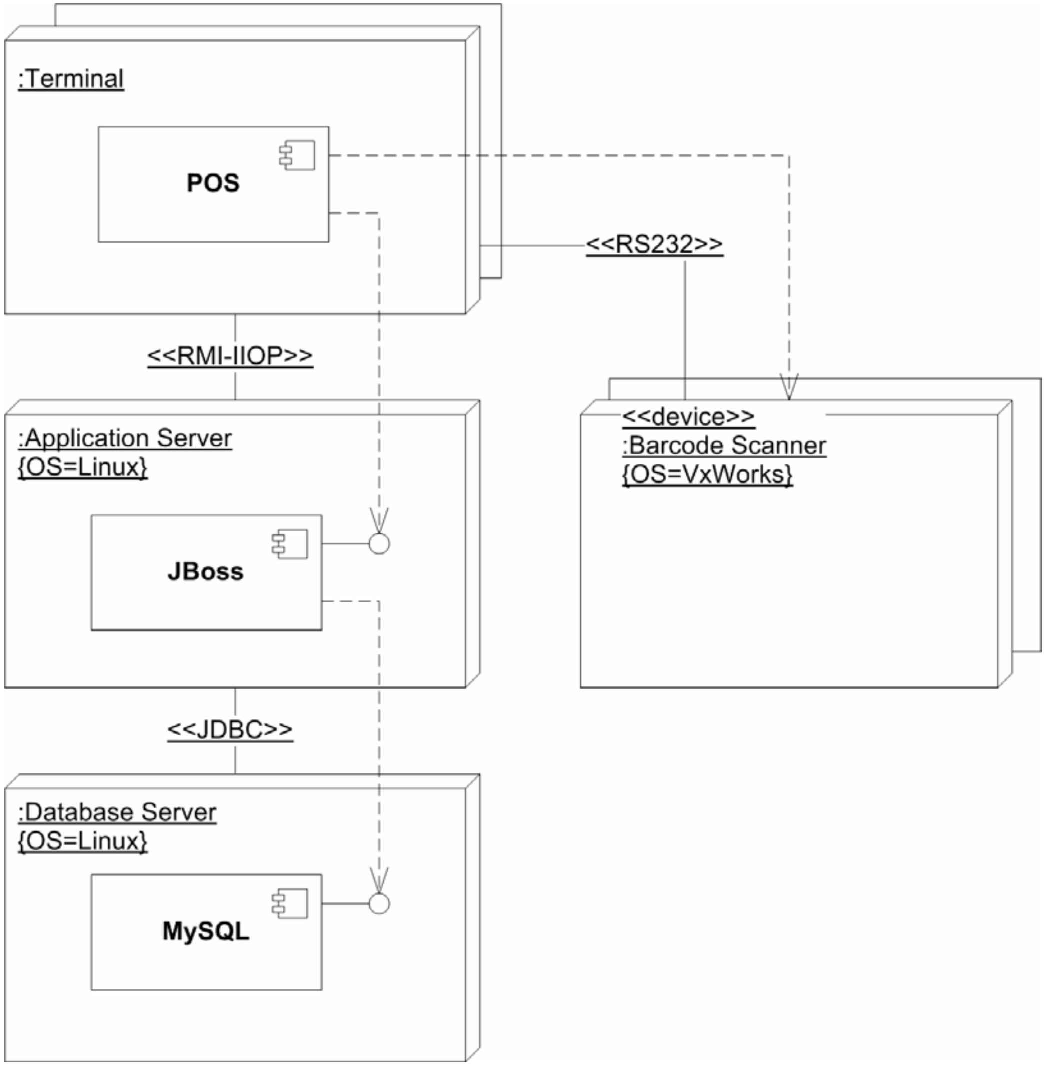
Common vocabulary

- stimulus cover ‘event’, ‘attack’, ...

Examples

Using the Next-POS system

NextGen POS Again



Availability

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

- *The MySQL server crashes during normal operation, the application server detects this and notifies user and maintainer, continues in degraded mode. A backup server is made available in one hour*

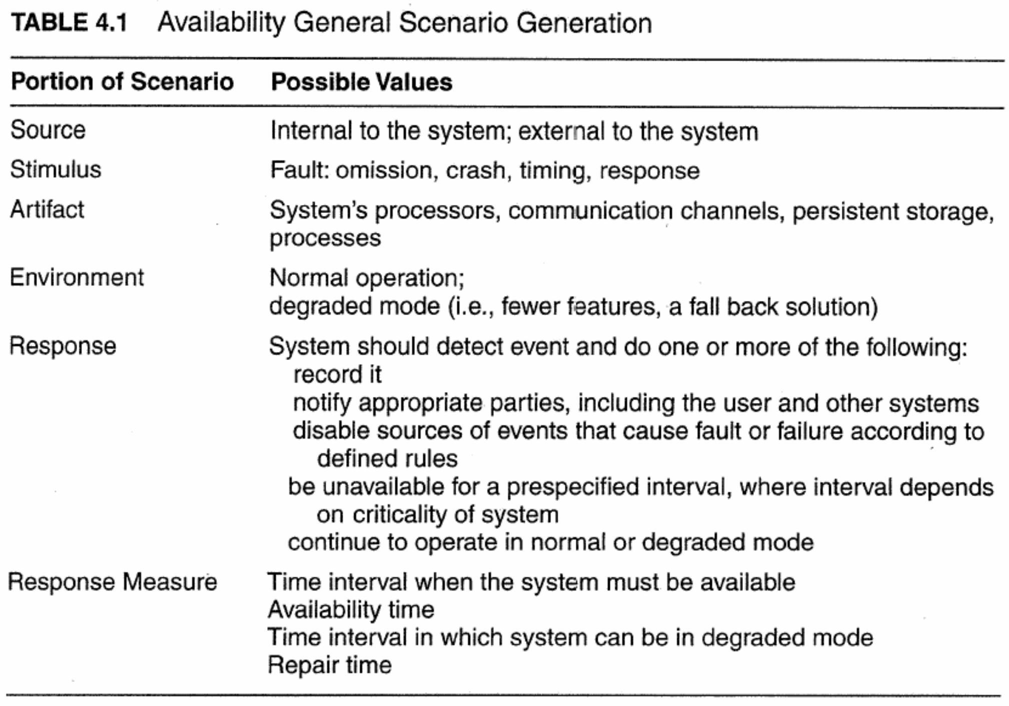


TABLE 4.1 Availability General Scenario Generation

Portion of Scenario	Possible Values
Source	Internal to the system; external to the system
Stimulus	Fault: omission, crash, timing, response
Artifact	System's processors, communication channels, persistent storage, processes
Environment	Normal operation; degraded mode (i.e., fewer features, a fall back solution)
Response	System should detect event and do one or more of the following: record it notify appropriate parties, including the user and other systems disable sources of events that cause fault or failure according to defined rules be unavailable for a prespecified interval, where interval depends on criticality of system continue to operate in normal or degraded mode
Response Measure	Time interval when the system must be available Availability time Time interval in which system can be in degraded mode Repair time

Modifiability

Concerned with the ease with which the system supports change

- *System administrator wants to add a new product category to system while system is operational. The administrator makes the modification with no downtime of the system and in less than 30 minutes*

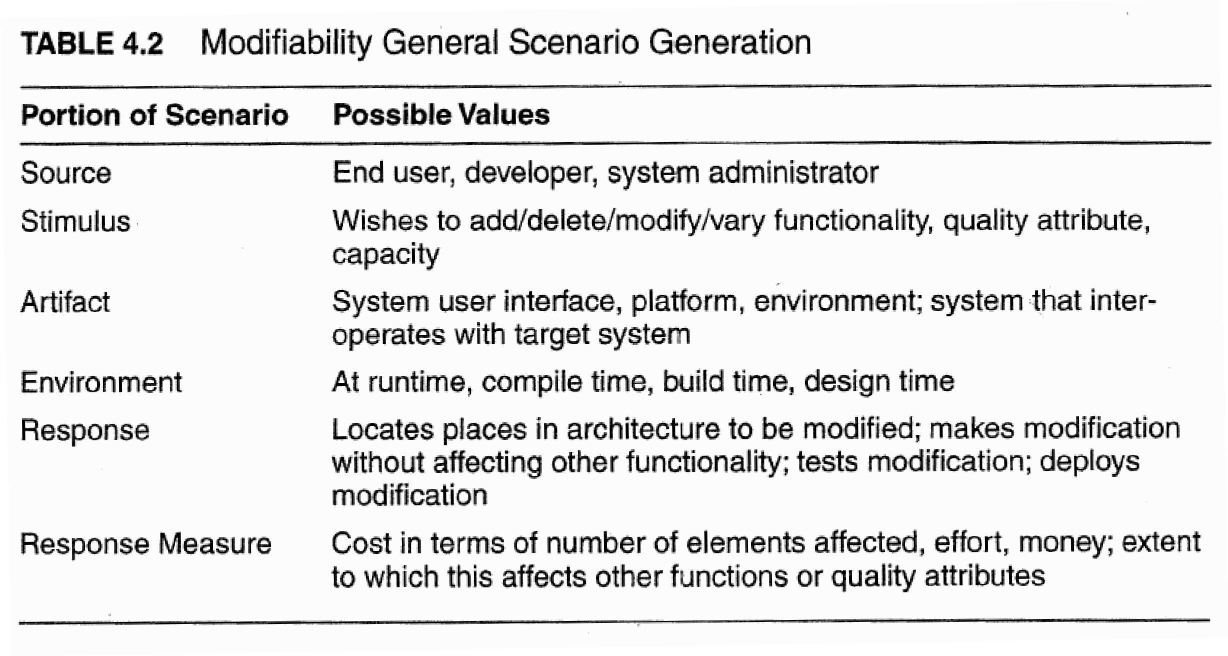


TABLE 4.2 Modifiability General Scenario Generation

Portion of Scenario	Possible Values
Source	End user, developer, system administrator
Stimulus	Wishes to add/delete/modify/vary functionality, quality attribute, capacity
Artifact	System user interface, platform, environment; system that interoperates with target system
Environment	At runtime, compile time, build time, design time
Response	Locates places in architecture to be modified; makes modification without affecting other functionality; tests modification; deploys modification
Response Measure	Cost in terms of number of elements affected, effort, money; extent to which this affects other functions or quality attributes

Performance

Concerned with how long it takes the system to respond when an event occurs

- *Users are initiating 1000 inventory transactions stochastically each minute, all of which are processed with an average latency of two seconds*

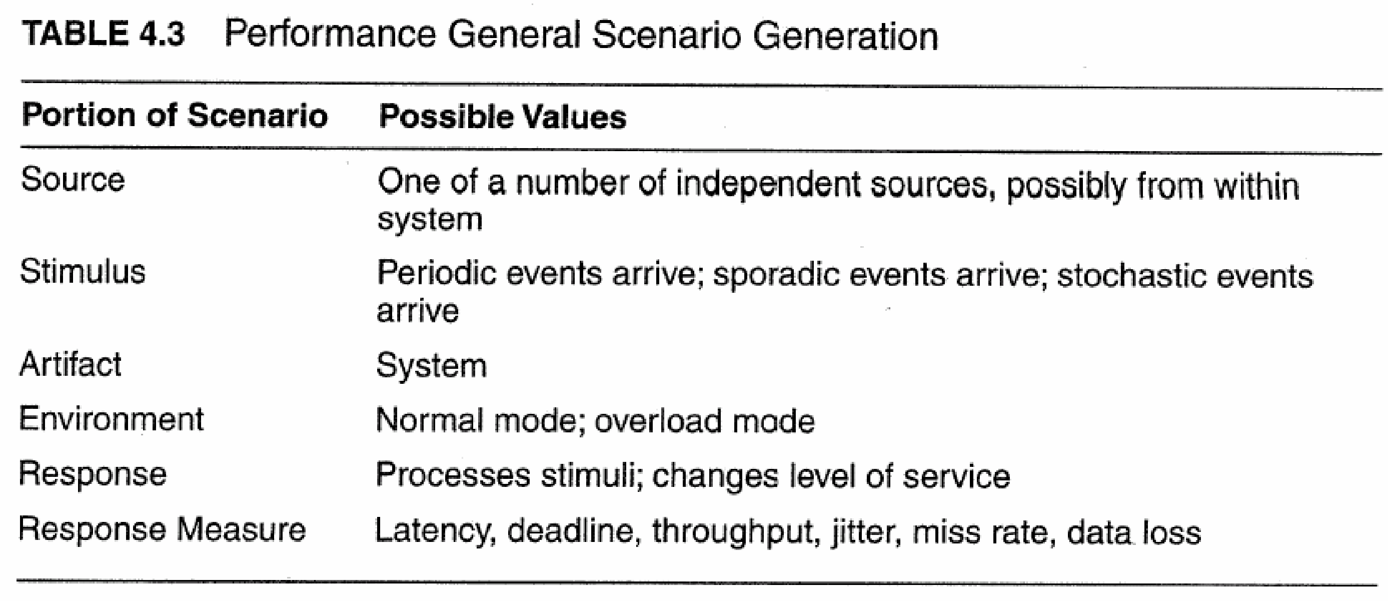


TABLE 4.3 Performance General Scenario Generation

Portion of Scenario	Possible Values
Source	One of a number of independent sources, possibly from within system
Stimulus	Periodic events arrive; sporadic events arrive; stochastic events arrive
Artifact	System
Environment	Normal mode; overload mode
Response	Processes stimuli; changes level of service
Response Measure	Latency, deadline, throughput, jitter, miss rate, data loss

Security

Concerned with the systems ability to withstand attacks/threats

- Nonrepudiation, confidentiality, integrity, assurance, availability, auditing

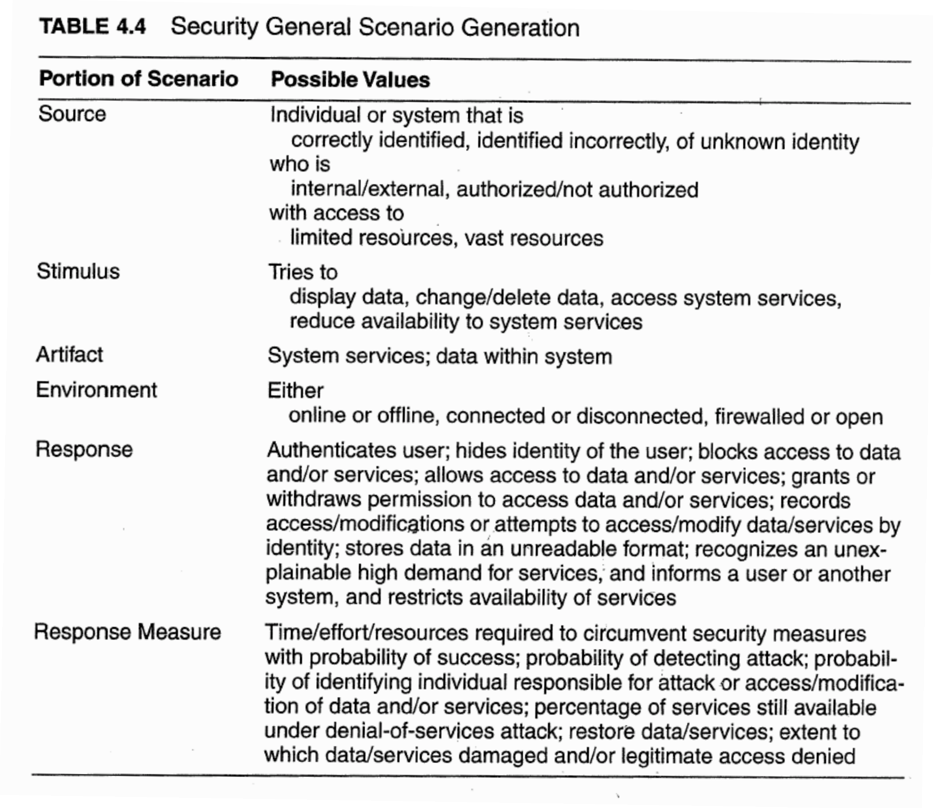


TABLE 4.4 Security General Scenario Generation

Portion of Scenario	Possible Values
Source	Individual or system that is correctly identified, identified incorrectly, of unknown identity who is internal/external, authorized/not authorized with access to limited resources, vast resources
Stimulus	Tries to display data, change/delete data, access system services, reduce availability to system services
Artifact	System services; data within system
Environment	Either online or offline, connected or disconnected, firewalled or open
Response	Authenticates user; hides identity of the user; blocks access to data and/or services; allows access to data and/or services; grants or withdraws permission to access data and/or services; records access/modifications or attempts to access/modify data/services by identity; stores data in an unreadable format; recognizes an unexplainable high demand for services, and informs a user or another system, and restricts availability of services
Response Measure	Time/effort/resources required to circumvent security measures with probability of success; probability of detecting attack; probability of identifying individual responsible for attack or access/modification of data and/or services; percentage of services still available under denial-of-services attack; restore data/services; extent to which data/services damaged and/or legitimate access denied

Testability

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

- *At POS component completion time an increment integrator performs integration test of the component with more then 90% coverage in less than a week*

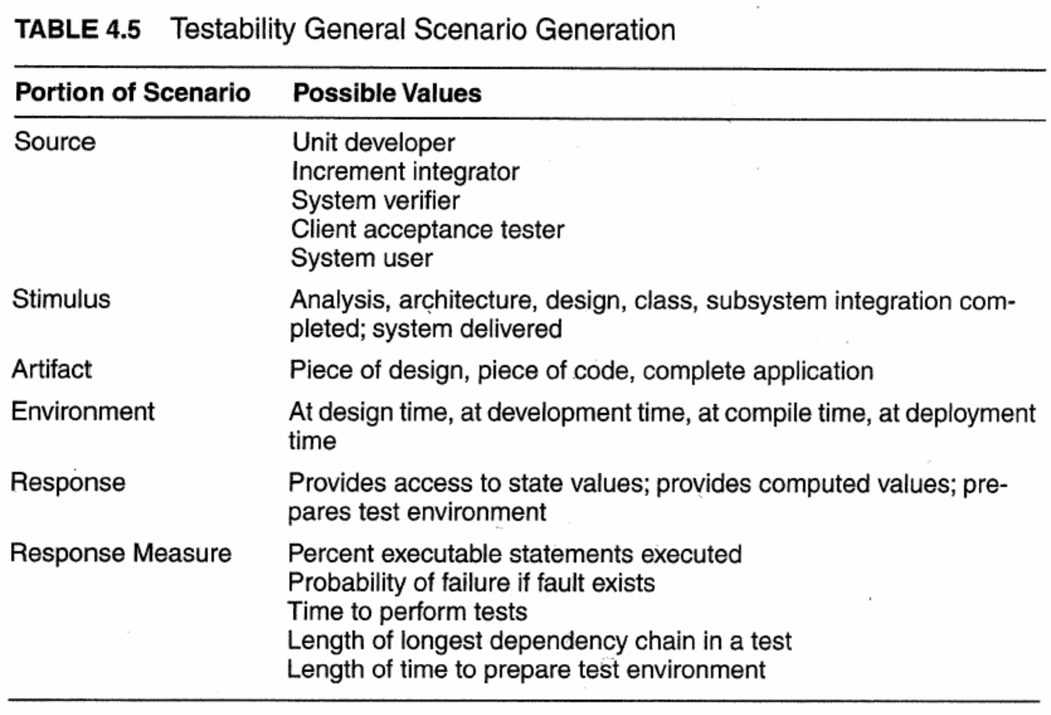


TABLE 4.5 Testability General Scenario Generation

Portion of Scenario	**Possible Values**
Source	Unit developer Increment integrator System verifier Client acceptance tester System user
Stimulus	Analysis, architecture, design, class, subsystem integration completed; system delivered
Artifact	Piece of design, piece of code, complete application
Environment	At design time, at development time, at compile time, at deployment time
Response	Provides access to state values; provides computed values; prepares test environment
Response Measure	Percent executable statements executed Probability of failure if fault exists Time to perform tests Length of longest dependency chain in a test Length of time to prepare test environment

Usability

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

- *An administrator wants to adapt the POS system at configure time in order to change localization; system changes in less than one minute*

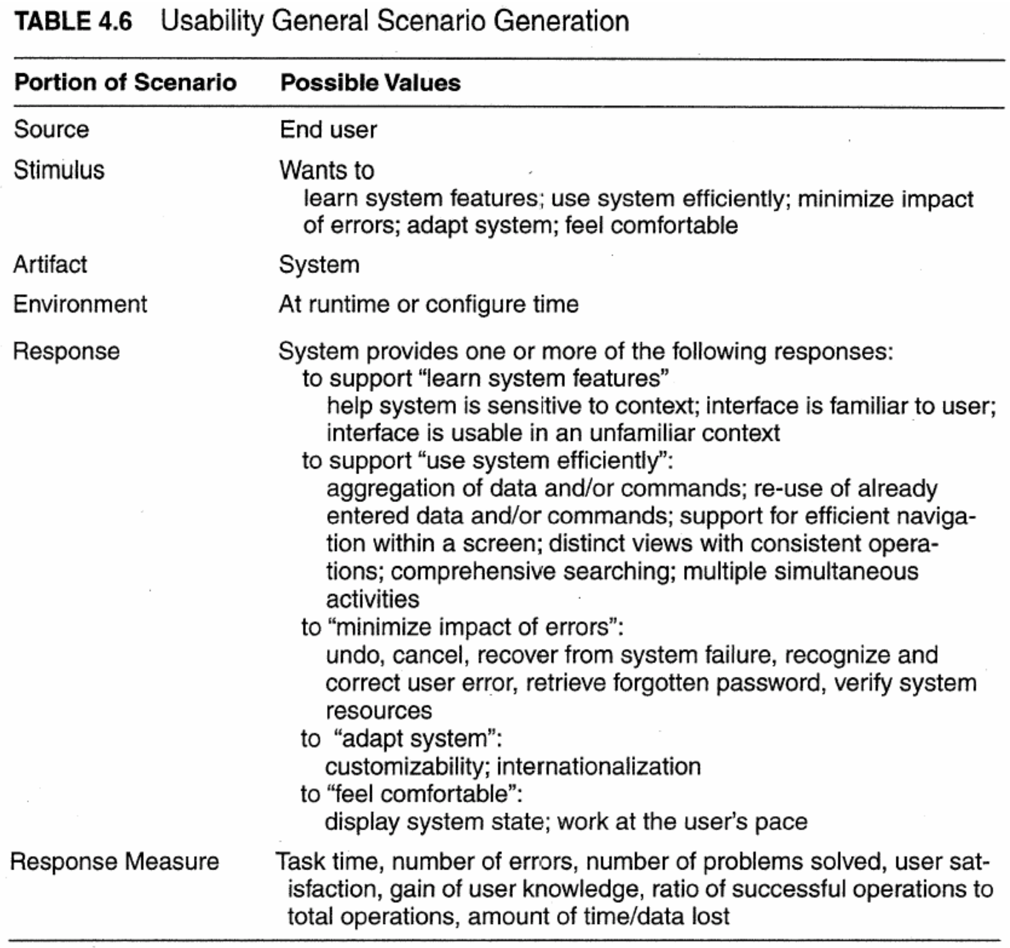


TABLE 4.6 Usability General Scenario Generation

Portion of Scenario	**Possible Values**
Source	End user
Stimulus	Wants to learn system features; use system efficiently; minimize impact of errors; adapt system; feel comfortable
Artifact	System
Environment	At runtime or configure time
Response	System provides one or more of the following responses: to support "learn system features" help system is sensitive to context; interface is familiar to user; interface is usable in an unfamiliar context to support "use system efficiently": aggregation of data and/or commands; re-use of already entered data and/or commands; support for efficient navigation within a screen; distinct views with consistent operations; comprehensive searching; multiple simultaneous activities to "minimize impact of errors": undo, cancel, recover from system failure, recognize and correct user error, retrieve forgotten password, verify system resources to "adapt system": customizability; internationalization to "feel comfortable": display system state; work at the user's pace
Response Measure	Task time, number of errors, number of problems solved, user satisfaction, gain of user knowledge, ratio of successful operations to total operations, amount of time/data lost

Other System Quality Attributes?

Some may be modeled by the six quality attributes

Others may need to be developed as general scenarios

Business Quality Attributes

Business goals which shape the architecture

- Qualities related to the business environment in which the system is being developed

Examples

- Time to market
- Cost/benefit
- Projected lifetime of system
- Targeted market
- Rollout schedule
- Integration with legacy systems

Also described with scenarios…

Architecture Quality Attributes

Quality directly related to the software architecture itself

Examples

- Conceptual integrity
  - Do similar things in similar ways
- Correctness and completeness
  - Does things right and does the right things
- Buildability
  - Ease of constructing a desired system

Scenarios again…

- Most suited to system qualities?

Discussion

Really operational?

- Check given stimulus
  - Whether response occurs
  - If response measure is as indicated?

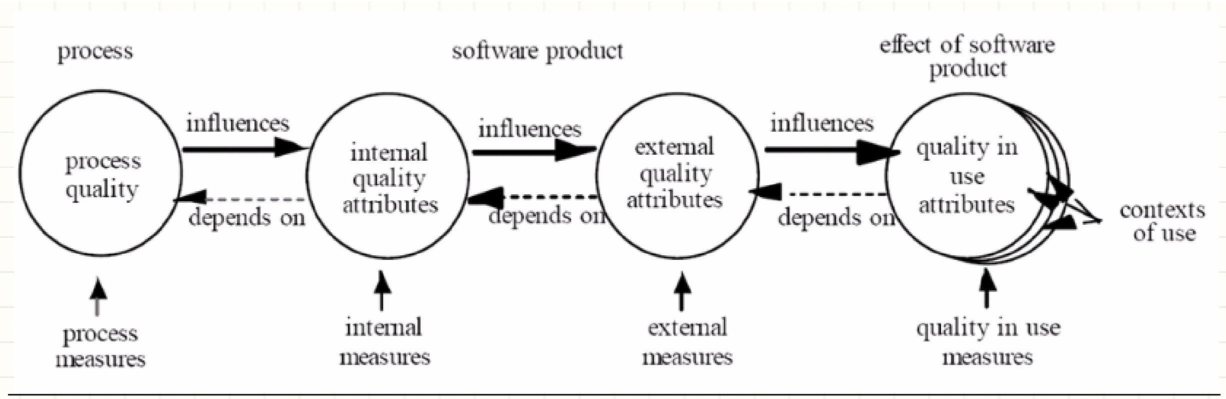
Number of scenarios

- When are you done?

Can all “non-functional” requirements be captured this way?

Another quality framework

Another View: ISO 9126



Internal and external quality attributes divided into characteristics

- Functionality
  - …
- Reliability
  - …
- Usability
  - …
- Efficiency
  - …
- Maintainability
  - Analyzability
  - Changeability
  - Stability
  - Testability
  - Maintainability compliance
- Portability
  - …

Characteristics are measured using metrics

- Internal metrics
  - measure intrinsic properties of software itself
- External metrics
  - measure behavior of computer-based system which includes the software
- In-use metrics
  - measures the effects of using the software in a specific context of use

Example: Changeability, External

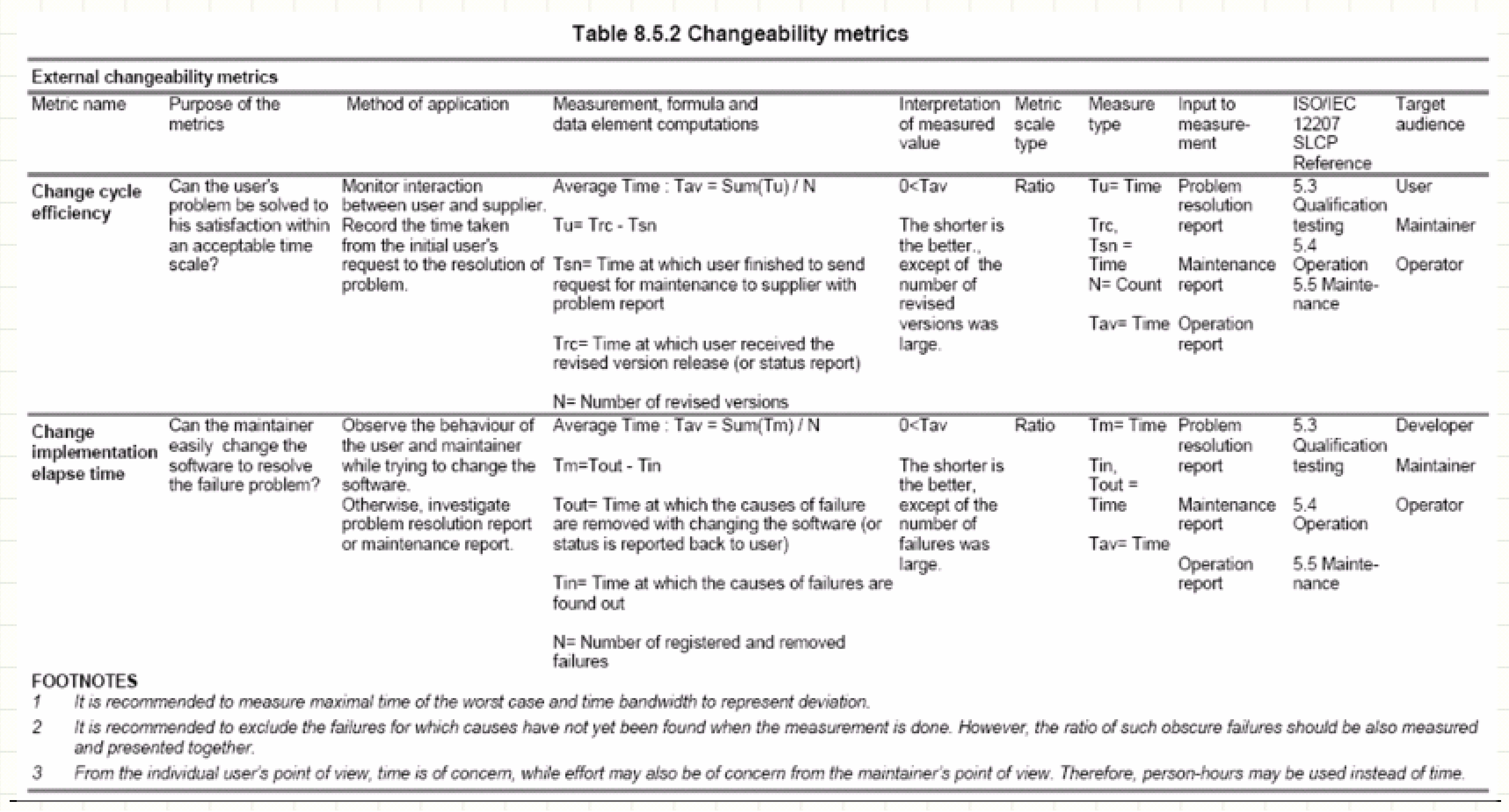


Table 8.5.2 Changeability metrics

External changeability metrics

Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measurement	ISO/IEC 12207 SLCP Reference	Target audience
Change cycle efficiency	Can the user's problem be solved to his satisfaction within an acceptable time scale?	Monitor interaction between user and supplier. Record the time taken from the initial user's request to the resolution of problem.	Average Time : Tav = Sum(Tu) / N Tu= Trc - Tsn Tsn= Time at which user finished to send request for maintenance to supplier with problem report Trc= Time at which user received the revised version release (or status report) N= Number of revised versions	0<Tav The shorter is the better., except of the number of revised versions was large.	Ratio	Tu= Time Trc, Tsn = Time N= Count Tav= Time	Problem resolution report Maintenance report Operation report	5.3 Qualification testing 5.4 Operation 5.5 Maintenance	User Maintainer Operator
Change implementation elapse time	Can the maintainer easily change the software to resolve the failure problem?	Observe the behaviour of the user and maintainer while trying to change the software. Otherwise, investigate problem resolution report or maintenance report.	Average Time : Tav = Sum(Tm) / N Tm=Tout - Tin Tout= Time at which the causes of failure are removed with changing the software (or status is reported back to user) Tin= Time at which the causes of failures are found out N= Number of registered and removed failures	0<Tav The shorter is the better, except of the number of failures was large.	Ratio	Tm= Time Tin, Tout = Time Tav= Time	Problem resolution report Maintenance report Operation report	5.3 Qualification testing 5.4 Operation 5.5 Maintenance	Developer Maintainer Operator

FOOTNOTES

1 *It is recommended to measure maximal time of the worst case and time bandwidth to represent deviation.*

2 *It is recommended to exclude the failures for which causes have not yet been found when the measurement is done. However, the ratio of such obscure failures should be also measured and presented together.*

3 *From the individual user's point of view, time is of concern, while effort may also be of concern from the maintainer's point of view. Therefore, person-hours may be used instead of time.*

Example: Changeability, Internal

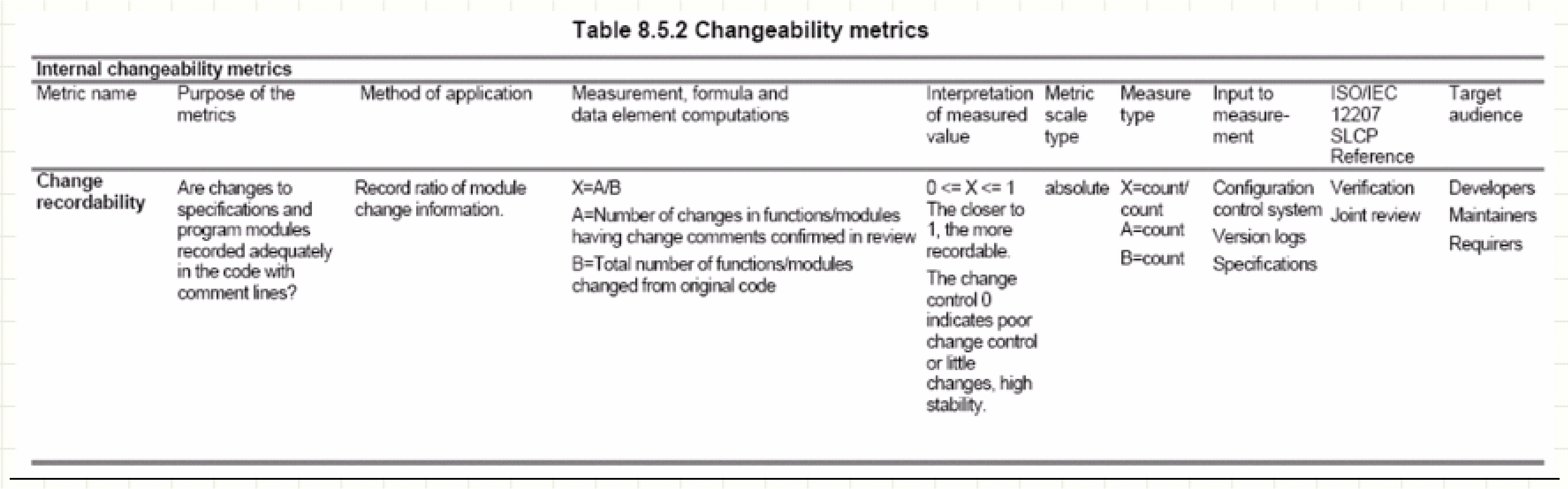


Table 8.5.2 Changeability metrics

Internal changeability metrics									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measurement	ISO/IEC 12207 SLCP Reference	Target audience
Change recordability	Are changes to specifications and program modules recorded adequately in the code with comment lines?	Record ratio of module change information.	X=A/B A=Number of changes in functions/modules having change comments confirmed in review B=Total number of functions/modules changed from original code	0 <= X <= 1 The closer to 1, the more recordable. The change control 0 indicates poor change control or little changes, high stability.	absolute	X=count/ count A=count B=count	Configuration control system Version logs Specifications	Verification Joint review	Developers Maintainers Requirers

“We want them all”

Qualities in conflict

The conflict of qualities

Many qualities are in direct conflict – they must be balanced !

- modifiability and performance
  - many delegations costs in execution speed – and memory footprint
- cost and reusability
  - highly flexible software costs time, effort, and money
- security and availability
  - availability through redundancy – increase opportunities of attack
- etc.

War story

Major EPJ suppliers

Architectural drivers of two vendors

- High modifiability and low time-to-market
  - Change requirements from the Sundhedsstyrelse
  - Physicians want to add new fields to objects

Two radically different architectures

- A: Run-time meta modelling
- B: Agile development process

Design Patterns in Perspective

Examples of Good turning Bad

Iteration

In a distributed system, the clients need to iterate over all order lines in a order object...

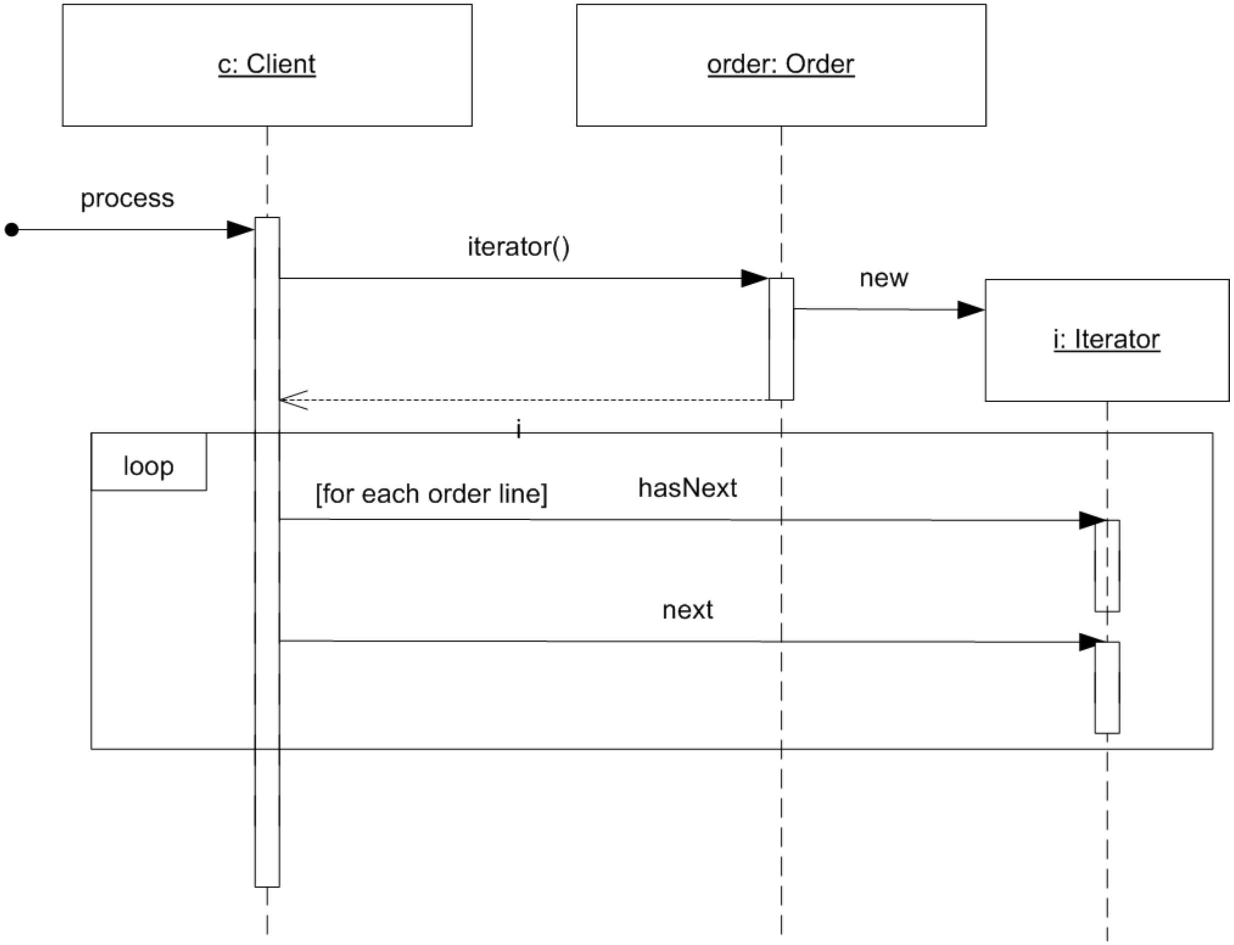
```
Iterator i = order.iterator();
while ( i.hasNext() ) {
  OrderLine entry = (OrderLine) i.next();
  [process entry]
}
```

Using RMI – Remote Method Invocation – the code looks exactly the same even when the Order object is on the server side !!! Great!

Iterator is a nice, flexible, design pattern ☺

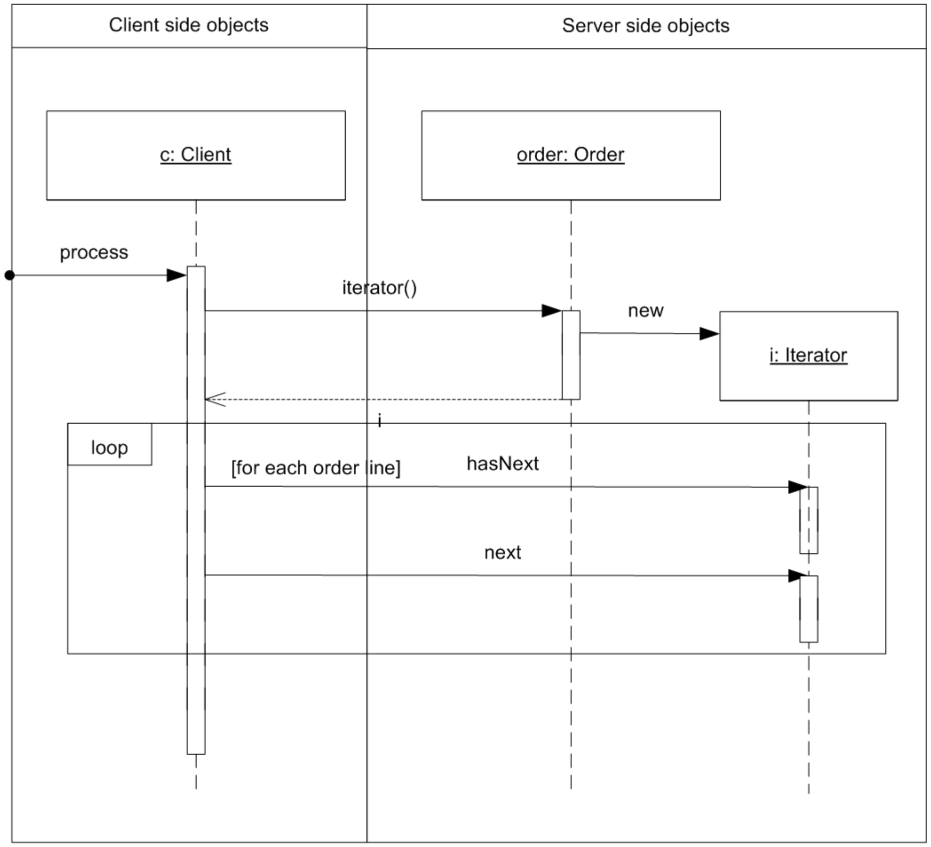
The sequence diagram

All is fine ...



... but

Let us consider the deployment of objects



... what about performance ?

Message-call is much more expense over a network

- 2002 data: 20.000 msg/sec local; 220 msg/sec remote
- 90-times slow-down – probably even worse today.

The iterator pattern ensures an extreme slow-down compared to transfer all order line objects in a single network package !

Another example...

Singleton

- Ensure a class only has one instance, and provide a global point of access to it.

However, a singleton in a distributed system is a major headache!

- one server becomes two servers with a load balancer for scalability
  - now there are two singletons !
- really only one singleton
  - a major performance bottle-neck to ensure the system will never ever scale !

Actually – many consider singleton an anti-pattern!

Discussion

Discussion

”It is the mapping of a system’s functionality onto software structures that determines the architecture’s support for quality attributes”.

That is, the **same** functionality can be made available for the end user using any of a number of **different** architectures !

Functionality and architecture are orthogonal (independent)

Discussion

The list of QA’s is a good checklist to consider when ”architecting” a software system:

- globally (typically modifiability, buildability, cost, security …)
- locally (typically performance, usability, …)

A key insight is that:

- **qualities appear anyway!!!**
- is the project going to control them or is uncoordinated, random, decisions made by individuals going to determine them?

War story

Major Danish software shop

Request for system X

- Considered one-time order...
  - Cost low ⇨ COTS reuse as much as possible

Customer happy ☹ ☺

- Maintaining it in its 15th’s year
- Major COTS vendors out of business
  - In-house reimplementations of old UNIX components

Design versus Architecture

Software architecture is an abstraction of a system

- so… when does architecture stop and design/implementation begin???

Architecture only considers information necessary for decision making and risk management – with respect to the most important quality attributes.