



AARHUS UNIVERSITET

Software Engineering and Architecture

Quality Attributes



Good or Bad?

- When I was taught OO programming and design, there was often statements like
- *"This is not good OO design"*
- Which was more or less equal to the statement
 - *I do not like it...*
- ***Theological statement*** 😞



What does “good” mean?

AARHUS UNIVERSITET

- 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?)

```

csdev@m31: ~/proj/qa-c-program
File Edit Tabs Help
csdev@m31:~/proj/qa-c-program$ ./a.out
#####
#   #   # #   #   #   #   #   #   #   #   #   #   #   #   #
# # ### ### ### # # # ##### ##### ### ### # # ### # ### # ##### # # # # #
# # #   #   # #   #   #   #   #   #   #   #   #   #   #
### ##### ### # ##### # ### ### # ### ##### ### # # # # # # ##### # ###
#   #   #   #   #   #   #   #   #   #   #   #   #   #   #
### ### ### ##### ##### # # # ##### ### # # # # # # ##### ##### ### ### #
#   #   #   #   #   #   #   #   #   #   #   #   #   #   #
##### ### ### ##### # # # ##### ### # ##### # ##### ### #
# #   #   #   #   #   #   #   #   #   #   #   #   #   #
# # # ##### ##### ### ### # ### ##### # # ### # ##### # ### ### #####
#   #   #   #   #   #   #   #   #   #   #   #   #   #   #
# # ### ### # # # ### # ### ##### # # ### ##### ##### # ### ### ### # #
# # #   #   #   #   #   #   #   #   #   #   #   #   #   #
##### ### # ### ##### # # ##### ### ### # ##### ##### ### # ##### #
#   #   #   #   #   #   #   #   #   #   #   #   #   #   #
# ##### # # ##### # ### ### ##### ### # # # ### # ##### ##### ##### # ###
#   #   #   #   #   #   #   #   #   #   #   #   #   #   #
#####
csdev@m31:~/proj/qa-c-program$

```



The need for measuring

AARHUS UNIVERSITET

- *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...

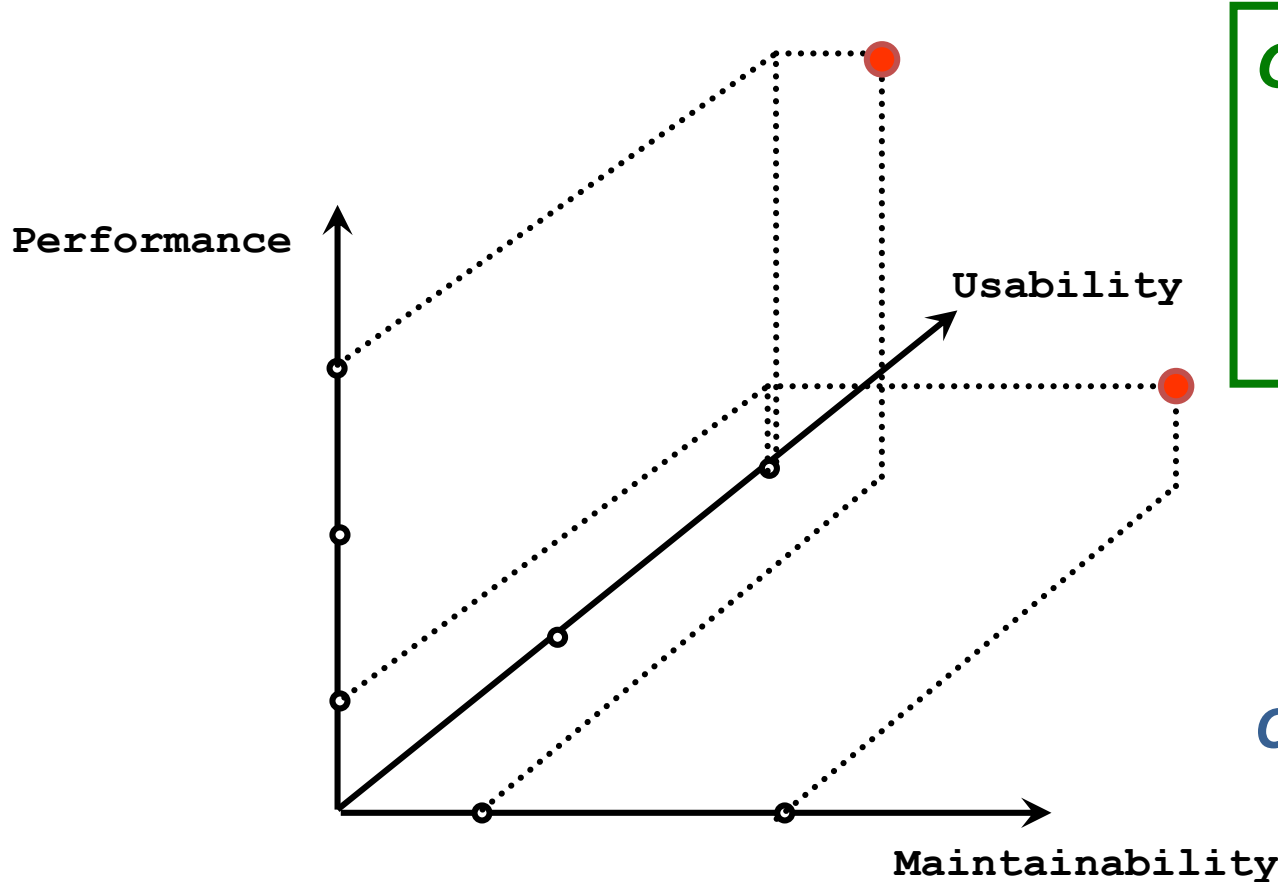


Quality Attributes

- 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**
- Quality attributes (da: kvalitets-attributter)



Measuring quality



Quality Framework

Quality Attribute

Metric

Measurement

Choose alternatives

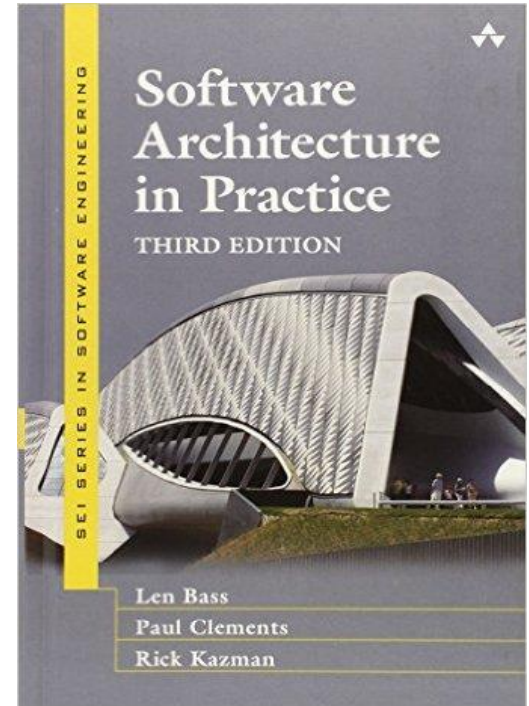


'Quality communities'

- 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*
- *We need to provide common ground*

SAiP's Contribution

- The book *Software Architecture in Practice* has defined a framework that allow different architecturally qualities to be expressed in a similar form: A quality framework
- **Quality Attributes:**
Set of qualities to consider
- **Quality Metric:**
A technique for measuring them



Quality framework (Bass et al.)

- System quality attributes
 - Availability
 - Modifiability
 - Performance
 - Security
 - Testability
 - Usability

 - Interoperability
 - Scalability
 - Deployability
- 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



The System Qualities

- Availability
 - Concerned with the **probability that the system will be operational when needed**
- Modifiability
 - Concerned with the **ease with which the system supports change**
- Performance
 - Concerned with **how long it takes the system to respond** when an event occurs



The System Qualities

- Security
 - Concerned with the systems **ability to withstand attacks/threats**
- Testability
 - Concerned with the **ease with which the software can be made to demonstrate its faults**
- Usability
 - Concerned with **how easy it is for the user to accomplish a desired task** and the kind of user support the system provides

Exercise

- 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

Which of these will have the greatest impact on your professional and personal lives?



AARHUS UNIVERSITET

“We want them all”

Qualities in conflict



The conflict of qualities

AARHUS UNIVERSITET

- 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.



AARHUS UNIVERSITET

Design Patterns in Perspective

Examples of Good turning Bad

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

```
public void processStream() {  
    getOrderFromServer().stream().forEach(this::processOrder);  
}
```

```
public void processJava8() {  
    List<OrderLine> order = getOrderFromServer();  
    for(OrderLine line : order) {  
        processOrder(line);  
    }  
}
```

```
public void processJava1() {  
    List<OrderLine> order = getOrderFromServer();  
    Iterator<OrderLine> iter = order.iterator();  
    while(iter.hasNext()) {  
        OrderLine line = iter.next();  
        processOrder(line);  
    }  
}
```

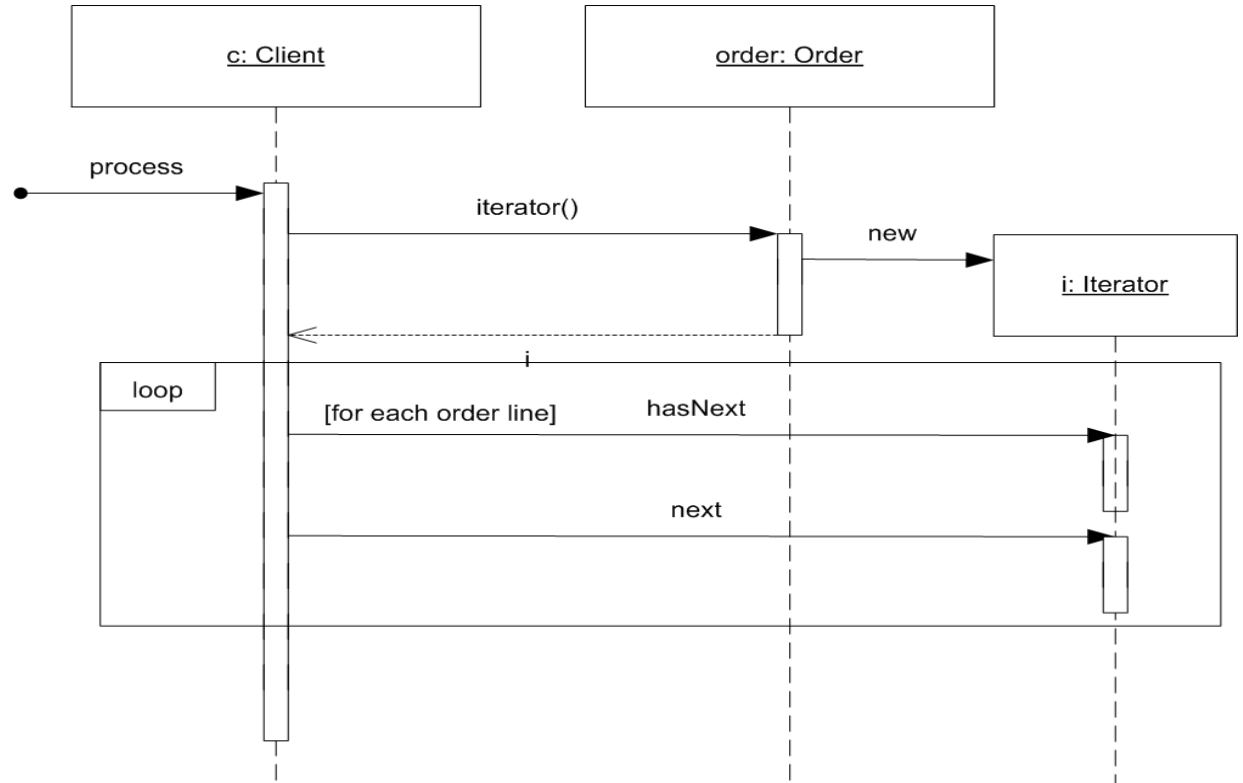
- Using Broker – the code looks exactly the same even when the Order and each OrderLine objects are on the server side !!! Great!
- ***Iterator*** is a nice, flexible, design pattern 😊



The sequence diagram

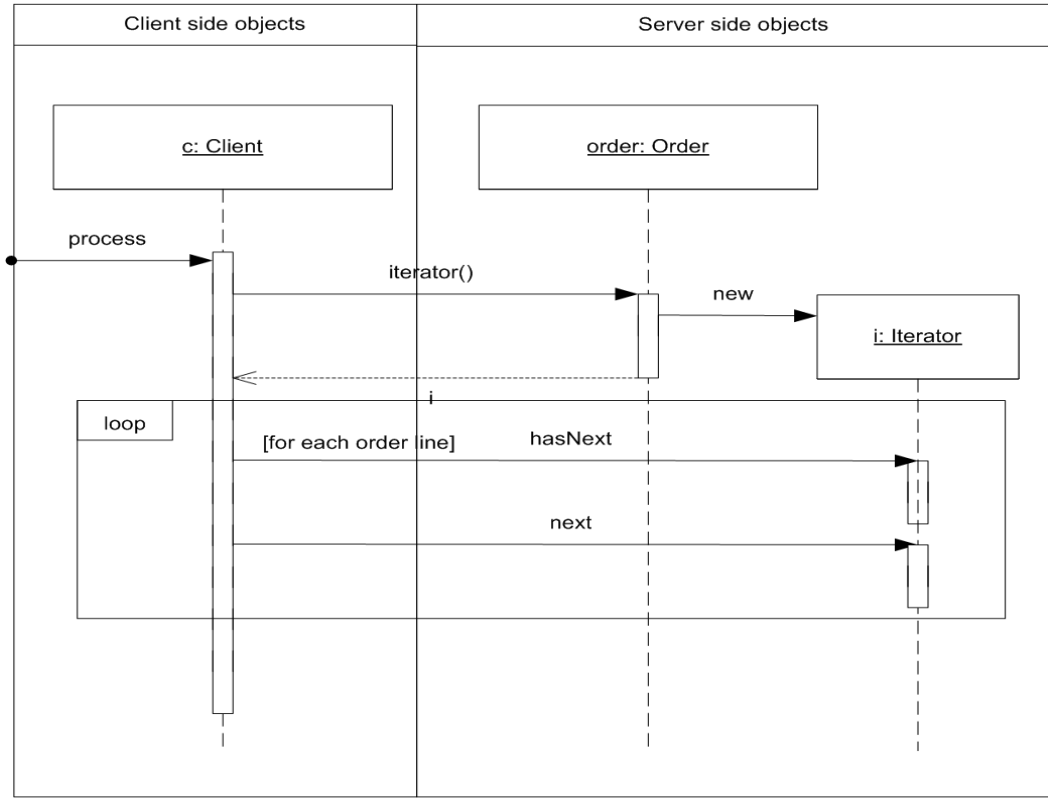
AARHUS UNIVERSITET

- All is fine ...



- Let us consider the **deployment** of objects

What's wrong?





... what about performance ?

- Message-call is much more expensive over a network
 - 2002 data: 20.000 msg/sec local; 220 msg/sec remote
 - 90-times slow-down
 - 2018 data on TeleMed:
 - Between a factor 12 to 275 times slow-down
 - (depends on geography of the two machines)
- The *iterator* pattern produces an extreme slow-down compared to transferring 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!



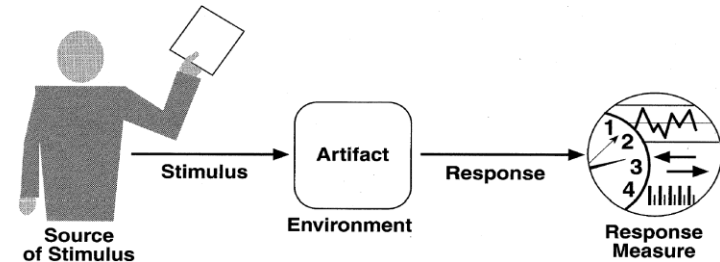
AARHUS UNIVERSITET

Quality Attribute Scenarios

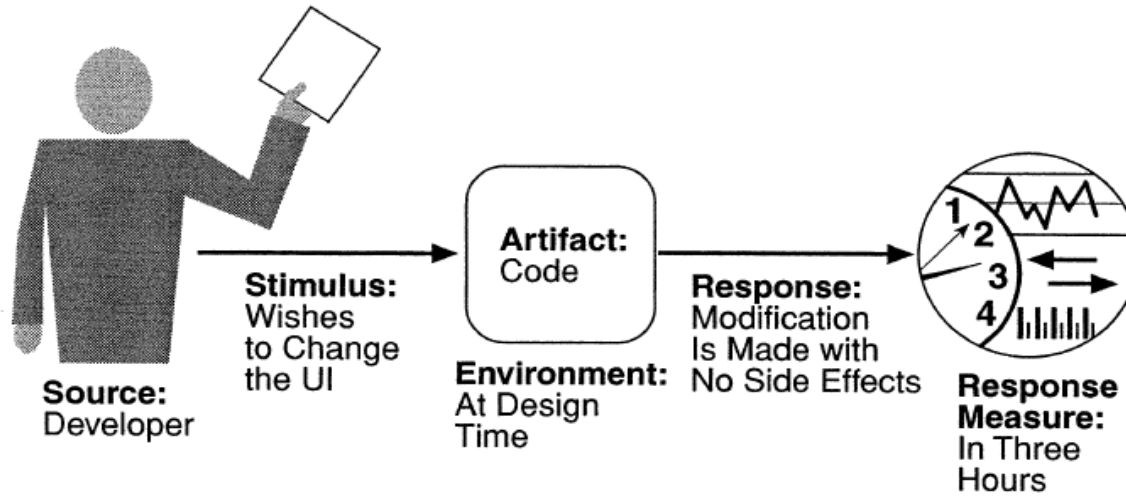
Measuring Modifiability

QaS: A Writing Template

- **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: Modifiability





- The keypoint of the template is
- 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)

- 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: <ul style="list-style-type: none"> ▪ Make modification ▪ Test modification ▪ Deploy modification
Response Measure	Cost in terms of the following: <ul style="list-style-type: none"> ▪ 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

- "In WoW world design phase, it should be easy to change a landscape feature of the world"
- How to formulate it using a QAS
 - *Just how easy, is easy?*

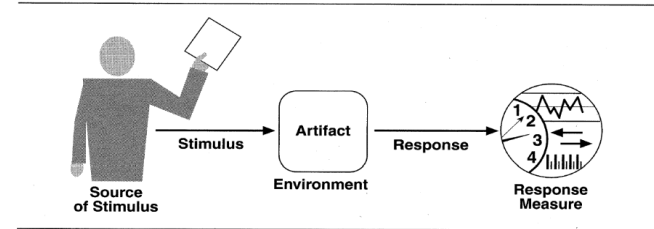


Table 7.1. Modifiability General Scenario

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: <ul style="list-style-type: none"> ▪ Make modification ▪ Test modification ▪ Deploy modification
Response Measure	Cost in terms of the following: <ul style="list-style-type: none"> ▪ 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

- Architectural qualities need to be specified as well as functional ones!
 - It is difficult to make them measurable, yes!
 - *Pretty measurable is much better than not measurable*
 - *The best is the worst enemy of the good...*
- Bass et al.'s QaS is a pretty good tool!
- Modifiability is measured in ***the cost of the change!***
 - Which is basically 'man hours'