AARHUS UNIVERSITET

Microservices and DevOps

Scalable Microservices

Quality Attributes ala Bass et al.

Henrik Bærbak Christensen

AARHUS UNIVERSITET

Motivation

- Why is *software architecture* interesting ???
- It is **not interesting** from a *functionality point of view!*
  - Almost any user requirement can be achieved by any architecture you like, including the ‘one-big-ball-of-mud’ !

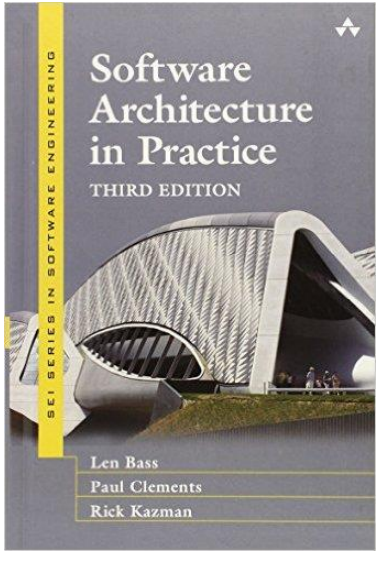
- *Architecture is essential to achieve quality attributes*
  - Performance, maintainability, security, availability, testability, …

- So – a short side-step into my ‘SAiP’ fagpakke…

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

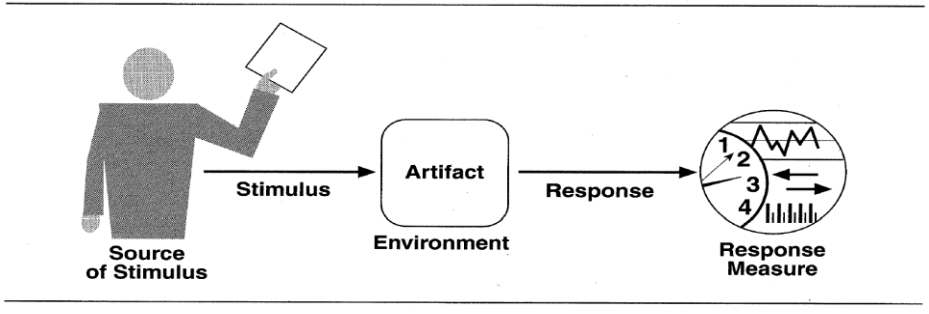


Quality framework (Bass et al.)

- Quality attributes
  - Availability
  - Interoperability
  - Modifiability
  - Performance
  - Security
  - Testability
  - Usability
- Other Quality attributes
  - Variability
  - Portability
  - Dev Distributability
  - Scalability
  - Deployability
  - Mobility
  - Monitorability
  - Safety
- Conceptual Integrity

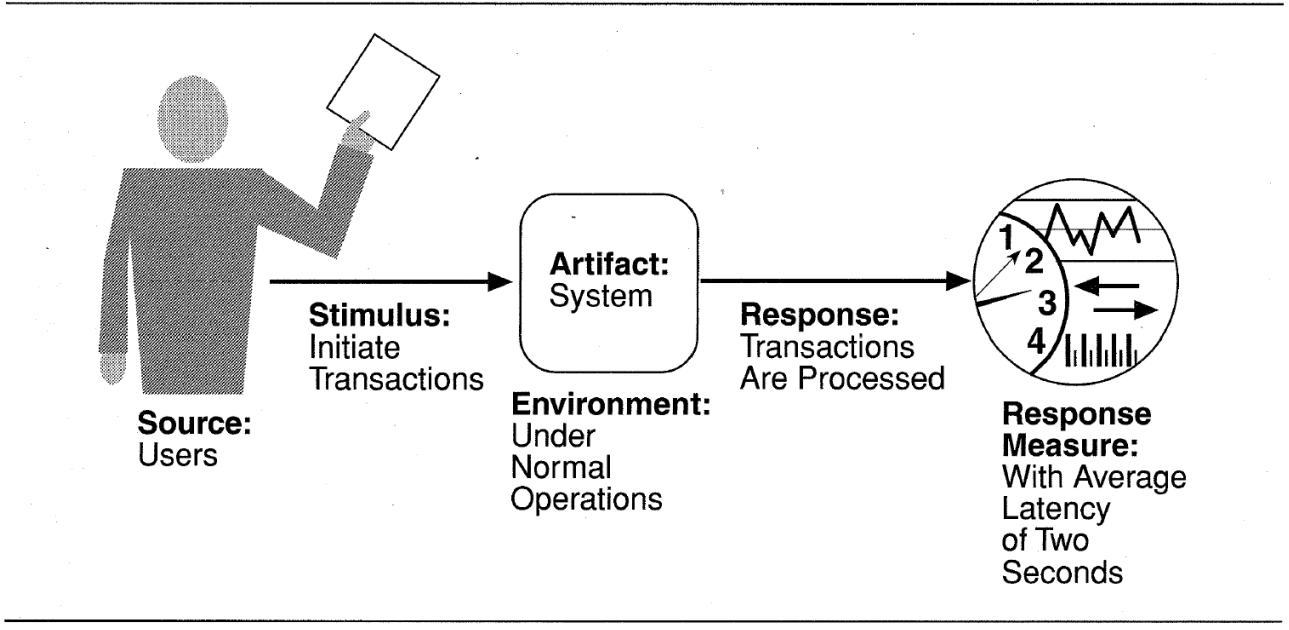
A writing template

- ***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.



AARHUS UNIVERSITET

Example: Performance



Selected QA

The Microservice style has specific QA focus…

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

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

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

Testability

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

Table 10.1. Testability General Scenario

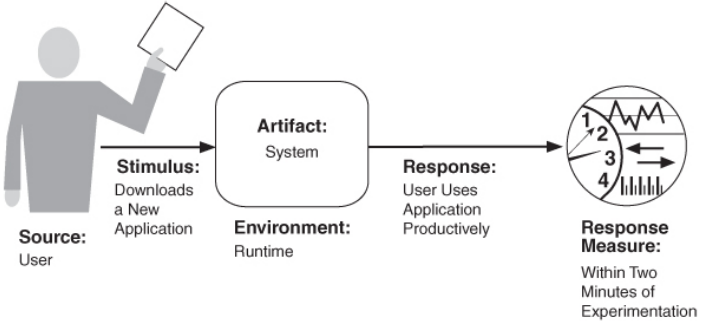
Portion of Scenario	Possible Values
Source	Unit testers, integration testers, system testers, acceptance testers, end users, either running tests manually or using automated testing tools
Stimulus	A set of tests is executed due to the completion of a coding increment such as a class layer or service, 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))

AARHUS UNIVERSITET

SkyCave Examples

QAS

- QAS capture *architectural 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)*



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

AARHUS UNIVERSITET

Example QAS

- *Modifiabilty Quality*
- *Customer* **(source)** wants to *switch to MongoDB* **stimulus)** in the *storage engine* **(artifact)** at *development time* **(environment)**. The *modifications are developed, tested, and delivered* with *no sideeffects* **(response)** within *8 staff hours* **(response measure)**.

- Exercise:
  - Does the SkyCave architecture fulfill this QAS?
  - Why/why not?

AARHUS UNIVERSITET

Example QAS

- *Performance Quality*
- *10.000 players* generate *two stochastic events pr second* on the *daemon* in *normal operations*. All events are *processed* with *95% quartile response time of max 50 ms*.

- Exercise:
  - Does the SkyCave architecture fulfill this QAS?
  - Why/why not?

AARHUS UNIVERSITET

Summary

- The Microservice Architectural style is in my opinion a style that strives for
  - Modifiability (low coupling, high cohesion)
  - Availability (design for failure)
  - Performance (horizontal and independent scaling)
  - Testability (continuous deployment, pipelines)
- Which – are concisely formulated using QAS
  - Because it emphasize the ***response measure***, meaning we talk in terms of numbers, much more than just ‘better/faster/more’…
- *Not required to use in this course, but… ☺*