Software Architectures and Quality

Tactics and Quality Attributes II: Tactics

Javier González Huerta

javier.gonzalez.huerta@bth.se

BLEKINGE TEKNISKA HÖGSKOLA · BTH ·

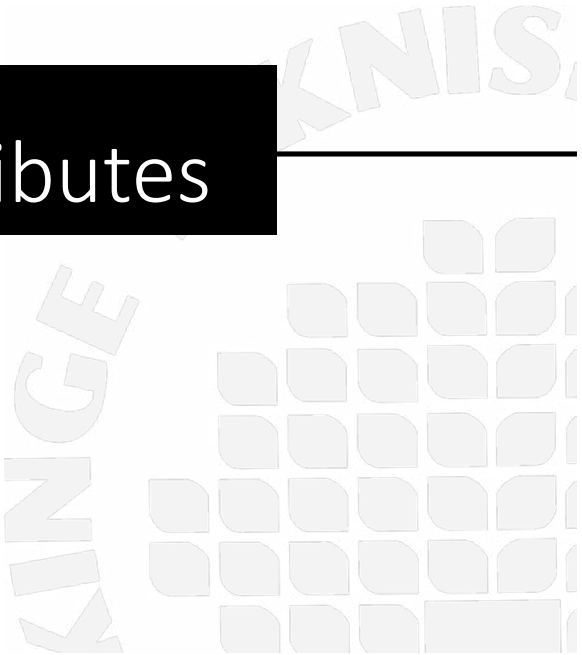
in real life

Goals

- Introduce the specification of quality scenarios
- Discuss how to specify non-functional requirements with general scenarios
- Analyze general scenarios for the most common quality attributes
- Understand the tactics as architects we can introduce to achieve a given Quality Attribute
- Analyze the tactics for the most common quality attributes

Run-Time & Development-time Quality Attributes

Part II: Tactics for Runtime Qualities



Runtime Qualities

Availability

- Whether the software is there and ready to carry out the tasks

Standard ISO/IEC25000 Definition: *"The degree to which a software component is operational and available when required for use."*

- Ability of the system to mask, handle or repair faults so that the cumulative outage time does not exceed a value in a period of time
- Usually is measured as:

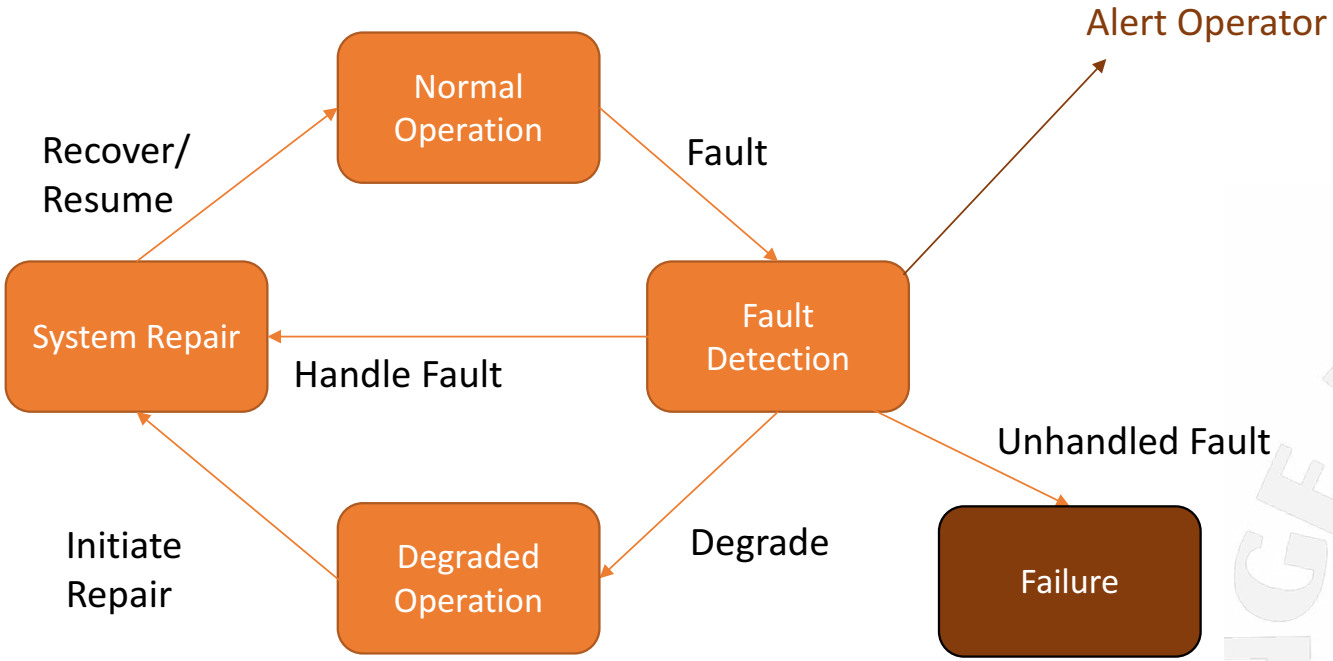
$$Availability = \frac{MTBF}{MTBF+MTTR}$$

- MTBF = Mean Time Between Failures
- MTTR = Mean Time To Repair

Fault vs Failure

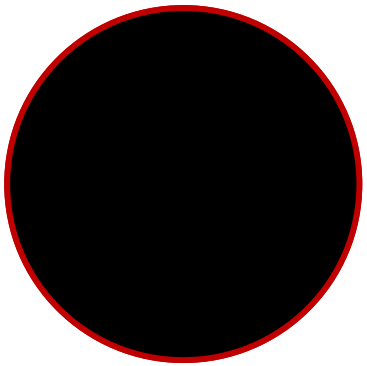
- Fault
  - Internal or external deviation from correct function
  - Cause of a failure
- Failure
  - When the system cannot provide its intended service
  - Caused by faults that cannot be handled / repaired

Availability Operation States

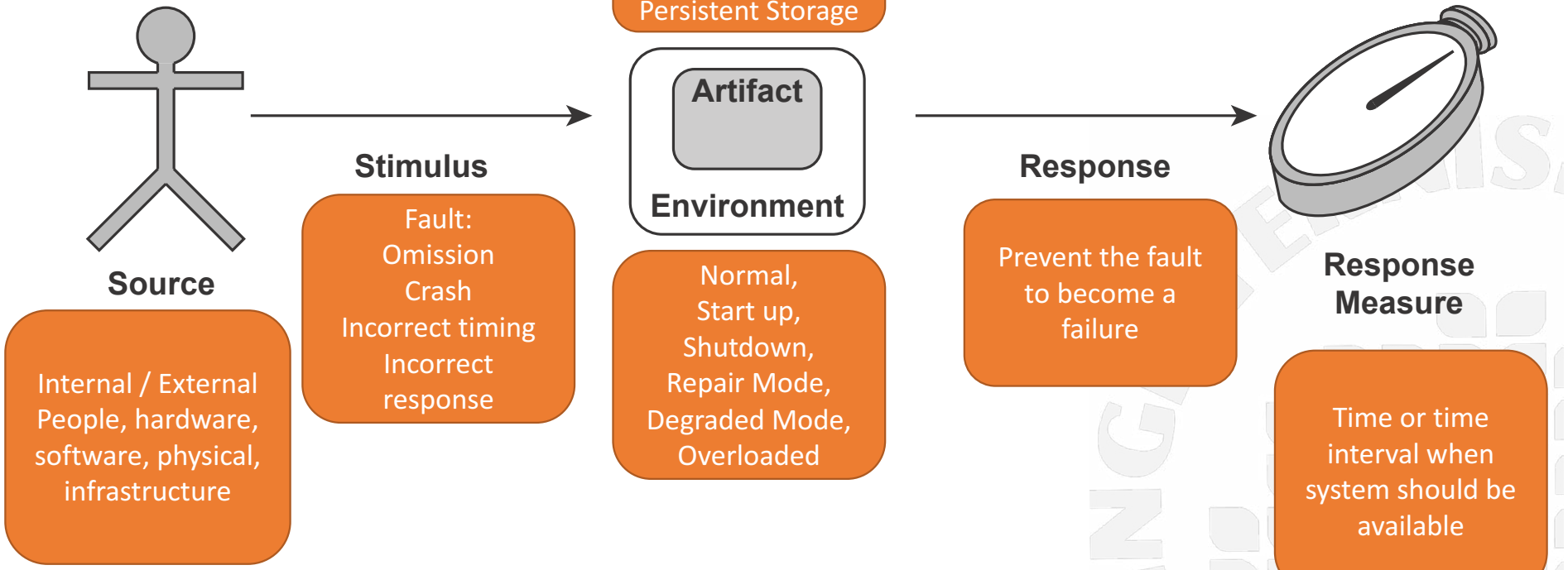


Reflection

- **Discuss during 2 minutes in groups of 2-3 persons how the availability general scenario will look like**



Availability General Scenario

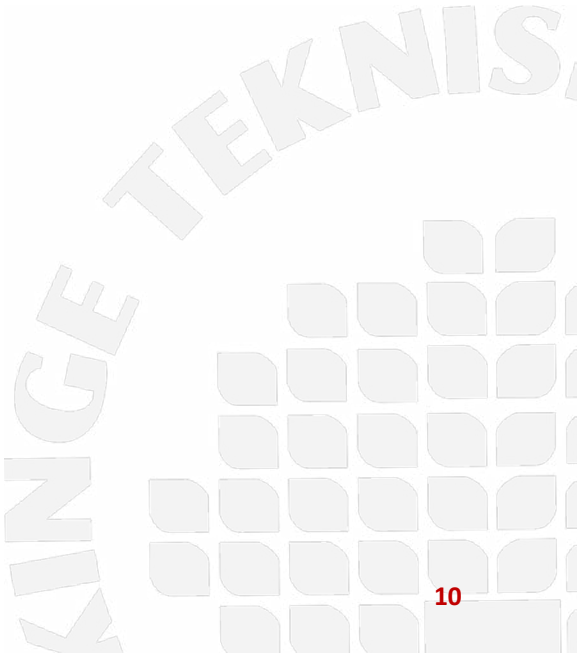


The nines rule

Availability	Downtime in 90 days	Downtime in a year
99.0%	21 h 36 min	3 d 15.6 h
99.9%	2 h 10 min	8 h 0 min 46 s
99.99%	12 min 58 s	52 min 34 s
99.999%	1 min 18 s	5 min 15 s
99.9999%	8s	32 s

Tactics for Availability

- Detect Faults
- Recover from Faults
- Prevent Faults

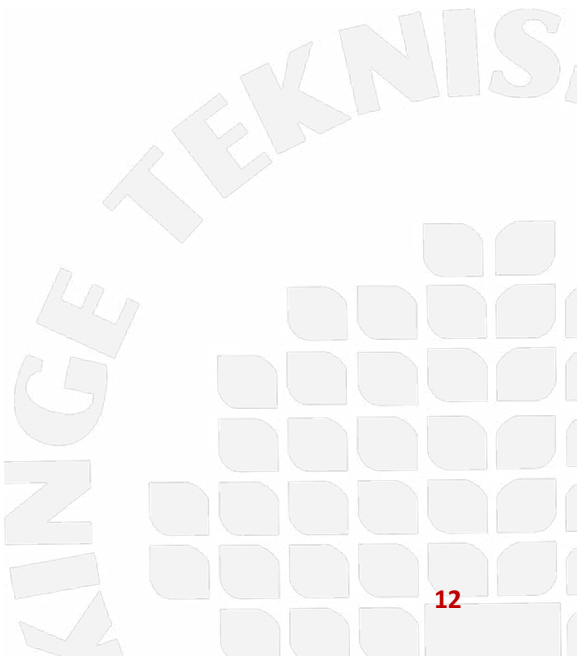


Availability Tactics: Detect Faults

- Ping/Echo
- Heartbeat (Watchdog)
- Monitor
- Timestamp
- Sanity Checking
- Condition Monitoring (e.g., checksum)
- Voting (TMR can also detect and report the fault)
- Exception detection
- Self test (invoked by the same component or by a monitor)

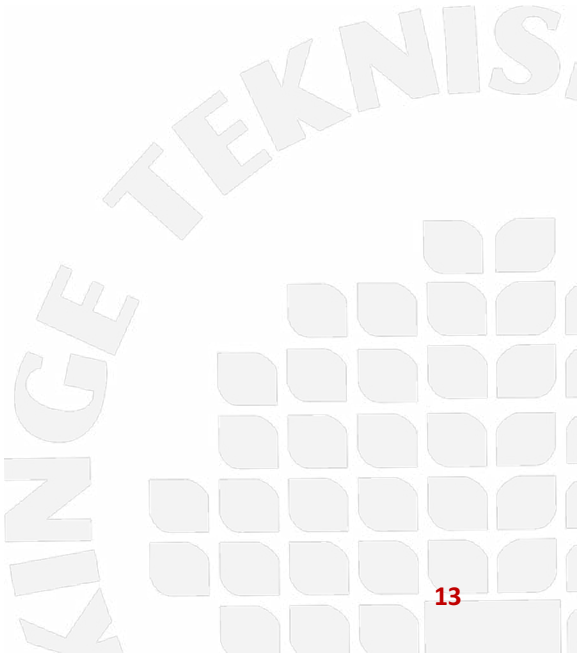
Availability Tactics: Recover from Faults

- Preparation and Repair
  - Active Redundancy
  - Passive Redundancy
  - (Cold) Spare
  - Exception handling
  - Rollback
  - Software upgrade
  - Retry
  - Ignore Faulty Behavior
  - Degradation
  - Reconfiguration
- Reintroduction
  - Shadow
  - State resynchronization
  - Escalating restart
  - Non-stop forwarding



Availability Tactics: Prevent Faults

- Removal from Service
- Transactions
- Predictive Model
- Exception Preventing
- Increase Competence



Runtime Qualities

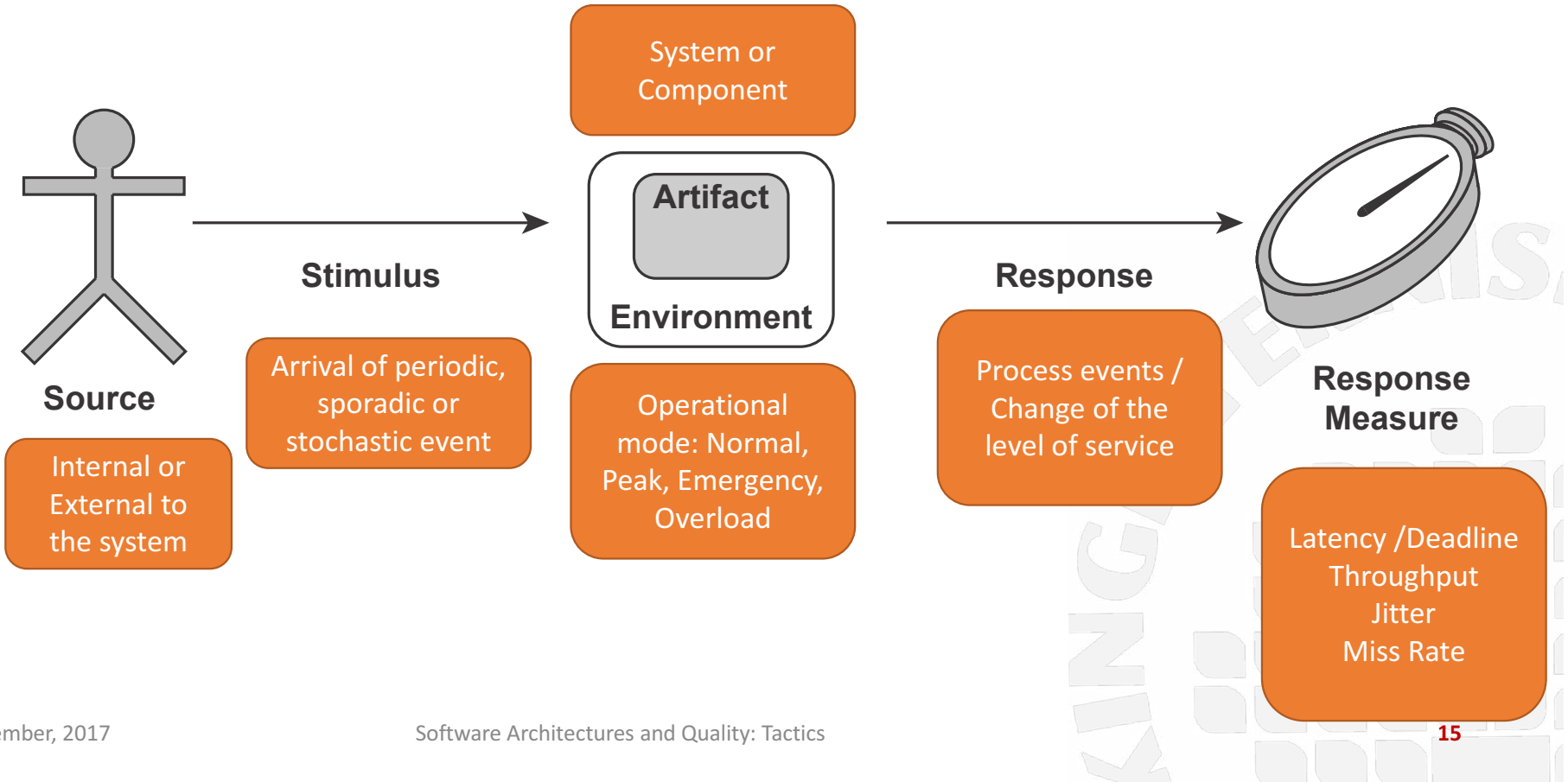
Performance

- It is the ability of the software to meet the timing requirements

Standard ISO/IEC25000 Definition: *"The degree to which the software product provides appropriate response and processing times and throughput rates when performing its function, under stated conditions."*

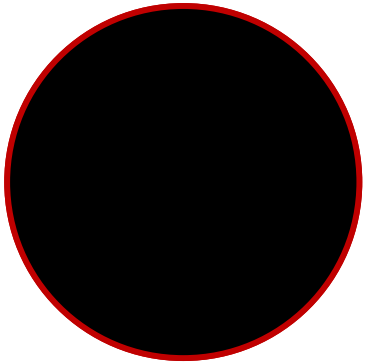
- Usually simplified to *Latency Time*
  - Time elapsed between firing an event and receiving a response
- Sometimes we also talk about resource utilization (memory consumption & CPU utilization)

Performance General Scenario



Reflection

- **Discuss during 2 minutes in groups of 2-3 persons about the tactics we can apply to improve or have performance (Latency / CPU consumption / Memory consumption under control)?**



Tactics for Performance

- Control Resource Demand
  - Manage sampling rate
  - Limit event response
  - Prioritize events
  - Reduce communication overhead (co-location)
  - Limit execution time (good enough results)
- Manage resources
  - Increase resources
  - Introduce concurrency
  - Multiple copies of computation (Load Balancer)
  - Multiple copies of data (*Cache*)
  - Limit queue sizes
  - Schedule resources

Runtime Qualities

Interoperability

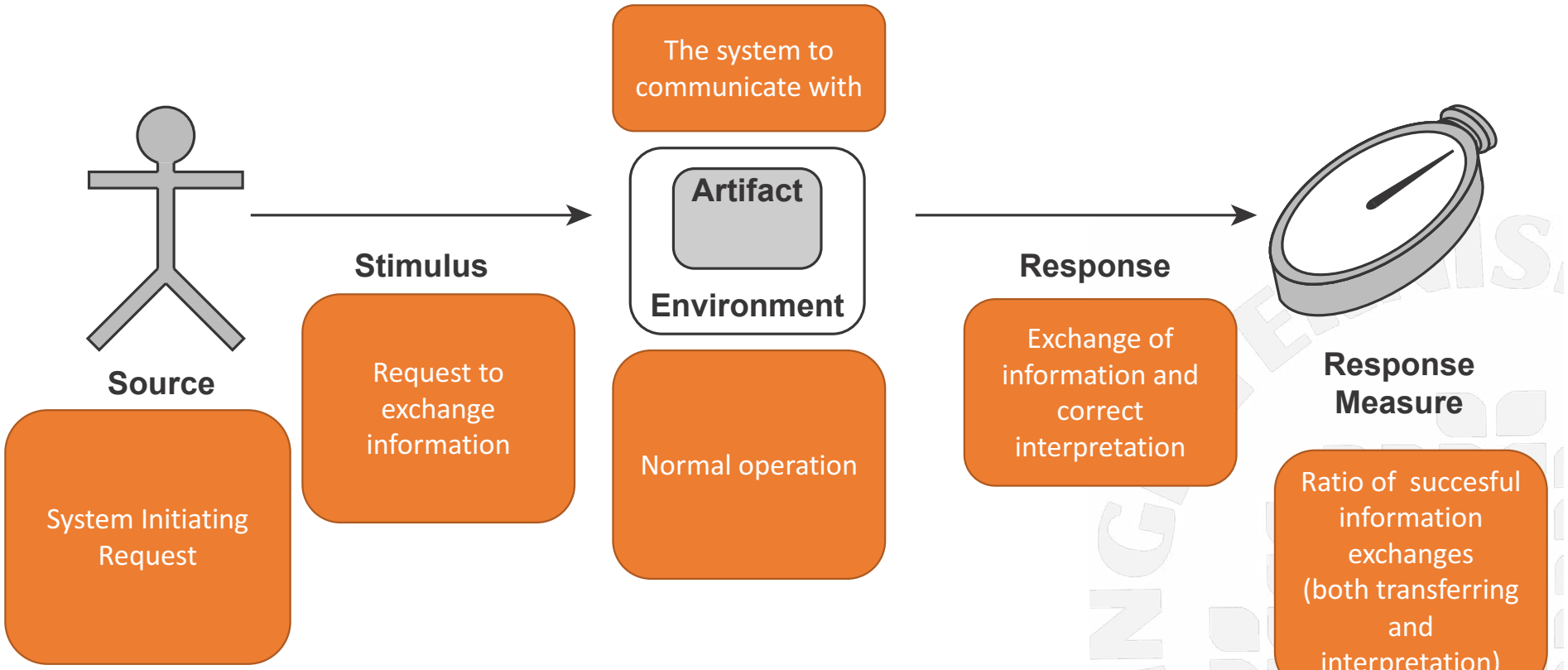
- Whether a system can interchange meaningful information and cooperate with other systems

Standard ISO/IEC25000 Definition: *"The degree to which the software product can be cooperatively operable with one or more other software products."*

Interoperability

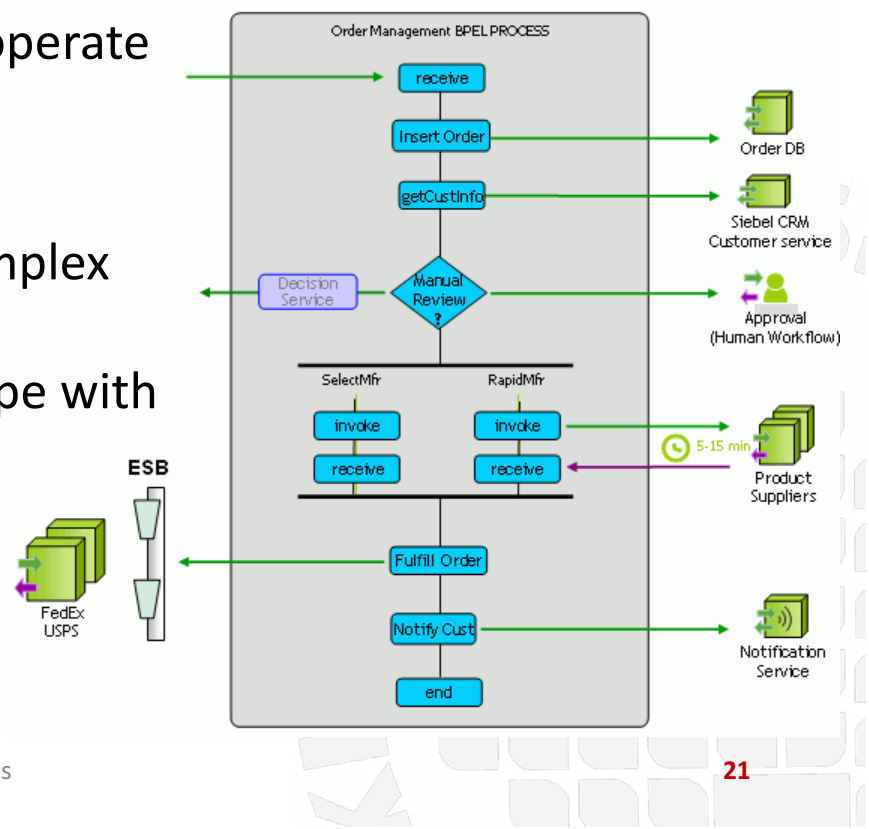
- Syntactic interoperability
- Semantic interoperability
- Interoperability standards:
  - IETF.org
  - Web Services/SOAP/XML –
  - REST/JSON

Interoperability General Scenario



Tactics for Interoperability

- Locate
  - Discover service: when the service that interoperate should be discovered at runtime
- Manage interfaces
  - Orchestrate: when the interoperation is a complex interaction
  - Tailor Interface: add or hide capabilities to cope with interoperations



Runtime Qualities

Security

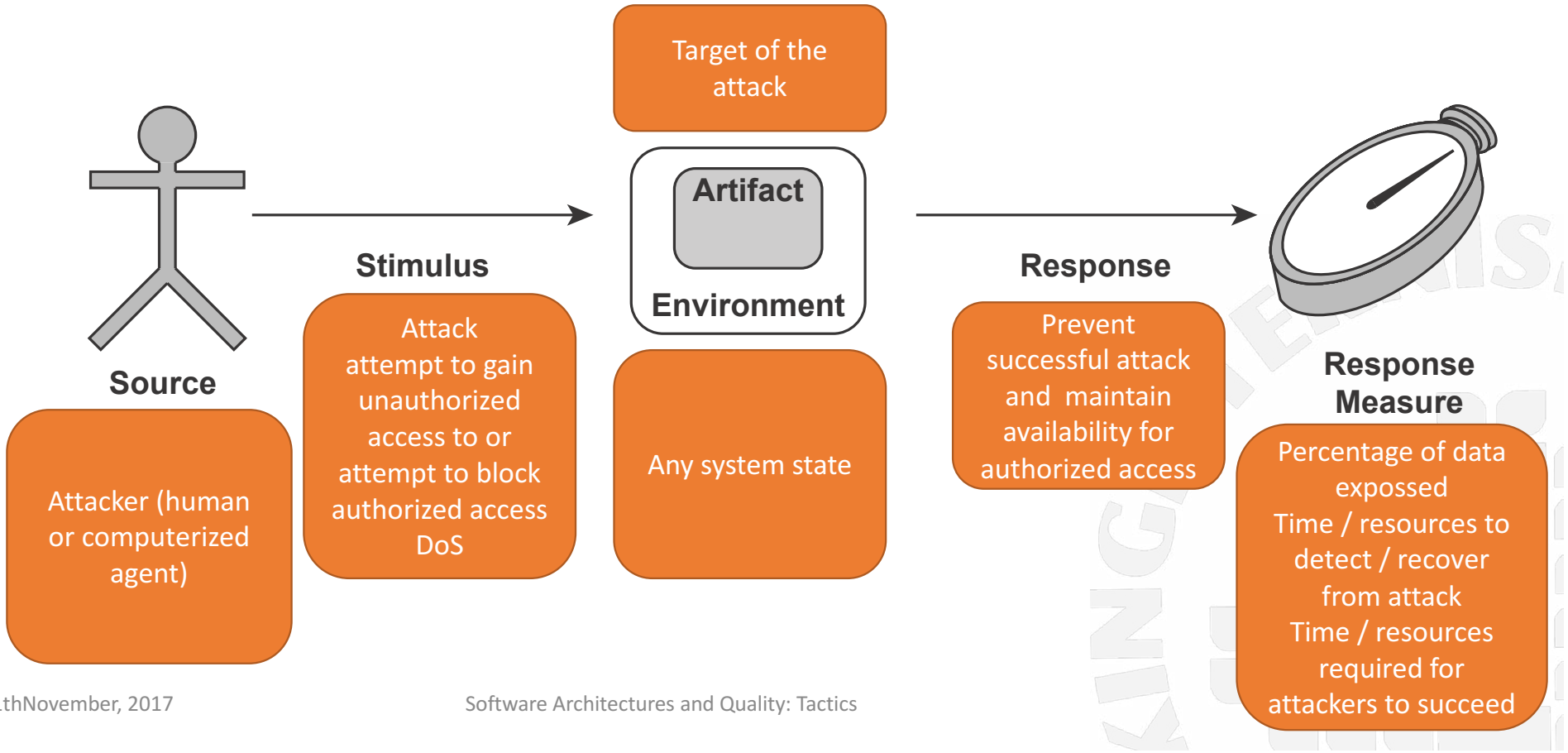
- Whether the software protect information and data to unauthorized access

Standard ISO/IEC25000 Definition: *"The protection of system items from accidental or malicious access, use, modification, destruction, or disclosure."*

- Confidentiality: Data protected to unauthorized access
- Integrity: Data not subject to unauthorized manipulation
- Availability*: System ready for its legitimate user

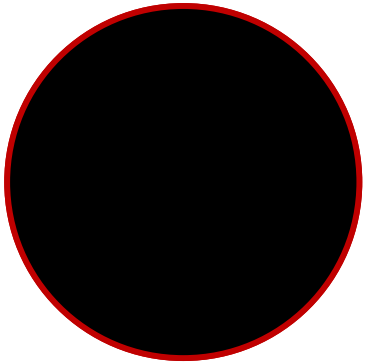
*Availability is seen also as a component of security

Security General Scenario



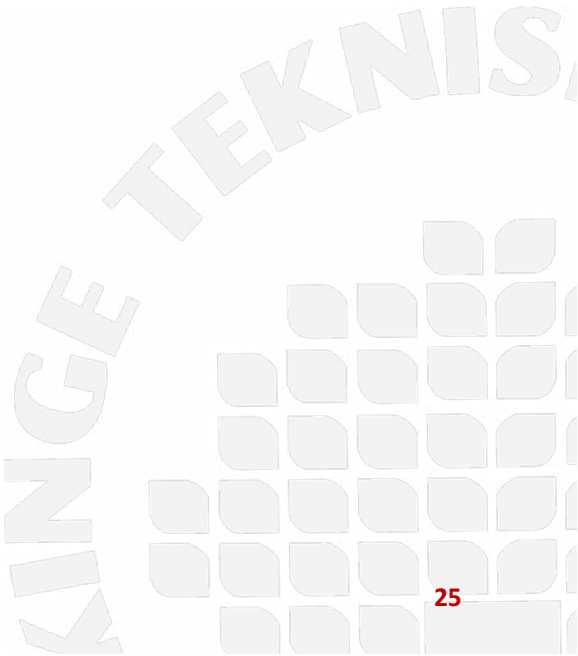
Reflection

- **Discuss during 2 minutes in groups of 2-3 persons about tactics for Security**



Tactics for Security

- Categories
  - Detect attacks
  - Resist attacks
  - React to attacks
  - Recover from Attacks

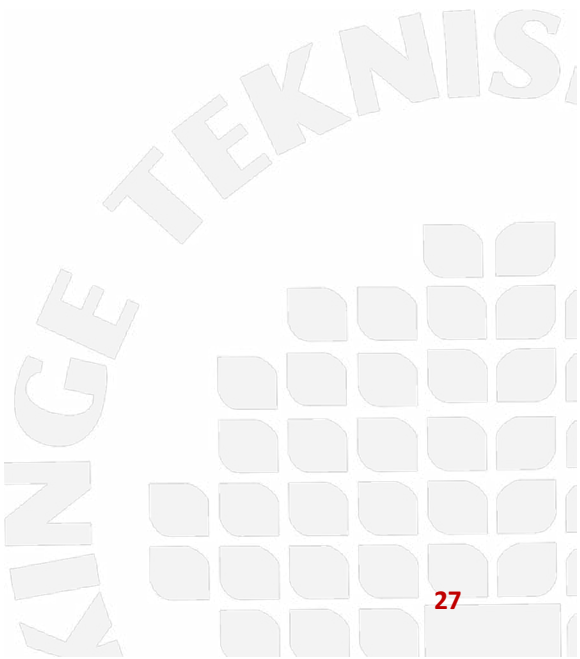


Tactics for Security

- Detect Attacks
  - Detect intrusion
  - Detect service denial
  - Verify Message Integrity
  - Detect Message Delay
- Recover from Attacks
  - Maintain audit trail (who did what?)
  - Restore (see availability)

Tactics for Security

- Resist attacks
  - Identify actors
  - Authenticate
  - Authorize
  - Limit access
  - Limit exposure
  - Encrypt data
  - Separate entities
  - Change default settings
- React to attacks
  - Revoke access
  - Lock computer
  - Inform actors



Runtime Qualities

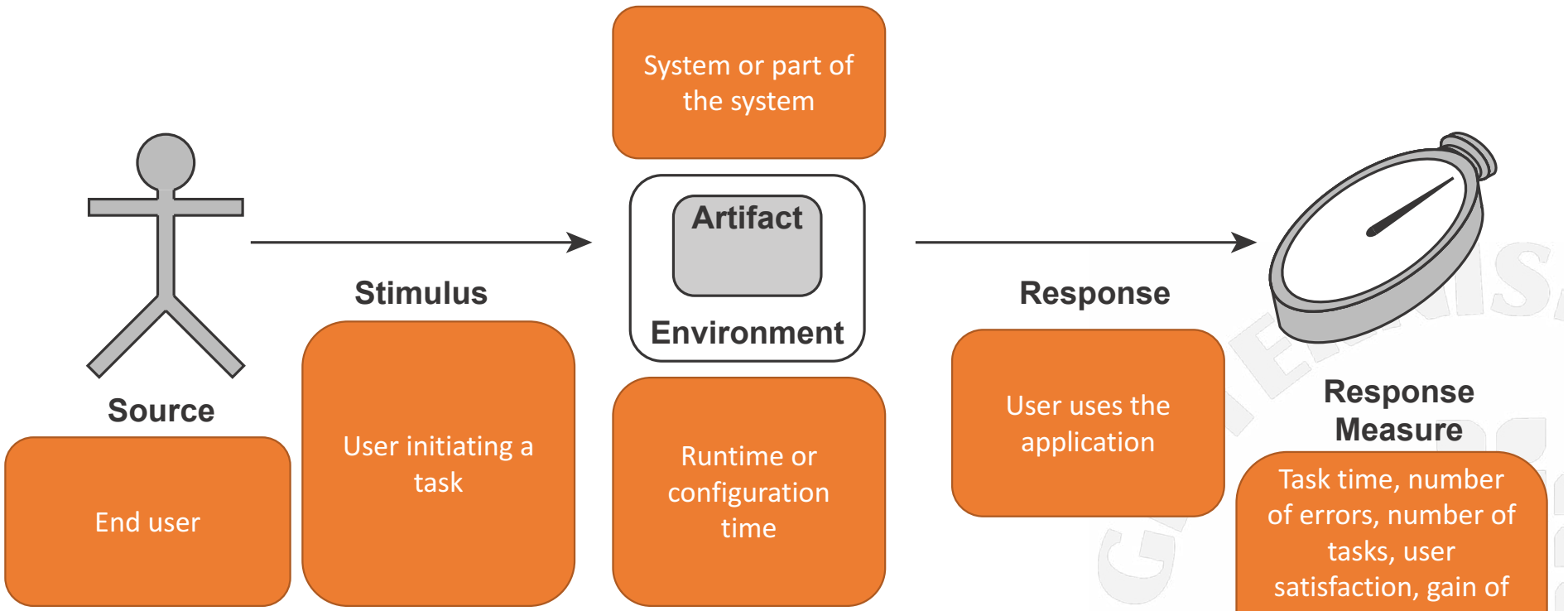
Usability

- How easy is for a user to accomplish a desired goal using the system

Standard ISO/IEC25000 Definition: "The degree to which specified users can achieve specified goals with effectiveness in use, efficiency in use and satisfaction in use in a specified context of use"

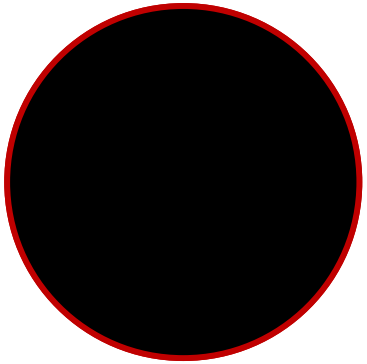
- Consider:
  - Learning the system features
  - Efficient use
  - Minimal impact of errors
  - Adapting to the user needs
  - Provide feedback in response

Usability General Scenario



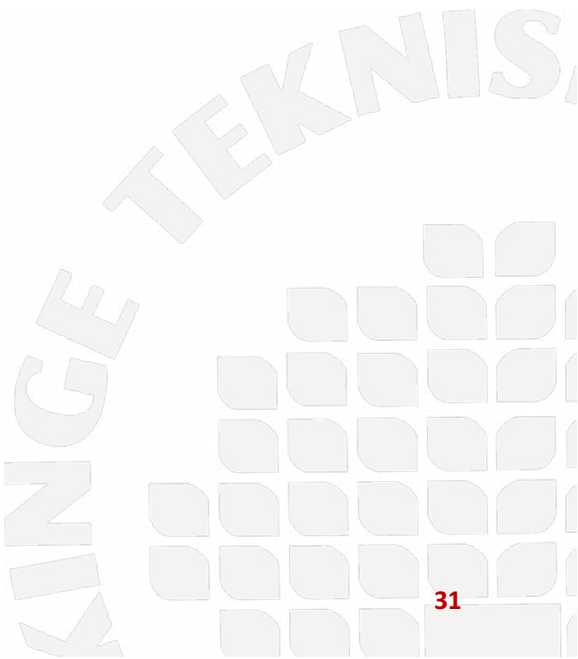
Reflection

- **Discuss during 2 minutes in groups of 2-3 persons about tactics for Usability**



Tactics for Usability

- Support User
  - Cancel
  - Undo
  - Pause/Resume
  - Aggregate
- Support System
  - Maintain task model
  - Maintain user model
  - Maintain system model



Runtime Qualities

Scalability

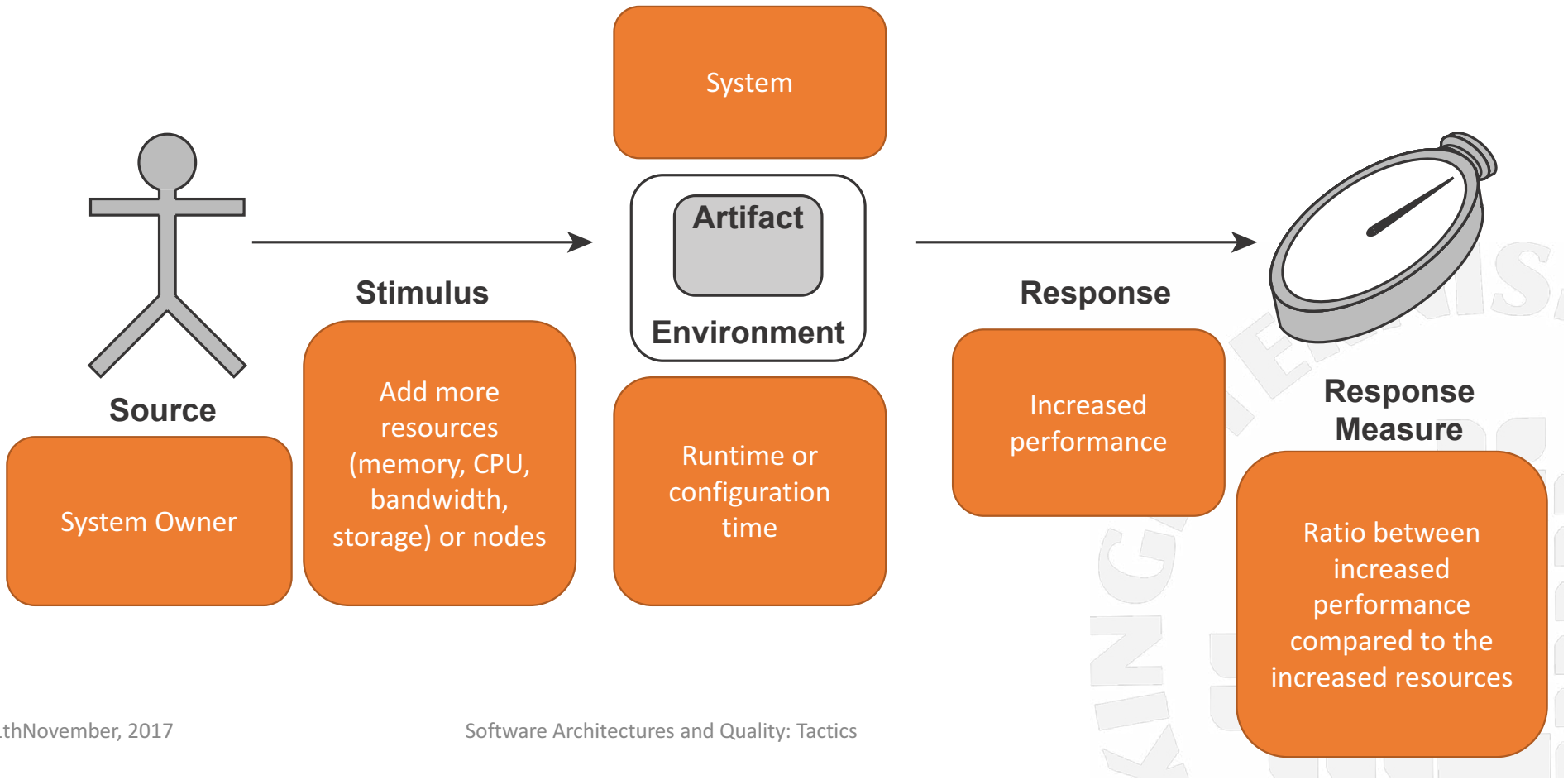
- How the system effectively manages [added] resources

Standard ISO/IEC25000 Definition*: "The degree to which the software product can be adapted for different specified environments without applying actions or means other than those provided for this purpose for the software considered.

*This version of ISO names scalability as adaptability

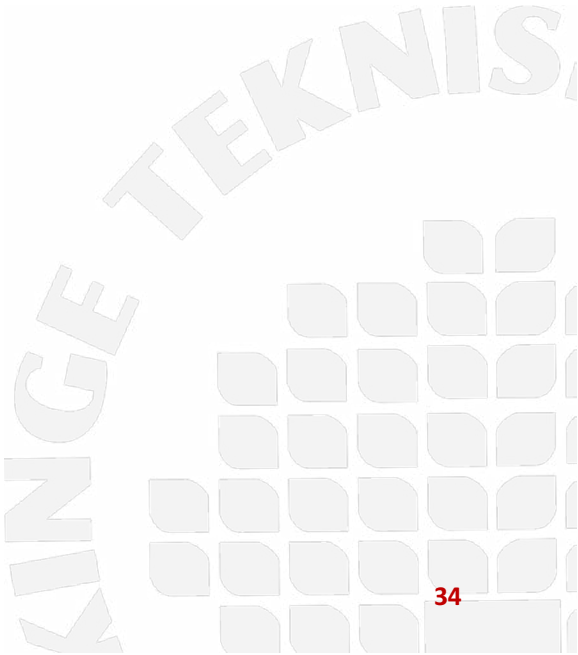
- Measures what happens for example when the number of requests grows
- If we are measuring how manages added resources, we will have to consider how impacts to other qualities

Scalability General Scenario



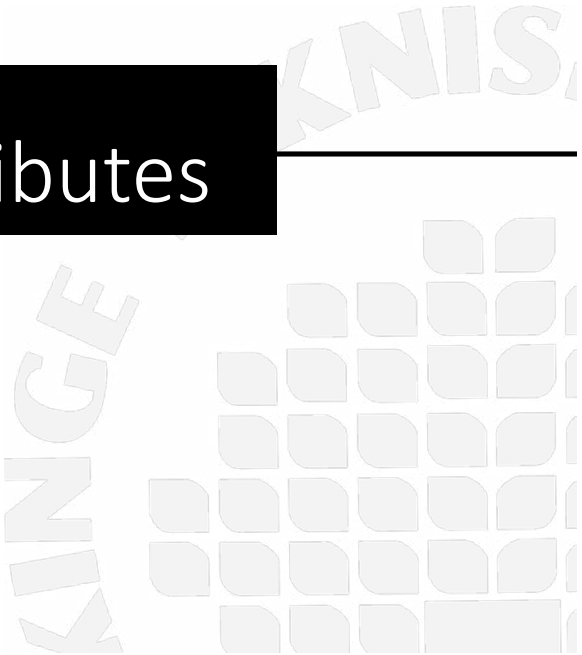
Scalability Tactics

- Not described in the book
- Parallelize computation
  - Threads, processes
- Asynchronous communication
- Non-waiting calls
- Shift of programming language paradigm
  - From Imperative (C++,Java, C#,C)
  - To Functional (F#,Haskell, Erlang, Go, Lisp)



Run-Time & Development-time Quality Attributes

Part III: Tactics for Development Time Qualities



Quick tour through some qualities: Design Time

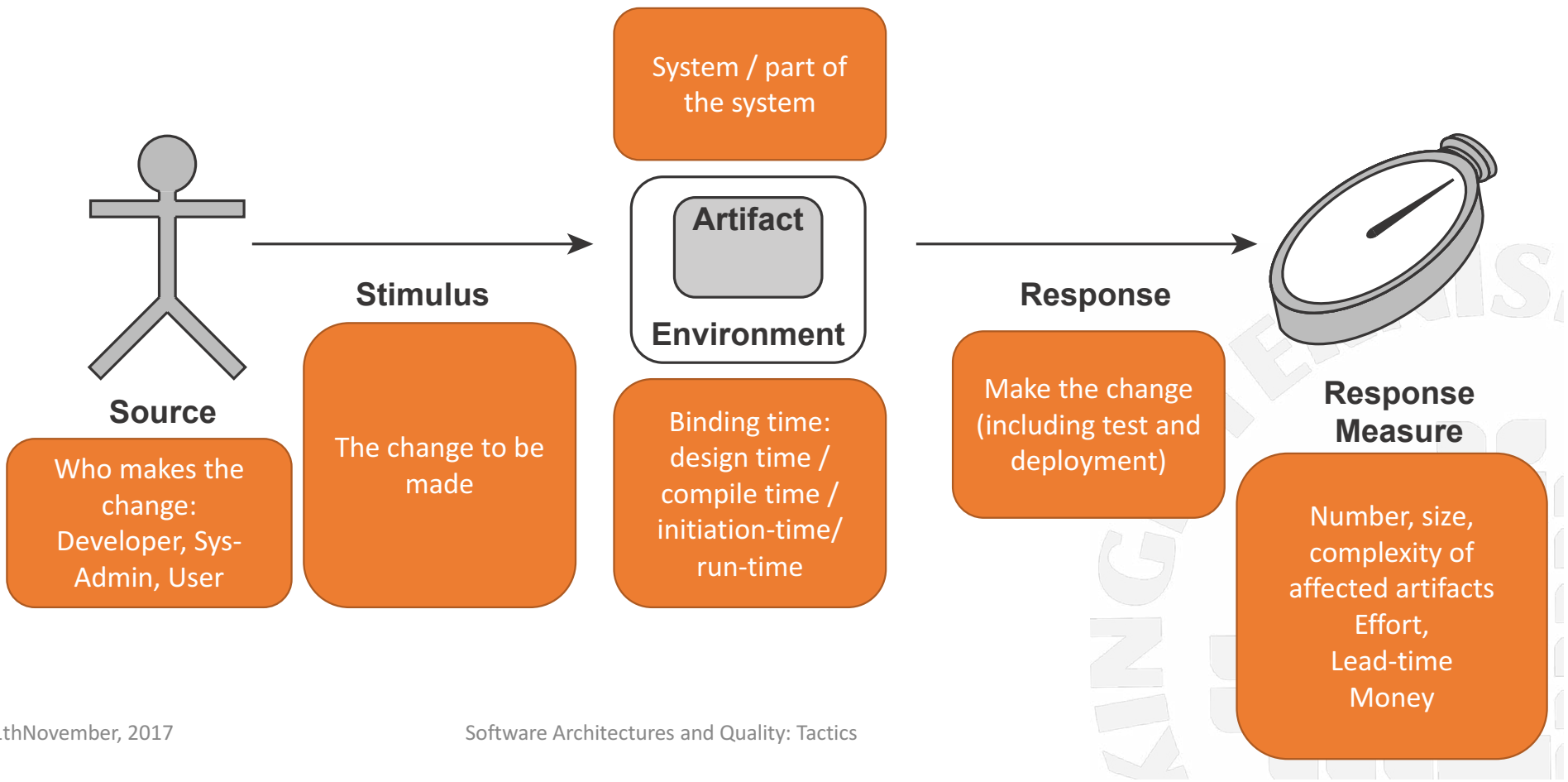
Modifiability/Maintainability

- The cost / effort required change the software system

Standard ISO/IEC25000 Definition (Maintainability): "The degree to which the software product can be modified. Modifications may include corrections, improvements or adaptation of the software to changes in environment, and in requirements and functional specifications. "

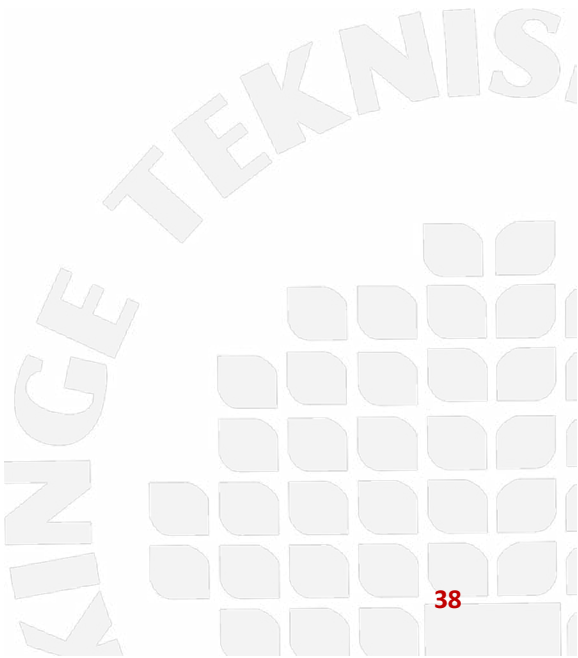
- Usually measured as the cost in money or effort in time to analyze, plan, design, develop and test a given change on the system

Modifiability General Scenario



Properties affecting modifiability

- **Size** of code concerned
- **Coupling** strength of explicit and implicit relations to other code
- **Cohesion**
  - uniformity of responsibility
  - freedom from (unexpected) side effects
- **Binding time** when change can be implemented



Tactics for Modifiability

- Reduce size of a module
  - Split module: avoid Swiss knifes
- Increase cohesion
  - Increase semantic coherence: avoid good classes
- Defer binding
- Reduce Coupling
  - Encapsulate
  - Use intermediary
  - Restrict dependencies
  - Refactor
  - Abstract common services

Defer Binding

Compare this two code snippets in terms of modifiability:

```
Main()
    {
        println(5*10);
    };
```

```
Main(argv, argc)
    {
      println(argv[0]*argv[1]);
    };
```

Quick tour through some qualities: Design Time

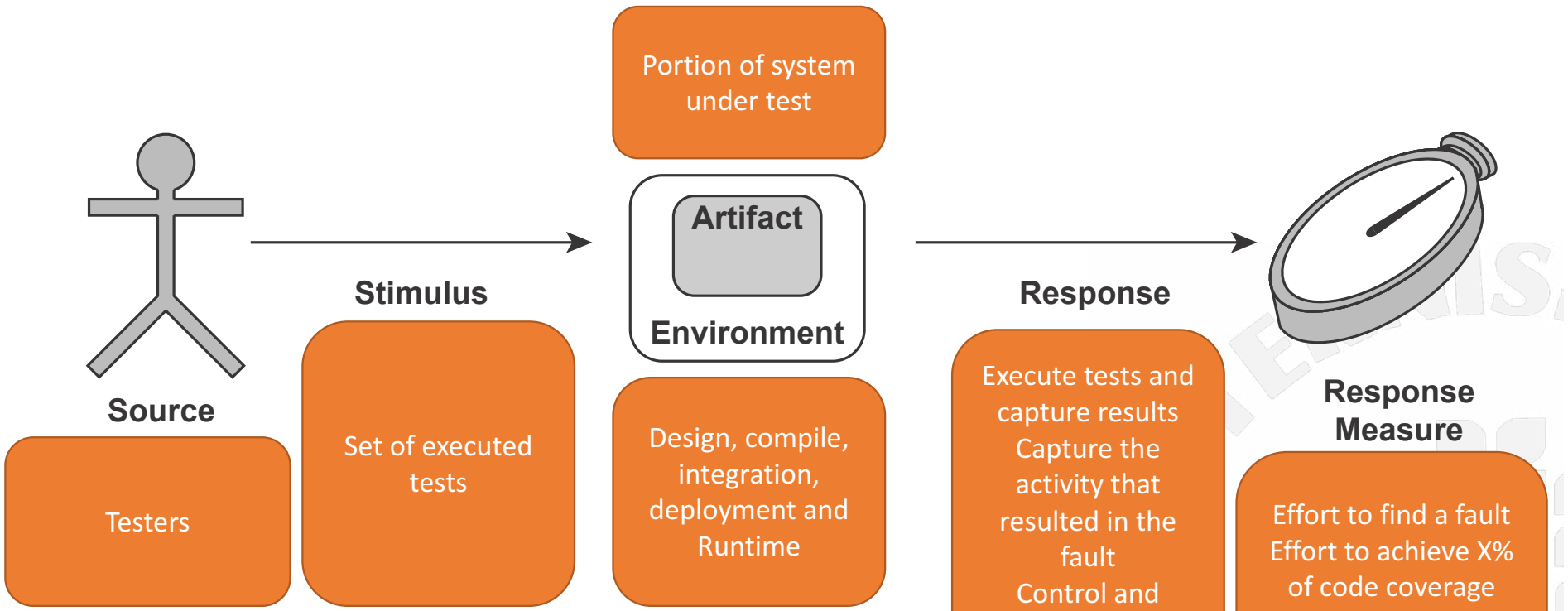
Testability

- How easy is finding software faults in the system

Standard ISO/IEC25000 Definition (Maintainability): "The degree to which the software product enables modified software to be validated."

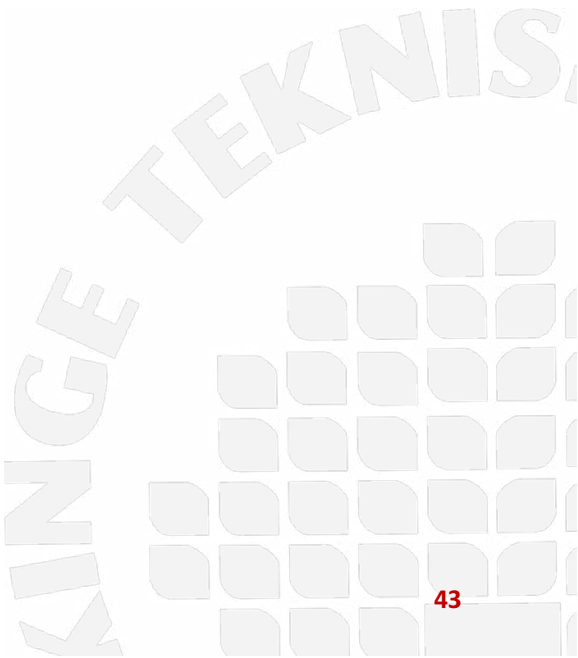
- Usually measured as the effort in time to test the software

Testability General Scenario



Testability Tactics

- Control and observe system state
  - Specialized testing interfaces
  - Record/Playback -> events/journal over state
  - Localized state storage
  - Abstract data sources
  - Sandbox
  - Executable Assertions (spot the faults)
- Limit complexity
  - Limit structural complexity
  - Limit non-determinism (random numbers, waiting times)



Software Architectures and Quality

Tactics and Quality Attributes

Javier González Huerta

javier.gonzalez.huerta@bth.se

BLEKINGE TEKNISKA HÖGSKOLA · BTH · in real life