Software Architectures and Quality

Tactics and Quality Attributes I: Quality Scenarios

Javier González Huerta

javier.gonzalez.huerta@bth.se

BLEKINGE TEKNISKA HÖGSKOLA · BTH ·

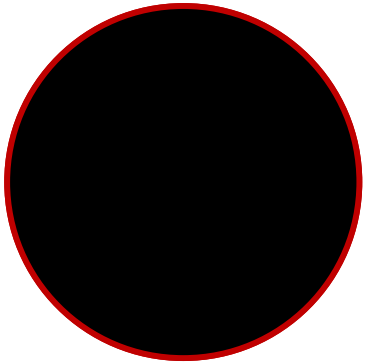
in real life

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

Reflection

- **Discuss during 2 minutes in groups of 2-3 persons about the difference between functional and non-functional requirements**



System Qualities: Non-Functional Requirements

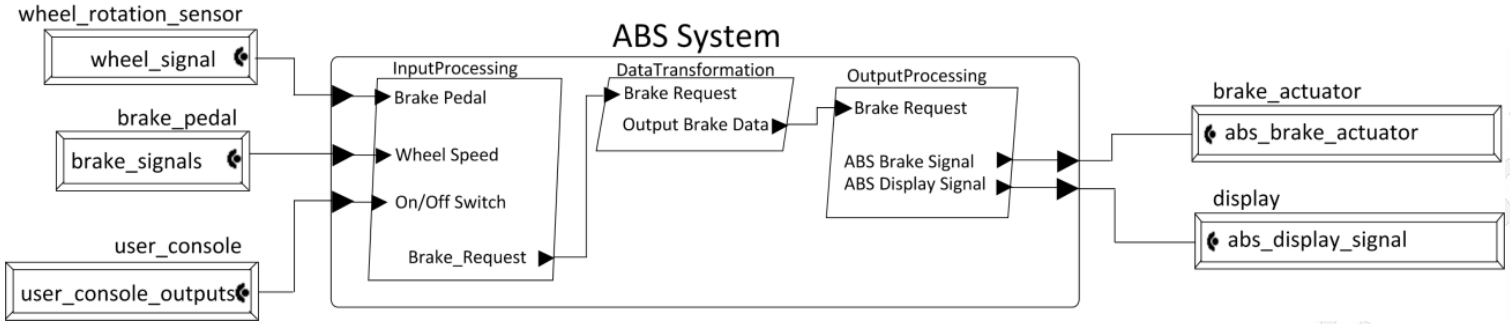
- **Functional Requirements:** Specify what the system software system should do (Functions / Features)
- **Non-functional Requirements:** Specify How the software system should do it
  - Specify the qualities that the system must have

System Qualities: Non-Functional Requirements

- Every system has quality characteristics
- Every functional requirement has some quality (non-functional requirements)
  - Often implicitly (dangerous)
  - Grouped into general system-wide quality requirements
  - Explicitly expressed

Implicit Non-Functional Requirements

- Functional Requirement:
  - The ABS System should activate the brakes based on the input from the brake pedal and the wheels rotation speed



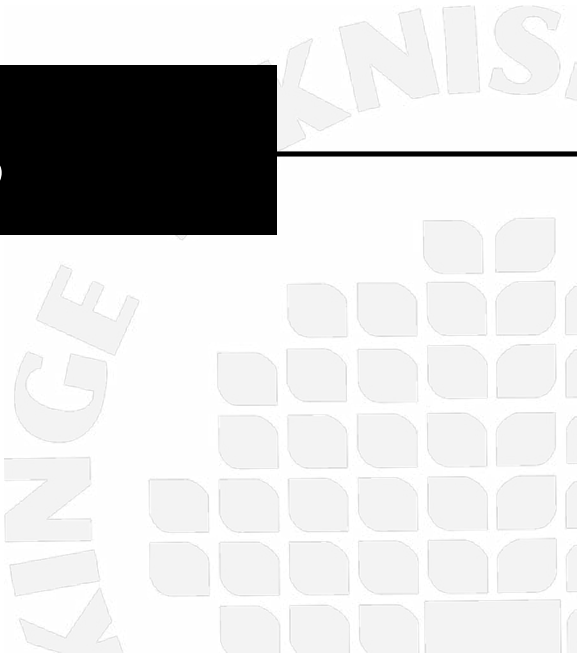
- If the computation takes forever,
  - Has the function really been performed?

Non-Functional Requirements Grouped per Function

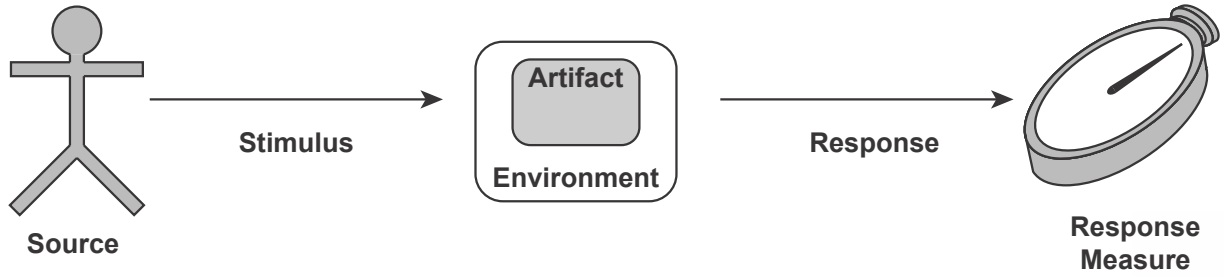
- Non-Functional Requirements can be expressed per function or grouped:
  - Per function: The ABSSystem response should be less than 1ms
  - Grouped: The latency for all the GUI-invoked functions should be less than 20ms (for every GUI-invoked functionality)

Tactics and Quality Attributes

Expressing Quality Attribute Scenarios



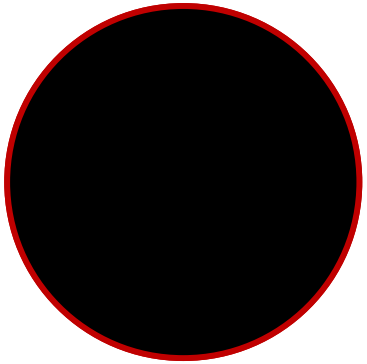
Specifying Non-Functional Requirements



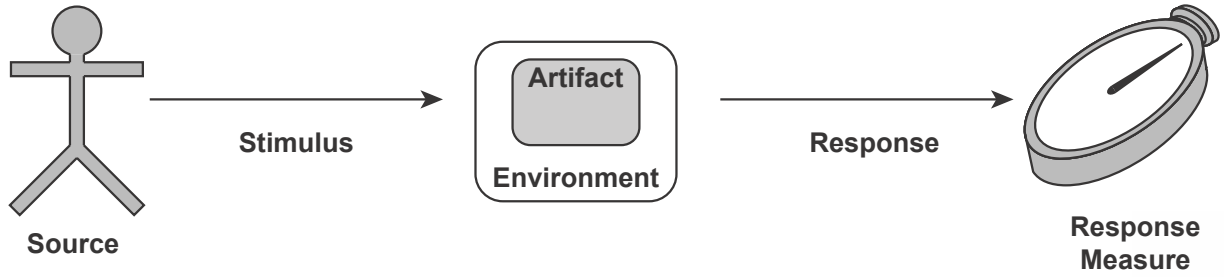
- We need to specify the non-functional requirements in a non-ambiguous, measurable way
- We initially handle these requirements in isolation, even though there will be interactions among them
  - Acting on one system quality can have positive or negative impacts on other system qualities

Reflection

- **Discuss during 2 minutes in groups of 2-3 persons about an example of a decision that potentially will have impact on more than one system quality**

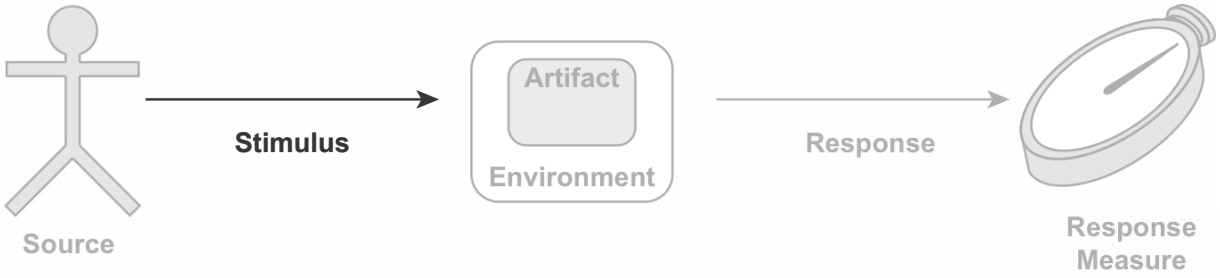


Quality Attribute Scenarios



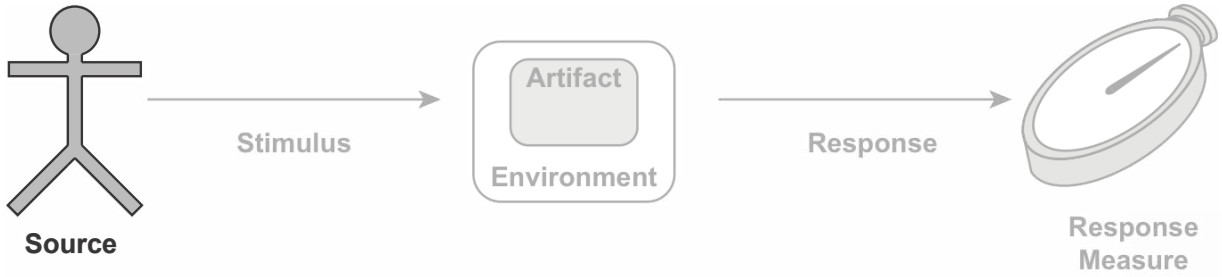
- We distinguish between:
  - **General quality attribute scenarios (AKA *General Scenarios*):**
    - To specify the way we specify / measure a given quality attribute
    - Potentially can be applied to any system
  - **Concrete quality attribute scenarios (AKA *Concrete Scenarios*):**
    - Specify the non-functional requirement / or a concrete measurement for a specific system

Quality Attribute Scenarios: Stimulus



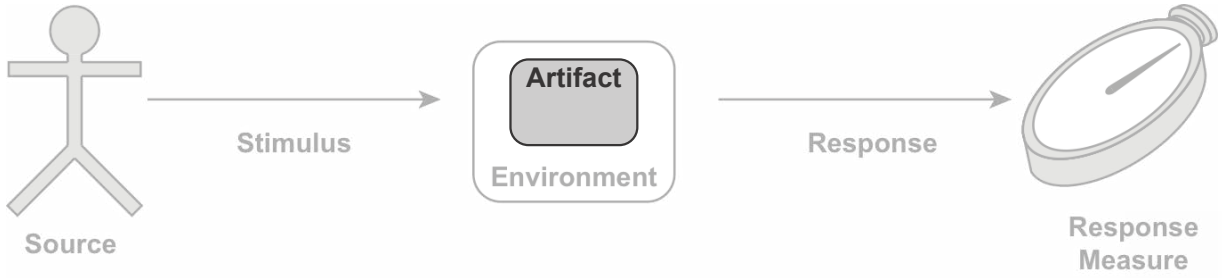
- **Stimulus:**
  - Event arriving to the system
  - Examples:
    - user interaction
    - request for modification
    - …

Quality Attribute Scenarios: Source



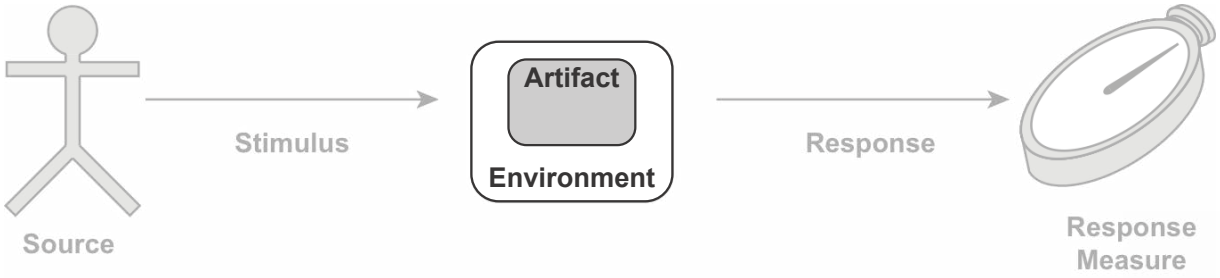
- **Source of Stimulus:**
  - The entity that that generates the stimulus
  - Examples:
    - User (for the user interaction stimulus)
    - Developer (for a request for modification scenario)
  - It might have an impact on how we treat the stimulus: authorized user vs hacker

Quality Attribute Scenarios: Artifact



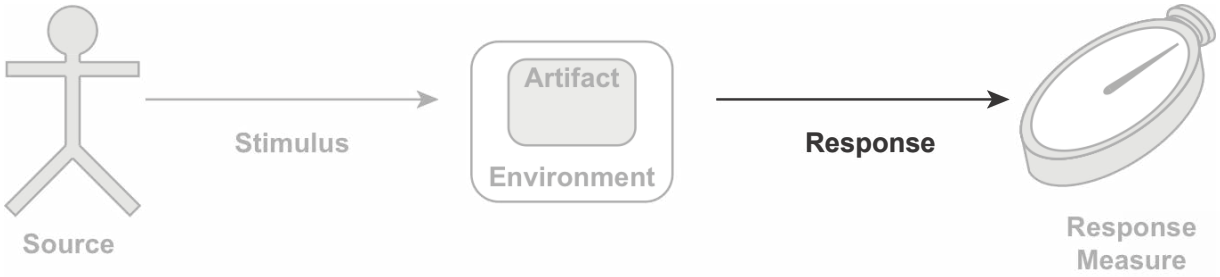
- **Artifact:**
  - some artifact that receives the stimulus and to which the requirement applies:
    - The system
    - Some portion of the system (module, set of modules)

Quality Attribute Scenarios: Environment



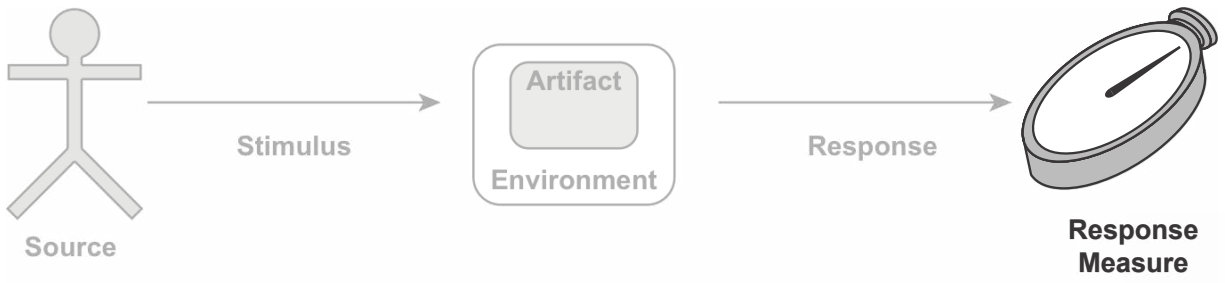
- **Environment:**
  - Describes the context on which the scenario is applicable
  - Qualifies the stimulus and or the artifact
    - It is not the same if a change request arrives before or after we have launched / deployed the version
  - Examples:
    - Mode of execution: Normal / Peak / Overloaded / Startup / Shutdown
    - Time: development time / run-time

Quality Attribute Scenarios: Response



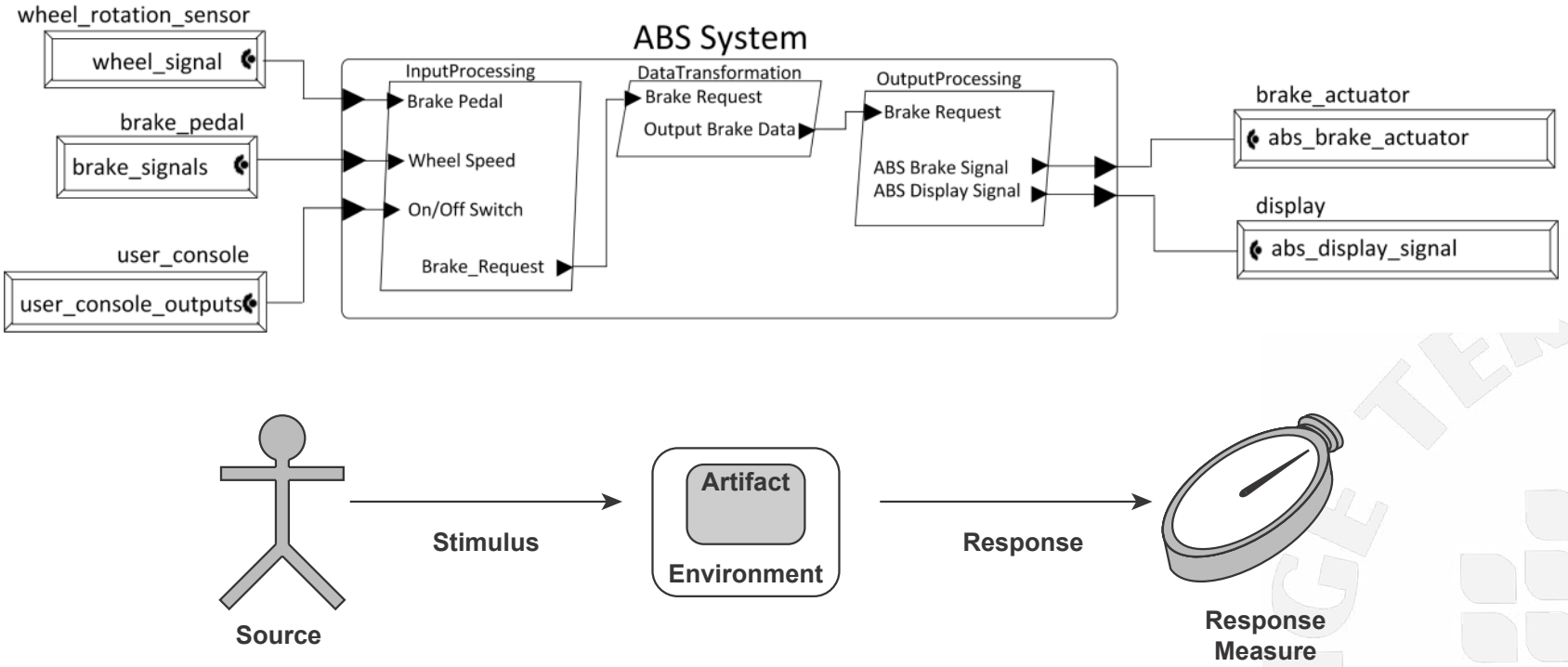
- **Response:**
  - How the system responds to the system in terms of responsibilities
    - For runtime qualities the responsibilities would be performed by the system
    - For design-time qualities the responsibilities would be performed by the developers
  - Examples:
    - **Performance:** The ABSSystem responds activating the brake actuator
    - **Modifiability:** The modification is completely implemented and tested

Quality Attribute Scenarios: Response Measure

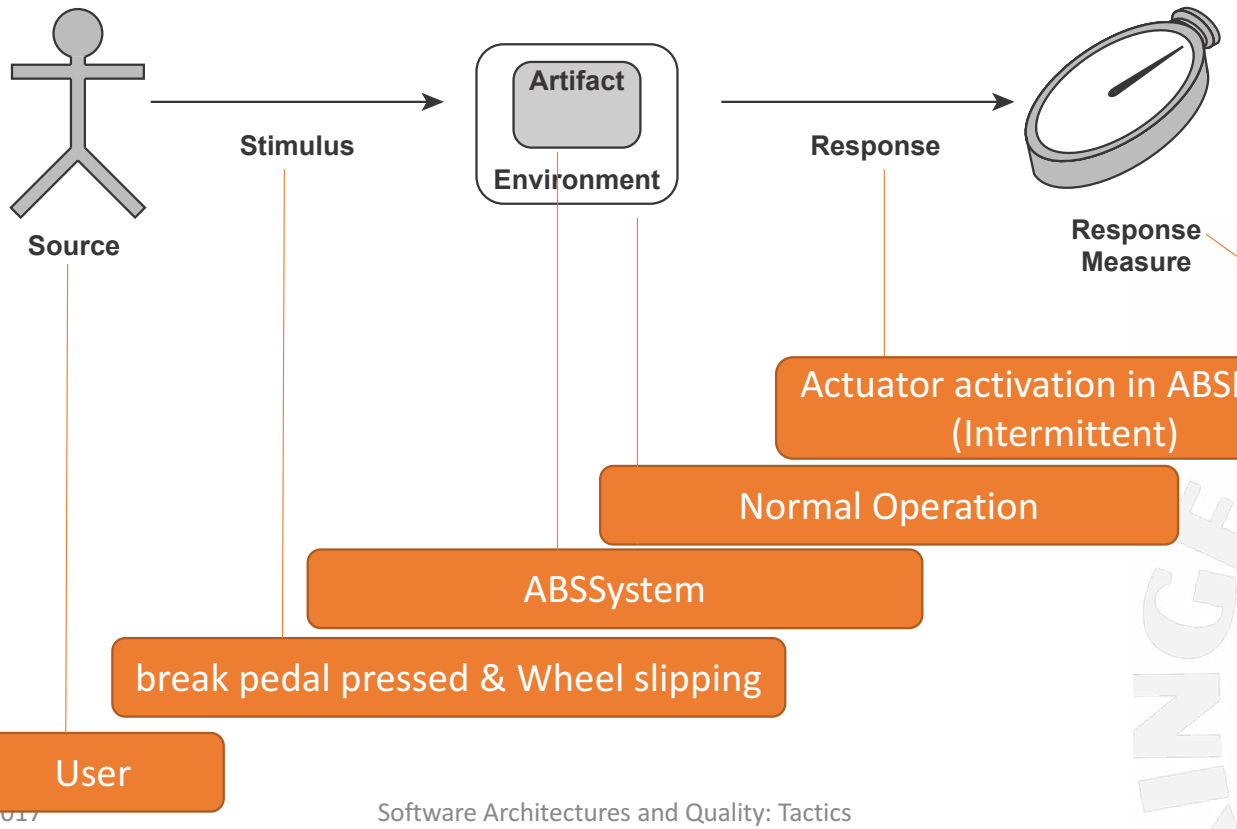


- **Response Measure:**
  - Measurable property of the system, to be able to test the degree of fulfillment of the scenario
  - Examples:
    - **Performance:** Latency in seconds
    - **Modifiability:** Average effort in hours

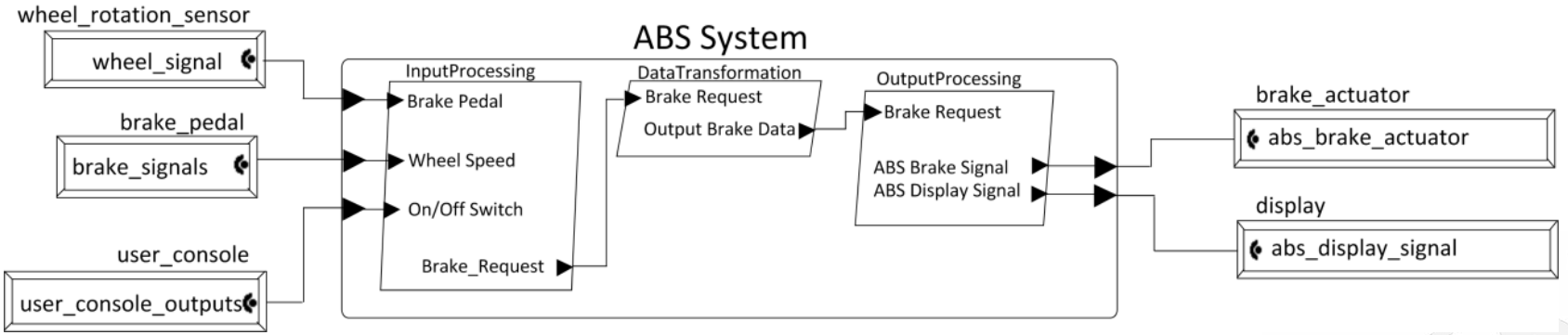
Performance General Scenario: Latency

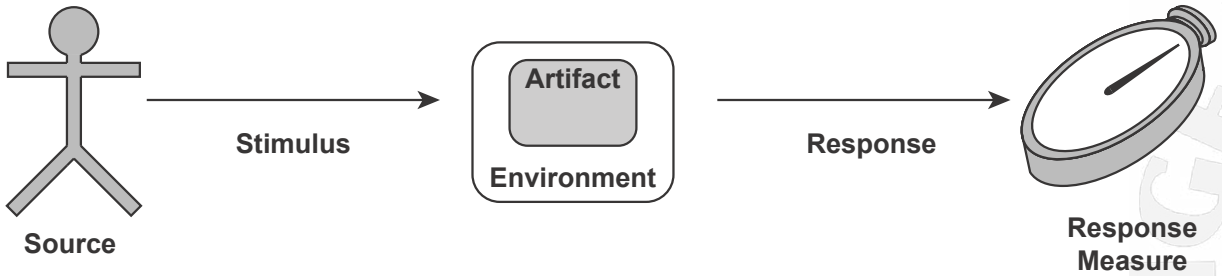


Performance General Scenario: Latency

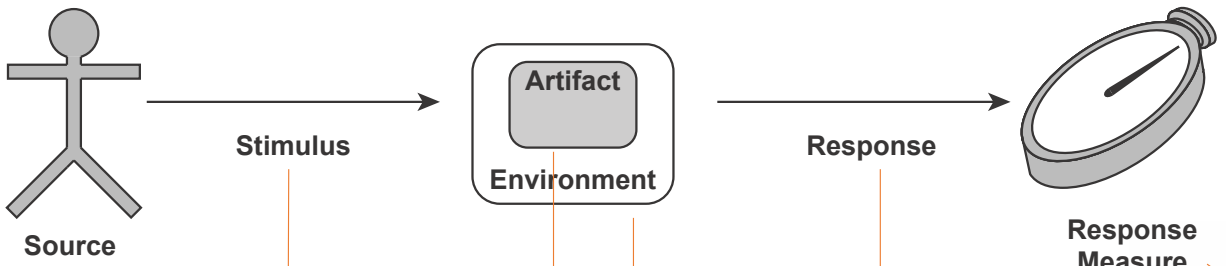


Modifiability General Scenario: Effort





Modifiability General Scenario: Effort



Developers

Change the break balance to avoid the slippery only in the wheel that slippers

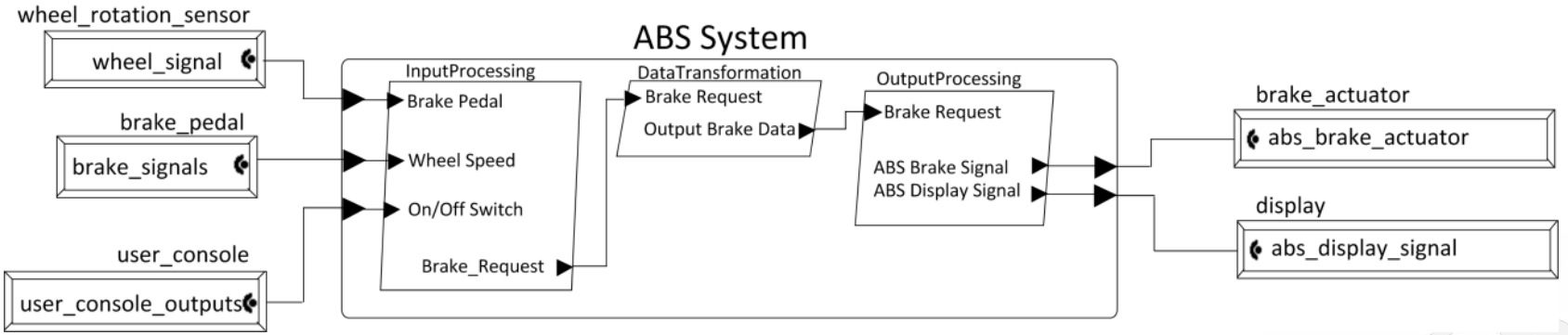
ABSSystem

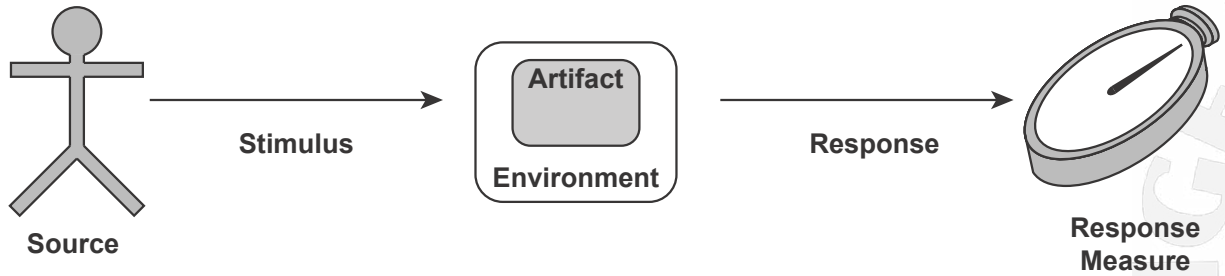
Development time

Implement the modification

Time in persons month

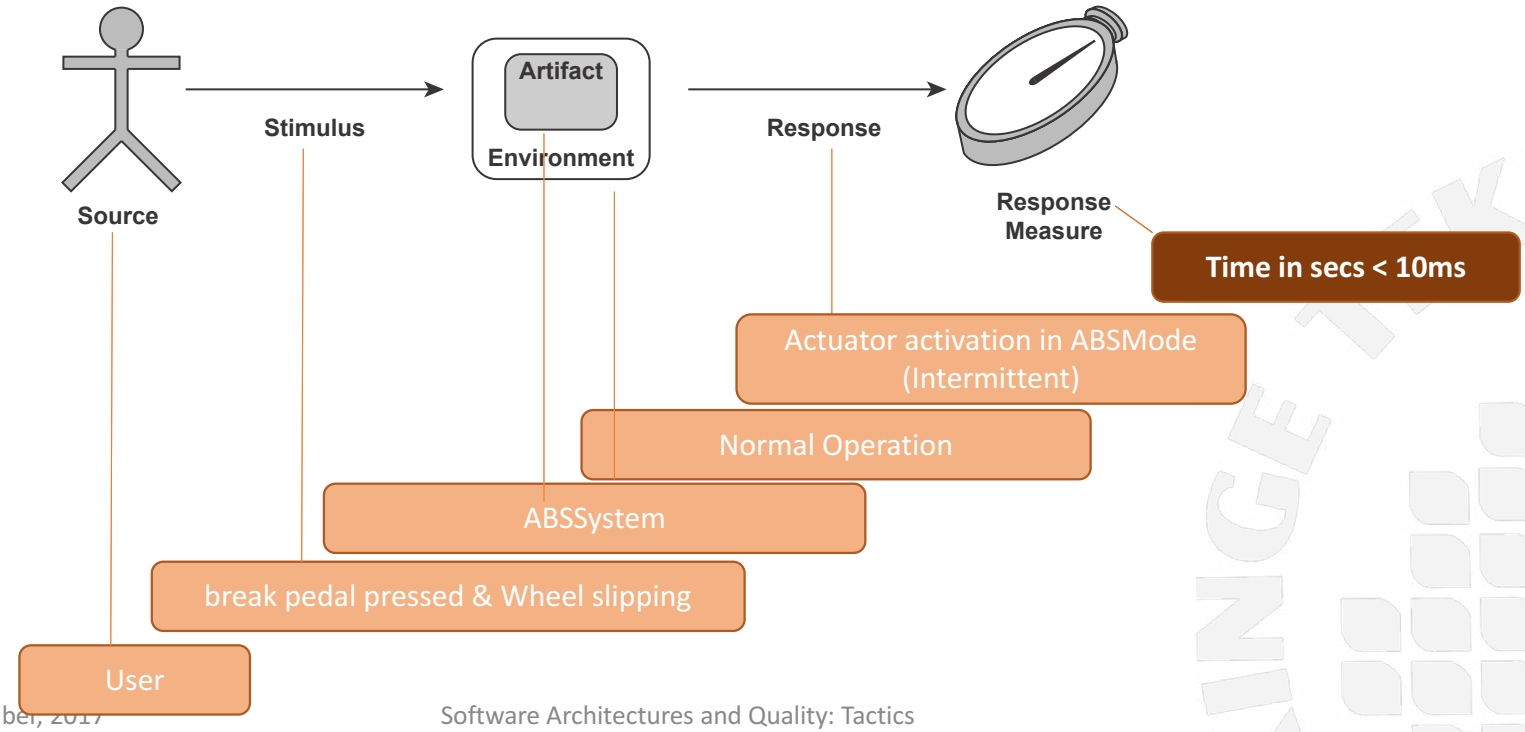
Performance General Scenario: Latency





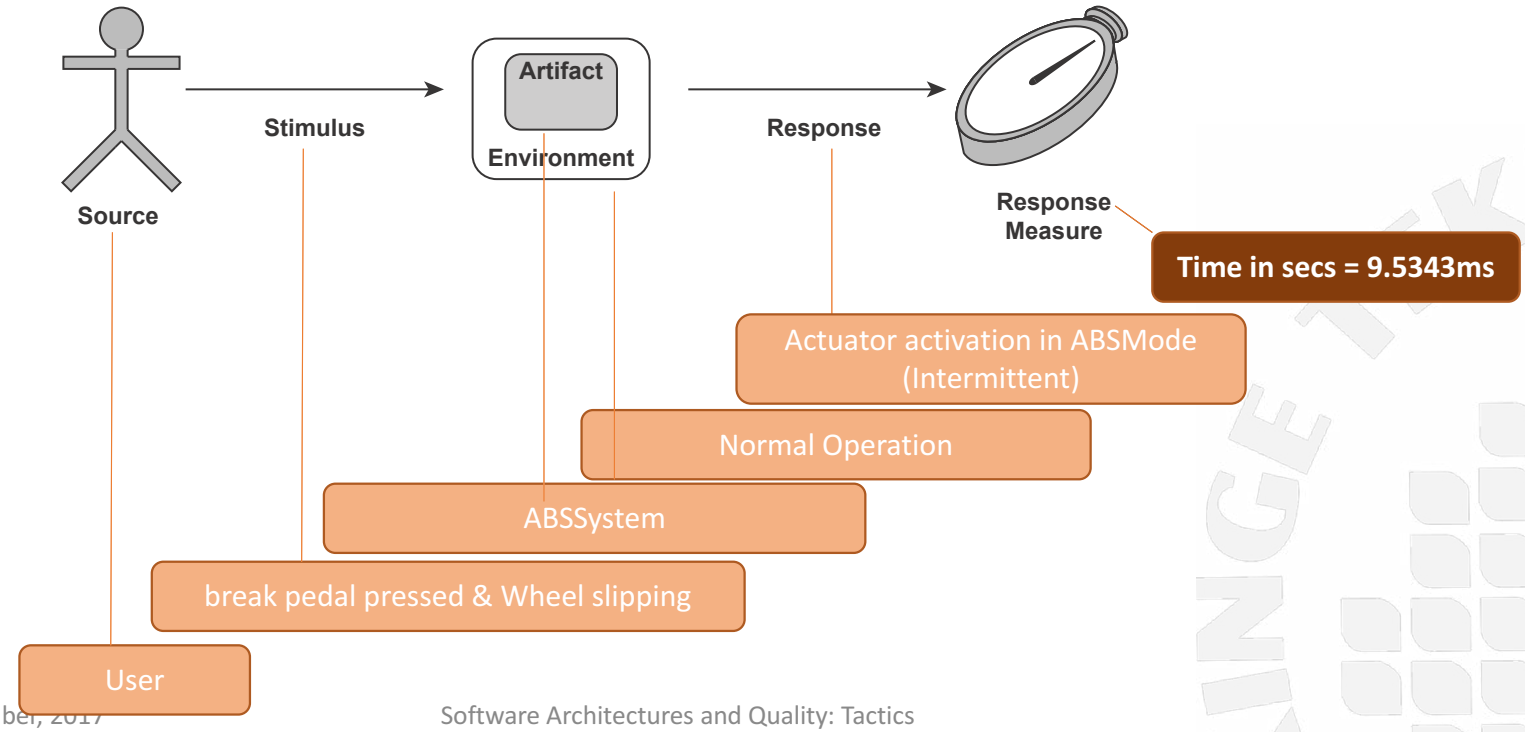
Quality Attribute Scenario: Expressing Requirements

- Non-Functional requirements are constraints on the values of a given quality attribute



Quality Attribute Scenario: Expressing Measurements / Predictions

- We can express the quality attribute levels as a result of a measurement / prediction



Software Architectures and Quality

Tactics and Quality Attributes I: Quality Scenarios

Javier González Huerta

javier.gonzalez.huerta@bth.se

BLEKINGE TEKNISKA HÖGSKOLA · BTH ·

in real life