Software Architecture in Practice

Chapter 4
Understanding Quality Attributes

Presenter: Dr. Hamed Vahdat-Nejad
http://perlab.birjand.ac.ir/

Architecture and Requirements

Requirements for a system come in a variety of forms: textual requirements, existing systems, use cases, user stories, and more.

No matter the source, all requirements encompass the following categories :

Functional requirements :These requirements state what the system must do , and how it must behave or react to runtime stimuli.

Quality attribute requirements : These requirements state the quality of the system in satisfying functional requirements.

Functional/Nonfunctional

- The quality attributes go beyond functionality , which is the basic statement of the system 's capabilities , services , and behavior .
- Systems are extended to add functionalities.
- Systems are frequently redesigned not because they are functionally deficient, but to improve the quality. The replacements are often functionally identical but because they are difficult to maintain or they are too slow or they have been compromised by hackers .

Functionality

- Functionality is the ability of the system to do the work for which it was intended.
- functionality does not determine architecture.
- We design our systems as structured sets of cooperating architectural elements modules , layers , classes , services , databases , apps and on and on
- Functionality is achieved by assigning responsibilities to architectural elements

Quality attribute

- A quality attribute (QA) is a measurable or testable property of a system that is used to indicate how well the system satisfies the needs of its stakeholders.

- You can think of a quality attribute as measuring the " goodness or quality "of a product along some dimension of interest to a stakeholder.

Constraints

- A constraint is a design decision with zero degrees of freedom .
- It's a design decision that's already been made .
- Examples include the requirement to use a certain programming language or to reuse a certain existing module .

Quality attributes considerations

From an architect's perspective , there are three problems with quality attributes:

1. The definitions provided for an attribute are not testable .

- It is meaningless to say that a system will be " modifiable . " Every system may be modifiable with respect to one set of changes and not modifiable with respect to another .
- A system may be robust (persistent) with respect to some faults and fragile with respect to others.
- Solution: Quality attribute scenarios

Quality attributes considerations

2. Discussion often focuses on which quality attribute a particular concern belongs to?

- Is a system failure due to a denial-of-service attack an aspect of availability, an aspect of performance, an aspect of security, or an aspect of usability ?
- All four attribute communities would claim ownership of a system failure due to a denial-of-service attack .
- All are, to some extent, correct.
- Solution: Quality attribute scenarios

Quality attributes considerations

3. Each attribute community has developed its own vocabulary.

- The performance community has events arriving at a system
- The security community has attacks arriving at a system
- The availability community has failures of a system and
- The usability community has user input.

All of these may actually refer to the same event, but they are described using different terms.

Quality attributes categories

There are two categories of quality attributes

1. Describe some property of the system at runtime such as availability, performance or usability .
2. Describe some property of the development of the system, such as modifiability or testability.

Balancing between attributes

- Quality attributes can never be achieved in isolation .
- The achievement of any one will have an effect sometimes positive and sometimes negative on the achievement of others.
- For example almost every quality attribute negatively affects performance.
- Portability requires independence from system, which enforces overhead to be executable on any system (Comparing to a specific platform). This overhead decreases the performance.
- The architecture should balance the target attributes (trade off)

Specifying Quality Attribute Requirements

- A quality attribute requirement should be unambiguous and testable .
- We use a common form to specify all quality attribute requirements.
- This has the advantage of emphasizing the commonalities among all quality attributes .
- It has the disadvantage of occasionally being a force-fit for some aspects of quality attributes .

Common form to specify a quality attribute

1- Stimulus: Describe an event arriving at the system .

Examples:

- An event for the performance attribute
- A user operation for the usability attribute
- An attack for the security attribute.
- A modification request for the modifiability attribute
- The completion of a development phase for testability attribute

Common form to specify a quality attribute

2- Stimulus source.

- A stimulus must have a source
- It must come from somewhere
- The source of the stimulus may affect how it is treated by the system
- A request from a reliable user is treated differently from a request from an unreliable user.

Common form to specify a quality attribute

3- Response

- How the system should respond to the stimulus
- The response consists of the responsibilities that the system (for runtime qualities) or the developers (for development-time qualities) should perform in response to the stimulus .
- Example: In a performance scenario, an event arrives (the stimulus) and the system should process that event and generate a response .
- Example: In a modifiability scenario, a request for a modification arrives (the stimulus) and the developers should implement the modification without side effects and then test and deploy the modification .

Common form to specify a quality attribute

4- Response measure

Determining whether a response is satisfactory

whether the requirement is satisfied is enabled by providing a response measure.

Example: For performance, delay or throughput should be measured

Example: for modifiability, it could be the required time for implementing, testing and deploying a modification.

Common form to specify a quality attribute

There are two more characteristics that are less important :

5- Environment

- The environment is the set of circumstances in which the scenario takes place.
- The environment acts as a qualifier on the stimulus.
- Example: a request for a modification that arrives after the code has been frozen for a release may be treated differently than one that arrives before the freeze.
- Example: A failure that is the fifth successive failure of a component may be treated differently than the first failure of that component.

Common form to specify a quality attribute

6- Artifact:

- The artifact is the portion of the system to which the requirement applies .
- Frequently this is the entire system, but occasionally specific portions of the system .
- Example: A failure in a data store may be treated differently than a failure in the metadata store .
- Example: Modifications to the user interface may have faster response times than modifications to the middleware.

Summarize (Quality attribute requirement)

We capture quality attribute requirements formally as six - part scenarios .

It is common to omit one or more of these six parts particularly in the early stages of thinking about quality attributes

However, knowing that all parts are there forces the architect to consider whether each part is relevant.

Summarize (Quality attribute requirement)

We capture quality attribute requirements formally as six - part scenarios :

1- Source of stimulus :Some entity (a human, a computer system, or any other actuator) that generated the stimulus .

2- Stimulus: A condition that requires a response when it arrives at a system.

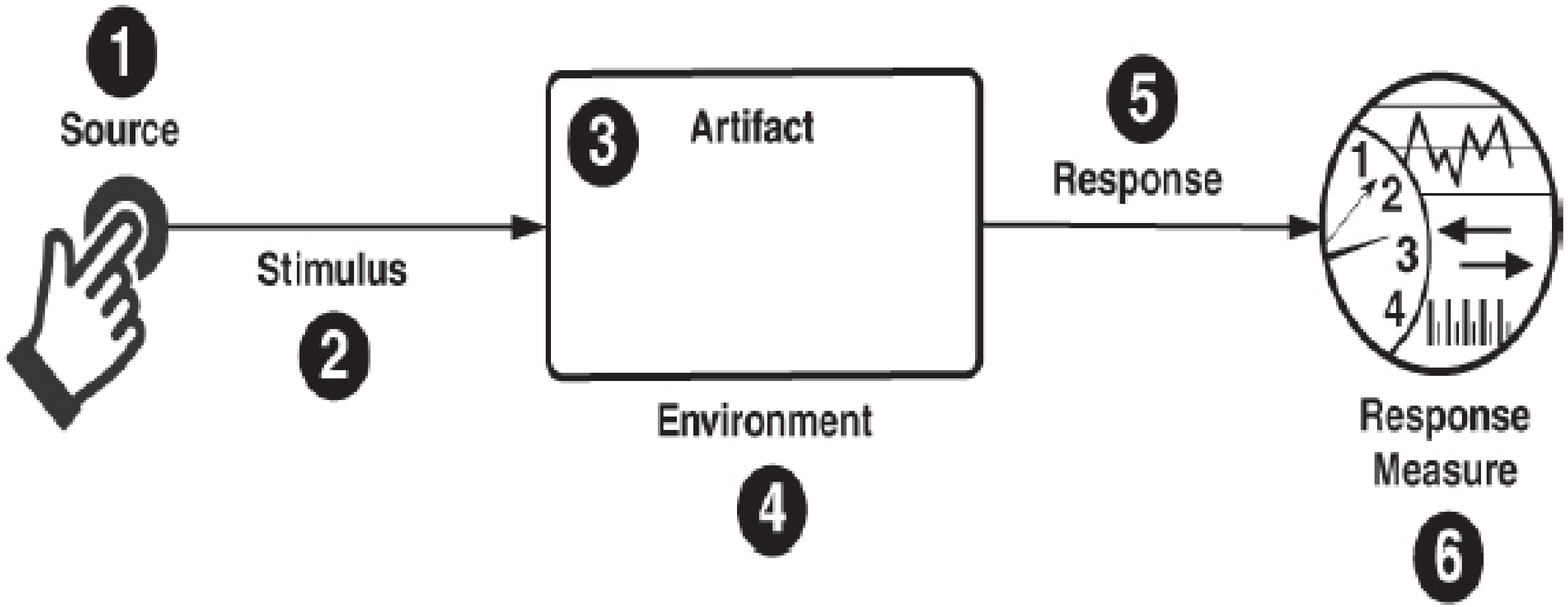
3- Environment: The stimulus occurs under certain conditions such as overload condition or normal operation or some other relevant states.

4- Artifact: A collection of systems, the whole system, or some piece or pieces of it.

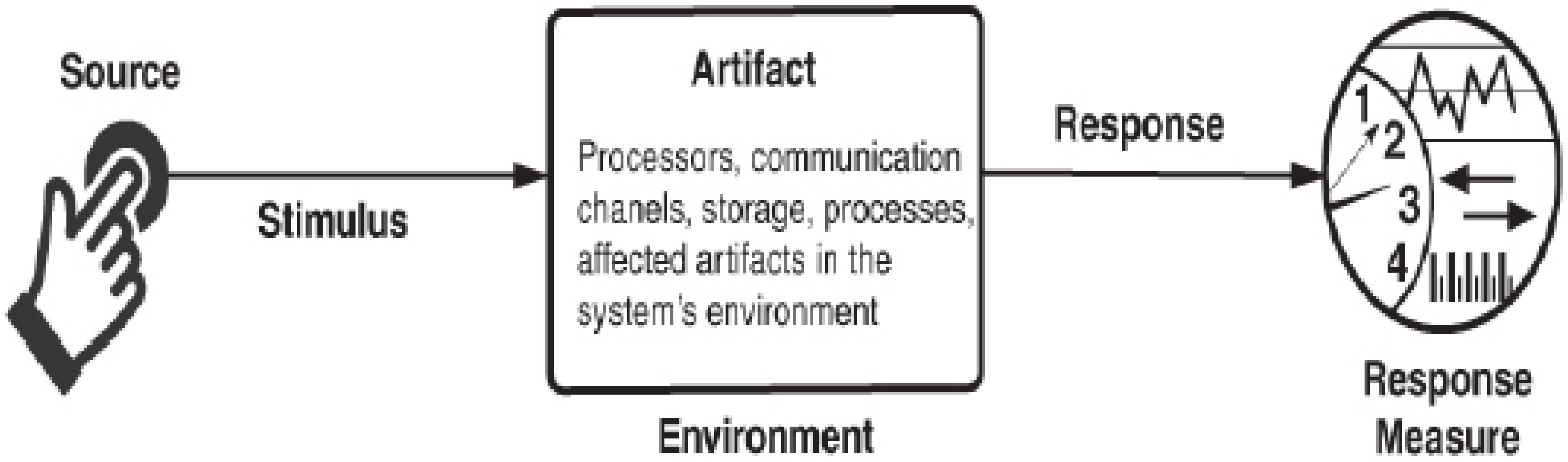
5- Response: The activity undertaken as the result of the arrival of the stimulus.

6- Response measure: The response should be measurable in some fashion so that the requirement can be tested.

General scenario scheme



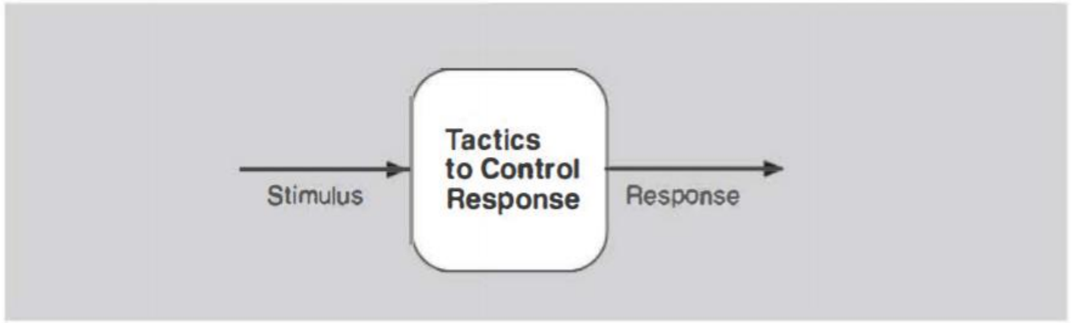
General scenario for availability


Source
Stimulus
Artifact
Processors, communication chanels, storage, processes, affected artifacts in the system's environment
Environment
Response
1
2
3
4
Response Measure

Internal/external: people, hardware, software, physical infrastructure, physical environment	Fault: omission, crash, incorrect timing, incorrect response	Normal operation, startup, shutdown, repair mode, degraded operation, overloaded operation	Prevent the fault from becoming a failure	Time or time interval when the system must be available
			Detect the fault	Availability perentage
			Recover from the fault	Time to detect the fault

Achieving Quality Attributes through Tactics

- Tactics are the techniques that an architect can use to achieve the required quality attributes.
- A tactic is a design decision that influences the achievement of a quality attribute response.
- Tactics directly affect the system's response to some stimulus.
- The focus of a tactic is on a single quality attribute response.
- Within a tactic there is no consideration of tradeoffs. Tradeoffs must be explicitly considered and controlled by the designer.
- In this respect tactics differ from architectural patterns where tradeoffs are considered in the patterns.



Tactics are intended to control responses to stimuli.

- The tactics overlap , and you frequently will have a choice among multiple tactics to improve a particular quality attribute .
- The choice of which tactic to use depends on factors such as tradeoffs among other quality attributes and the cost to implement.
- The tactics are design techniques such as
    - Replication for the availability attribute,
    - Decentralization for scalability
    - A scheduling resources strategy (Shortest-job-first or Round-Robin) for the performance attribute.
    - Use an intermediary (layers , brokers , and proxies, etc.) for the modifiability attribute.