# P02. *Quality Attribute, Verifiability and Quality Attribute Scenario*

**2014**

**Sungwon Kang**

# 2. 품질속성, 검증가능성, 품질속성시나리오

2.1 품질속성

2.2 검증가능성

2.3 품질속성 시나리오

기능은 아키텍처에  명시적으로 영향을 주지 못하고,

제약사항은 너무나 구체적인 방식으로 아키텍처에 영향을 미친다.

요구사항 중에 미묘한 방법으로 아키텍처  결정에 영향을 주는 것이 품질 속성이다.

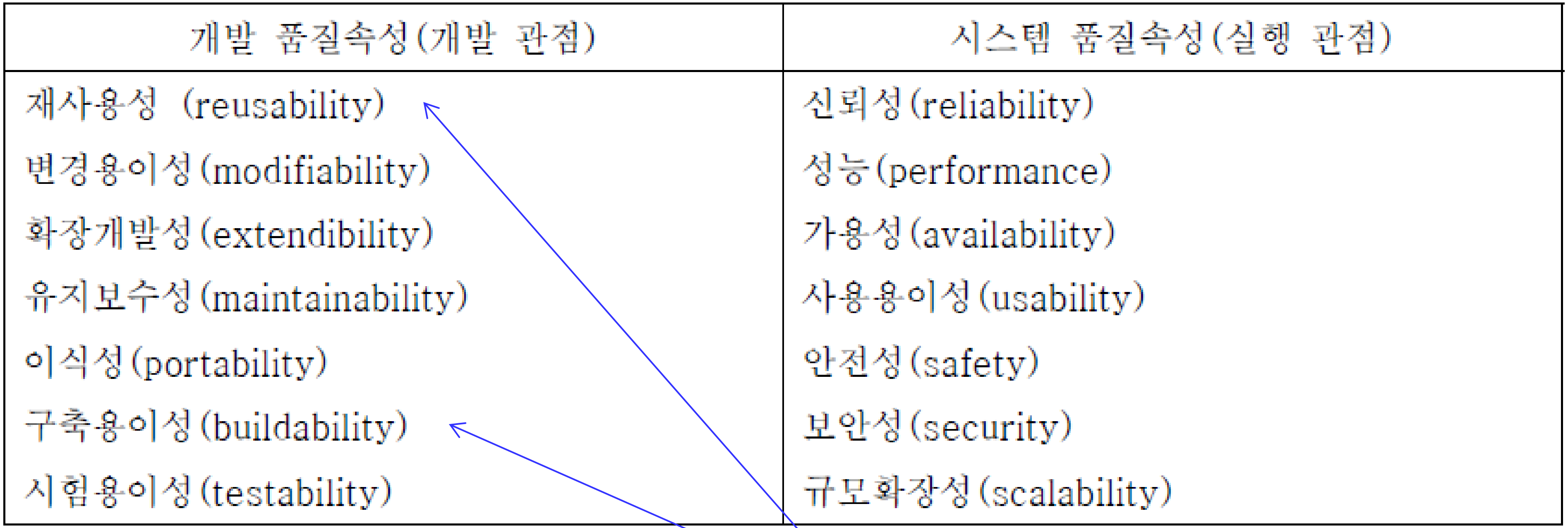
# 2.1 품질속성

- Quality attributes examples :
  - performance
  - time of development
  - modifiability
  - coordination of work teams
- Functionality is largely orthogonal to the structure of the architecture and quality attribute requirements.
- Architectural decisions determine the degree to which a system meets its quality attribute requirements.

Based on [Kazman 04]

# 2.1 품질속성

## 표 2-1. 품질속성의 분류



| 개발 품질속성(개발 관점) | 시스템 품질속성(실행 관점) |
|---|---|
| 재사용성 (reusability) | 신뢰성(reliability) |
| 변경용이성(modifiability) | 성능(performance) |
| 확장개발성(extendibility) | 가용성(availability) |
| 유지보수성(maintainability) | 사용용이성(usability) |
| 이식성(portability) | 안전성(safety) |
| 구축용이성(buildability) | 보안성(security) |
| 시험용이성(testability) | 규모확장성(scalability) |

understandability

Often implicitly adopted

# 2.2 검증가능성

- Quality attribute names by themselves are ***not*** enough.
  - It is *meaningless* to say that the system shall be “modifiable”.
    - Every system is modifiable with respect to *some* set of changes and not with respect to some *other* set of changes.

- For years people have unsuccessfully tried to understand quality attributes by decomposing them.  For example:

  Maintainability
  - Analyzability
  - Changeability
  - Stability
  - Testability

Based on [Kazman 04]

# 2.2 검증가능성

“Einstein told the physicists (and philosophers):
you must first say what you mean by simultaneity,
and this you can only do by showing
how the statement ‘two events are simultaneous’ is verified.
But in so doing you have then also established the meaning fully
and without remainder.” – [Boyd 91] p42

☛ 검증할 수 없는 주장은 의미 있는 주장이 아니다

☛ 의미 있는 요구사항은 측정 가능하여야 한다.

- A stronger form of this principle
  - ☛ Falsifiability Principle - Karl Popper

# 2.3 품질속성 시나리오

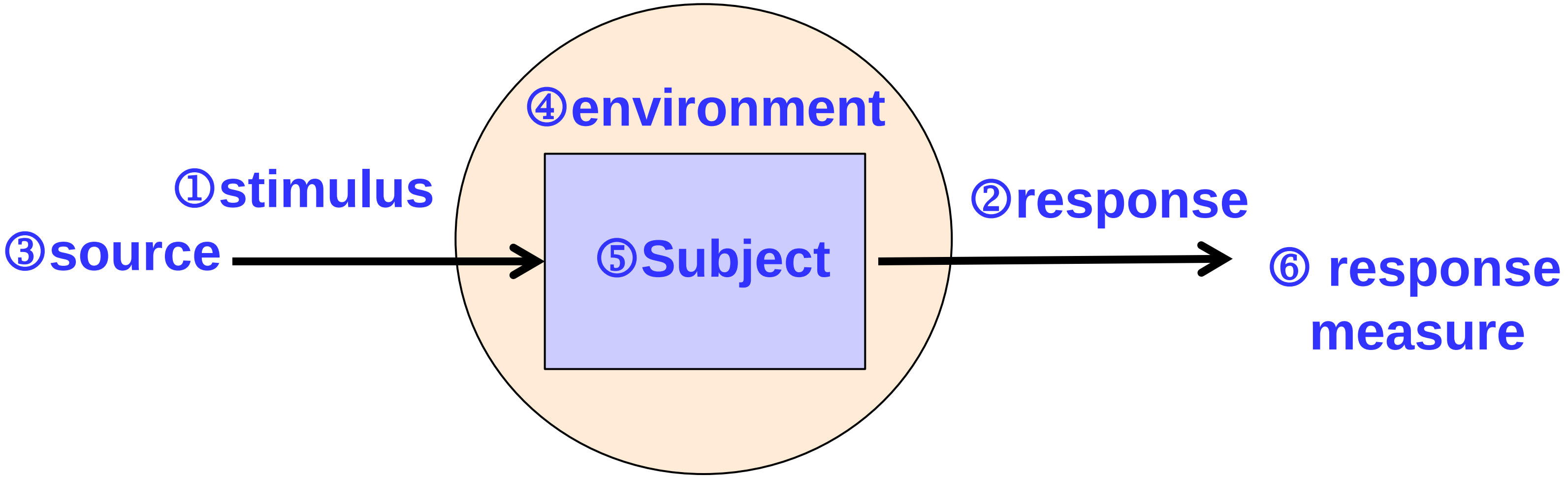
- Then how can you make quality attribute verifiable?
  - ☛ Use *quality attribute scenarios* as a means to characterize quality attributes.

Six parts scenario

- A quality attribute scenario consists of six parts:
  ① Stimulus – condition that effecting the system.
  ② Response – activity as a result of the stimulus.
  ③ Source of stimulus – The entity that generated the stimulus.
  ④ Environment – the condition under which the stimulus occurred.
  ⑤ Artifact stimulated – the artifact that was stimulated.
  ⑥ Response measure – the measure by which the system's response will be evaluated.

Based on [Kazman 04]

# Why the six parts scenario approach works



- To measure quality, need to specify:

  => Quality of what (⑤Subject)?

  => Need (response) ⑥measure

  => Need ②response from the subject

  => Need trigger (or ①stimulus) to the subject

  => There is an initiator (③source) of the stimulus

  => The subject resides surrounded by its ④environment.

# General Scenarios

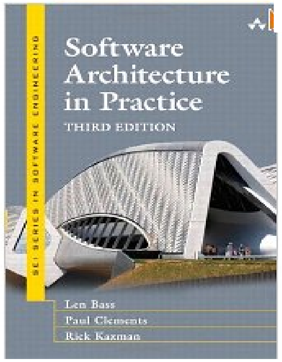
- **What if you need more guidance than the six parts?**

  ☛ *General scenarios* are those scenarios that are system independent.

  – represent quality attribute characterizations

- General scenarios can be used to create concrete scenarios which are specific to a particular system.

- General six-part scenarios exist for:
  1) performance
  2) modifiability
  3) availability
  4) usability
  5) interoperability
  6) testability
  7) security


SEI SERIES IN SOFTWARE ENGINEERING
Software Architecture in Practice
THIRD EDITION
Len Bass
Paul Clements
Rick Kazman

[Bass 13]

# Performance (1/3)

- **Definition**: *Performance* is about timing: how long it takes the system to respond when an event occurs.
- Areas of concern:
  - **Sources of events** vary: interrupts, messages, requests from users, transactions, and so forth.
  - **Arrival rates and patterns** vary: sporadic, periodic, stochastic, or some combination.
  - **Response time**: elapsed time from event arrival to system reaction.

Based on [Kazman 04]

# 성능 설계 전략 (2/3)

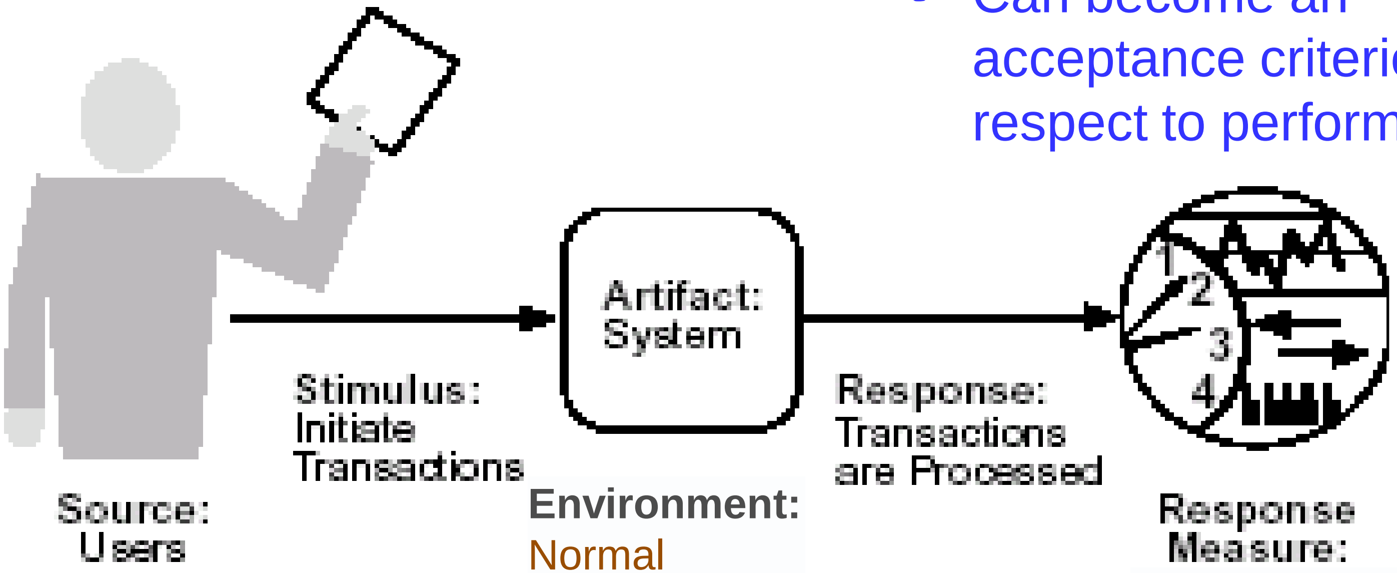
- **General Scenario Considerations**:

| | |
|---|---|
| **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** | Processes events, change level of services |
| **Response Measure** | Latency, deadline, throughput, jitter, miss rate |

# 성능 설계 전략 (3/3)

- 성능품질속성 시나리오 예 - *Users initiate 1000 transactions per minute stochastically under normal operations and these transactions are processed with an average latency of 2 seconds.*

☛ Can become an acceptance criterion with respect to performance



Environment:
Normal
Operation

Average Latency of
2 seconds

# Modifiability (변경용이성) (1/3)

- **Definition**: *Modifiability* is about the cost of change and refers to the ease with which a software system can accommodate changes.
- Areas of concern:
  - Identify **what can change**
    - functions, platforms, hardware, operating systems, middleware, systems it must operate with, protocols, and so forth
    - quality attributes: performance, reliability, future modifiability, and so forth
  - **When** will the change be made and **who** makes it?

Based on [Kazman 04]

# 변경용이성 설계 전략 (2/3)

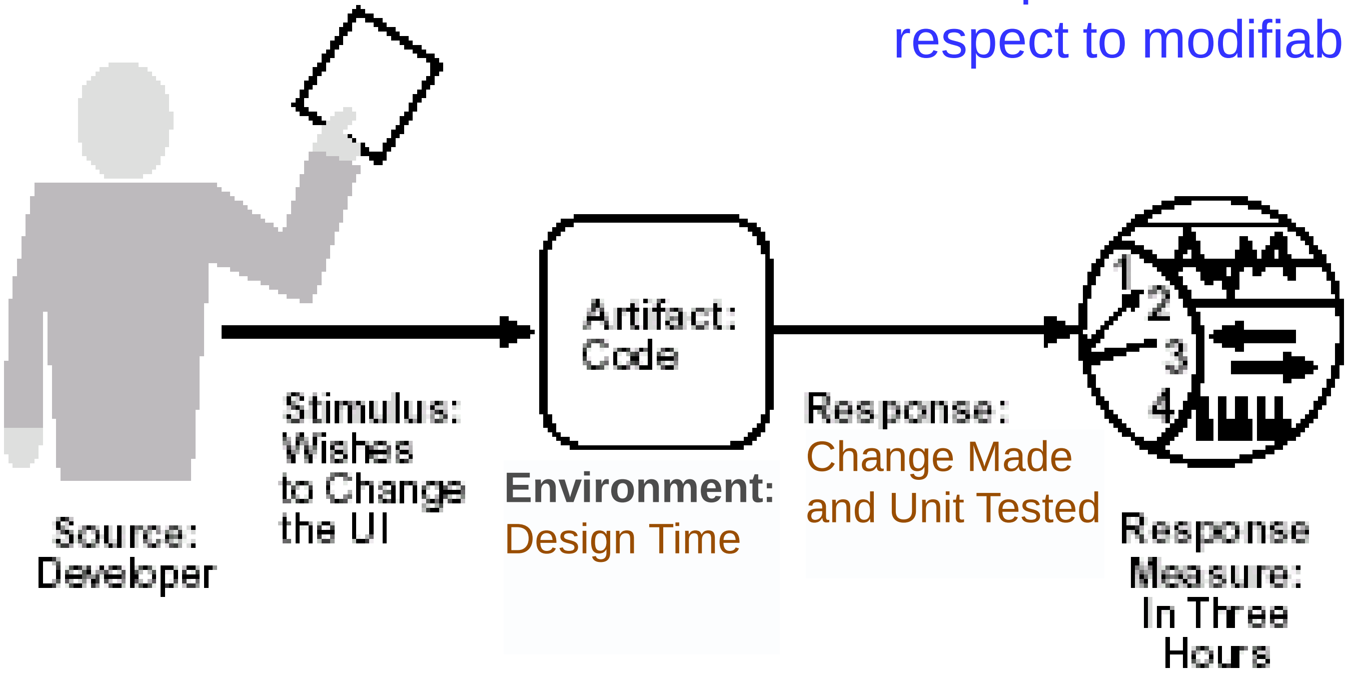
- **General Scenario Considerations**:

| | |
|---|---|
| **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, configuration, ...** |
| **Environment** | Run time, compile time, build time, initiation time, design time |
| **Response** | One or more of the following:<br>- Make modification<br>- Test modification<br>- Deploy modification |
| **Response Measure** | Cost in terms of the following:<br>- Number, size, complexity of affected artifacts<br>- Effort<br>- Calendar time<br>- Money (direct outlay or opportunity cost)<br>- Extent to which this modification affects other functions or quality attributes<br>- New defects introduced |

# 변경용이성 설계 전략 (3/3)

- 변경용이성 품질속성 시나리오 예 - *A developer wishes to change the UI code at design time. The modification is made with no side effects, in three hours*.

☛ Can become an acceptance criterion with respect to modifiability



Source: Developer

Stimulus: Wishes to Change the UI

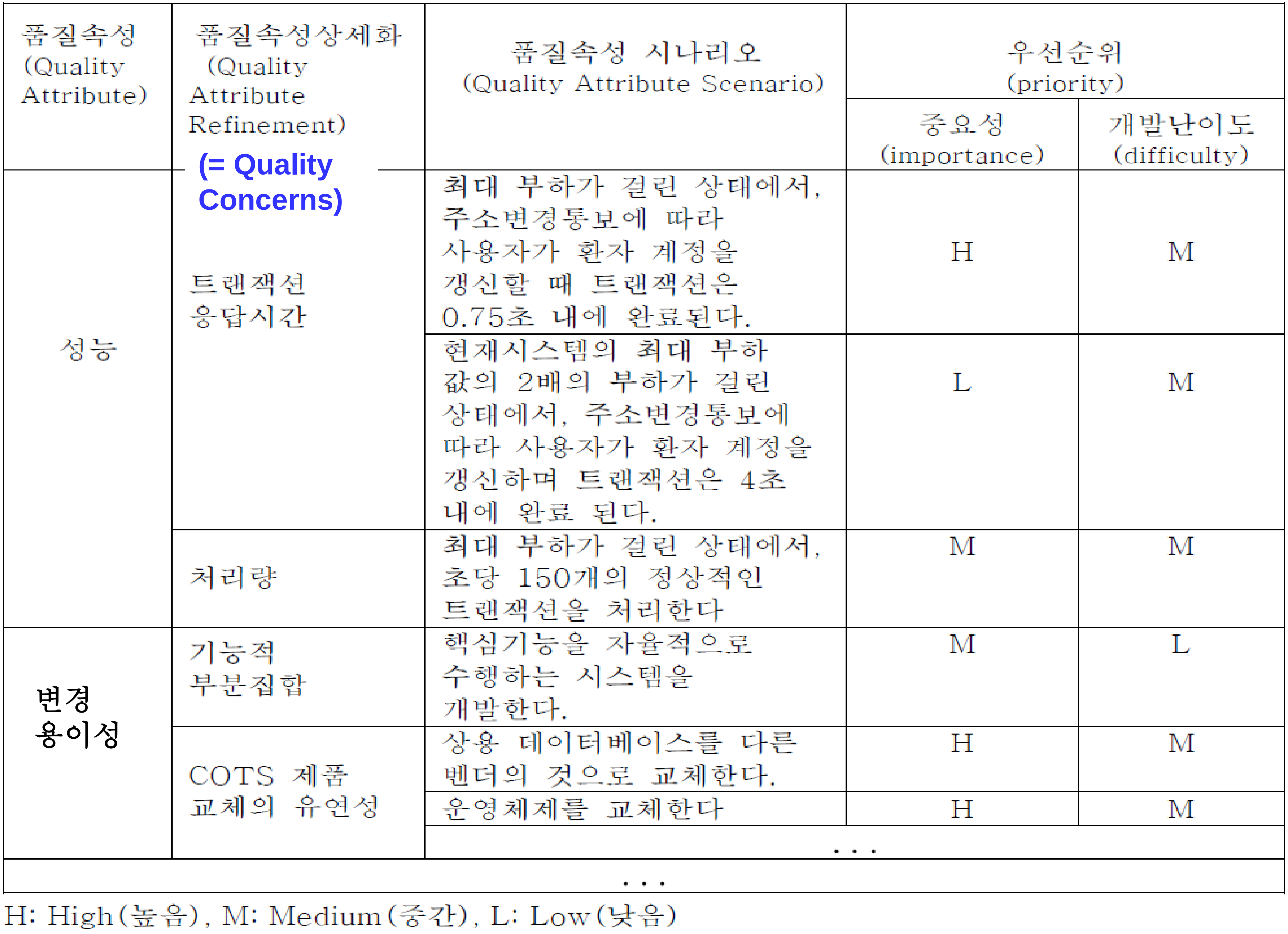
Artifact: Code

Environment: Design Time

Response: Change Made and Unit Tested

Response Measure: In Three Hours

표 2-2. 품질속성 트리 ([Bass 13] p. 307)



| 품질속성<br>(Quality Attribute) | 품질속성상세화<br>(Quality Attribute Refinement)<br>(= Quality Concerns) | 품질속성 시나리오<br>(Quality Attribute Scenario) | 우선순위<br>(priority) | |
|---|---|---|---|---|
| | | | 중요성<br>(importance) | 개발난이도<br>(difficulty) |
| 성능 | 트랜잭션 응답시간 | 최대 부하가 걸린 상태에서, 주소변경통보에 따라 사용자가 환자 계정을 갱신할 때 트랜잭션은 0.75초 내에 완료된다. | H | M |
| | | 현재시스템의 최대 부하 값의 2배의 부하가 걸린 상태에서, 주소변경통보에 따라 사용자가 환자 계정을 갱신하며 트랜잭션은 4초 내에 완료 된다. | L | M |
| | 처리량 | 최대 부하가 걸린 상태에서, 초당 150개의 정상적인 트랜잭션을 처리한다 | M | M |
| 변경 용이성 | 기능적 부분집합 | 핵심기능을 자율적으로 수행하는 시스템을 개발한다. | M | L |
| | COTS 제품 교체의 유연성 | 상용 데이터베이스를 다른 벤더의 것으로 교체한다. | H | M |
| | | 운영체제를 교체한다 | H | M |
| | | . . . | | |
| . . . | | | | |

H: High(높음), M: Medium(중간), L: Low(낮음)

# Lab 1. 요구사항 명세

- **아키텍처 문서의 A와 B를 작성**

Step 1. 요구사항 추출

Step 2. 요구사항의 정제

- 품질속성 시나리오의 도출(QAW)

- 시스템 개요에 시스템 컨텍스트 포함

- 모든 요구사항/제약사항에 id 부여

# Questions?