

아키텍처 요구사항 작성 가이드

목차

- ❖ 아키텍처 요구사항 산출물 개요
- ❖ Project Overview
- ❖ System Overview
- ❖ Architectural Drivers
 - Functional Requirements
 - Quality Requirements
 - Constraints

아키텍처 요구사항 산출물

Chapter	Section
2. Project Overview	2.1 Project Background 2.2 Business Context Diagram 2.3 Stakeholder List 2.4 Business Goal List
3. System Overview	3.1 System Context Diagram 3.2 External Entity List 3.3 External Interface List 3.4 System Feature List
4. Architectural Drivers	4.1 Use Case Model 4.2 Quality Attribute Scenario 4.3 Constraint
5. Top Level Design Description	
6. Component Design Description	
7. Architecture Evaluation	

Basic Concepts

2. Project Overview

Business
Context Diagram

Stakeholder

Business Goal

Why do
stakeholders
need a system?

3. System Overview

System
Context Diagram

**System
Feature**

What features does the system
have to achieve the goal?

4. Architectural Drivers

How can the features be
provided by the system?

Use Case

**QA
Scenario**

Constraint

PROJECT OVERVIEW

2. Project Overview

2.1 Project Background

2.2 Business Context Diagram

2.3 Stakeholder List

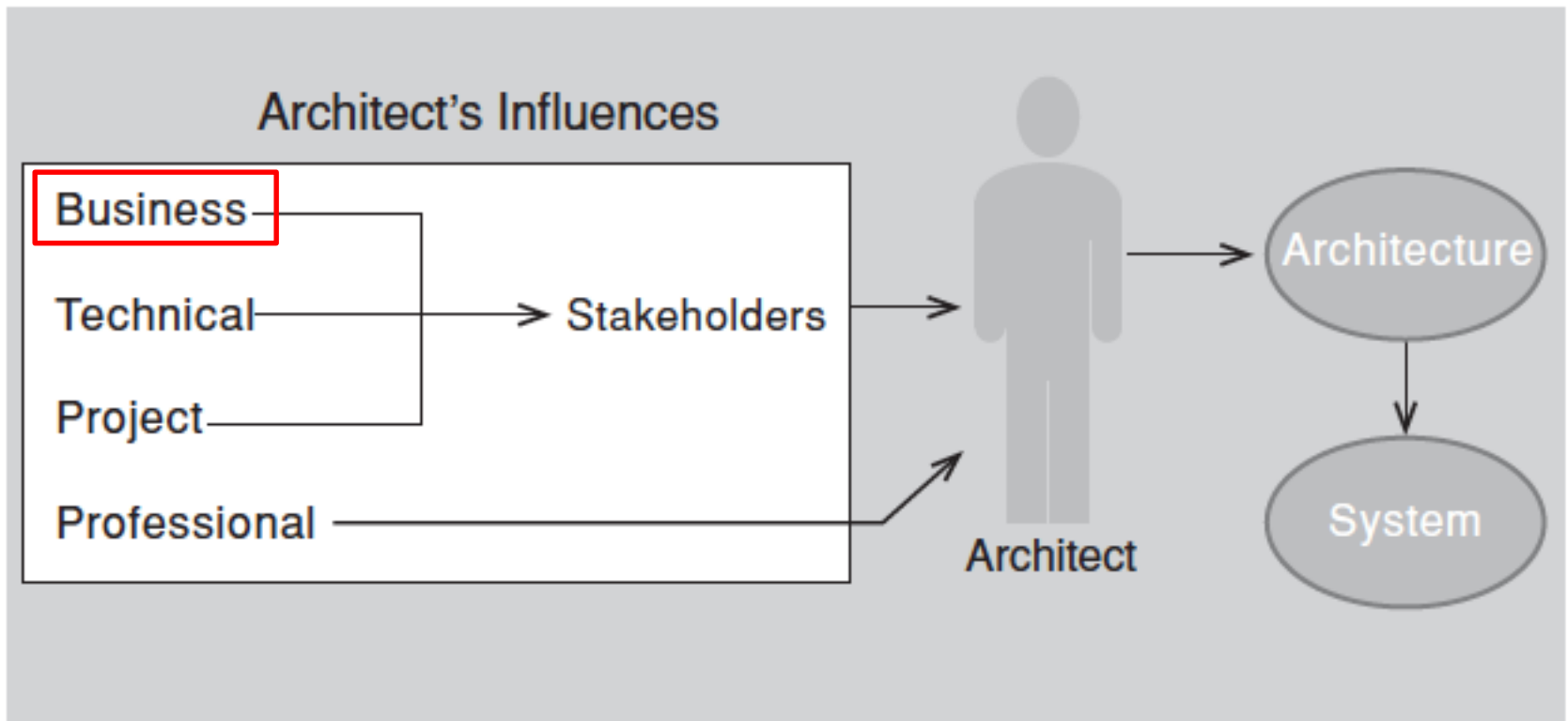
2.4 Business Goal List

Project Background

- ❖ Describe the project and its purpose and scope
- ❖ That is, describe the background information on the domain of your system in order to help stakeholders understand the project and the system.

How is Architecture Influenced?

- ❖ A software architecture is a result of business and social influences, as well as technical ones

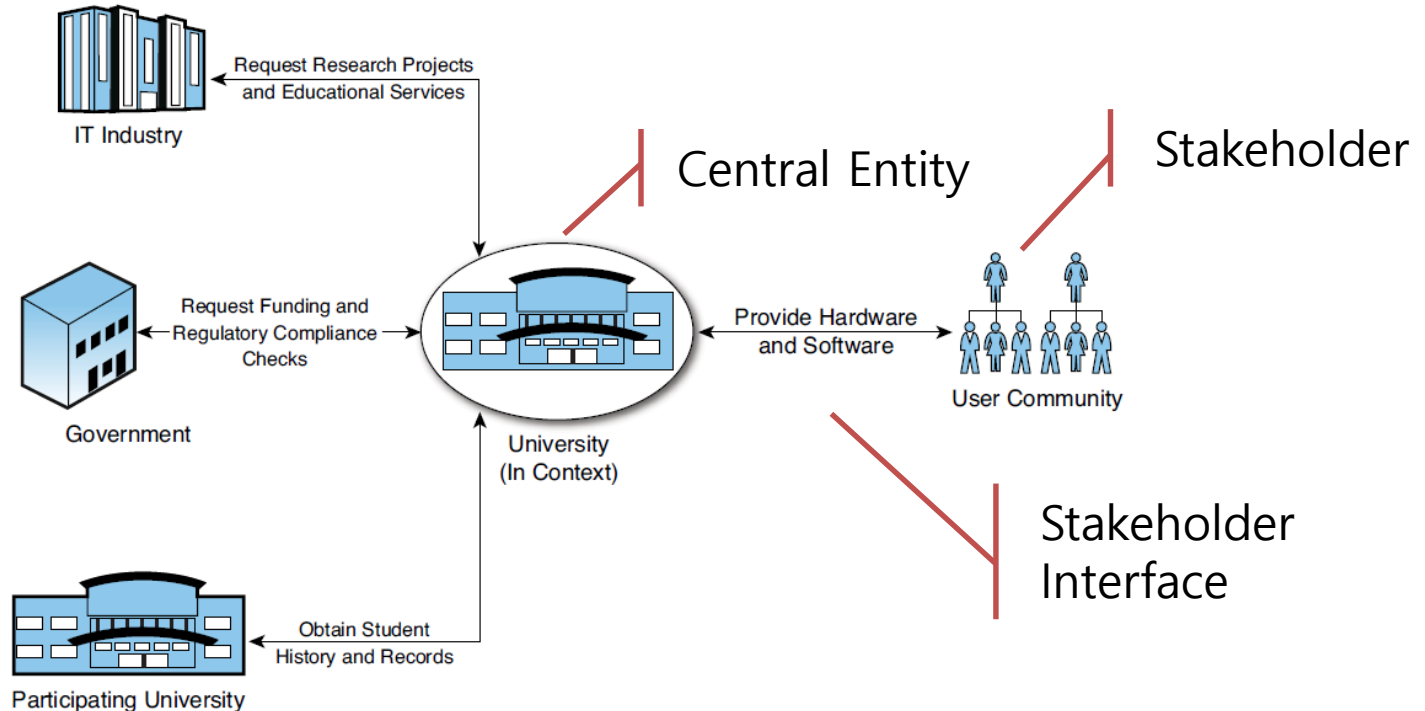


Business Context

- ❖ Systems are created to satisfy the business goals of one or more stakeholders/organizations
- ❖ Customers have their own goals for acquiring a system, usually involving some aspect of making their lives easier or more productive.
- ❖ Other organizations involved in a project's life cycle, such as subcontractors or government regulatory agencies, have their own goals dealing with the system.
- ❖ Development organizations want to make a profit, or capture market, or stay in business.
- ❖ Architects need to understand who are the stakeholders of the system and what their goals are because many of these goals will have a profound influence on the architecture.

Business Context Diagram

- ❖ It provides an organizational-level view of
 - how organizations/stakeholders are related to each other,
 - what information exchange between them
- ❖ When you are building particular software for a university



Practical software architecture: moving from system context to deployment(2015)

Stakeholders

- ❖ A party having a right, share or claim in a system or in its possession of characteristics that meet that party's needs and expectations
- ❖ Identify the stakeholders throughout its life cycle.
 - Users, operators, trainers, maintainers, disposers,
 - Supporters, developers, producers,
 - Acquirer and supplier organizations,
 - Parties responsible for external interfacing entities or enabling systems,
 - Regulatory bodies and members of society.



Stakeholder List

- ❖ A stakeholder is a person, group, or entity with an interest in or concerns about the realization of the architecture.

Category	Description
Acquirers	Oversee the procurement of the system
Developers	Construct and deploy the system from specifications
Testers	Test the system to ensure that it is suitable for use
Assessors	Oversee the system's conformance to standards and legal regulation
Suppliers	Build and/or supply the hardware, software, or infrastructure on which the system will run
System Administrators	Run the system once it has been deployed
Users	Define the system's functionality and ultimately make use of it
Support staff	Provide support to users for the product when it is running
Maintainers	Manage the evolution of the system once it is operational

Software systems architecture-working with stakeholders using viewpoints and perspectives (2005)

Stakeholders:

A Software Product Development Project

A company, an educational software supplier, wants to develop a product that will be used by college lecturers to manage their schedules of classes.

The company has formed a partnership with a local college and has obtained some venture capital funding for the product.

The system will run on PCs and will be inexpensive and easy to operate.

- Acquirers include senior management and product managers at the Company, the educational partner, and representatives from the venture capitalists.
- Developers and maintainers are product development staff from the Company
- Users are college lecturers and administrative staff
- Support staff might be provided by the Company and/or colleges that buy the product.

Stakeholders: A Partnered Development

A company, an established financial organization, wishes to expand its presence on the Internet with the ability to market a range of financial services to members of the public.

These services are aimed at residents of the country where the company is based, as well as international customers.

The company plans to contract out the development and operation of the system to an established Web developer.

- Acquirers include senior managers who will authorize funding for the project.
- Users include members of the public, who will access the Web site along with internal administrative staff, who will carry out its back-office functions.
- Developers and system administrators are staff from the Web development company.
- Assessors include the Company's internal accounting and legal staff, as well as external financial regulators from any country in which Company wants to trade.
- Support staff are provided by the Company and/or the Web development company.

Business Goals

- ❖ No organization builds a system without a reason
 - The organization's leaders want to further the mission and ambitions of their organization and themselves.
 - Business goals are synonymous with mission objectives

- ❖ Business goals are the primary influencing factors on the architecture
 - Business goal should be captured explicitly because they often imply ASRs that would otherwise go undetected until it is too late or too expensive to address them
 - For example, "What are our ambitions about market share for this product, and how could the architecture contribute to meeting them?"

List of Business Goals

Category	Goal Examples
Managing <u>product's quality and reputation</u>	System helps <ul style="list-style-type: none">• Improve branding• Reduce recalls• Support certain types of users• Improve quality and testing support and strategies
Meeting <u>financial objectives</u>	System meets financial objectives through <ul style="list-style-type: none">• Revenue generation• Business process efficiency• Reduced training costs• Higher shareholder dividends
<u>Organization's growth and continuity</u>	System promotes growth and continuity through <ul style="list-style-type: none">• Market share increase• Product line creation and success• International sales• Long-term business sustenance

List of Business Goals

Category	Goal Examples
Meeting responsibility to employees	System fulfills responsibilities to employee through <ul style="list-style-type: none">• Improved <u>operator safety and reduced workload</u>• Opportunity for learning new development skills
Meeting responsibility to society	System fulfills responsibilities to a society through <ul style="list-style-type: none">• <u>Compliance with laws and regulations</u>, particularly those related to ethics, safety, security, and privacy• Green computing
Meeting responsibility to country	System fulfills responsibilities to a country through <ul style="list-style-type: none">• <u>Compliance with export controls</u>• <u>Regulatory conformance</u>

Business Goals

❖ 각 이해관계자 별로 시스템 개발의 목표/필요성이 제시되어야 함

- 판매: 고객 만족도 개선, 시장 점유율 확대, 매출 증대
- 운영: 운영 비용 절감
- 사용: 업무 효율성 개선
- 검사/인증: 검사/인증 편의성, 검사/인증 비용 절감

- 시스템 개발: 개발 편의성 개선, 개발 비용 절감
- 시스템 유지보수: 유지보수 편의성 개선, 비용 절감

Expressing Business Goals

- ❖ All business goals are expressed clearly, in a consistent fashion, and contain sufficient information to enable their shared understanding by relevant stakeholders
- ❖ Examples
 - The project manager has the goal that his family's stock in the company will rise by 5 %
 - The portfolio manager has the goal that the company will make the portfolio 30 % more profitable.
 - The project manager has the goal that customer satisfaction will rise by 10 %

<goal-subject> has the goal that <goal-object> achieve <goal>

- in the context of <environment> and
- will be satisfied if <goal-measure>

SYSTEM OVERVIEW

3. System Overview

3 System Overview

3.1 System Context Diagram

3.2 External Entity List

3.3 External Interface List

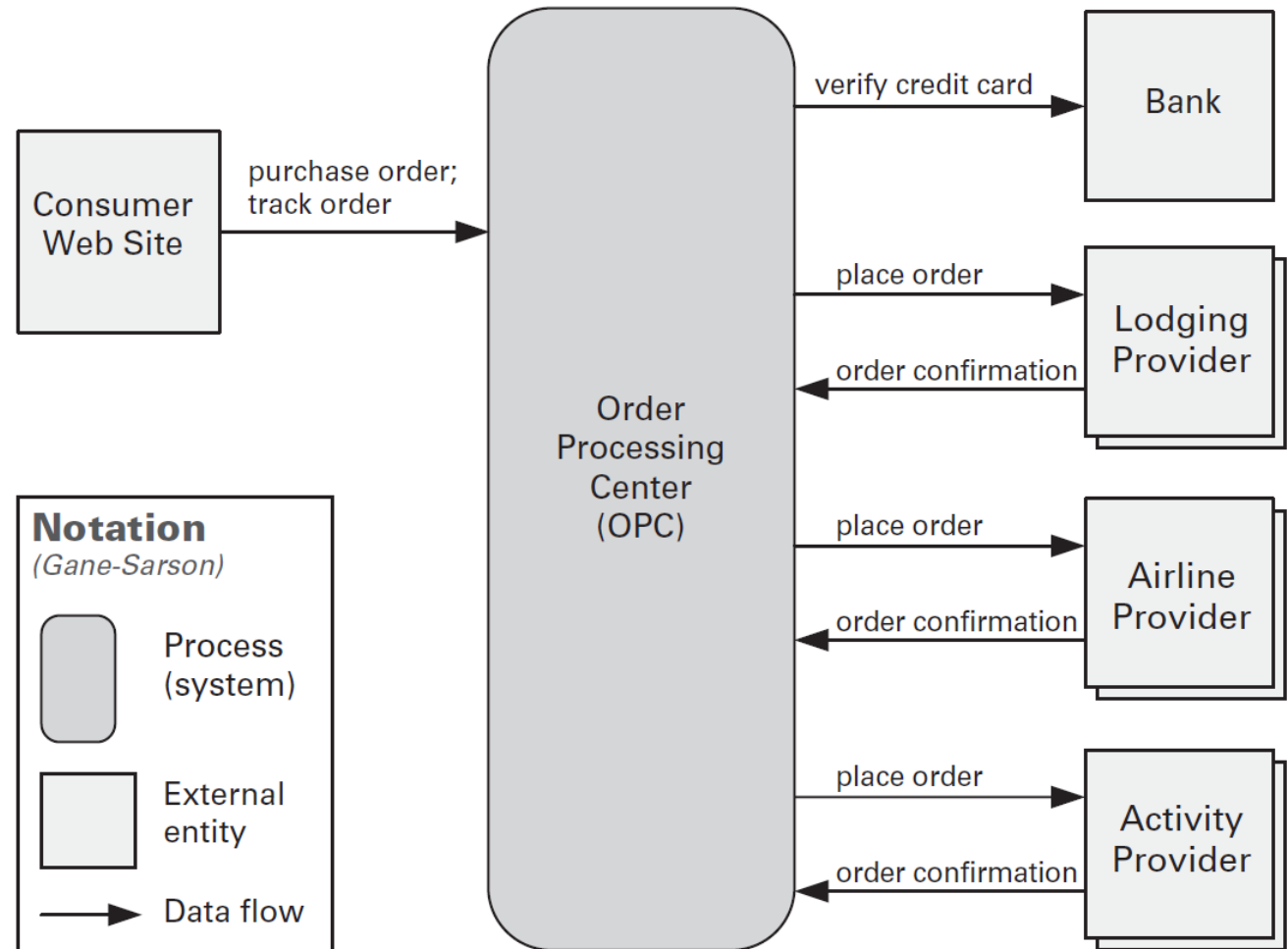
3.4 System Feature List

System Context Model

- ❖ Within the System Context, the IT System is brought into focus to implement the business goals
- ❖ System context defines
 - What is in scope → **System Features**
 - What is out of scope → **External Entities**
 - How the system related to its environment → **External Interfaces**
- ❖ A good context model is an essential part of an effective architecture document

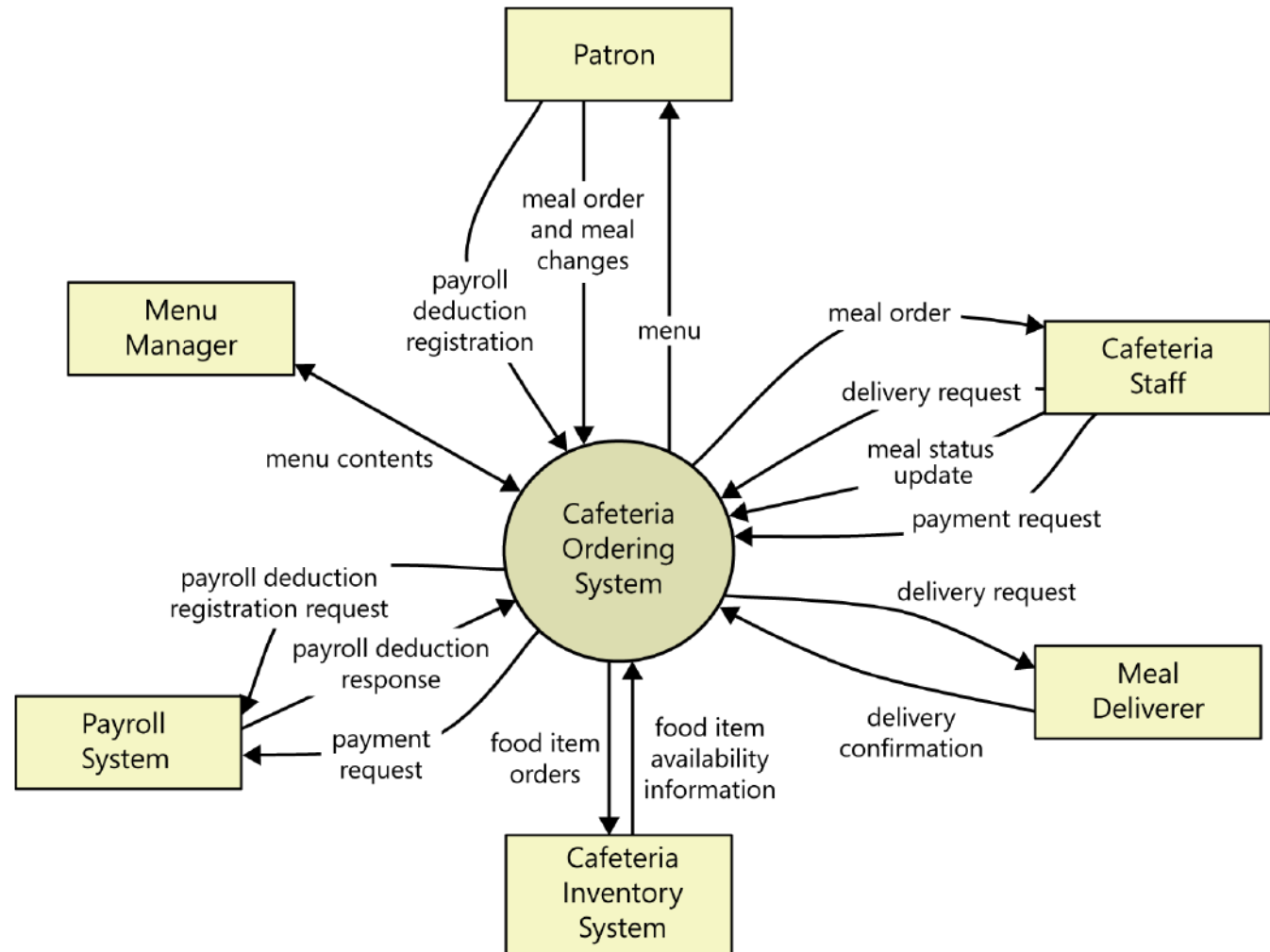
System Context Diagram

❖ Order Processing Center



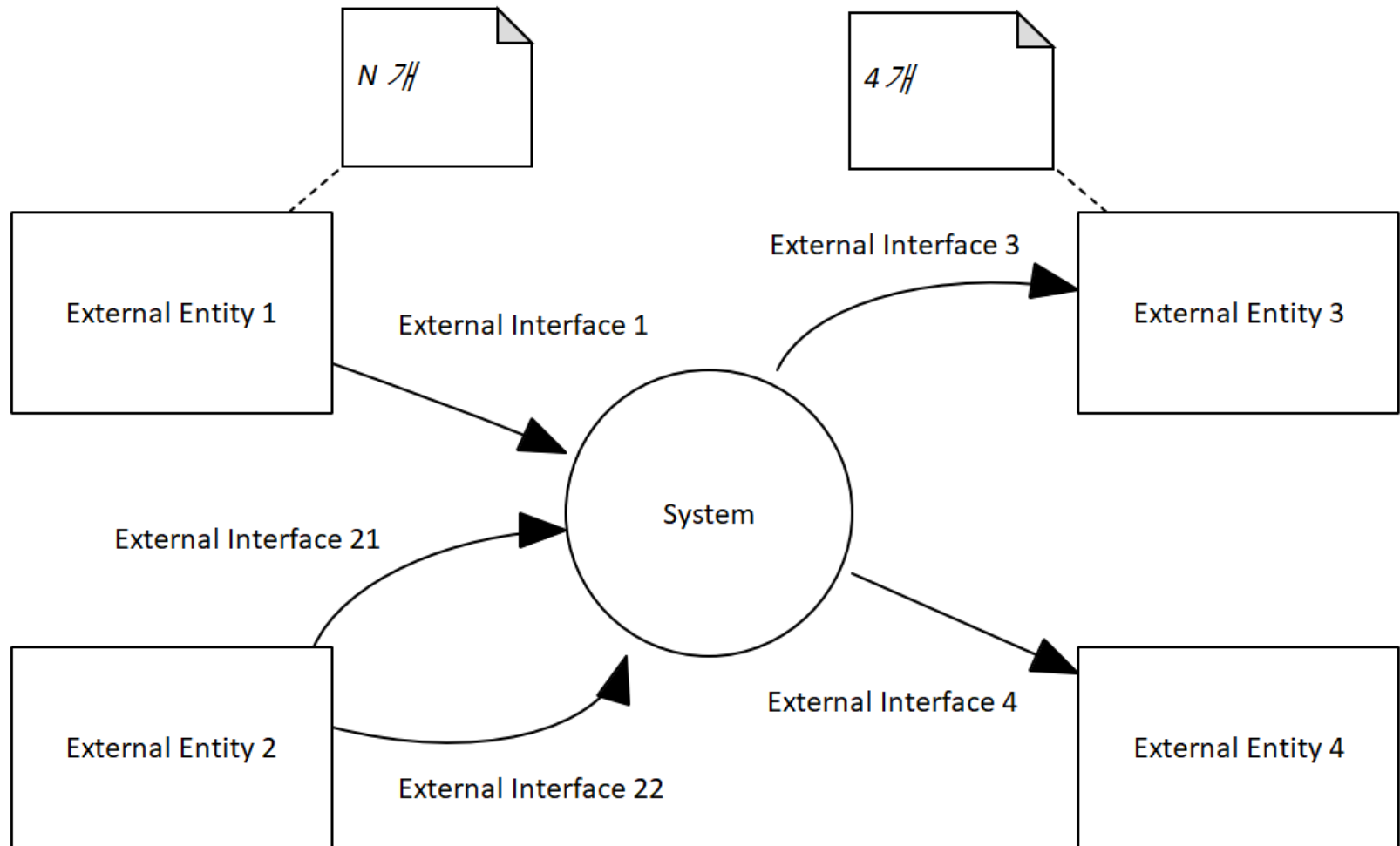
System Context Diagram

❖ Cafeteria Ordering System



Software requirements, 3rd edition(2013)

3.1 System Context Diagram



3.2 External Entity

- ❖ An external entity is any person, system, or device with which the system directly interacts

Kind	Description
User	User, a class of user, or some other person or role
External system	Another system that runs <u>in another organization</u>
Internal system	Another system that runs <u>in the same organization</u> as the system being modeled
Gateway component	Gateway or other implementation component that has the effect of <u>hiding other systems</u> (internal or external)
Data store	<u>Data store that is external to the system</u> (e.g., a shared database, data warehouse)

Nature and Characteristics of External Entities

- ❖ The quality of external entities, such as reliability, availability, or performance may significantly affect the architecture of the system

A travel booking system exchanges information with many other systems located around the world. Some of these systems may be only intermittently available, because of time zone differences or because they are more liable to failure

The travel system's interfaces with external systems will therefore need to be carefully designed for **reliable** operation

- All failed interactions should be automatically retried a configurable number of times
- These retry attempts should be logged to a database so that operational staff can monitor trends.
- It should be possible to restart very large transfers that fail partway through from the point of failure rather than having to retransmit the whole file.

3.3 External Interface

- ❖ For each external entity, all interfaces between it and the system should be identified

Kind	Description
Data provider or consumer	The external entity <u>supplies data</u> directly to the system or receives data directly from it
Event provider or consumer	The external entity <u>publishes events that this system wishes to be notified of</u> , or this system publishes events that the external entity wishes to be notified
Service provider or consumer	The external entity is <u>requested to perform some action by the system</u> or requests some action of the system, and the system may return data and/or status information in response to the request

Nature and Characteristics of External Interfaces

- ❖ The interface will have a significant effect on the architecture of the system. (e.g., reliability of the interface, volume and frequency of data/event/service)

Kind	Description
Data provider or consumer	The content, scope, and meaning of the data to be transferred
Event provider or consumer	Events of interest, their meaning and content, and the volume and timing of their occurrence
Service provider or consumer	<u>Syntax of the request</u> (name and any parameters), the <u>actions to be taken</u> , <u>any data to be returned</u> , any ack, status, or error information that may be returned, any exception actions to be taken by either side

System Context Diagram: Guidelines

- ❖ A pure context diagram does not disclose any architecture detail about the system
 - it just appears as an undecomposed block
 - although in practice, context diagrams may show some internal structure of the system being put in context.

- ❖ Interfaces
 - Context diagrams do not show any temporal information, such as order of interactions or data flow.
 - They do not show the conditions under which data is transferred, stimuli fired, messages transmitted, and so on.
 - Each interface needs to be “assigned” to one of the system’s architecture elements.

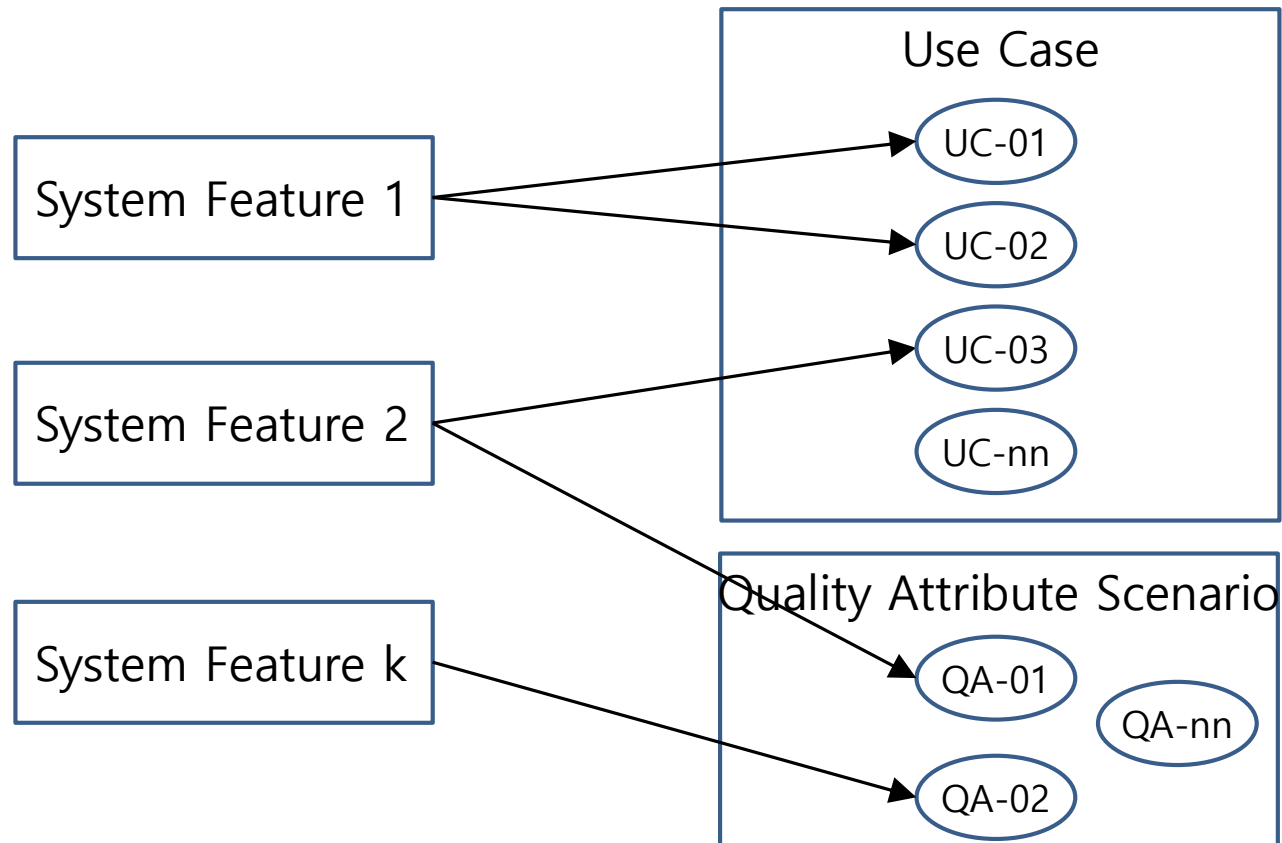
3.4 System Feature

- ❖ Business goals are refined into system features by which the goals are realized.

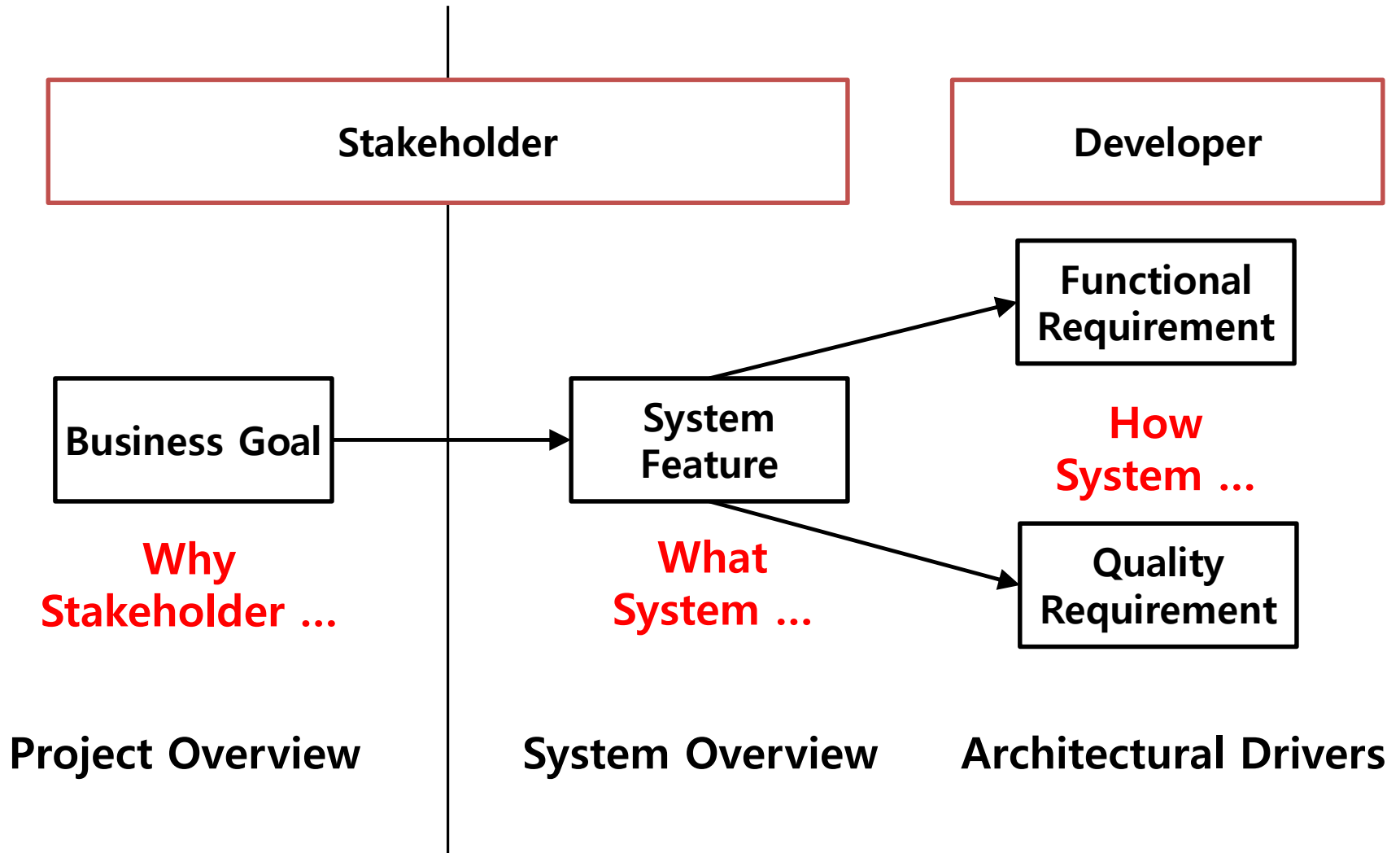
Business Goal	System Feature
Expand by entering new and emerging geographic markets	Support international languages
	Comply with regulations that have an impact on life-critical systems such as fire alarms
Open new sales channels in the form of VARs	Support hardware devices from different manufacturers
	Support conversions of nonstandard units used by the different hardware devices

System Feature vs System Requirements

- ❖ You define high-level requirements that the system need to provide system features.
- ❖ System requirements(UC/QA) are derived from system features



Business Goal vs Feature vs Requirement



Goal – Feature – Requirement: 예

- ❖ 건물주: 엘리베이터 관련 안전사고가 1년에 1건 이내로 발생하여 건물의 명성이 실추되지 않는 것을 목표로 가지고 있다.
 - Business Goal: 명성 실추를 유발하는 사고 10% 감소
 - System Feature: 안전사고 방지
 - Requirement: 안전 사고 장비에 대한 사전 점검 시점 안내, 이상 상황 실시간 모니터링

- ❖ 엘리베이터 운영 업체: 엘리베이터 시스템의 관리를 통해 최적의 정기 점검 주기를 찾아서 점검에 필요한 인건비를 10% 절감하는 것을 목표로 가지고 있다.
 - Business Goal: 인건비 10% 절감
 - System Feature: 최적의 점검 주기 제시
 - Requirement: 장비 중요도 별 점검 주기 결정

Goal – Feature – Requirement: 예

시스템 관리자	시스템 관리자는 웹 인터페이스 통해 엘리베이터 시스템을 관리할 수 있어야 한다	R
시스템 관리자	엘리베이터 사용자는 엘리베이터가 Idle 시 지상 1층(출퇴근시간) 또는 요청 예상 층(그 외 시간)으로 엘리베이터가 사전에 이동되길 원한다	R
시스템 관리자	시스템 관리자는 엘리베이터의 운행량을 고려한 엘리베이터의 유지보수활동이 필요한 시점을 안내 받길 원한다	F
엘리베이터 사용자	엘리베이터 사용자는 층에서 버튼을 누른 후 엘리베이터의 평균 대기 시간이 3분이내길 원한다	F
건물 소유주	건물 소유주는 각 엘리베이터의 전기요금이 월 10,000원 이내이길 원한다	F
건물 소유주	건물 소유주는 각 엘리베이터의 유지보수 비용이 월 200,000원 이내이길 원한다	F

ARCHITECTURAL DRIVERS

4. Architectural Driver

4.1 Use Case Model

4.1.1 Use Case Diagram

4.1.2 Actor List

4.1.3 Use Case List

4.1.4 *UC-01 Title* Description

4.1.5 *UC-02 Title* Description

...

4.2 Quality Attribute Scenario

4.2.1 QA Scenario List

4.2.2 *QA-01 Title* Scenario

4.2.3 *QA-02 Title* Scenario

...

4.3 Constraint

4.3.1 Business Constraint List

4.3.2 Technical Constraint List

What is Architectural Driver?

- ❖ To begin designing the architecture of a software-intensive system, architects need the key requirements that are most likely to affect the fundamental structure of the implementation.
- ❖ These key requirements will determine the structure of the system; they are the architectural drivers.
- ❖ Architectural drivers are not all of the requirements for a system, but they are an early attempt to identify and capture those requirements that are most influential to the architect making early design decisions. → Architecturally Significant Requirements(ASR)
- ❖ Uncovering the architectural drivers as early as possible is critical because these early architectural decisions are binding for the lifetime of a system

Architectural Drivers

- ❖ Architectural drivers consist of
 - Architecturally significant functional requirements,
 - Architecturally significant quality requirements,
 - Technical constraints and business constraints
- ❖ Each of these exerts forces on the architect and influences the early design decisions that the architect makes

Driver	Representation	Description
AS FR	Use Case Model	High-level functional requirements from the user perspective
AS QR	QA Scenario	Properties that the system must possess, such as availability, performance, and so forth
Constraints		Fixed premade decisions that are in place before development begins

AS: Architecturally Significant

Capturing Architecturally Significant Requirements

- ❖ One of the most powerful techniques is to define and apply scenarios to your architecture
- ❖ An architectural scenario is a concise description of a situation along with a definition of the response required of the system
- ❖ Just as with requirements, scenarios can be divided into the group of functional scenarios and the group of system quality attribute (or nonfunctional) scenarios.

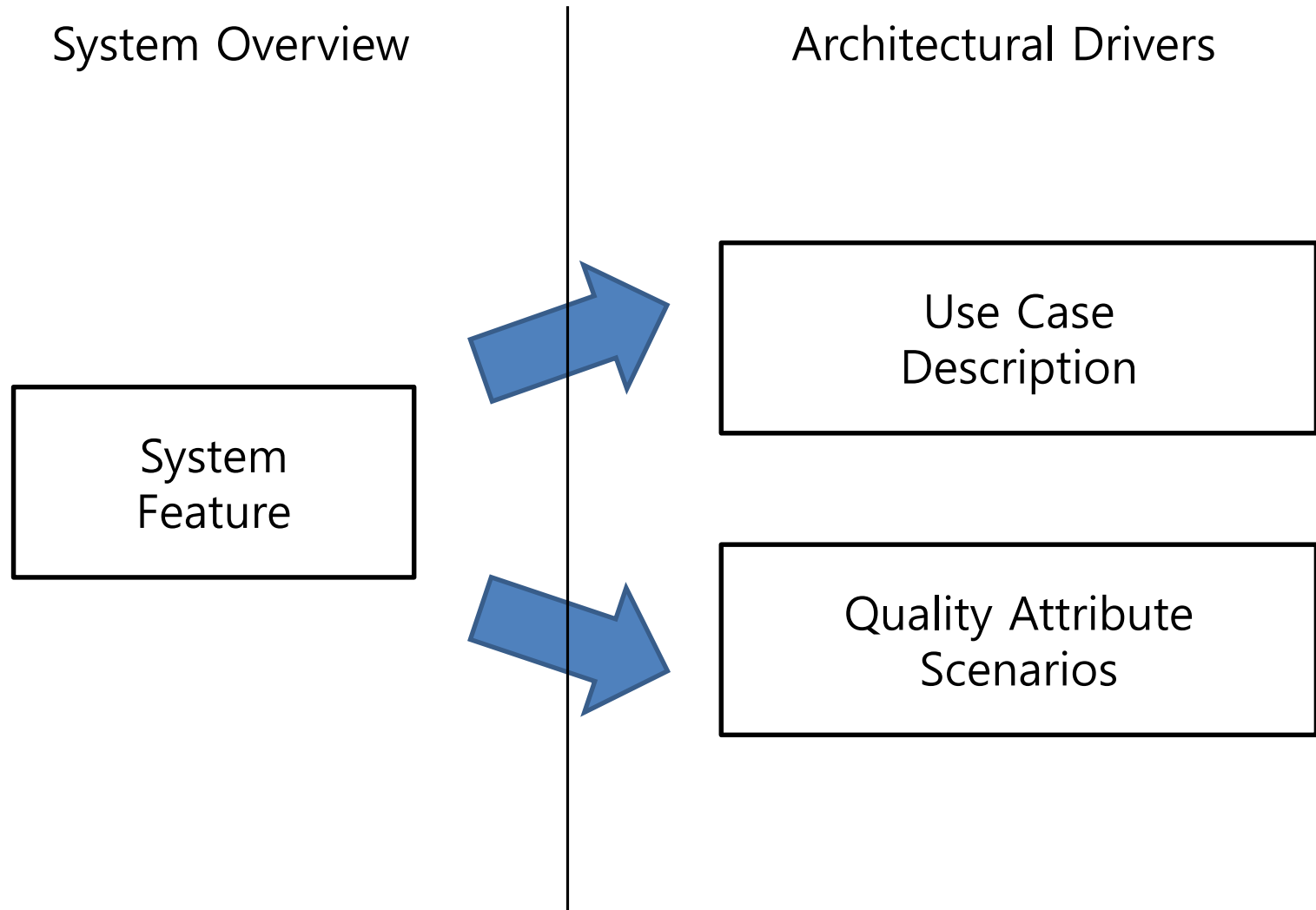
Scenarios

- ❖ Functional scenarios are nearly always defined in terms of a sequence of external events that the system must respond to in a particular way. ➔ Use Case Scenario
- ❖ In contrast, quality scenarios are defined in terms of how the system should react to a change in its environment in order to exhibit one or more quality properties.
- ❖ There are as many types of system quality scenarios as there are quality properties the ability of the system
 - to be modified to provide a new function, ➔ modifiability
 - to cope with a particular type of peak load, ➔ performance
 - to protect critical information even if some of the security infrastructure is compromised, and so on. ➔ security

Uses for Scenarios

- ❖ You can use scenarios in a number of ways within the architecture definition process.
 - Providing input to architecture definition
 - Evaluating the architecture: Scenarios are a primary input into almost any process of architectural evaluation
 - Communicating with stakeholders: discussion of a scenario and how the system can meet the situation described is a very useful vehicle for communicating with all types of stakeholders.
 - Finding missing requirements: Another benefit of creating scenarios is that they often reveal what is missing as well as the suitability of what already exists
 - Driving the testing process: Scenarios help highlight the things that are important to your stakeholders, thus providing a tremendously useful guide for where to focus testing activity

System Feature vs Architectural Drivers



Prioritizing Architectural Drivers

- ❖ Once each driver(UC or QA) is identified, its priority can be analyzed based on the importance and difficulty
- ❖ Importance: For business value,
 - High: must-have requirement,
 - Medium: important but would not lead to project failure were it omitted.
 - Low: a nice requirement to have but not something worth much effort
- ❖ Difficulty: For architectural difficulty
 - High: meeting the architecture driver will be highly difficult
 - Medium: meeting the architecture driver will be somewhat difficult
 - Low: meeting the architecture driver will be not difficult

Software architecture in practice, 3rd edition(2012)

ARCHITECTURAL DRIVERS USE CASE MODEL

Use Case Model

- ❖ Use cases describe functional requirements of the system.
- ❖ Each use case defines a set of interactions between actors and the system we intend to design.
- ❖ Use cases model the interaction between the system and the actors at a coarse-grained level of abstraction.
- ❖ This can help in establishing the scope of the project and in guiding architects through the initial decomposition and structuring of the system

Use Case Model

- ❖ Use cases can help establish system context or scope
 - That is, what is inside the design space and what is outside of the design space.
 - While functionality has less influence on structure, establishing a clear context is an essential first step in design.

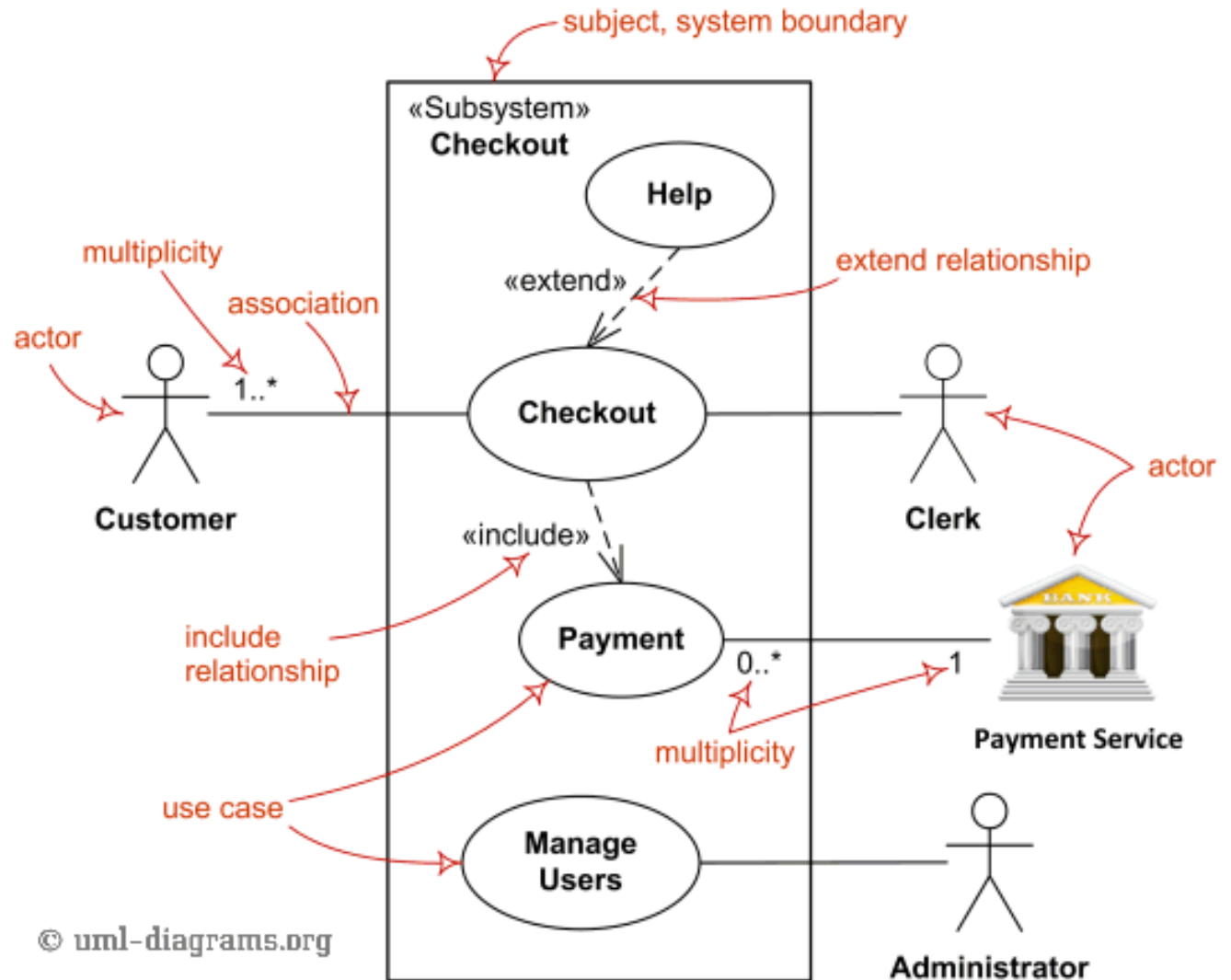
- ❖ Use case models serve
 - as a communication vehicle and encourage dialog
 - between technical and nontechnical stakeholders.

- ❖ Initially in use cases,
 - the system is treated as a black box and
 - the use case describes the requirement—what is needed—not how the system delivers the services.

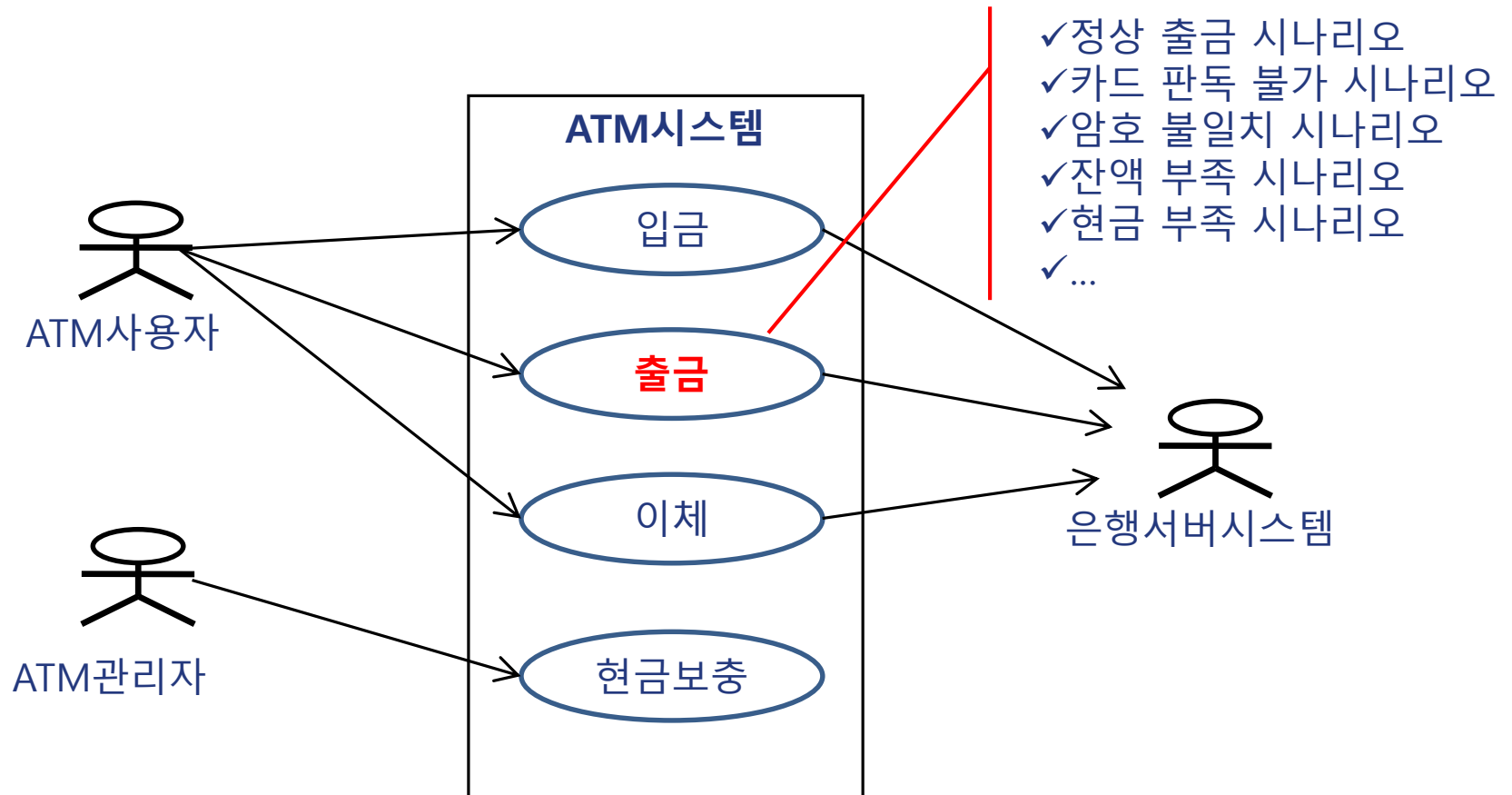
Architecting software intensive systems-A practitioner's guide(2008)

Use Case Model

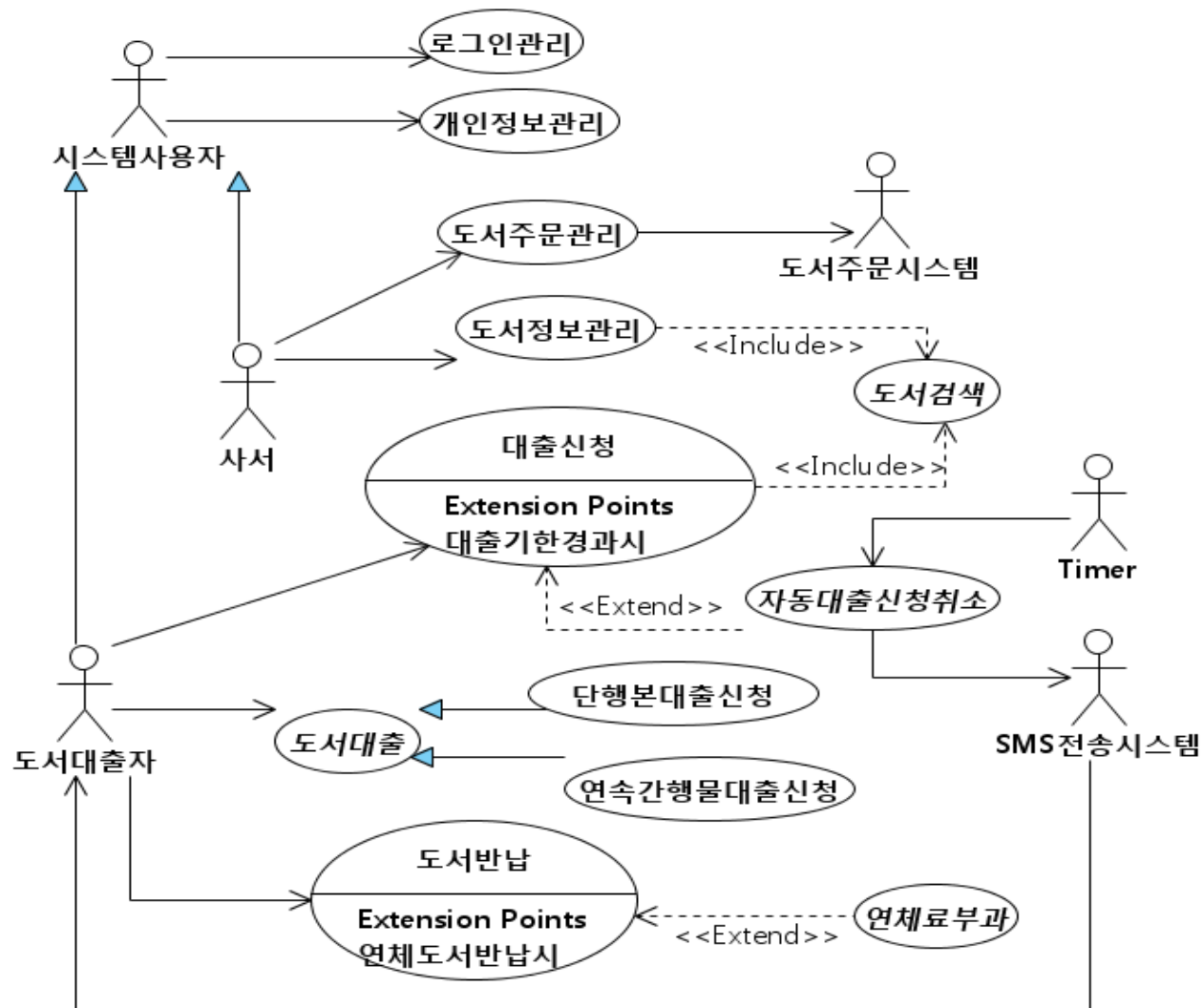
❖ Use Case Diagram



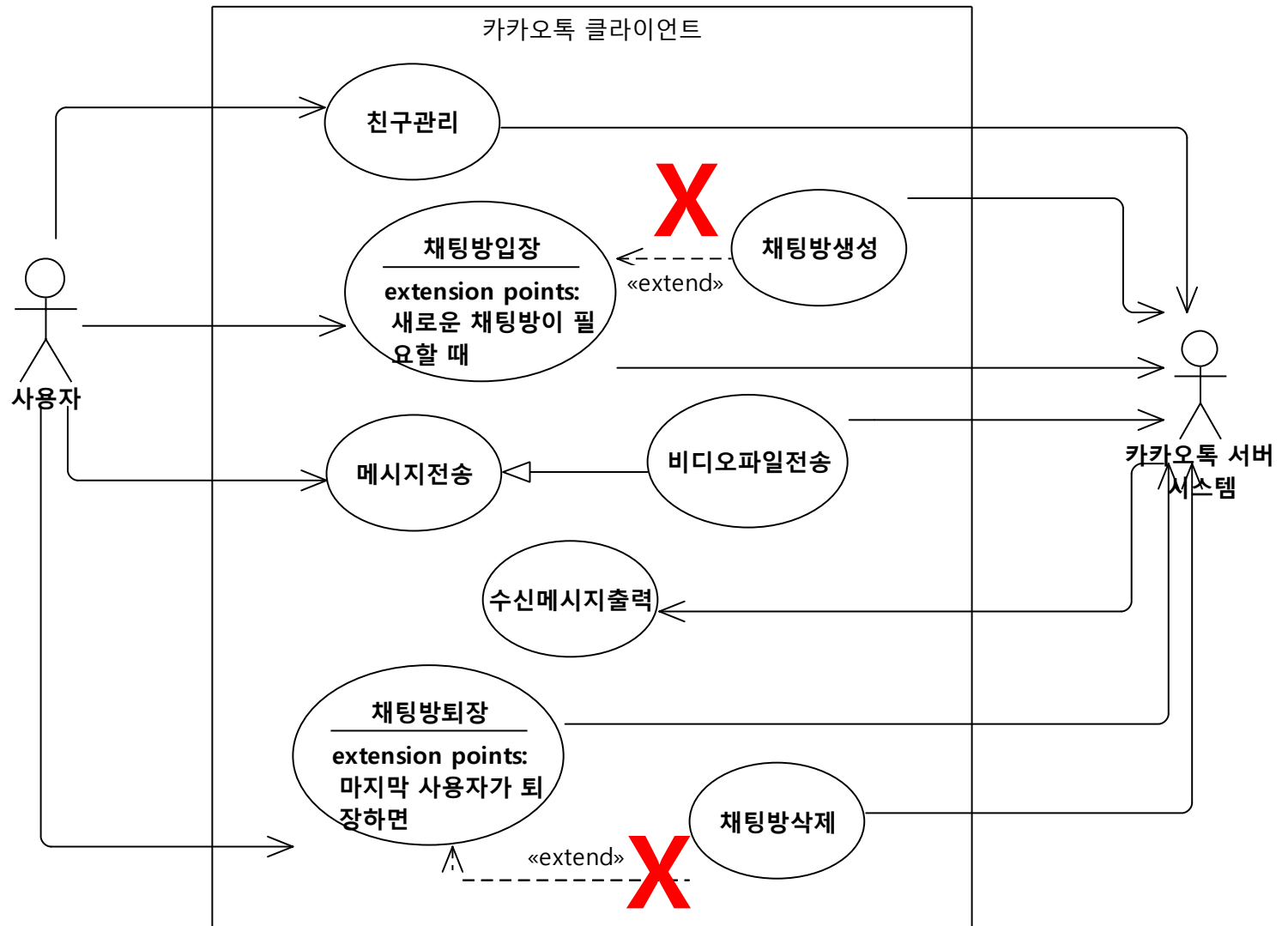
Use Case Model: Example



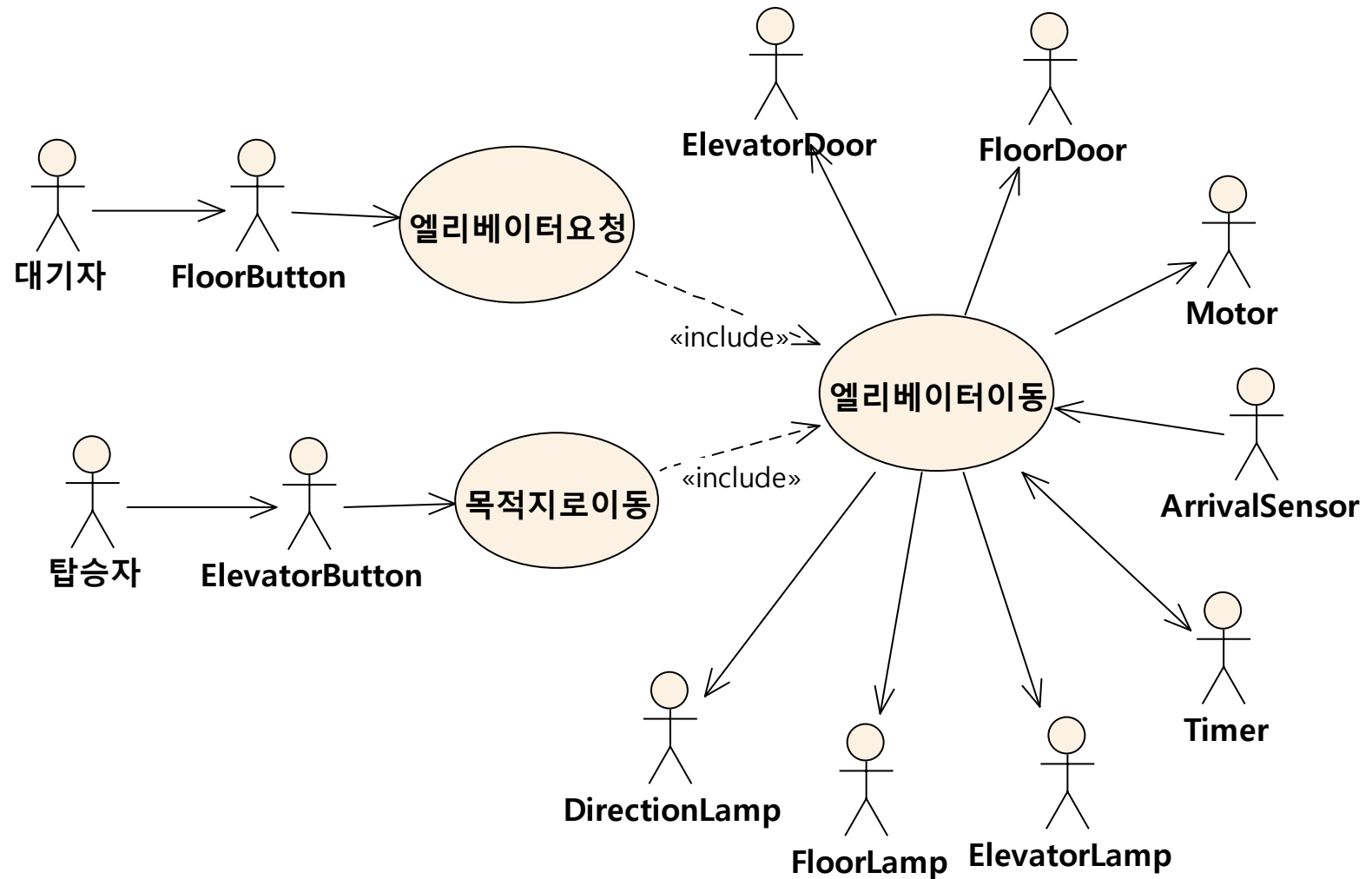
Use Case Model: Example



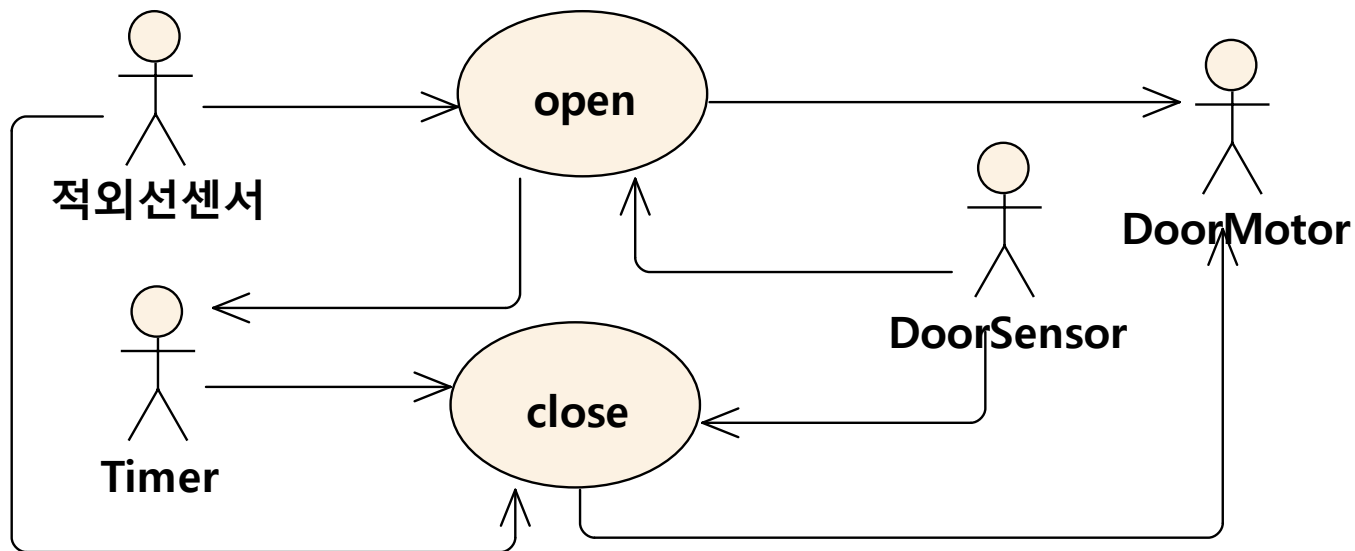
Use Case Model: Example



Use Case Model: Example



Use Case Model: Example



Guidelines for Use Case Model

- ❖ Use cases are generally written in narrative language that can be understood by all stakeholders.
- ❖ Use cases are not models for functional decomposition, nor should designers use them to describe how the system provides services.
- ❖ Use case models describe what is needed in a system in terms of functional responses to given stimuli.
- ❖ A use case is initiated by an actor, and then goes on to describe a sequence of interactions between actors and the system that, when taken together, model systemic functional requirements.
- ❖ Use cases may also include variants of the normal operation that describe error occurrences, detection, handling and recovery, failure modes, or other alternative behaviors

Guidelines for Use Case Model

❖ Actor

- System과 interaction하는 모든 요소가 Actor로 식별되었는가?
- System Context Model의 External Entity와 일치하는가?
- System의 개발 범위 외부에 존재하는가?

❖ Use Case

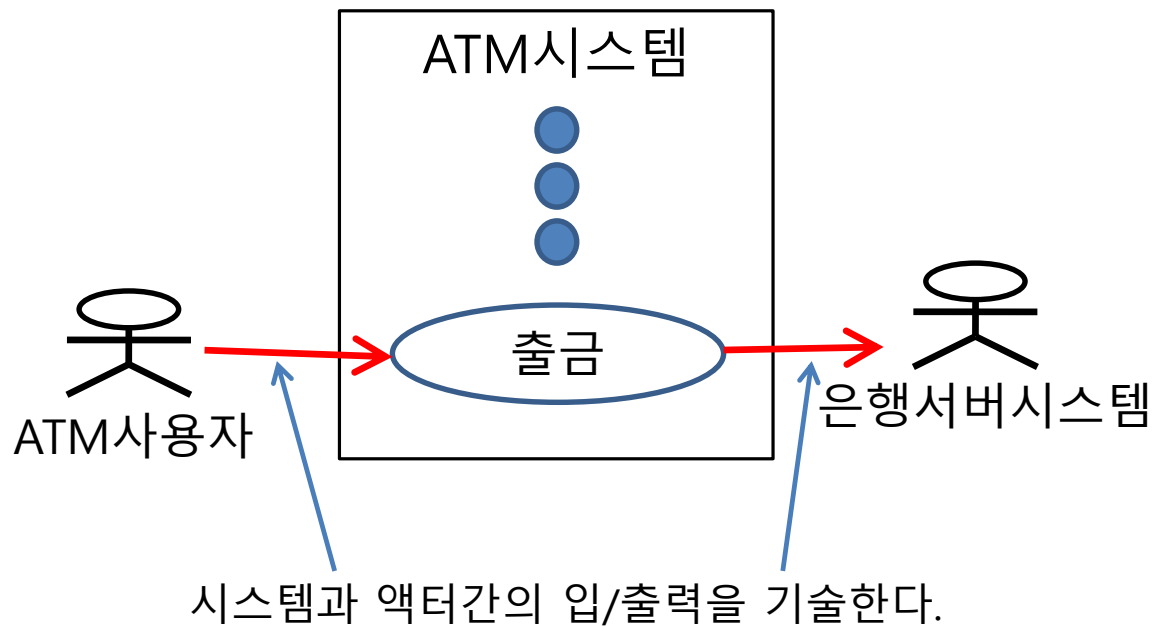
- 시스템이 제공할 기능을 의미하는가?
- 사용자 관점의 개별적인 기능 단위를 표현하고 있는가?
- 모든 기능 요구사항이 Use case로 식별되었는가?
- 사용자 관점의 궁극적인 결과를 바탕으로 명명되었는가?
- 다양한 시나리오를 하나의 Use case로 정의하였는가?

❖ Association

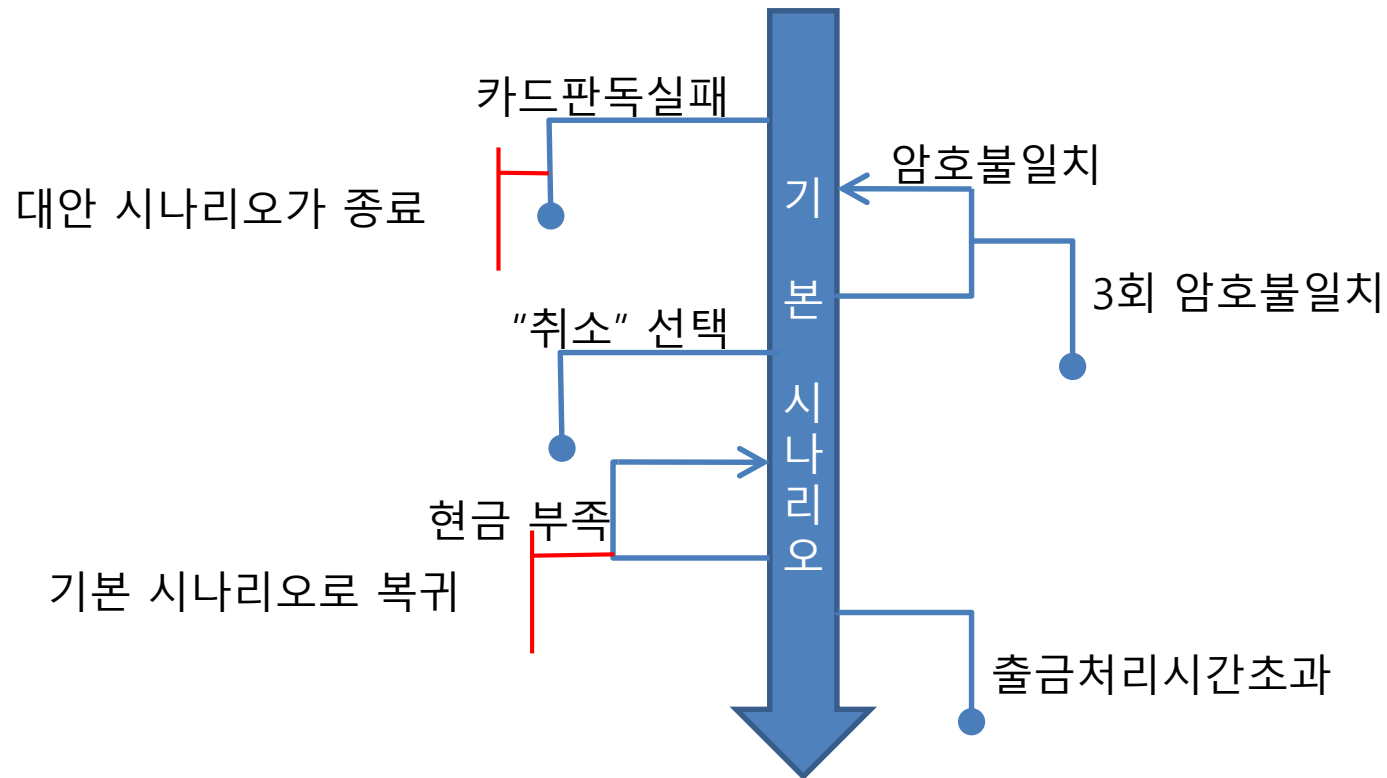
- Use case를 위하여 필요한 interaction의 대상 Actor와 모든 Association이 정의되었는가?
- 각 Association이 System과 Actor간의 실제적인 interaction을 의미하는가?
- Association의 방향이 interaction의 방향과 일치하는가?

Use Case Scenario

- ❖ Scenario specifies the specific interaction between the system and the associated actors



Basic Scenario vs. Alternative Scenarios

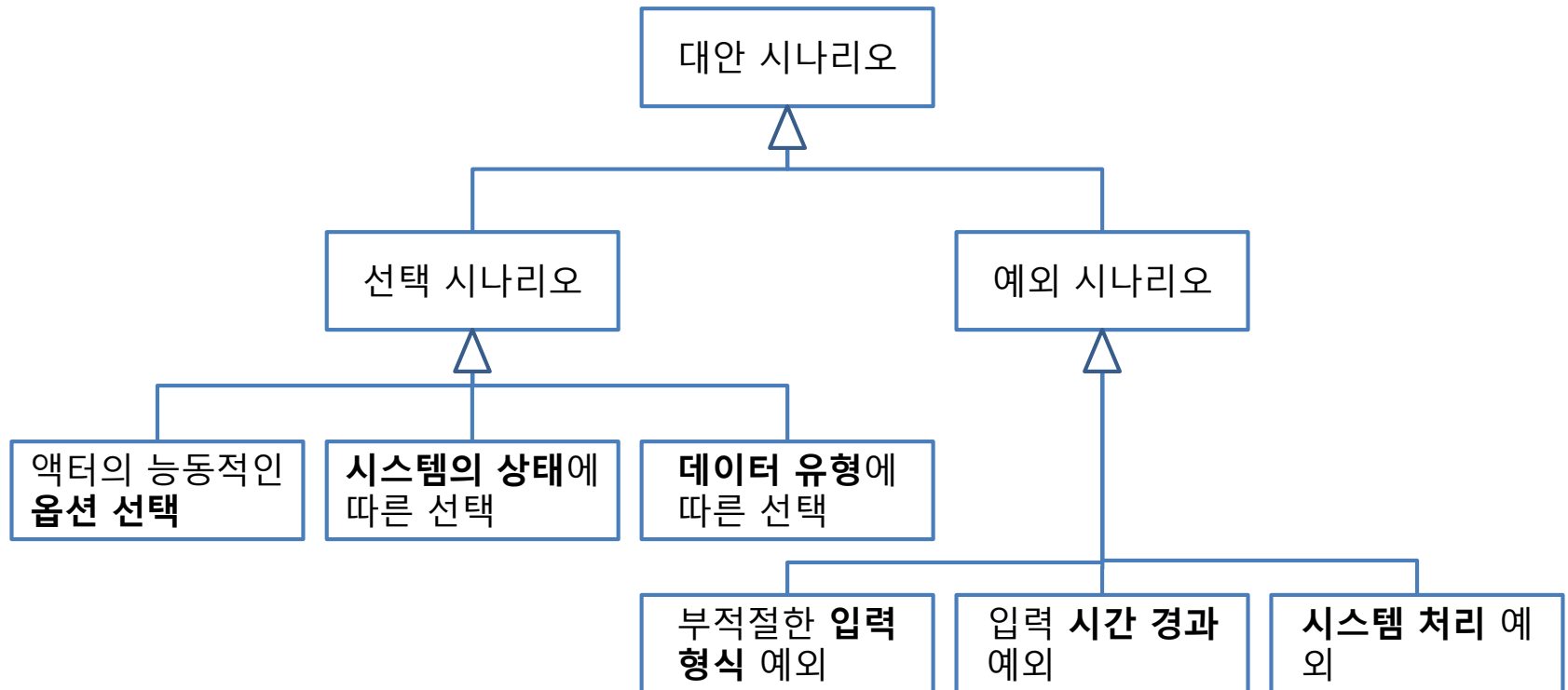


Use Case Scenario

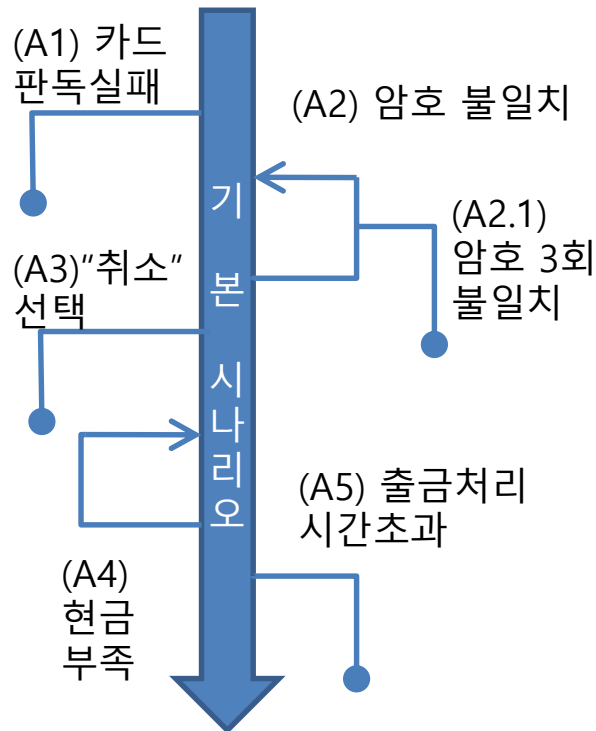
❖ 출금 유스케이스의 시나리오 명세 예

1. **ATM사용자**는 카드입력 장치에 카드를 삽입한다.
2. **시스템**은 삽입된 카드를 판독한다.
3. 시스템은 메뉴 화면을 출력한다.
4. ATM사용자는 "출금"을 선택한다.
5. 시스템은 암호 입력 화면을 출력한다.
6. ATM사용자는 암호를 입력한다.
7. 시스템은 입력된 암호의 정확성을 점검한다.
8. 시스템은 출금 금액 입력 화면을 출력한다.
9. ATM사용자는 인출금액을 입력한다.
10. 시스템은 **은행서버시스템**에게 출금요청을 한다.
11. 은행서버시스템은 요청된 출금에 대한 처리 결과를 시스템에게 통보한다.
12. 시스템은 카드와 지폐를 배출하고, 영수증은 인쇄한다.
13. ATM사용자는 카드, 지폐, 영수증을 수령한다.
14. 시스템은 지폐 배출 문을 닫는다.

Alternative Scenarios



Documenting Scenarios



기본 시나리오

1. ATM사용자는 카드입력 장치에 카드를 삽입한다.
2. **시스템**은 삽입된 카드를 판독한다.
3. 시스템은 메뉴 화면을 출력한다.
4. ATM사용자는 "출금"을 선택한다.
5. **DO**
6. 시스템은 암호 입력 화면을 출력한다.
7. ATM사용자는 암호를 입력한다.
8. 시스템은 입력된 암호의 정확성을 점검한다.
9. **WHILE** (암호가 부정확함)
10. 시스템은 출금 금액 입력 화면을 출력한다.
11. ATM사용자는 인출금액을 입력한다.
12. 시스템은 은행서버시스템에게 출금요청을 한다.
13. 은행서버시스템은 요청된 출금에 대한 처리 결과를 시스템에게 통보한다.
14. 시스템은 카드와 지폐를 배출하고, 영수증은 인쇄한다.
15. ATM사용자는 카드, 지폐, 영수증을 수령한다.
16. 시스템은 지폐 배출 문을 닫는다.

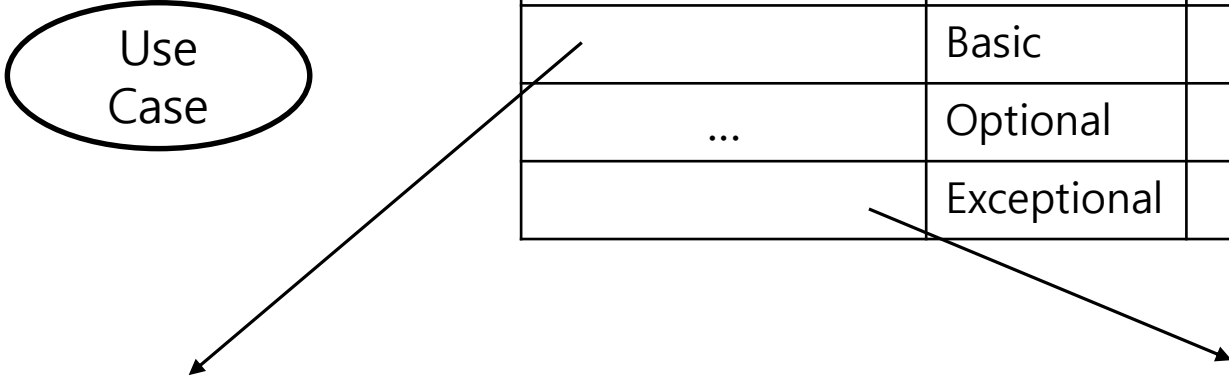
Documenting Scenarios

대안 시나리오
A1: 카드판독 실패 시 (2)에서 카드 판독이 안 될 때 1. 시스템은 카드 판독 실패 화면을 출력한다. 2. 시스템은 카드를 배출시킨다.
A2.1: 3회 암호 불일치 시 (8)에서 3회 부정확한 암호가 입력되었을 때 1. 시스템은 3회 암호가 부정확함을 출력한다. 2. 시스템은 카드를 배출시킨다.
A3: 취소 입력 시 (10)에서 "취소" 버튼을 누를 때 1. 시스템은 카드를 배출시킨다.
A4: 현금 부족 시 (11)에서 인출금액보다 적은 지폐가 있을 때 1. 최대 인출가능 금액을 출력한다. 2. 기본 시나리오 (10)으로 간다.
A5: 출금 처리 시간 초과 시 (12)에서 제한된 시간 내에 처리결과가 오지 않을 때 1. 시스템은 출금이 안됨을 출력한다. 2. 시스템을 카드를 배출시킨다

Guidelines for Use Case Scenario

- ❖ Focus on interactions which involve quality attributes such as performance, modifiability, security, ...
- ❖ Include **each interaction** with **all the actors** associated with the use case
- ❖ Each step should be written in active voice with the subject of the system or an associated actor.
- ❖ Each step should describe the behavior of the system or an associated actor, but not both.
- ❖ Each step should describe the interaction clearly.
- ❖ Use terms that can be understood by stakeholders.
Don't use technical terms that can be only understood by developers.

Use Case Specification



The diagram illustrates the relationship between a Use Case and its specification. An oval labeled "Use Case" has an arrow pointing to the "Scenario Title" column of the table below. Another arrow points from the "Exceptional" row of the table to the "Flow of events" section of the table on the right.

Scenario Title	Kind	Description
	Basic	
...	Optional	
	Exceptional	

Pre-condition	
Post-condition	
Flow of events	

○ ○ ○

Pre-condition	
Post-condition	
Flow of events	

ARCHITECTURAL DRIVERS QUALITY ATTRIBUTE SCENARIOS

Quality Requirements and Architecture

❖ Functionality does not determine architecture.

- That is, given a set of required functionality, there is no end to the architectures you could create to satisfy that functionality.
- At the very least, you could divide up the functionality in any number of ways and assign the sub-pieces to different architectural elements.
- In fact, if functionality were the only thing that mattered, you wouldn't have to divide the system into pieces at all;

❖ Instead, we design our systems

- as structured sets of cooperating architectural elements (layers, components, classes, databases, apps, threads, peers, tiers, and on and on)
- to support a variety of other purposes (i.e. quality attributes).

Software architecture in practice, 3rd edition(2012)

Quality Requirements and Architecture

- ❖ Systems are frequently redesigned
- ❖ not because they are functionally deficient,
- ❖ but because
 - they are difficult to maintain, port, or scale;
 - or they are too slow;
 - or they have been compromised by hackers
- ❖ Among the drivers, quality requirements are the ones that shape the architecture the most significantly.

Performance Requirement

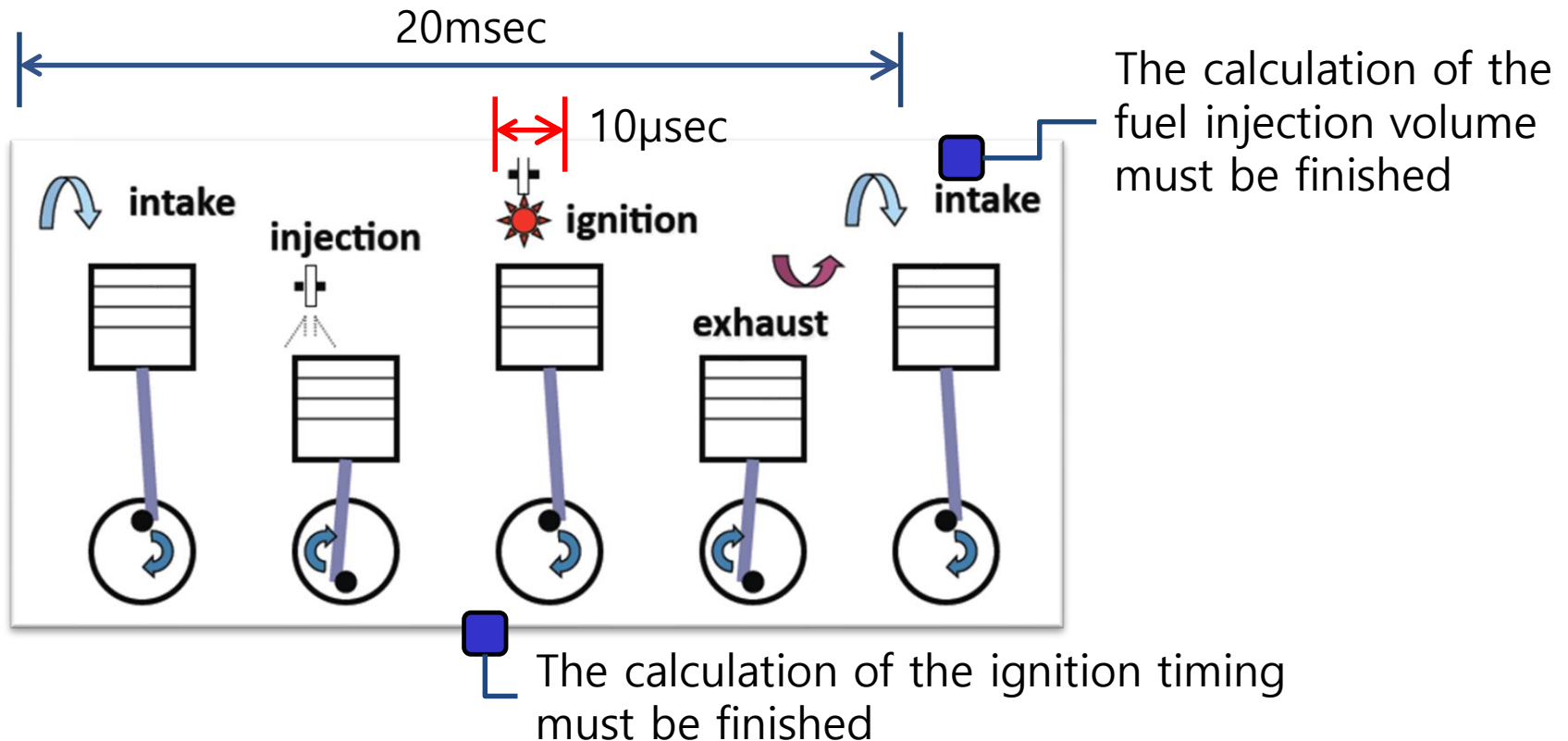
- ❖ A requirement that defines the extent or how well, and under what conditions, a function or task is to be performed.

- ❖ These are quantitative requirements of system performance and are verifiable individually.
 - Response time for a transaction(average, maximum)
 - Throughput (for example, transactions per second)
 - Capacity (for example, the number of customers or transactions the system can accommodate)
 - Degradation modes (what is the acceptable mode of operation when the system has been degraded in some manner)
 - Resource use: memory, disk, communications, and so forth

Performance Requirement

❖ Example

- In the case of 6000rpm, one cycle is 20msec.
- Timing precision of the ignition is 10μsec. order.



Reliability Requirement

- ❖ Reliability: the probability of **failure-free operation** for a specified period of time in a specified environment.
- ❖ Failure: An unacceptable effect or behavior under permissible operating conditions.
- ❖ Reliability can be quantitatively evaluated
 - Mean Time Between Failures (MTBF)
 - Mean Time To Repair (MTTR)
 - Accuracy – specify precision (resolution) and accuracy (by some known standard) that is required in the systems output.
 - Maximum bugs or defect rate – usually expressed in terms of bugs/KLOC (thousands of lines of code), or bugs/function-point.

Availability Requirement

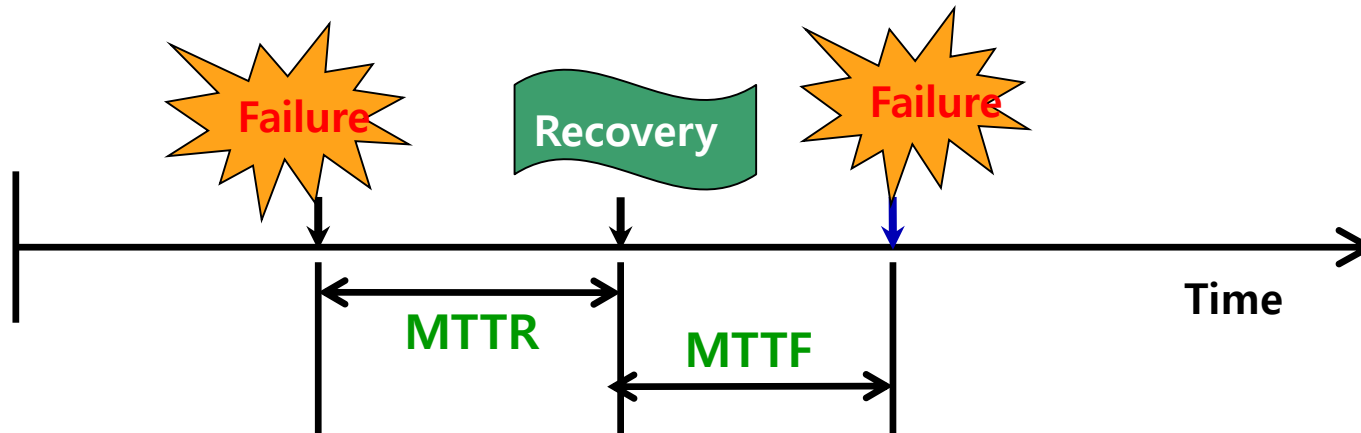
The degree to which a system or component is **operational and accessible when required for use**

[IEEE 610]

$$\text{Availability} = \frac{\text{MTTF}}{\text{MTTF} + \text{MTTR}}$$

MTTF : Mean Time To **F**ailures

MTTR : Mean Time To **R**epair



Availability Requirement

가용성	연간 장애 시간	주당 장애 시간
98%	7.3일	3시간 22분
99%	3.65일	1시간 41분
99.8%	17시간 30분	20분 10초
99.9%	8시간 45분	10분 5초
99.99%	52분 30초	1분
99.999%	5분 25초	6초

Monthly Uptime Percentage	Service Credit Percentage
99.0% ~ 99.95%	10%
~ 99.0%	30%

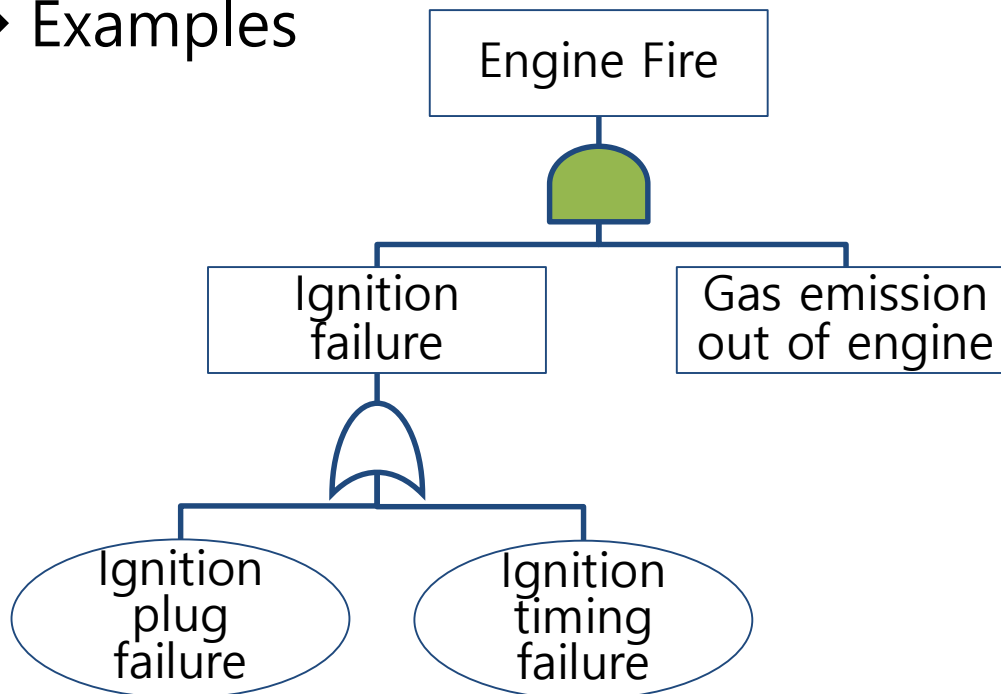
<http://aws.amazon.com/ko/ec2-sla/>

Safety Requirement

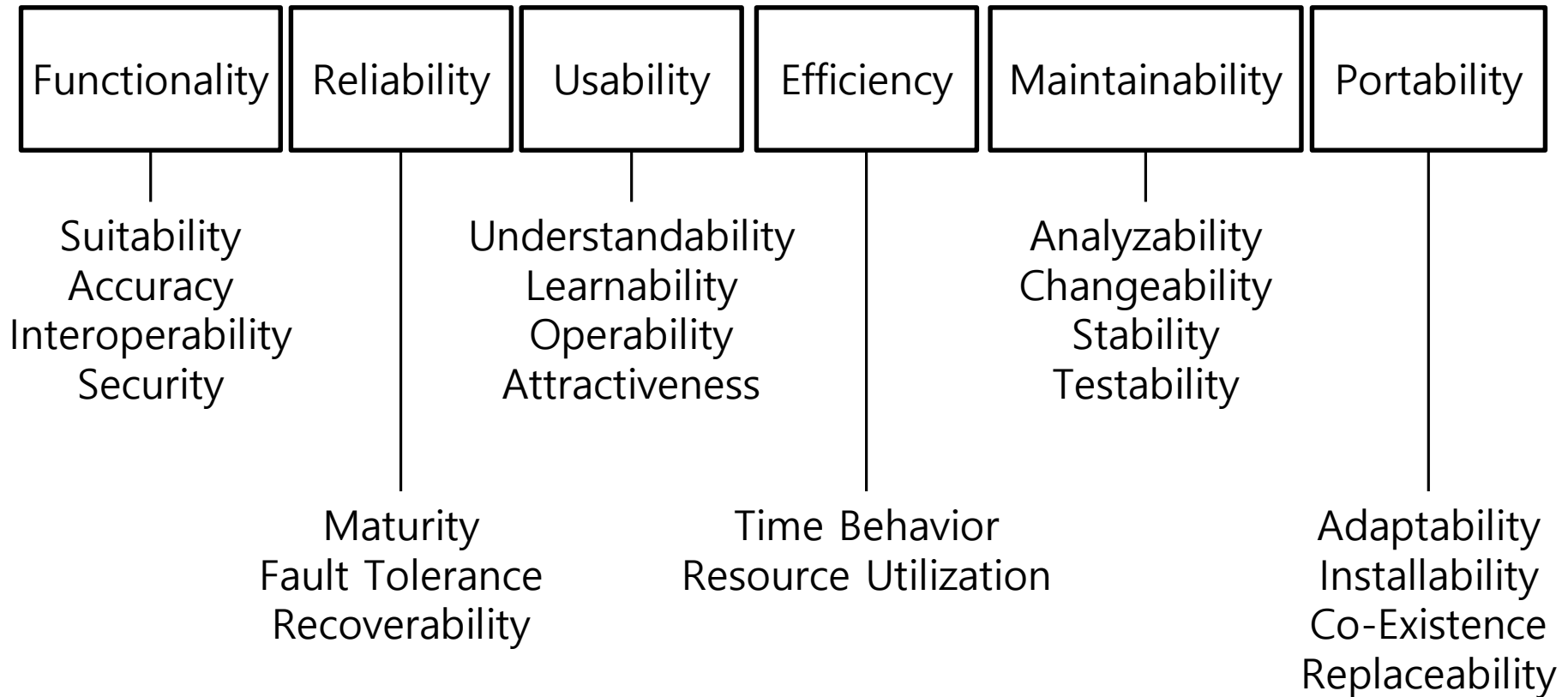
Freedom from those conditions that can cause **death, injury**, occupational illness, or damage to or loss of equipment or property, or **damage to the environment**

[MIL-STD-882C]

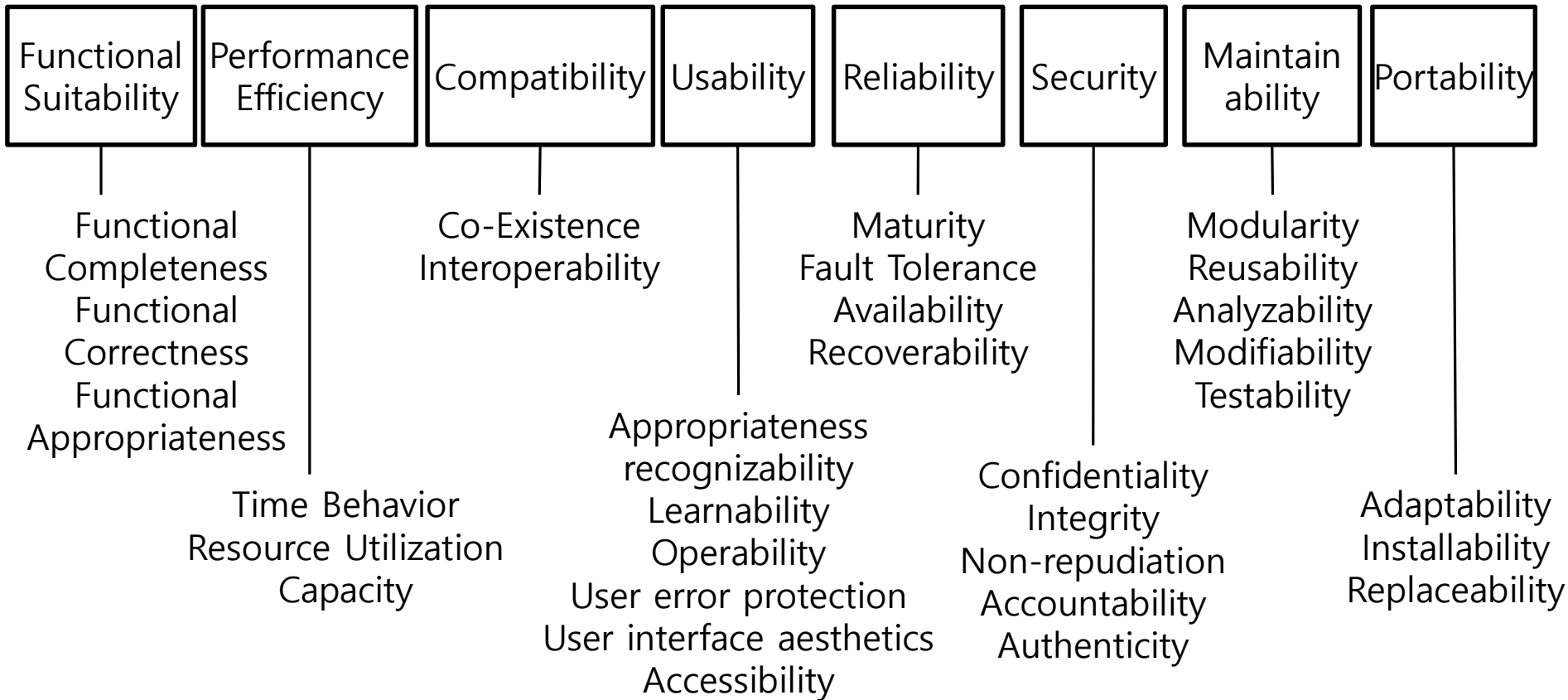
❖ Examples



ISO 9126-1:2001 Quality Model



ISO/IEC 25010:2011 Quality Model



ISO/IEC 25010:2011 Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models

ISO/IEC 25010:2011 Quality Model

Characteristics	Description
Functional Suitability	degree to which a product or system provides <u>functions that meet stated and implied needs</u> when used under specified conditions
Performance Efficiency	performance relative to the <u>amount of resources</u> used under stated conditions
Compatibility	degree to which a product, system or component can <u>exchange information</u> with other products, systems or components, and/or <u>perform its required functions</u> , while sharing the same hardware or software environment
Usability	degree to which a product or system can be used by specified users <u>to achieve specified goals with effectiveness, efficiency and satisfaction</u> in a specified context of use

ISO/IEC 25010:2011 Quality Model

Characteristics	Description
Reliability	degree to which a system, product or component <u>performs specified functions</u> under specified conditions for a specified period of time
Security	degree to which a product or system <u>protects information and data</u> so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization
Maintainability	degree of effectiveness and efficiency with which a product or system <u>can be modified</u> by the intended maintainers
Portability	degree of effectiveness and efficiency with which a system, product or component can <u>be transferred from one hardware, software or other operational or usage environment to another</u>

Other Important Quality Attributes

❖ Variability

- a special form of modifiability.
- refers to the ability of a system and its supporting artifacts to support the production of a set of variants that differ from each other in a preplanned fashion

❖ Scalability

- Vertical scalability (scaling up): adding more resources to a physical unit, such as adding more memory to a single computer
- Horizontal scalability (scaling out): adding more resources to logical units, such as adding another server to a cluster of

System Feature → Quality Requirement

❖ Quality requirements are derived from system features

System Feature	QA Kind	Quality Requirement
Support international languages	Modifiability	A developer is able to <u>package a version of the system with new language support</u> in 80 person-hours.
Comply with regulations that have an impact on life-critical systems such as fire alarms	Performance	A life-critical alarm should be reported to the concerned users <u>within 3s of the occurrence of the event</u> that generated the alarm.
Support hardware devices from different Manufacturers	Modifiability	A field engineer is able to <u>integrate a new field device</u> into the system at runtime <u>with no downtime or side effects</u> .
Support conversions of nonstandard units used by the different hardware devices	Modifiability	A system administrator configures the system at runtime to <u>handle the units from a newly plugged in field device</u> with <u>no downtime or side effects</u> .

Functional Requirement vs Quality Requirement

- ❖ In practice, quality attribute requirements and functionality are usually intimately intertwined.
 - It is impossible and meaningless to say a system “shall have high performance.”
 - Without associating the performance to some specific behavior in the system, architects cannot hope to design a system to satisfy this need.

Architecting software intensive systems-A practitioner's guide(2008)

- ❖ Suppose a functional requirement: “The game shall change view modes when the user presses the <C> button”
 - Performance: How fast should the function be?
 - Security: How secure should the function be?
 - Modifiability: How modifiable should the function be?

Designing software architecture-A practical approach(2016)

Quality Requirement Description

- ❖ A system will be “modifiable” → ambiguous
 - because every system is modifiable with respect to some changes and not modifiable with respect to others.

- ❖ A system shall have “high performance”. → ambiguous
 - This kind of requirement is non-descriptive and puts forth only general notions that are impossible for architects to design or measure in the system once implemented.
 - What kind of performance does this refer to?
 - Does performance mean response time, throughput, or something else?

Quality Requirement Description

- ❖ For architects to fully understand quality requirements, more detailed descriptions required

- ❖ Quality attributes

- Measurable or testable properties of a system
- that are used to indicate how well the system satisfies the needs of its stakeholders.

- ❖ How to express the qualities unambiguously?

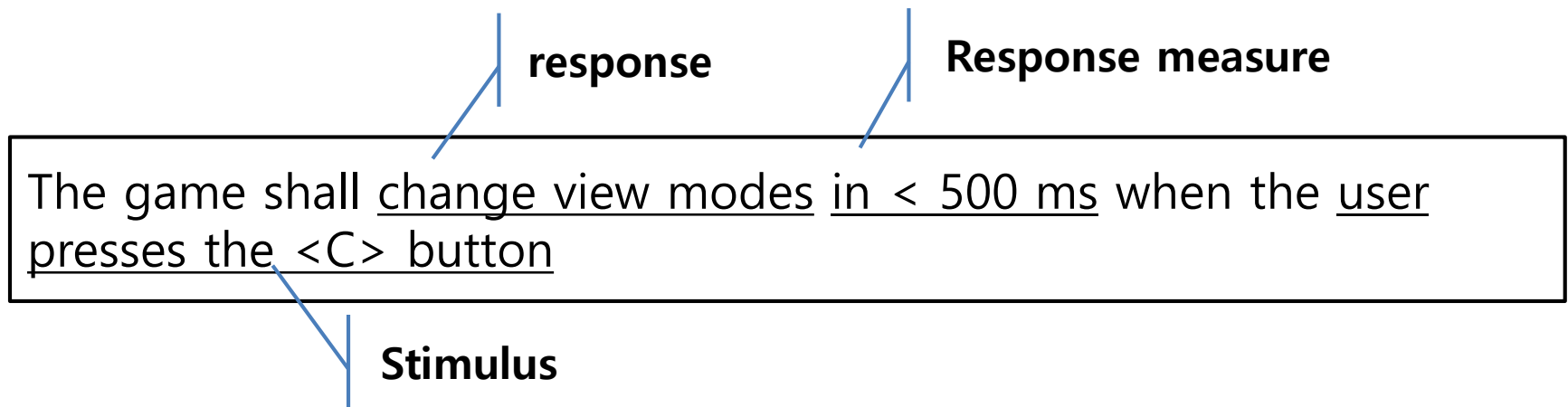
- ❖ Solution: quality attribute scenarios

Quality Attribute Scenario

- ❖ The best way to discuss, document, and prioritize quality requirement is as a set of scenarios
- ❖ Scenario describes the system's response to some stimulus.
- ❖ Example of modifiability
 - One can specify the modifiability response measure you would like to achieve (say, elapsed time or effort) in response to a specific change request.
 - "a change to update shipping rates on the e-commerce website is completed and tested in less than 1 person-day of effort"

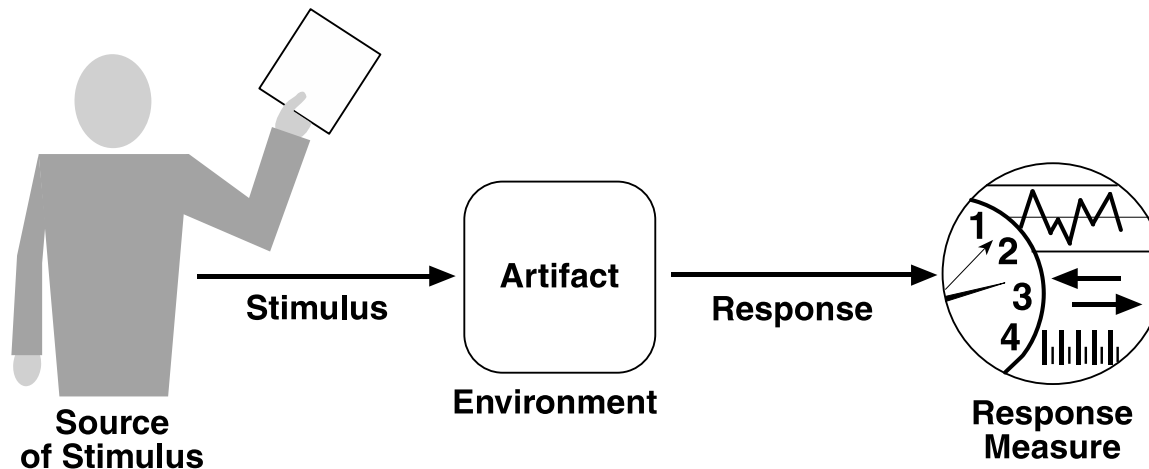
Basic Quality Attribute Scenario

- ❖ A short description of how(**measure**) a system is required to respond(**response**) to some event(**stimulus**).
- ❖ e.g.) performance scenario



Complete Quality Attribute Scenario

- ❖ In addition to **stimulus**, **response** and **response measure**, complete QA scenario adds three other parts:
 - The **source** of the stimulus: the use
 - The **artifact** affected: the entire system
 - The **environment**: in normal operation, startup, degraded mode, or some other mode?
- ❖ In total, there are six parts of a complete QA scenario.



Complete Quality Attribute Scenario

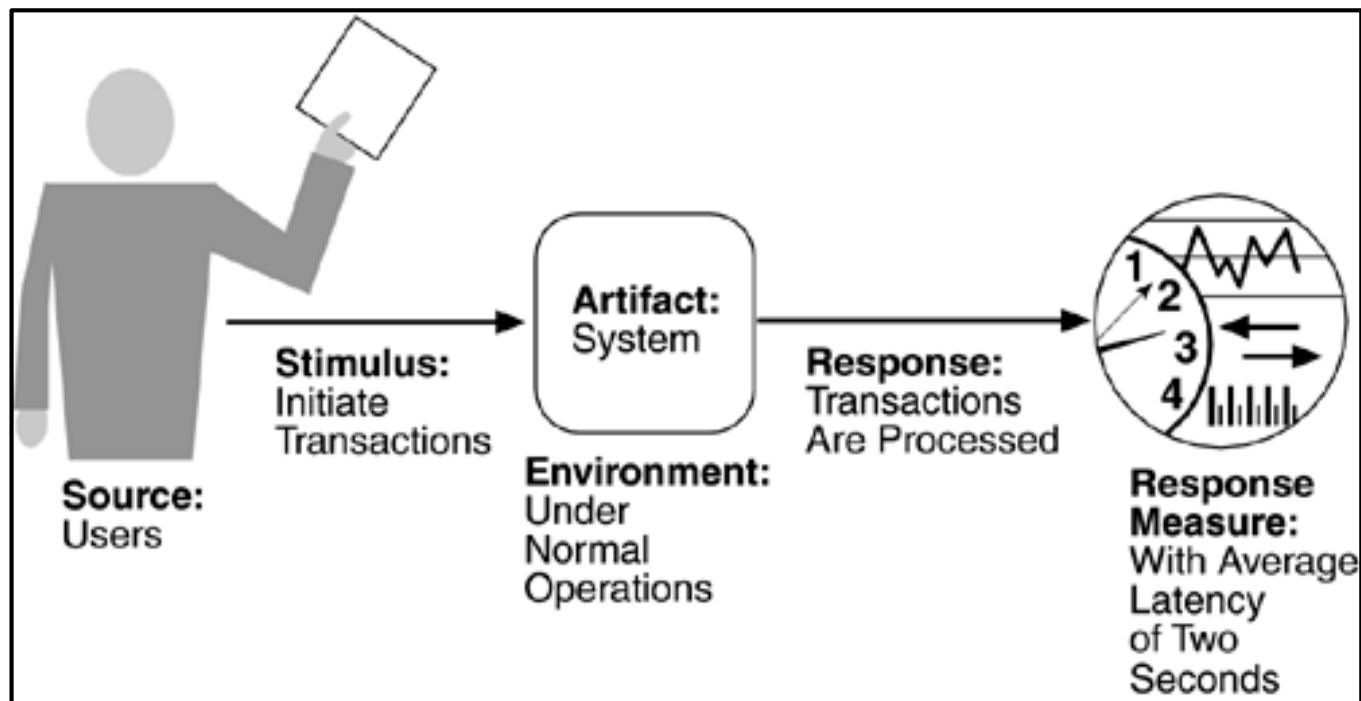
- ❖ Six-part quality attribute scenario framework for recording, negotiating, and analyzing quality attribute requirements.

Stimulus	The stimulus is a condition that requires a response when it arrives at a system
Source of the stimulus	This is some entity (a human, a computer system, or any other sensors/actuators) that generated the stimulus
Artifact	Some artifact is stimulated. This may be a collection of systems, the whole system, or some piece or pieces of it
Environment	The stimulus occurs under certain conditions.
Response	The response is the output generated or activity undertaken as the result of 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.

Architecting software intensive systems-A practitioner's guide(2008)

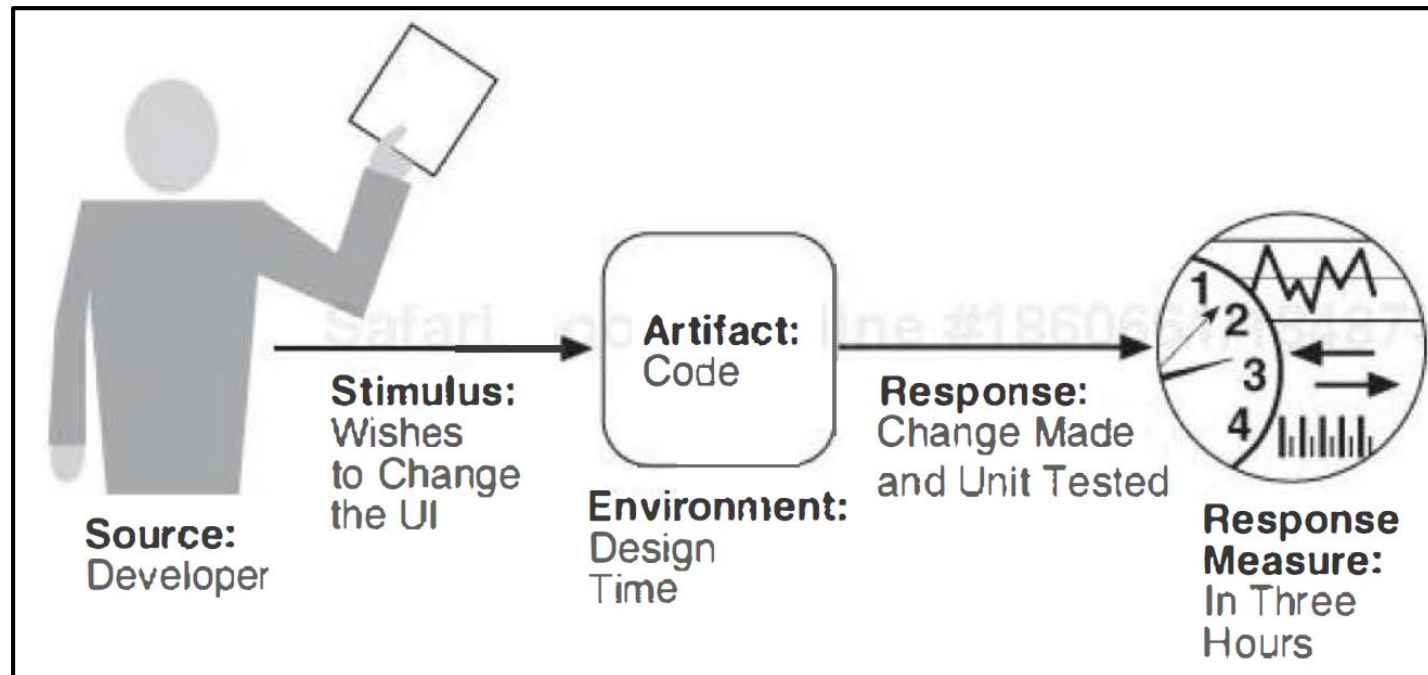
Sample Performance Scenario

- ❖ Users<**Source**> initiate transactions<**Stimulus**> under normal operations<**Environment**>.
- ❖ The system<**Artifact**> processes the transactions<**Response**> with an average latency of two seconds<**Response Measure**>.



Sample Modifiability Scenario

- ❖ The developer<**Source**> wishes to change the user interface <**Stimulus**> by modifying the code<**Artifact**> at design time<**Environment**>.
- ❖ The modifications are made with no side effects <**Response**> within three hours<**Response Measure**>.



Stimulus

- ❖ Phenomenon, event, situation, etc., that
 - prompts the system or stakeholders(developers)
 - to take some action
 - in some way
- ❖ It might be a user request, an event or interrupt, an error, a request for change, and so forth.
- ❖ Specific stimulus can be different depending on QA
 - Run-time QA
 - ✓ Performance, Reliability: an arrival of event
 - ✓ Security: an attack
 - ✓ Usability: user operation
 - Development-time QA
 - ✓ Modifiability: request for change
 - ✓ Testability: completion of a phase of development

Response

- ❖ How the system or stakeholder 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.
- ❖ In a performance scenario,
 - Stimulus: an event arrives
 - Response: the system should process that event and generate a response.
- ❖ In a modifiability scenario,
 - Stimulus: a request for a modification arrives
 - Response: the developers should implement the modification without side effects and then test and deploy the modification.

Stimulus and Response

Quality	Stimulus	Response
Performance	Arrival of a periodic, sporadic, or stochastic event	Process events, change level of service
Availability	Fault: omission, crash, incorrect timing, incorrect response	Prevent the fault from becoming a failure: Detect the fault, Recover from the fault
Modifiability	A directive to add / delete / modify functionality, or change a quality attribute, or technology	One or more of the following: <ul style="list-style-type: none">• make modification• test modification• deploy modification

Response Measure

- ❖ Determining whether a response is satisfactory - whether the requirement is satisfied - is enabled by providing a response measure.
- ❖ Examples
 - For performance, it could be a measure of latency or throughput
 - For modifiability, it could be the labor or wall clock time required to make, test, and deploy the modification.

Quality	Examples of Response Measure
Performance	Latency, deadline, throughput, jitter, miss rate
Availability	Availability percentage (e.g. 99.999%), Time to detect the fault, Time to repair the fault
Modifiability	Cost in terms of size of affected artifacts and effort

Environment

- ❖ The environment is the set of circumstances in which the scenario takes place.
- ❖ Examples
 - 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.
 - A failure that is the fifth successive failure of a component may be treated differently than the first failure of that component

Quality	Examples of Environment
Performance	Operational mode: normal, emergency, peak load, overload
Availability	Normal operation, startup, shutdown, repair mode, degraded operation, overloaded operation
Modifiability	Runtime, compile time, build time, initiation time, design time

Artifact

❖ Some artifact is stimulated.

- Frequently this is the entire system, but occasionally specific portions of the system

❖ Examples

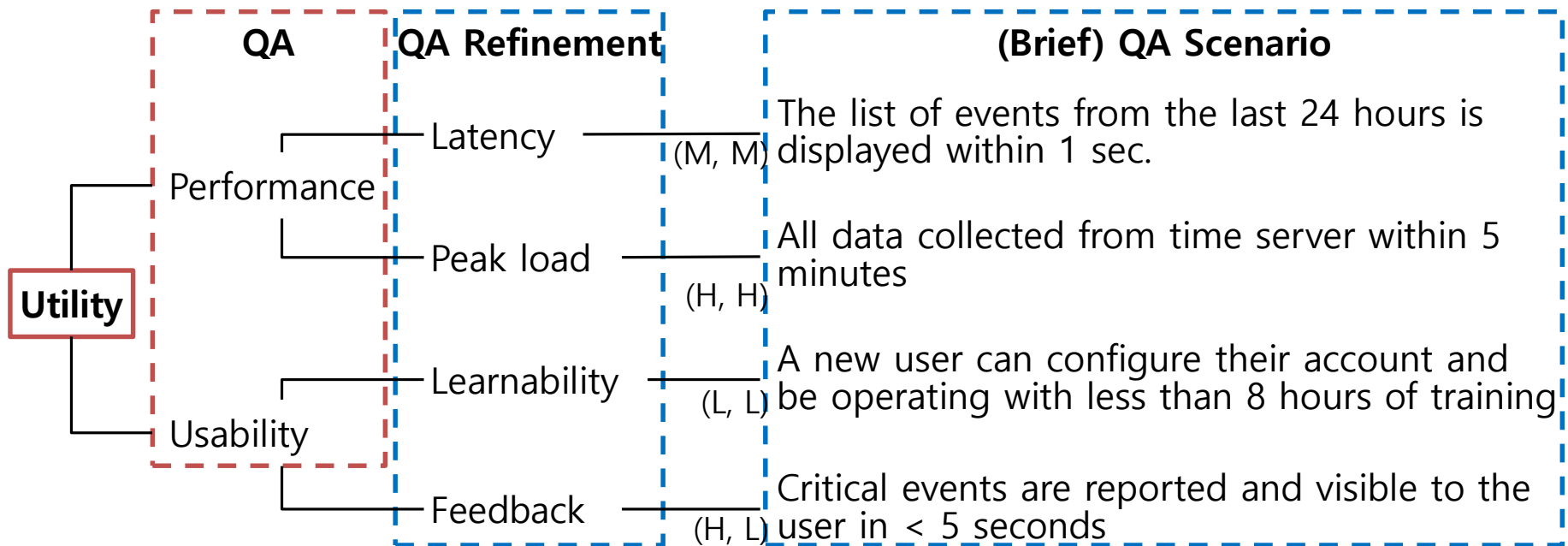
- A failure in a data store may be treated differently than a failure in the metadata store.
- Modifications to the user interface may have faster response times than modifications to the middleware.

Quality	Examples of Artifact
Performance	System or one or more components in the system
Availability	Processors, communication channels, persistent storage, processes
Modifiability	Code, data, components/interfaces, resources, configurations, ...

Utility Tree

- ❖ Word "utility" as the root node.
- ❖ We then elaborate this root node by listing the major quality attributes that the system is required to exhibit
- ❖ Under each QA, record a specific refinement of that QA.
- ❖ Under each refinement, describe (brief) QA scenarios.

Software architecture in practice, 3rd edition(2012)



QA Scenario Check List

Stimulus	Artifact에 대한 Response를 유발할 수 있는 Stimulus인가? QA에 따른 타당한 Stimulus인가? Stimulus를 객관적으로 생성할 수 있을 정도로 구체적인가?
Source of the stimulus	Stimulus에 대한 타당한 Source인가? QA에 따른 타당한 Source인가?
Artifact	Stimulus의 대상이면서 Response의 대상인가?
Environment	Artifact에 대한 조건을 기술하였는가? Response 및 Response Measure에 대한 조건을 기술하였는가?
Response	Stimulus에 대한 Artifact의 결과를 기술하였는가? QA에 따른 타당한 Response인가? Response의 발생 여부를 객관적으로 판단할 정도로 구체적인가?
Response Measure	Response에 대하여 가능한 Measure인가? QA에 따른 타당한 Measure인가? Measure의 수치에 대한 객관적인 평가가 가능한가?

QA Scenario: Example

QA Type	신뢰성
Description	차량형태, 번호판형태에 무관하게 차량번호를 정확하게 → 99.9%로 인식해야 한다.
Source of Stimulus	번호판카메라 → 얼마나 많은 수의 카메라
Stimulus	차량번호영상 및 스냅샷 → 일정 시간 동안에 몇 개, 어떤 유형의 번호판
Artifact	차량번호 영상분석 소프트웨어 → 컴포넌트/모듈 (OCR:Optical Characterized Recognition)
Environment	차량입출차시 → Response Measure에 영향을 주는 시스템 환경
Response	인식된 차량번호
Response Measure	차량번호 인식률 99.9% 이상 → 인식률 계산 방법은? 99.9 %는 충분한가? 1,000대 중 1대는 부정확함을 허용할 수 있는가?

QA Scenario: Example

QA Type	Performance
Description	번호판 촬영하면 차량 번호를 인식하여 빠른 시간 → 1초 안에 차단기를 열어야 한다.
Source of Stimulus	카메라로부터 번호판 촬영 영상 입력 → 번호판 카메라 000 개
Stimulus	번호판 영상으로부터 차량 번호 추출 → 00개 유형의 번호판 이미지
Artifact	스마트 주차 관리 시스템
Environment	스마트 주차 관리 시스템 정상 동작 중 → 가정하는 시스템 부하 상태 설명 필요
Response	차량 번호를 추출하여 차단기를 열어준다 → 차량 번호를 인식하고 차단기에게 열기 명령을 전송
Response Measure	번호판 영상으로부터 차량 번호를 추출한 뒤 차단기를 열기까지 1초 이내 에 완료되어야 한다 → 번호판 이미지 입력부터 차단기 열기 명령 전송까지 평균 1초

QA Scenario

QA Type	Availability
Description	주차장 시설물 이상 발생에 대해 감지하여 관리자와 소유주에게 알림을 보낸다.
Source of Stimulus	시스템에 등록된 주차장에 위치한 시설물 상태 확인 → 각종 장치 구체적 적시(개수 까지)
Stimulus	주차장 시설 이상 발생 → 이상의 유형 구체화 필요(예, 응답 없음, 비정상 값 수신)
Artifact	스마트 주차 관리 시스템
Environment	스마트 주차 관리 시스템 정상 동작 중
Response	주차장 시설물에서 이상 감지되면 시스템 관리자, 시설 관리자, 주차장 소유주에게 즉시 알림 발생한다. → 장애 발생 여부를 분석하고 장애 발생시 해당 주차장 시설 관리자, 해당 주차장 소유주, 관제 센터 관리자에게 장애 발생 통보
Response Measure	1분 간격으로 주차장 기기 상태에 대해 주기적으로 체크하고, 이상이 감지되면 1초 이내에 관리자와 소유주에게 알림을 보내야 한다. → 이상 발생 후 000 초 이내에 통보

QA Scenario

QA Type	Availability
Description	고객의 호텔 및 객실 출입이 차단될 경우 고객 불만이 크게 발생할 수 있으므로, 시스템 중단 시간의 최소화 필요
Source of Stimulus	고객, 장비 → 장애 탐지 또는 복구 대상을 구체적으로 명시
Stimulus	Crash, Incorrect response
Artifact	시스템, 체크인/아웃 Pole, Door Controller → 시스템
Environment	Normal Operation
Response	시스템 중단 없이 동작 → 장애 탐지에 대한 통보 또는 장애 복구 후의 시스템 상태를 구체적으로 명시
Response Measure	가용성 99.99% → 장애 탐지 보고인 경우와 장애 복구인 경우가 다름; 장애 탐지만으로는 가용성 보장 불가능함

QA Scenario

QA Type	Modifiability
Description	새로운 유형의 번호판 인식 지원을 위한 스마트 주차 관리 시스템 변경 시 기존 시스템 변경을 최소화하고, 개발 및 테스트 시간이 짧아야 한다.
Source of Stimulus	스마트 주차 관리 시스템 개발팀 → 개발팀이 개발팀에게 요청하는가?
Stimulus	이미지 차량번호 인식 부분 변경 → 새로운 유형의 번호판 도입 - 이미지 Crop 영역 수정 - 차량 번호 인식 AI 엔진 학습
Artifact	스마트 주차 관리 시스템 (차량 번호 인식하는 부분)
Environment	정부 지침 변경에 따라 주차장 소유주가 새로운 번호판 인식 지원을 결정한다. 새로운 유형의 번호판 인식 지원에 따른 스마트 주차 관리 시스템 업데이트를 요청한다. → 개발/테스트 환경
Response	코드 수정, Unit Test, 시스템 통합 테스트, 차량 번호 인식 테스트 완료.
Response Measure	새로운 유형의 번호판 인식 지원에 따른 스마트 주차장 관리 시스템 변경은 코드 수정부터 테스트 완료까지 1 M/W (Man/Week) 안에 완료되고, 시스템 동작에 부작용은 발생하지 않는다 → Response 임

ARCHITECTURAL DRIVERS CONSTRAINTS

Constraint

- ❖ A constraint is fixed premade decisions before design begins
 - Business constraints limit decisions about people, process, costs, and schedule.
 - Technical constraints limit decisions about the technology we may use in the software system; Externally imposed limitation on system requirements, design, or implementation or on the process used to develop or modify a system
- ❖ Each of these exerts forces on the architect and influences the design decisions that the architect makes
- ❖ Constraints limit choice, but some constraints simplify the problem and can make it easier to design a satisficing architecture

Design It! – From programmer to software architect(2017)

Business Constraints

- ❖ Business constraints are indirect constraints on the design space.
- ❖ They are indirect in that business constraints do not specify that a particular technology is used to design or build a system
- ❖ But they impose cost, schedule, regulatory, legal, marketing, and other similar demands that will influence the design of the system

Business Constraints

Kind	Description
Cost limitations	How much over what period of time can be spent on the system or product?
Schedule limitations	What are the delivery schedules? One delivery? Incremental? What functionality must be delivered at what point in time?
Regulatory restrictions and demands	Are there any regulations imposed on the system, product, or organization designing and building the system, or the customer stakeholders' organization?
Legal restrictions and demands	Are there any legal impositions placed on the system, product, or organization designing and building the system, or the customer stakeholders' organization?
Market restrictions and demands	Does the target market impose any restrictions or demands on the system or product, especially if it could prevent entry into another market?
Organizational restrictions and demands	Do any of the organizations involved in the project have policies, processes, resources or lack thereof, or structural issues that could impose restrictions or demands on the design or construction of the system or product?
Logistical issues	Are there logistical issues such as deployment, transportation, supplier/supply chain, and similar that could impact the design of the system?

Architecting software intensive systems-A practitioner's guide(2008)

Technical Constraints

- ❖ A constraint is an unchangeable design decision
- ❖ Technical constraints have direct influence on the design because they specify that a particular product, tool, language, OS, platform, network, protocol, algorithm, and so forth must be used in the system.
- ❖ In practice, technical constraints are often more subtle and may direct the use of commercial products, design methods, algorithms, specific design structures, hardware, operating systems, tools, languages, and so forth.

Technical Constraints

- ❖ Specific technologies, tools, languages, and databases that must be used or avoided.
 - Restrictions because of the product's operating environment or platform, such as the types and versions of web browsers or operating systems that will be used.
- ❖ Required development conventions or standards.
 - For instance, if the customer's organization will be maintaining the software, the organization might specify design notations and coding standards that a subcontractor must follow.
- ❖ Backward compatibility with earlier products and potential forward compatibility, such as knowing which version of the software was used to create a specific data file.

Technical Constraints

❖ Standards

- Standards may address technology problems, such as those that define the physical mechanisms for linking computers together, or they may address business problems, such as those that define the syntax and semantics of a certain class of business messages.
- Open standards(ISO, IEEE, W3C, ...), Proprietary standards(MS, Oracle, ...), De facto standards, Organizational standards

❖ Guidelines and Strategies

- These are less formal than standards and take the form of advice or recommended practice.
- Compliance with these is not mandatory, and you and your stakeholders will need to assess the need to comply in all cases.

❖ Policies

- Policies start to define processes that must be followed in order to meet stakeholder needs.
- You may need to comply with preexisting policies (security policies are a common example), or alternatively your architectural analysis may identify the need to produce some policies of your own

Software systems architecture-working with stakeholders using viewpoints and perspectives(2005)

Technical Constraints

Kind	Description
operating system(s)	Are there any constraints to use a particular OS? Are there any constraints to support multiple OSs?
platform(s)	Are there any constraints to use particular platform(s)?
languages(s)	Is there a constraint to use a particular language?
Peripheral or network hardware	Are there any constraints that specify that particular peripheral devices or network hardware be used?
Commercial products	Is there a constraint that specific commercial hardware and software products be used?
Tools and methods	Are there any constraints that specify that certain tools (e.g., design/programming tools) or technical methods be used?
Protocols, interfaces, standards	Are there any constraints that specify that certain protocols, interfaces, or standards be used or adhered to during development?
Legacy hardware and software	Are there any constraints that indicate that the new system/product must utilize or interact with any legacy hardware or software systems or elements?

Architecting software intensive systems-A practitioner's guide(2008)

Technical Constraints

- ❖ Limitations or compliance requirements imposed by regulations or other business rules.
- ❖ Hardware limitations such as timing requirements, memory or processor restrictions, size, weight, materials, or cost.
- ❖ Restrictions because of the size of the display, as when running on a tablet or phone.
- ❖ Physical restrictions because of the operating environment or because of characteristics or limitations of the users.
- ❖ Existing interface conventions to be followed when enhancing an existing product.
- ❖ Interfaces to other existing systems, such as data formats and communication protocols.
- ❖ Standard data interchange formats used, such as XML, or RosettaNet for e-business

Design Constraint: Code Metrics

유형	메트릭	허용 최대값				
		근거				
		MISRA	SCR-G	JPL	JSF	HIS
크기	Lines of Code(LOC)	80	200	60	200	50
	Comment Frequency	50%	30%	-	-	-
복잡도	Cyclomatic Complexity(CC)	15	20	-	20	10
	Number of Execution Paths(NPath)	75	-	-	-	80
	Number of Structuring Levels	6	6	-	-	4
결합도/ 모듈화	Number of Parameters	-	8	6	6	5
	Fan In	-	8	-	-	5
	Fan Out	-	10	-	-	7
	Number of Calling Levels	8	-	-	-	4

* MISRA: MISRA Report 5, Software Metrics

* SCR-G: 무기체계 소프트웨어 개발 및 관리 매뉴얼, 소프트웨어 신뢰성/보안성 시험 절차

* JPL: JPL(Jet Propulsion Lab.) Coding Standard for the C

* JSF: Joint Strike Fighter Air Vehicle C++ Coding Standards

* HIS: HIS(Audi, BMW 등 5개 자동차 업체 그룹) Source Code Metrics

Implementation Constraints

❖ Coding styles

	Coding Styles
Java	<ul style="list-style-type: none">● Oracle: Code Conventions for the Java Programming Language● Google Android: Code Style Guidelines for Contributors
C++	<ul style="list-style-type: none">● Google C++ Style Guide● GCC Coding Conventions

❖ Coding standards

	Coding Standards
Java	<ul style="list-style-type: none">● CWE-660: Weaknesses in Software Written in Java
C++	<ul style="list-style-type: none">● MISRA C++: 2008● 방위사업청 무기체계 소프트웨어 코딩 규칙
C	<ul style="list-style-type: none">● MISRA C: 2012● 방위사업청 무기체계 소프트웨어 코딩 규칙

Constraint: Examples

Technical Constraints	Business Constraints
Programming Language Choice Anything that runs on the JVM.	Team Composition and Makeup Team X will build the XYZ component
Operating System or Platform It must run on Windows, Linux, and BeOS.	Schedule or Budget It must be ready in time for the Big Trade Show and cost less than \$80,000.
Use of Components or Technology We own DB2 so that's your database.	Legal Restrictions There is a 5GB daily limit in our license

Requirement vs Constraint

❖ Requirement

- Solution ideas that describe one particular way the user envisions meeting a need.
- Requirements can be negotiated depending on the situation

❖ Constraint

- Restrictions on the design or implementation choices available to the developer.
- Constraints can be imposed by external stakeholders, by other systems that interact with the system.
- Constraints should be respected and generally non-negotiable

Constraint Description

- ❖ To capture a constraint, describe the decision and its origin in a brief statement.

Constraint	Type	Origin	Context
Must ship by the end of Q3.	B	Brown	Avoids end of fiscal year budget issues.
Must be developed as open source software.	B	James	The City has an Open Data Policy that citizens can have access to source.
Must build a browser-based web application	T	Maria	Decreases concerns about software delivery and maintenance
Must support latest Firefox web browser	T	City IT	Officially supported Browser
Must be served from a Linux server.	T	City IT	City uses Linux and open source where possible

Q&A
