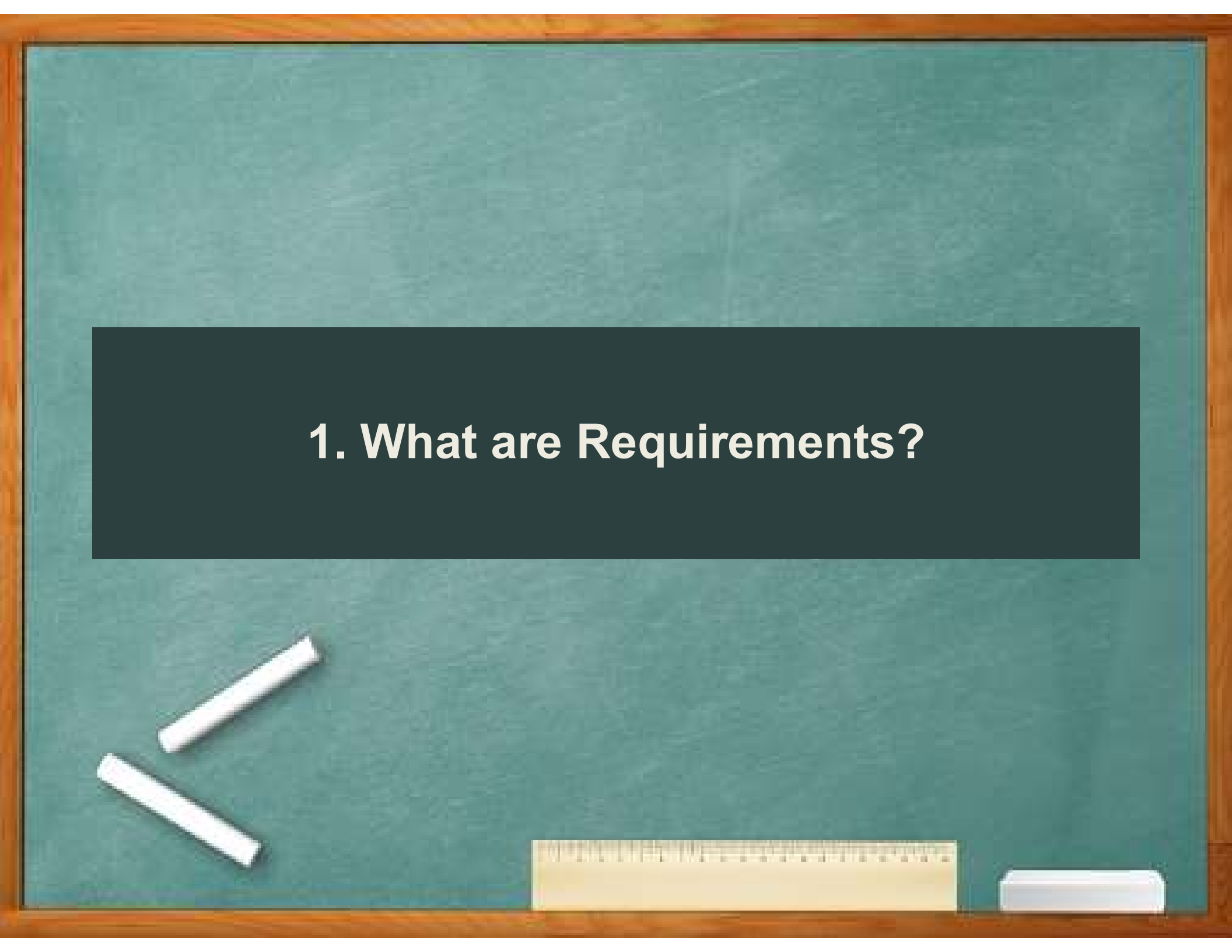
요구공학 및 품질속성
Requirements Engineering & Quality Attribute

서강대학교 기술경영전문대학원
박수진
psjdream@sogang.ac.kr

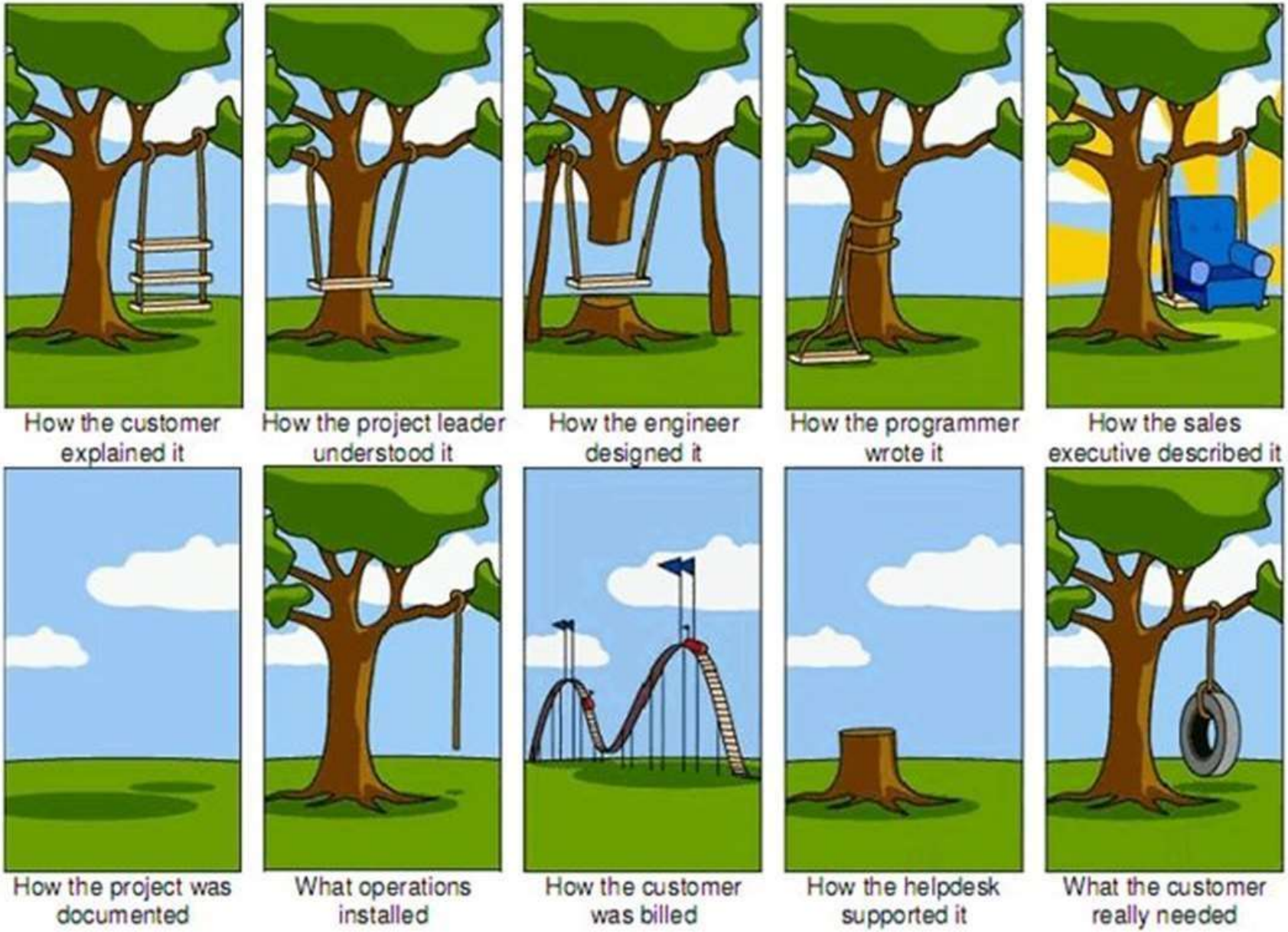
과정 구성

Day	주제	시간
Day 1	1. What are Requirements?	1.5hr
	2. Requirements Engineering Process Overview	1hr
	3. Feasibility Study	0.5hr
	4. Requirements Elicitation	1hr
	5. Requirements Analysis	3.5hr
	Exercise 1. Use Case Diagram 완성하기	0.5hr
	Exercise 2. Use Case Specification 작성하기	1hr
Day 2	Exercise 2. Review	1.5hr
	6. Quality Attribute Workshop	0.5hr
	7. Quality Attribute Scenario	2hr
	8. Requirements Specification	1.5hr
	9. Requirements Validation	1hr
	10. Requirements Management	1hr

1. What are Requirements?



Prologue


How the customer explained it
How the project leader understood it
How the engineer designed it
How the programmer wrote it
How the sales executive described it
How the project was documented
What operations installed
How the customer was billed
How the helpdesk supported it
What the customer really needed

What are Requirements?

- Requirements are defined during the early stages of a system development as a specification of what should be implemented or as a constraint of some kind on the system. (Ian Sommervile)
- They may be:
  - a user-level facility description,
  - a detailed specification of expected system behaviour,
  - a general system property,
  - a specific constraint on the system,
  - information on how to carry out some computation,
  - a constraint on the development of the system.
- A requirement is an expression of desired behaviour.

Types of Requirements

- **User requirements → Higher Level Requirements**
  - Statements in natural language, diagrams of the services the system provides and its operational constraints
  - Written for(from) customers
  - Defined

- **System requirements → More detailed Requirements**
  - Structured document setting out detailed descriptions of the system's functions, services and operational constraints.
  - Define what should be implemented to support user requirements
  - May be part of a contract between clients and contractors
  - Specified

User and System Requirements: Examples

User Requirement Definition

진료과별 월말 약제비 현황 보고서 생성

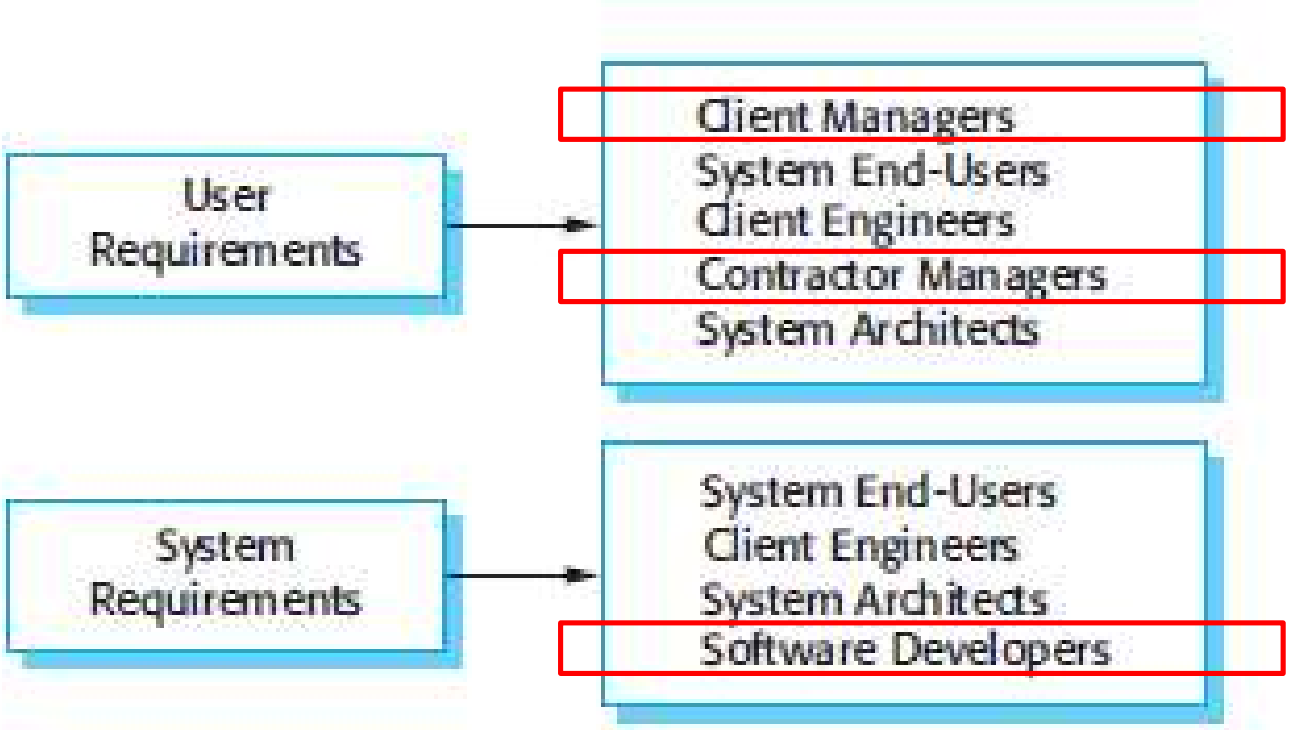
1. The MHC-PMS shall generate monthly management reports showing the cost of drugs prescribed by each clinic during that month.

System Requirements Specification

1.1 On the last working day of each month, a summary of the drugs prescribed, their cost, and the prescribing clinics shall be generated.
1.2 The system shall automatically generate the report for printing after 17.30 on the last working day of the month.
1.3 A report shall be created for each clinic and shall list the individual drug names, the total number of prescriptions, the number of doses prescribed, and the total cost of the prescribed drugs.
1.4 If drugs are available in different dose units (e.g., 10 mg, 20 mg) separate reports shall be created for each dose unit.
1.5 Access to all cost reports shall be restricted to authorized users listed on a management access control list.

* MHC-PMS: Mental Health Care Patient Management System

Readers of Different Types of Requirements Specification



Functional vs. Non-Functional Requirements

- **Functional requirements**
  - Statements of services which the system should provide
  - How the system should react to particular inputs
  - How the system should behave in particular situations
  - May state what the system should not do

- **Non-functional requirements**
  - Constraints on the services or functions offered by the system
    - Timing constraints
    - Constraints on the development process
    - Standards
  - Often apply to the system as a whole rather than individual features or services

- **Domain requirements**
  - Requirements that come from the application domain of the system
  - Reflect characteristics of the target domain
  - May be functional or non-functional or the both

Functional Requirements

- Describe functionality or system services.
- Depend on the type of software, expected users and the type of system where the software is used.
- Functional user requirements may be high-level statements of what the system should do.
- Functional system requirements should describe the system services in detail.

Functional Requirements: Examples

- Functional requirements for the MHC-PMS
  - A user shall be able to search the appointments lists for all clinics.
  - The system shall generate each day, for each clinic, a list of patients who are expected to attend appointments that day.
  - Each staff member using the system shall be uniquely identified by his or her 8-digit employee number. →user authentication에 대한 system의 behavior

Revisited: User and System Requirements: Examples

User Requirement Definition → **Functional User Requirement**

1. The MHC-PMS shall generate monthly management reports showing the cost of drugs prescribed by each clinic during that month.

System Requirements Specification → **Functional System Requirement**

1.1 On the last working day of each month, a summary of the drugs prescribed, their cost, and the prescribing clinics shall be generated.

1.2 The system shall automatically generate the report for printing after 17.30 on the last working day of the month.

1.3 A report shall be created for each clinic and shall list the individual drug names, the total number of prescriptions, the number of doses prescribed, and the total cost of the prescribed drugs.

1.4 If drugs are available in different dose units (e.g., 10 mg, 20 mg) separate reports shall be created for each dose unit.

1.5 Access to all cost reports shall be restricted to authorized users listed on a management access control list.

* MHC-PMS: Mental Health Care Patient Management System

Non-Functional Requirements

- **Define system properties and constraints**
  - Reliability, Response time, Storage requirements
  - Constraints on I/O device capability
  - System representations, Etc.

- **The challenge of NFRs**
  - Hard to model
  - Usually stated informally
    - Often contradictory, difficult to enforce during development
    - Difficult to evaluate for the customer prior to delivery
  - Hard to make them measurable requirements
    - We'd like to state them in a way that we can measure how well they've been met

- Often called **Quality Attributes** or **Quality Requirements**
  - Often called just the "-ilities"
  - Non-functional requirements may be more critical than functional requirements.
  - If these are not met, the system is totally useless.

- Critical systems often include non-functional requirements into mandatory requirements.
  - ex. 의료기기/ 무기제어/ 원자력 발전.. 생명 혹은 자산의 손실을 야기할 수 있는 system들

Classification of Non-Functional Requirements

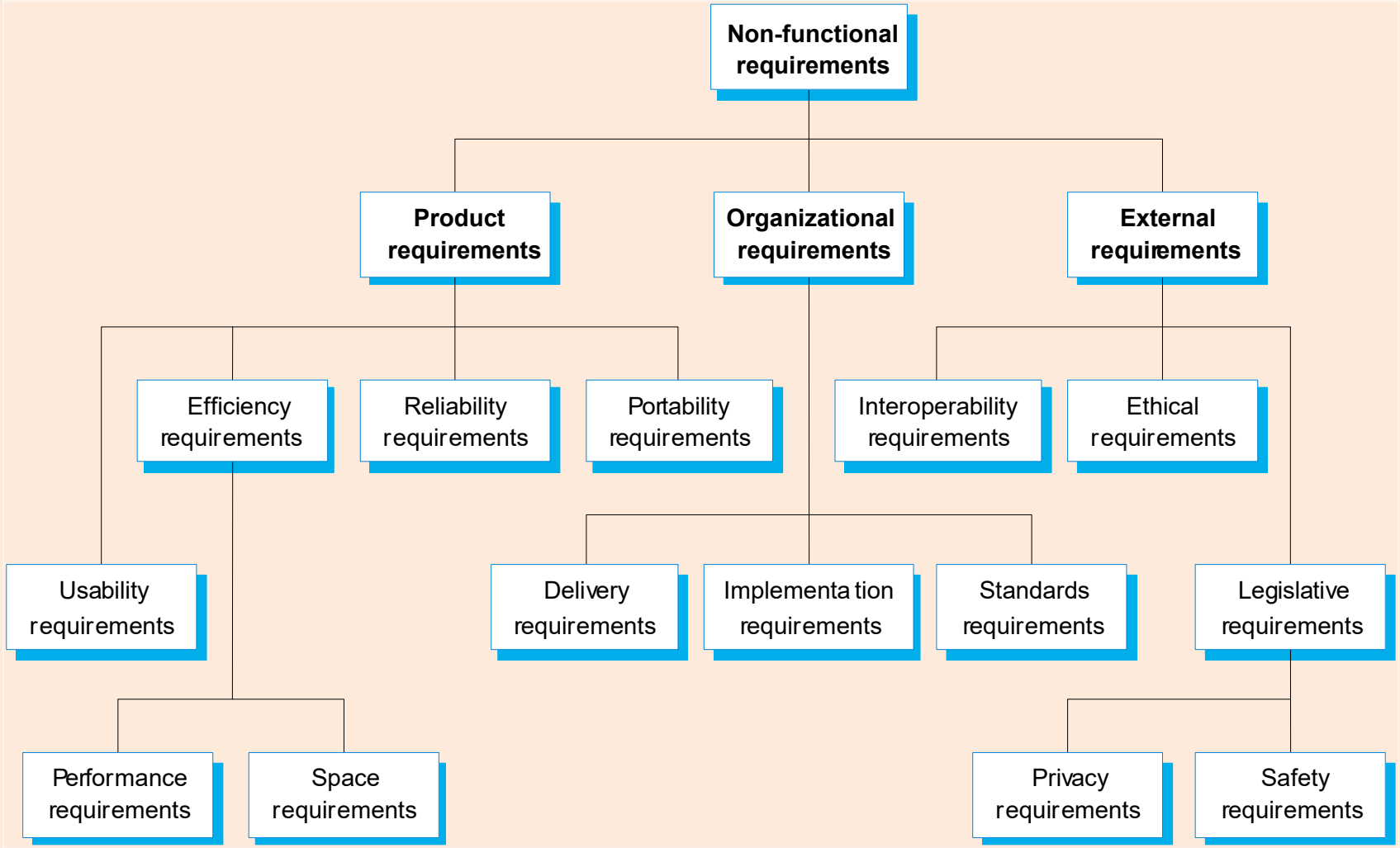
- Three types of non-functional requirements(Ian Sommervile)
  - **Product requirements**
    - Specify that the delivered product must behave in a particular way
    - e.g., execution speed, reliability, etc.
  - **Organizational requirements**
    - Requirements which are a consequence of organizational policies and procedures
    - e.g., process standards, implementation requirements, etc.
  - **External requirements**
    - Requirements which arise from the factors external to the development process
    - e.g. interoperability requirements, legislative requirements, etc.

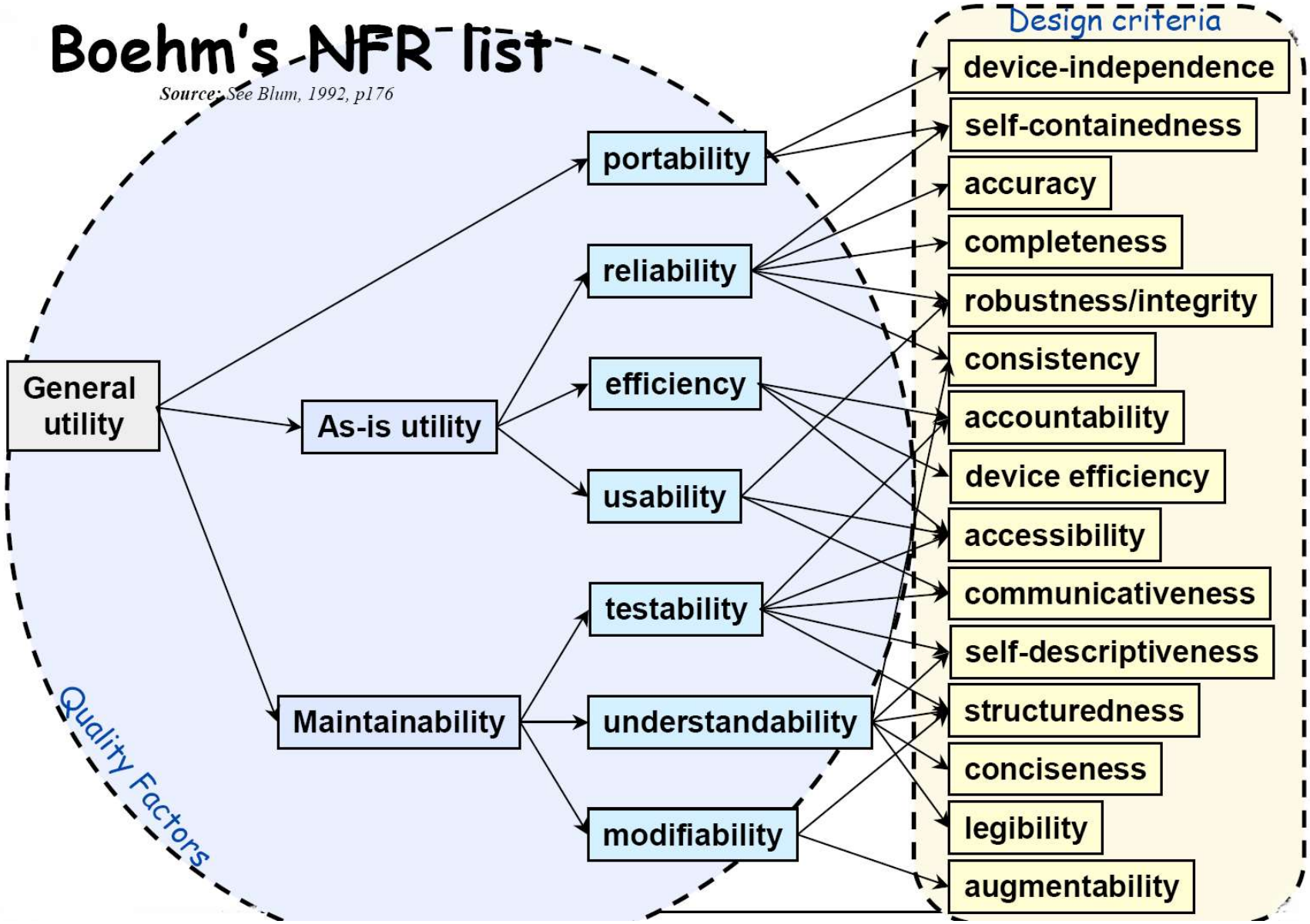
Non-functional Requirements: Example

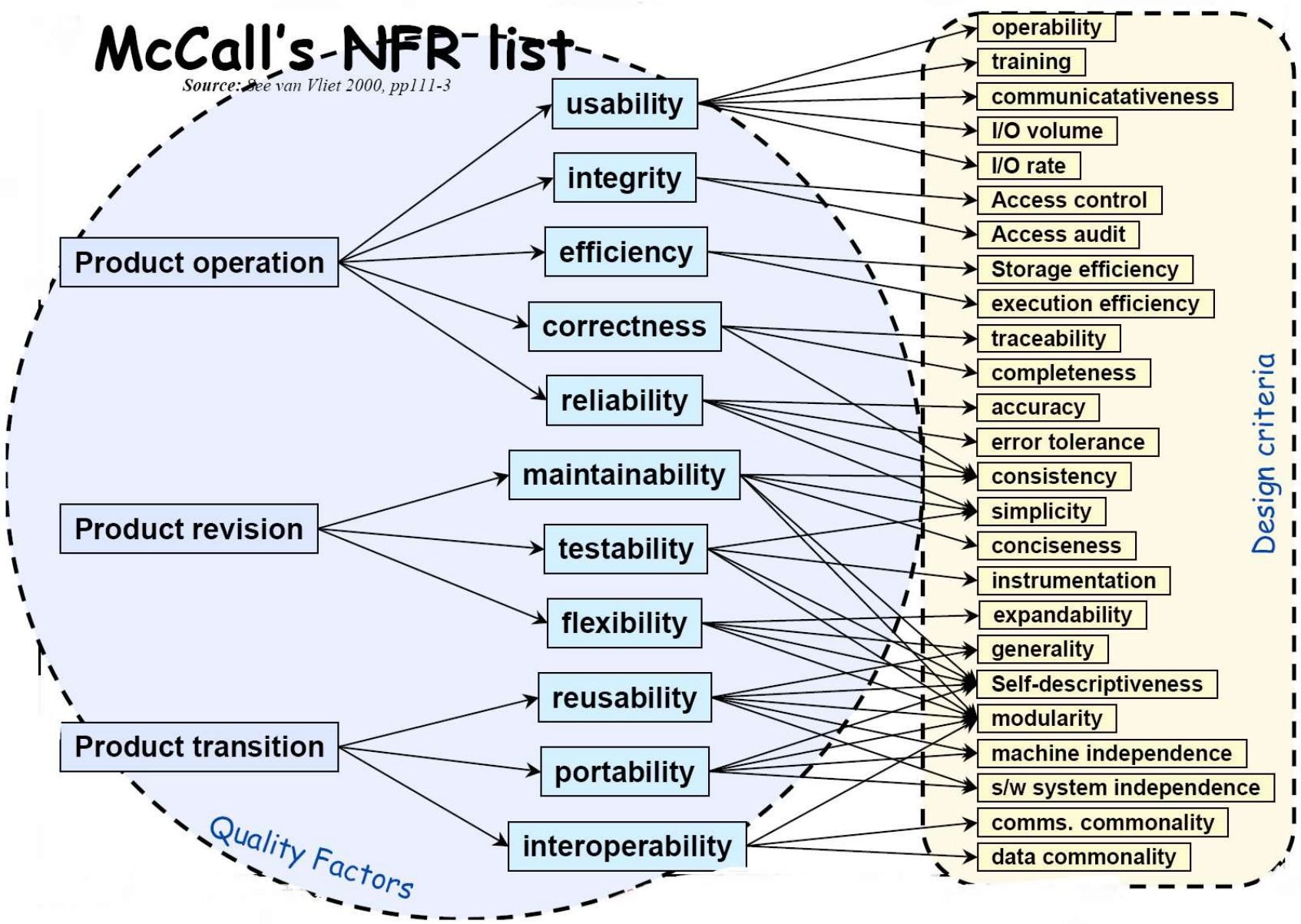
- MHC-PMS (Mental Health Care Patient Management System)
  - Product requirements
    - The MHC-PMS shall be available to all clinics during normal working hours (Mon–Fri, 0830–17.30). Downtime within normal working hours shall not exceed five seconds in any one day. → Availability
    - The system should be easy to use by medical staff and should be organized in such a way that user errors are minimized. → Usability(But not a good specification)
  - Organizational requirements(제품 자체의 품질과는 무관)
    - Users of the MHC-PMS system shall authenticate themselves using their health authority identity card. → Security

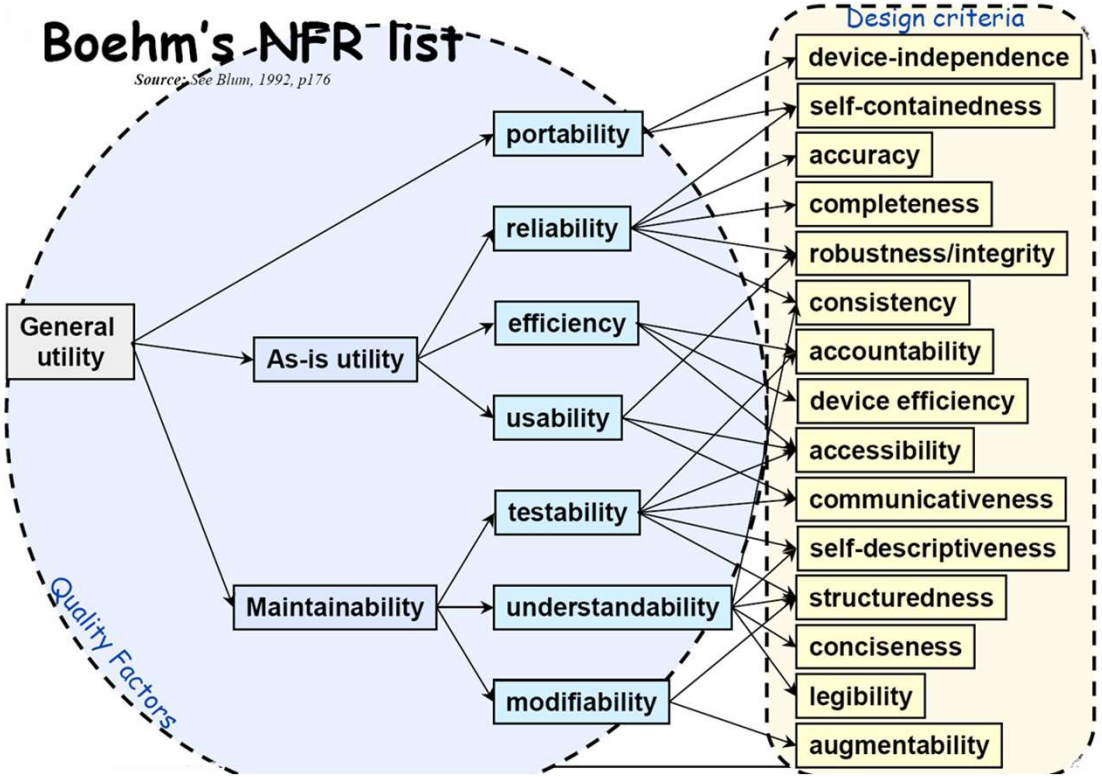
  - Each staff member using the system shall be uniquely identified by his or her 8-digit employee number. →security issue 관련 Non-functional Requirements 범주에 넣을 수도 있으나, user authentication에 대한 system의 behavior로 보고 Functional Requirements 범주에 포함 시킬 수도 있음
  - External requirements
    - The system shall implement patient privacy provisions as set out in HStan-03-2006-priv. → Standard compliance

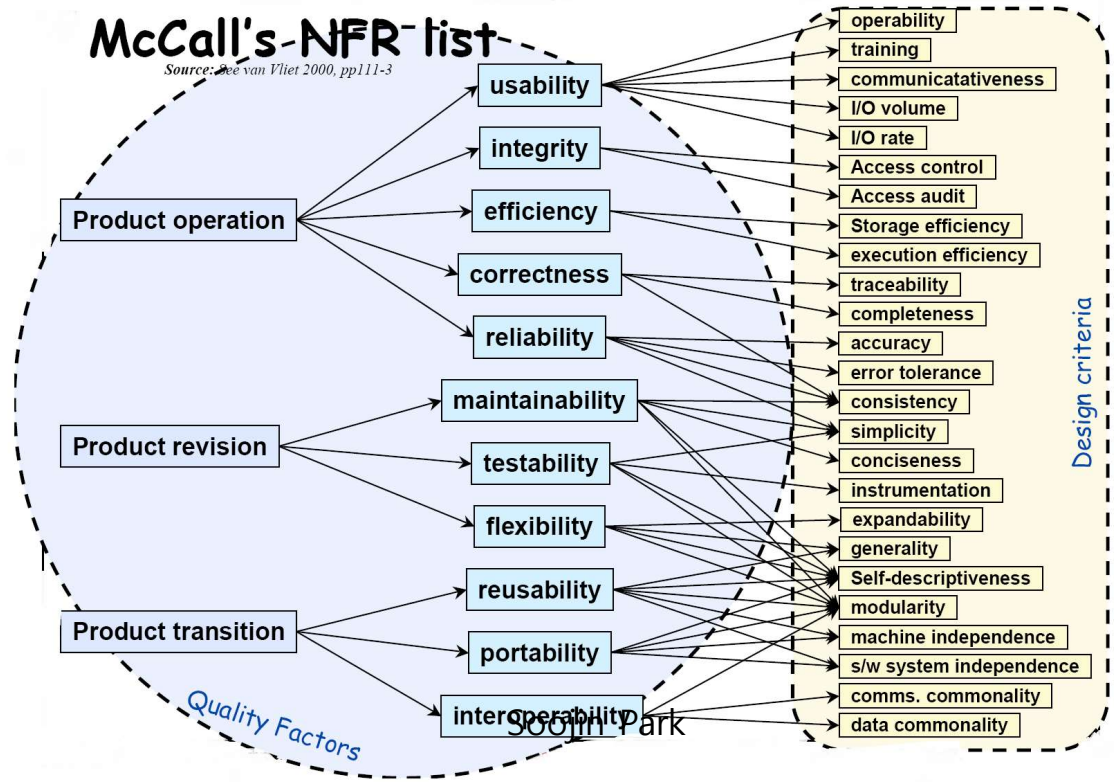
3 Types of Non-Functional Requirements




Boehm's NFR list
Source: See Blum, 1992, p176
General utility
As-is utility
Maintainability
portability
reliability
efficiency
usability
testability
understandability
modifiability
Quality Factors
Design criteria
device-independence
self-containedness
accuracy
completeness
robustness/integrity
consistency
accountability
device efficiency
accessibility
communicativeness
self-descriptiveness
structuredness
conciseness
legibility
augmentability


McCall's-NFR-list
Source: See van Vliet 2000, pp111-3
Product operation
Product revision
Product transition
usability
integrity
efficiency
correctness
reliability
maintainability
testability
flexibility
reusability
portability
interoperability
Quality Factors
operability
training
communicatativeness
I/O volume
I/O rate
Access control
Access audit
Storage efficiency
execution efficiency
traceability
completeness
accuracy
error tolerance
consistency
simplicity
conciseness
instrumentation
expandability
generality
Self-descriptiveness
modularity
machine independence
s/w system independence
comms. commonality
data commonality
Design criteria


Boehm's NFR list
Source: See Blum, 1992, p176
General utility
As-is utility
Maintainability
portability
reliability
efficiency
usability
testability
understandability
modifiability
Quality Factors
Design criteria
device-independence
self-containedness
accuracy
completeness
robustness/integrity
consistency
accountability
device efficiency
accessibility
communicativeness
self-descriptiveness
structuredness
conciseness
legibility
augmentability


McCall's NFR list
Source: See van Vliet 2000, pp111-3
Product operation
Product revision
Product transition
usability
integrity
efficiency
correctness
reliability
maintainability
testability
flexibility
reusability
portability
interoperability
Quality Factors
Design criteria
operability
training
communicatativeness
I/O volume
I/O rate
Access control
Access audit
Storage efficiency
execution efficiency
traceability
completeness
accuracy
error tolerance
consistency
simplicity
conciseness
instrumentation
expandability
generality
Self-descriptiveness
modularity
machine independence
s/w system independence
comms. commonality
data commonality

FURPS+

- Classification system devised by Robert Grady at Hewlett-Packard
- FURPS
  - Functionality
    - ✓ Main product feature
    - ✓ Architecturally Significant Functional Requirements: Auditing, Licensing, Localization, Mail, Online help, Security, System management, Printing, etc.
  - Usability
  - Reliability
  - Performance
  - Supportability: testability, adaptability, maintainability, compatibility, configurability, installability, scalability, and localizability

Architecturally Significant→ 어느 하나의 function에만 적용되는 것이 아니라, 다양한 function들을 cross-cutting 하여 적용되어야 하는 기능들

- The "+" in FURPS+ also helps us to remember concerns such as:
  - Design requirements
  - Implementation requirements
  - Interface requirements
  - Physical requirements(HW requirements)

Non-functional Requirements Implementation

홍채인식을 활용한 보안 기능 강화 → Security Issue → Non-functional Requirements

업계 관계자는 "갤럭시S8 출시로 생체인증이 다시 활성화될 것으로 기대된다"며 "홍채인식과 공인인증서를 접목한 방식은 안전하면서도 편리한 서비스"라고 말했다. 우리은행 관계자도 "노트7때 선보였던 홍채인증 서비스가 편의성에 힘입어 사용자들의 호응을 얻었다"며 서비스 재개에 대한 기대감을 숨기지 않았다.

반면 보안에 대한 우려는 여전히 존재한다. 아이디와 비밀번호의 경우, 유출이 되더라도 변경할 수 있지만 생체 정보는 바꿀 수가 없다. 이에 삼성전자는 지문과 홍채 등의 생체 정보를 네트워크를 통해 서버로 보내지 않고, 단말기 내부에 저장하는 방식을 택했다. 한국인터넷진흥원 관계자는 "갤럭시S8은 생체 정보를 단말기 내부의 트러스트존에 암호화해 저장하고, 등록된 것과의 비교작업을 거친다"며 "외부로 정보가 유출되지 않아 보안성을 갖췄다"고 말했다.

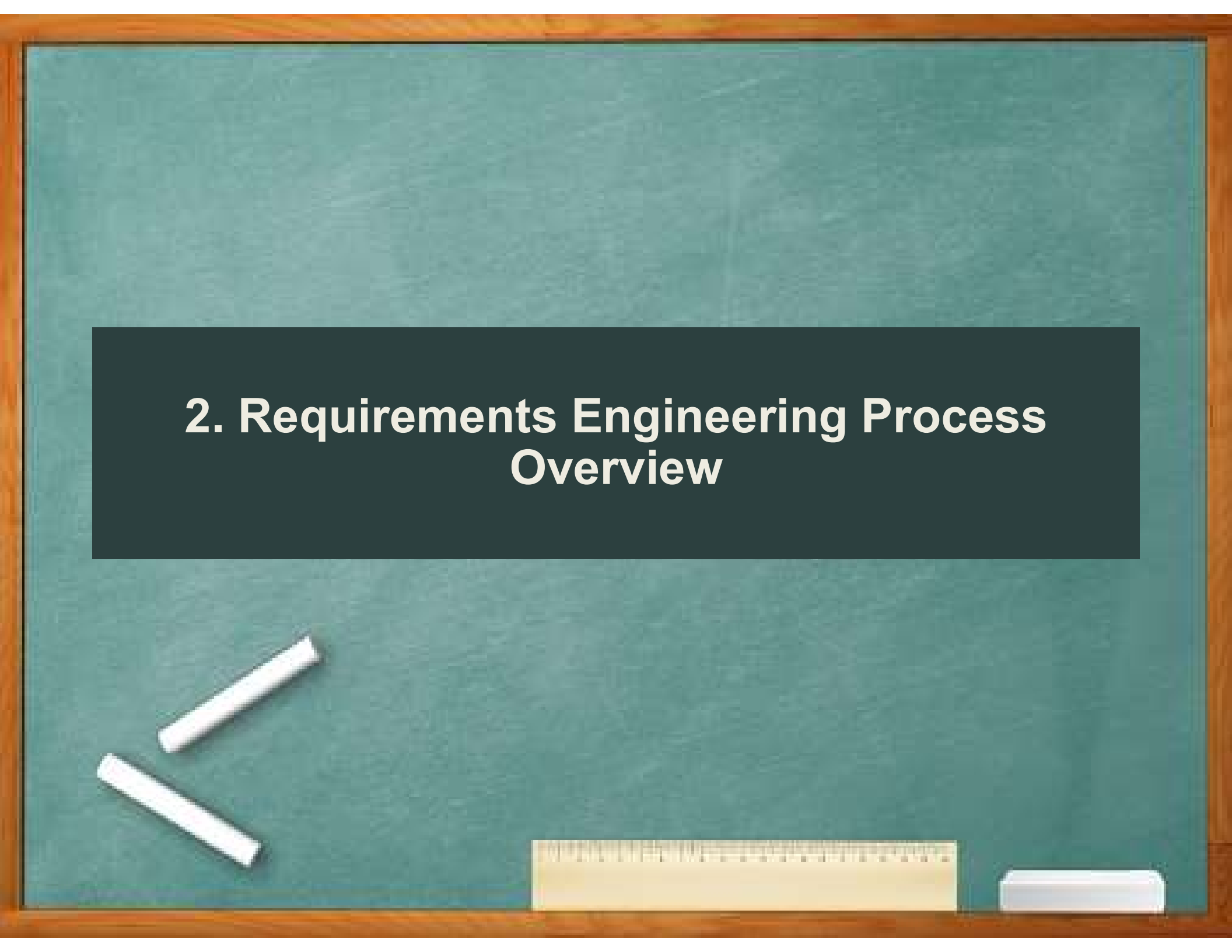
→ 여러 가지의 Functional Requirements를 파생시킴

Functional Requirements vs. Non-functional Requirements

Functional Requirements	Non-functional Requirements
• 내/외부로부터의 stimuli에 대한 System Behavior를 기술함	• FR이 시스템 상에 implement되었을 때의 expectation level을 기술함
• 각 requirement가 시스템에 구현되거나(PASS), 혹은 그렇지 못했거나(FAIL), 둘 중 한가지 경우로 판단 가능	• Quality Attribute별로 시스템이 만족시켜야 하는 range를 만족시켰는지를 검증 가능 → 각 요구사항이 How much/many에 대한 답을 계측 가능한 숫자형태로 명세 되어야 함
• 각 기능적 요구사항은 **특정 기능에 국한됨**	• 비기능적 요구사항은 **시스템 전체 범위에 적용됨**
→ 각 기능적 요구사항의 구현은 개발자의 몫	→ 비기능적 요구사항의 구현은 Architect의 몫
→ 기능적 요구사항의 변경 파급효과는 상대적으로 시스템의 특정 컴포넌트(들)에 제한적	→ 비기능적 요구사항의 변경은 전체 시스템의 architecture 변경으로 이어질 수 있음
• 모든 기능적 요구사항을 시스템에 implement하여 만족시키는 것이 1차적 목표→ 시스템의 성쇠를 판가름할 수는 없음	• 시스템에 implement된 **비기능적 요구사항의 수준에 따라 시스템의 성공과 실패가 결정됨**

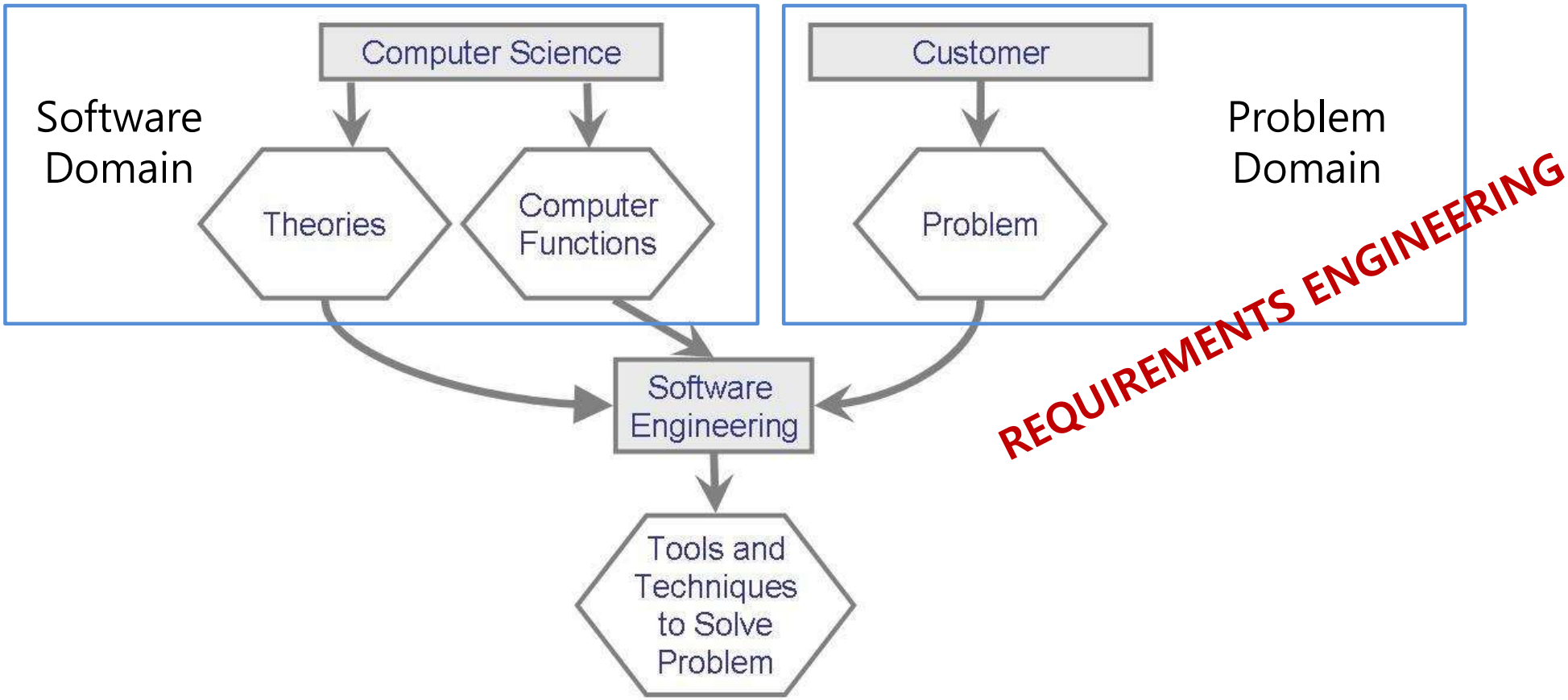
Software Architect의 Capability

2. Requirements Engineering Process Overview



Requirements Engineering

- Relationship between computer science and software engineering & requirements engineering



Requirements Engineering

Quality means fitness-for-purpose.
Cannot say anything about quality unless you understand the purpose.

Communication is as important as the analysis.

Not a phase or stage

- **Requirements Engineering (RE)** is a set of activities concerned with identifying and communicating the purpose of a software-intensive system, and the contexts in which it will be used. Hence, RE acts as the bridge between the real world needs of users, customers, and other constituencies affected by a software system, and the capabilities and opportunities afforded by software-intensive technologies

and partly about what is possible

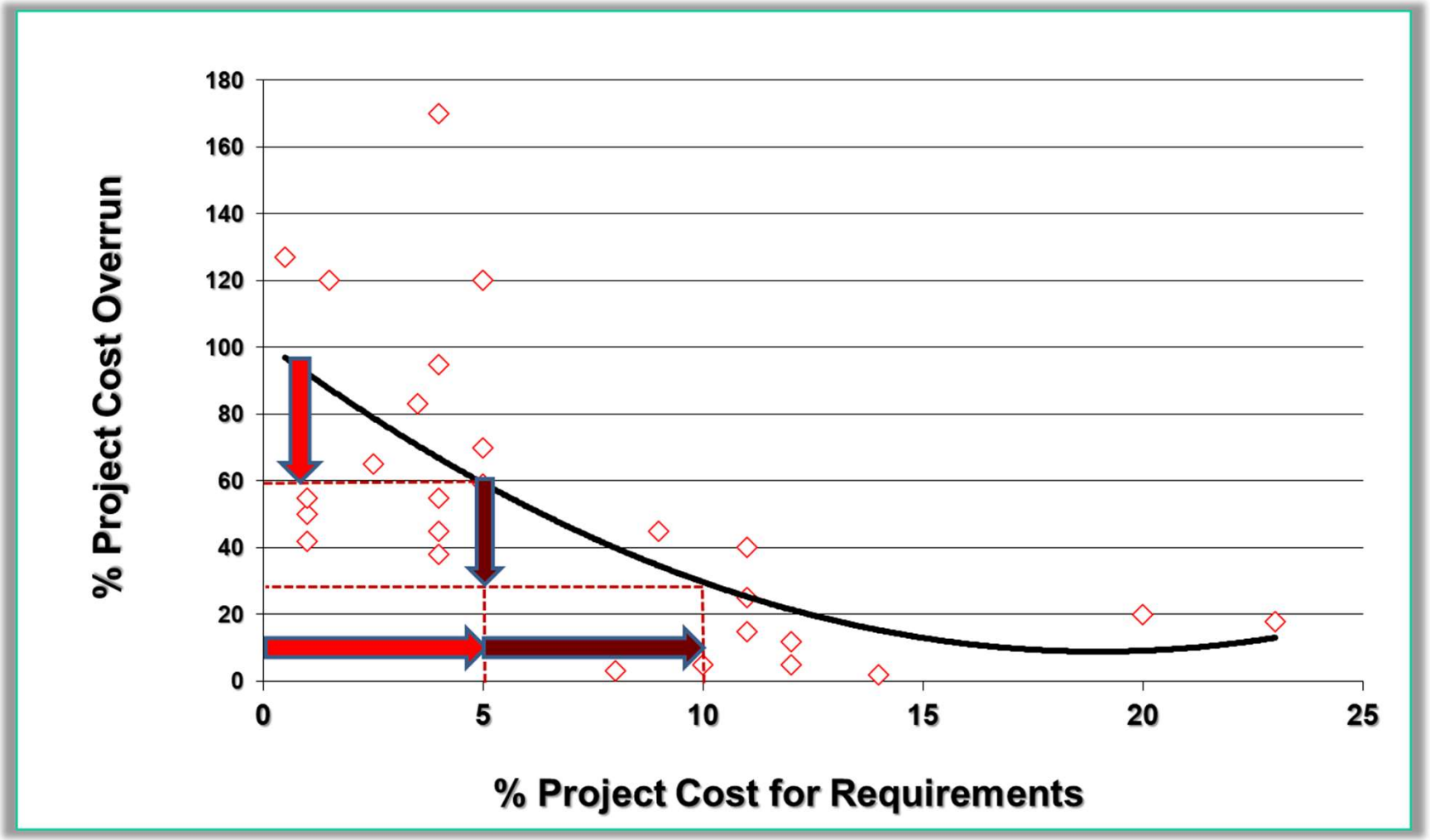
Designers need to know how and where the system will be used.

Requirements are partly about what is needed.

Need to identify all the stakeholders - not just the customer and user.

Are Requirements Important?

- **How Much Time Should We Spend on Requirements?**

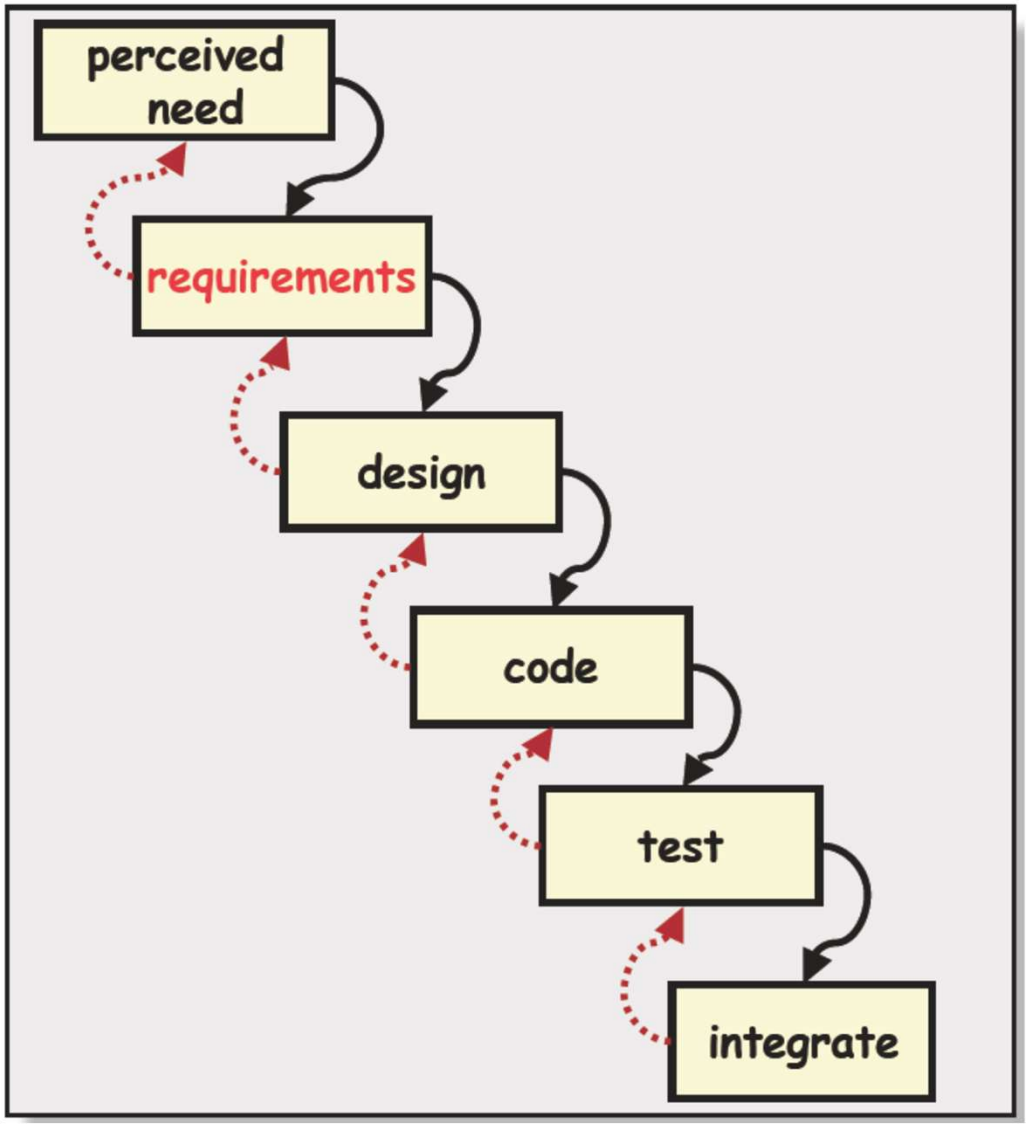


SDLC and RE Process

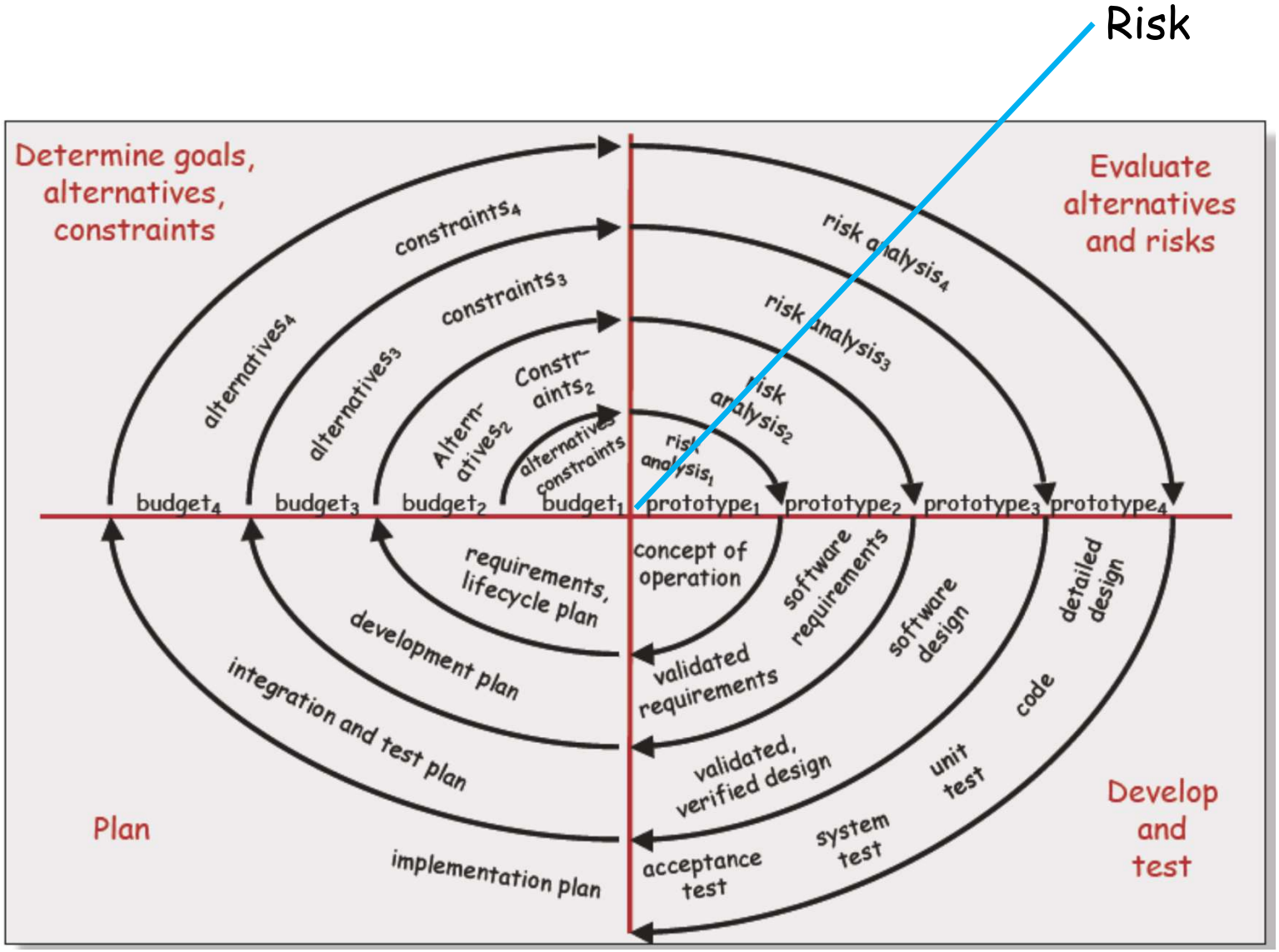
- Requirements engineering process should be adapted to a specific SDLC.
  - **RE process + Development process**

- **Software Development LifeCycle (SDLC)** models
  - Waterfall
  - Incremental, Evolutionary
  - Spiral, Iterative, Agile, RUP

Waterfall Model



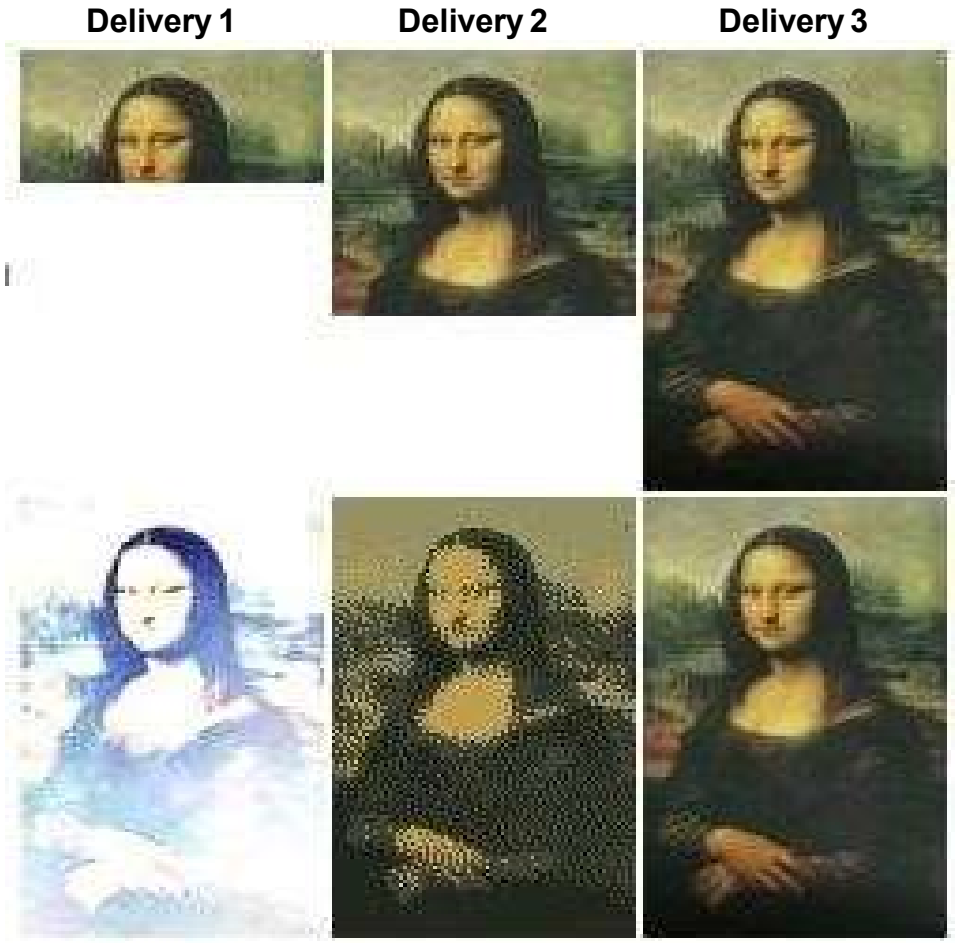
The Spiral Model

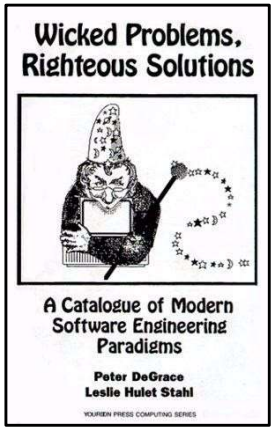


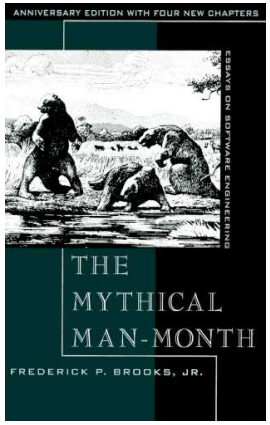
Incremental vs. Iterative(Evolutionary) Development

Incremental(점진적) Plan

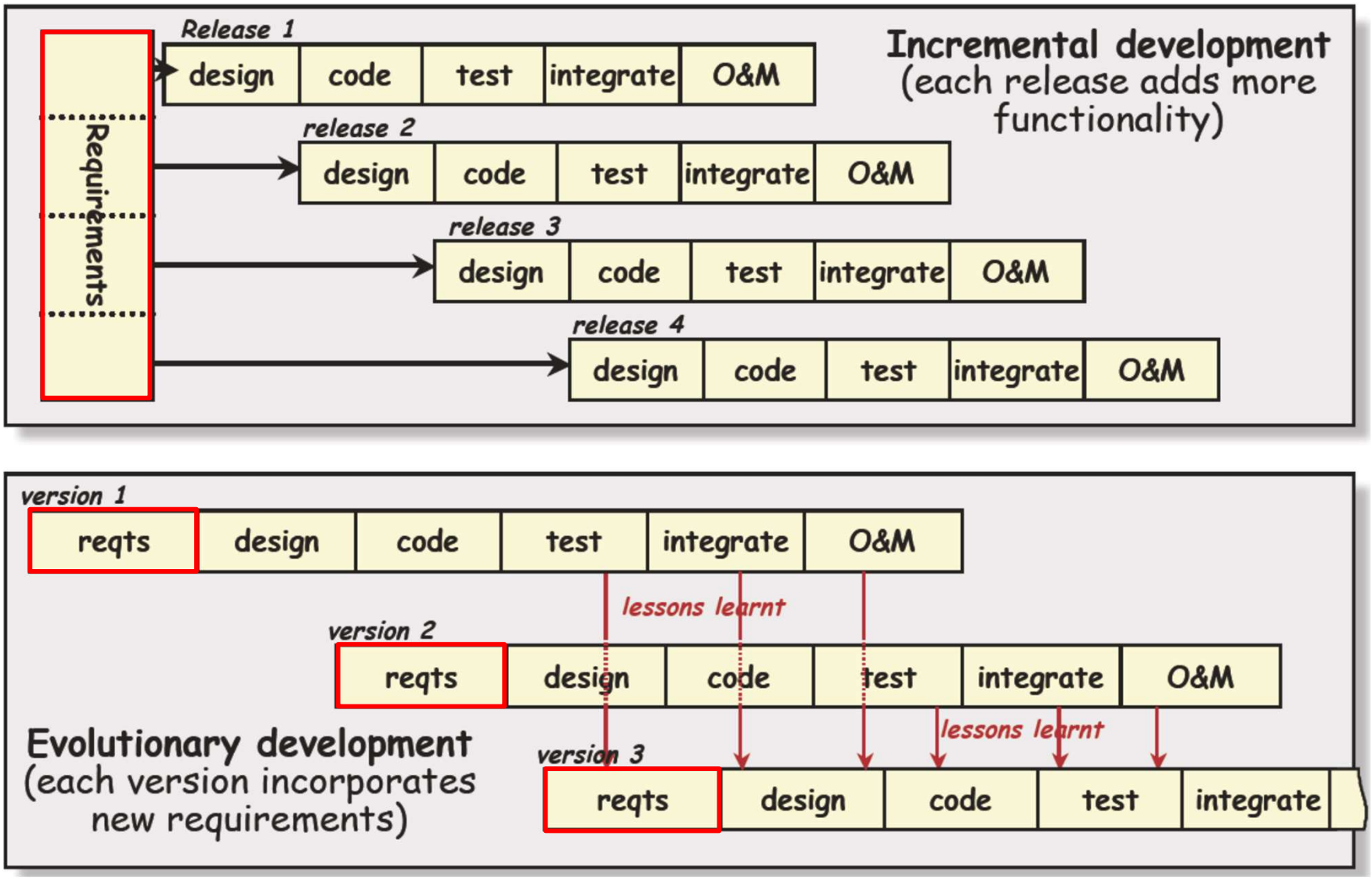
Iterative(반복적) Plan



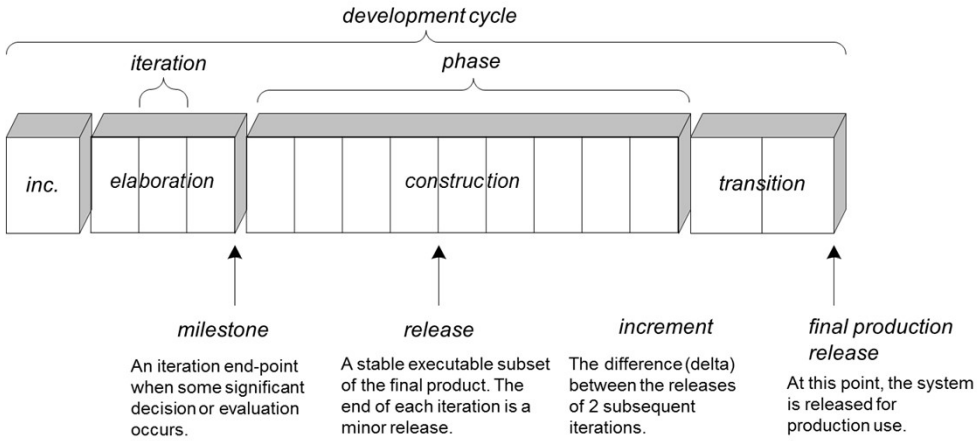


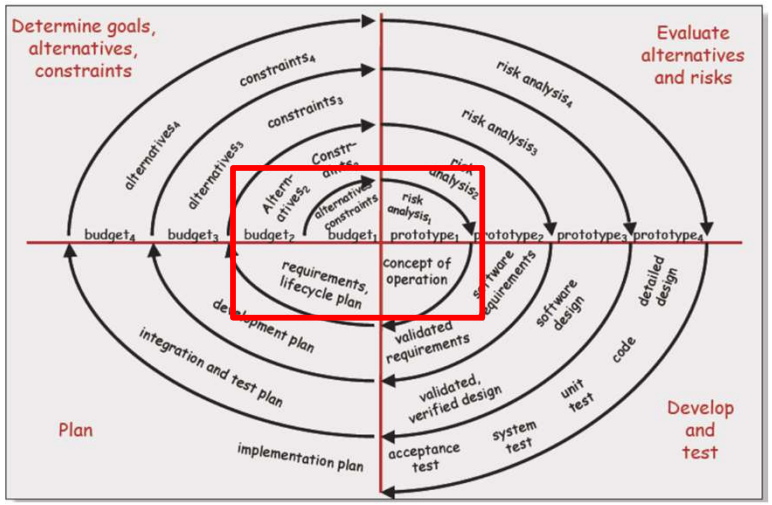


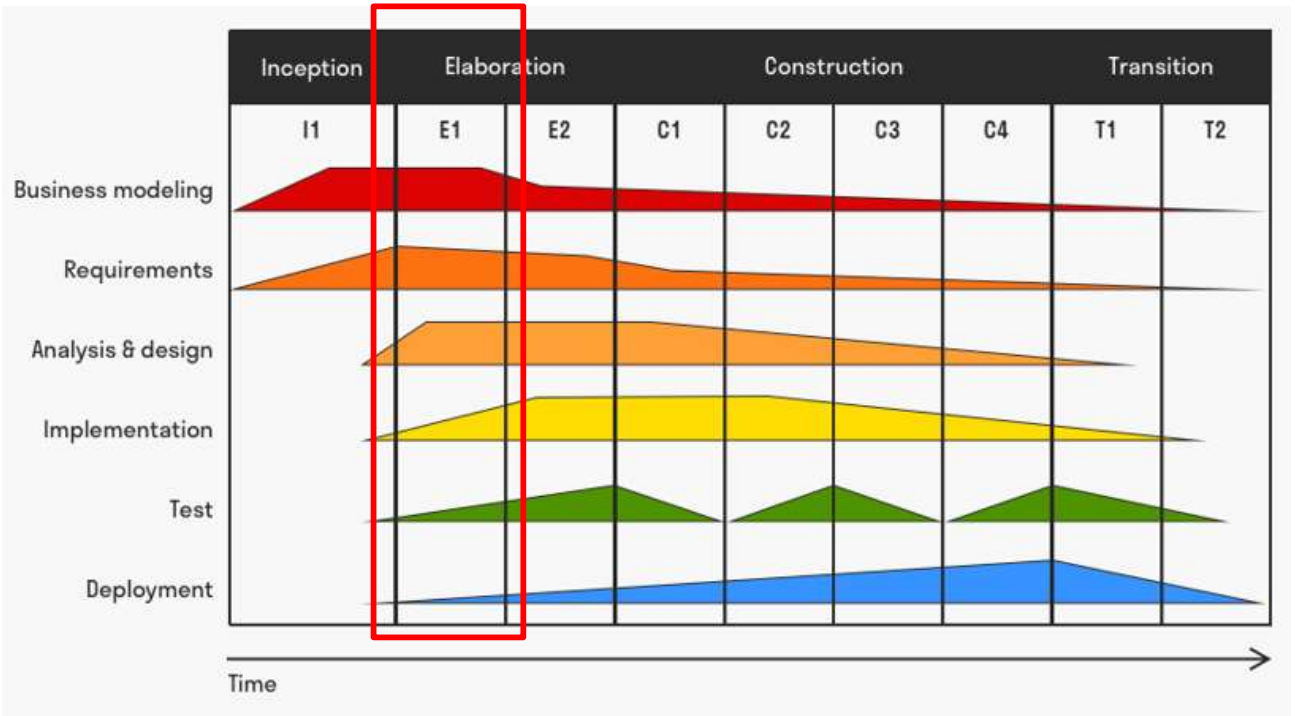
Phased Lifecycle Models



Evolved into (Rational) Unified Process

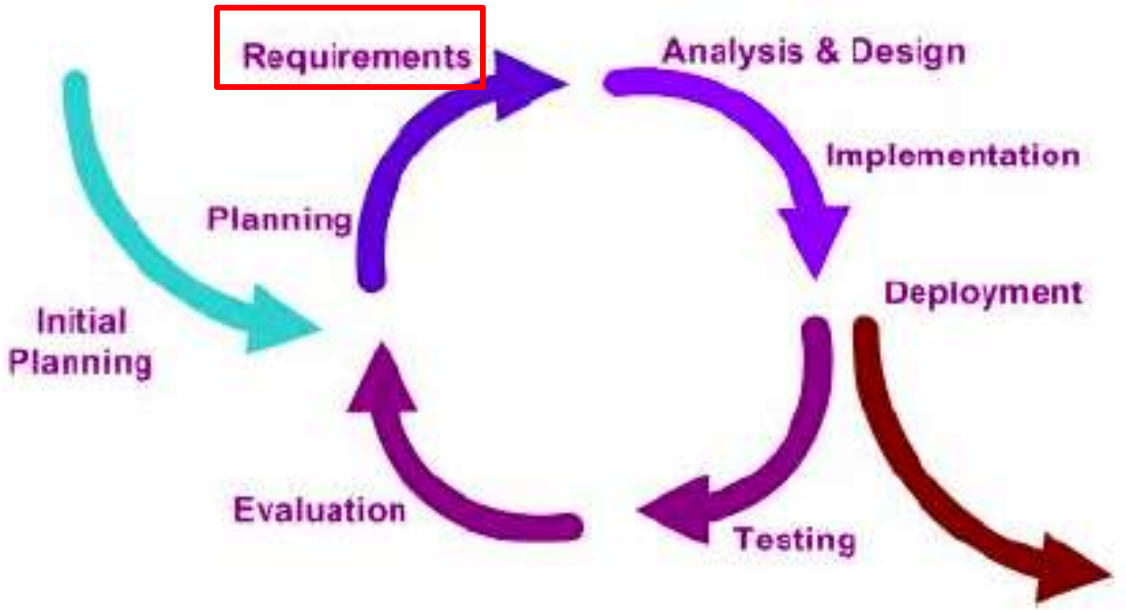






Agile Models: XP, Scrum, Crystal…

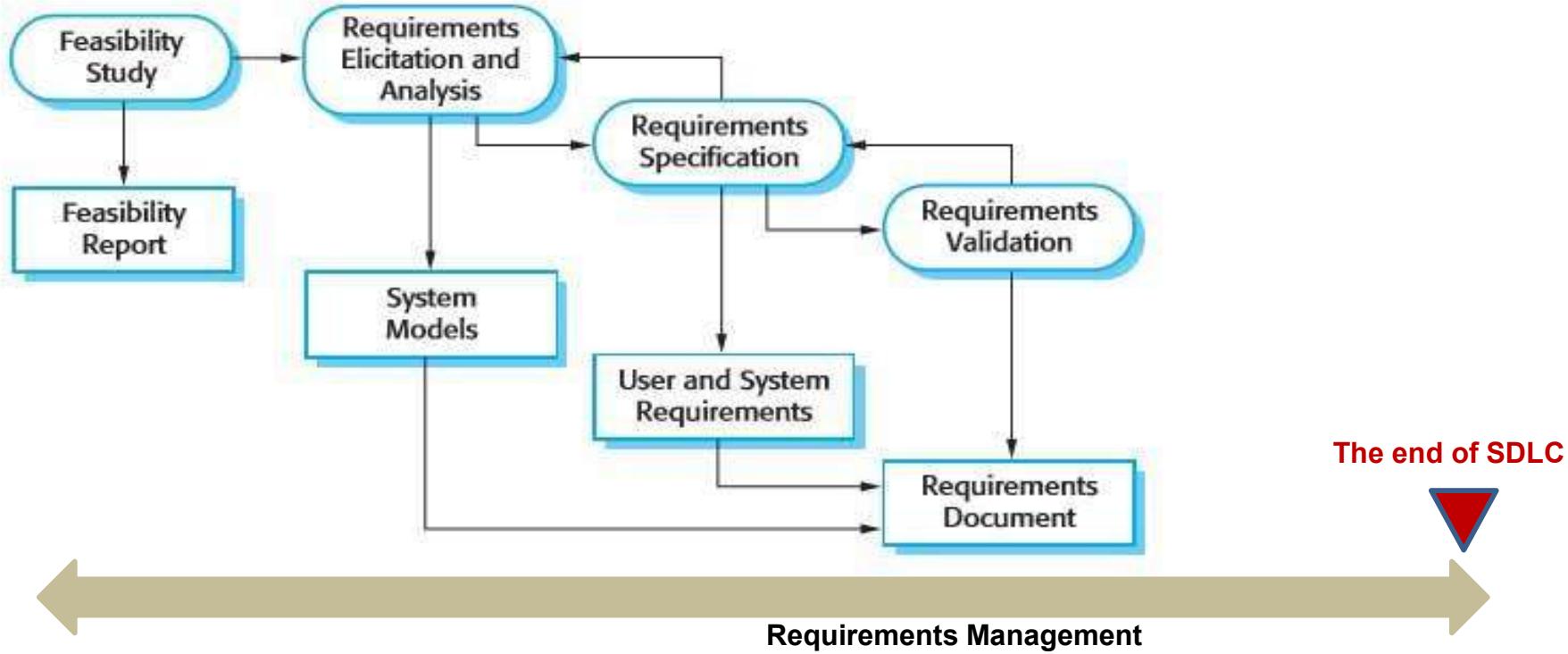
- **Basic Philosophy of Agile**
  - Individual over processes and tools
  - Working software over documentation
  - Customer collaboration over contract negotiation
  - Responding to change over following a plan



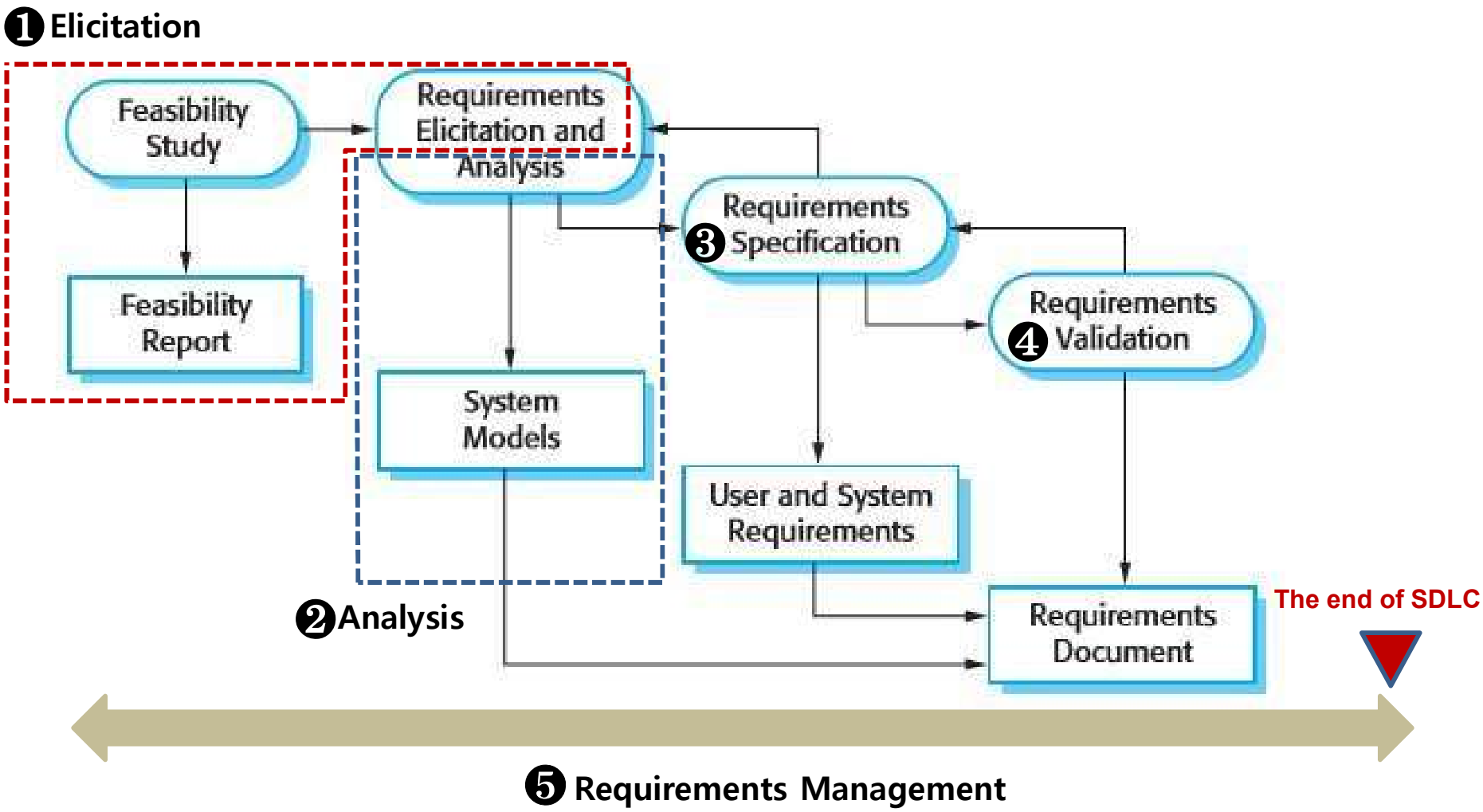
Requirements Engineering Processes(Ian Sommervile)

- **Requirement engineering processes** vary depending on
  - Application(target) domain
  - People involved
  - Organization developing the requirements
  - Software development processes used

- **Generic activities** common to all RE processes :



Requirements Engineering Process in this course



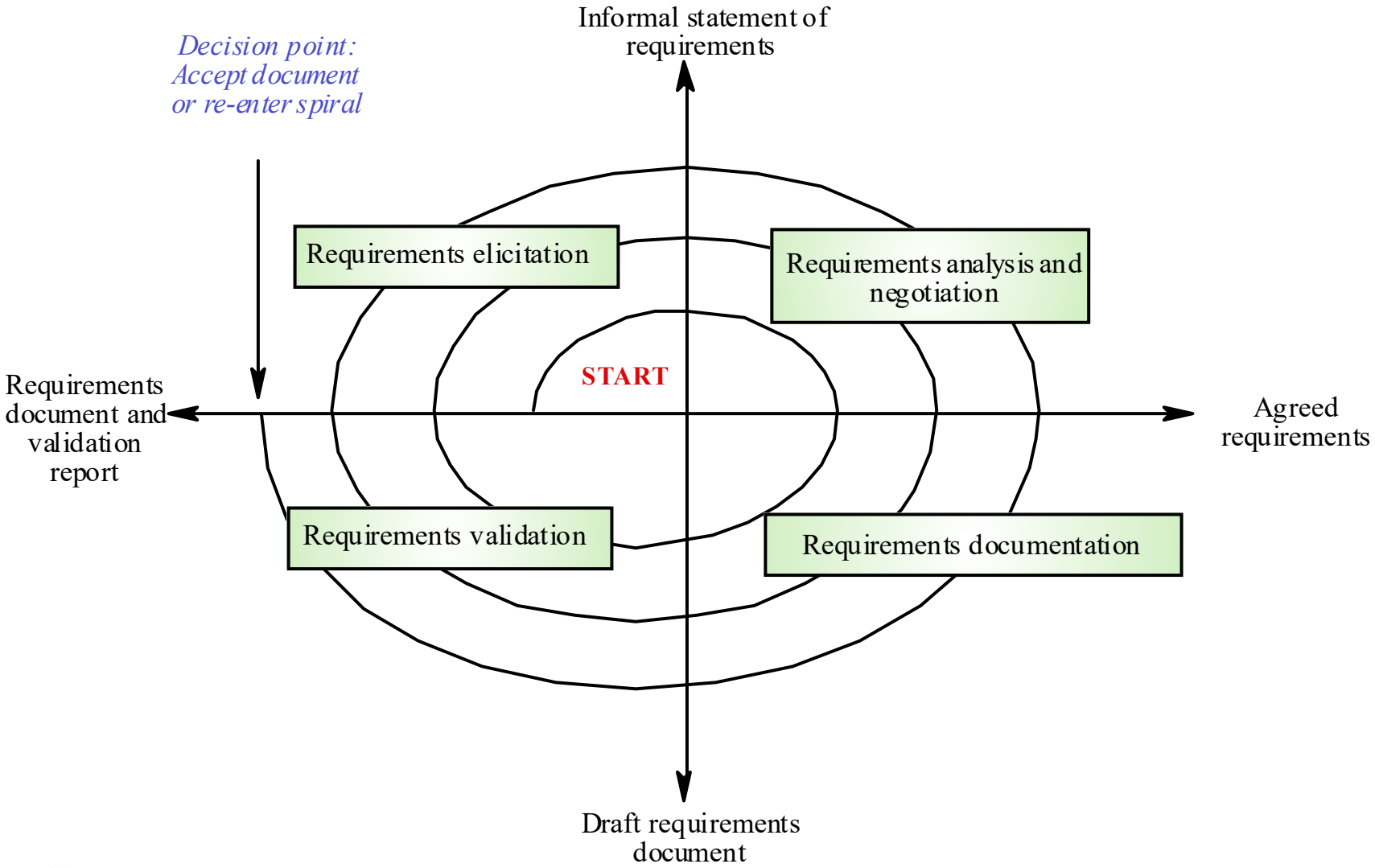
Requirements Engineering Processes

- 5 generic activities common to all processes
  - Requirements elicitation
  - Requirements analysis
  - Requirements specification
  - Requirements validation
  - Requirements management

요구사항 추출/분석/명세/검증/관리

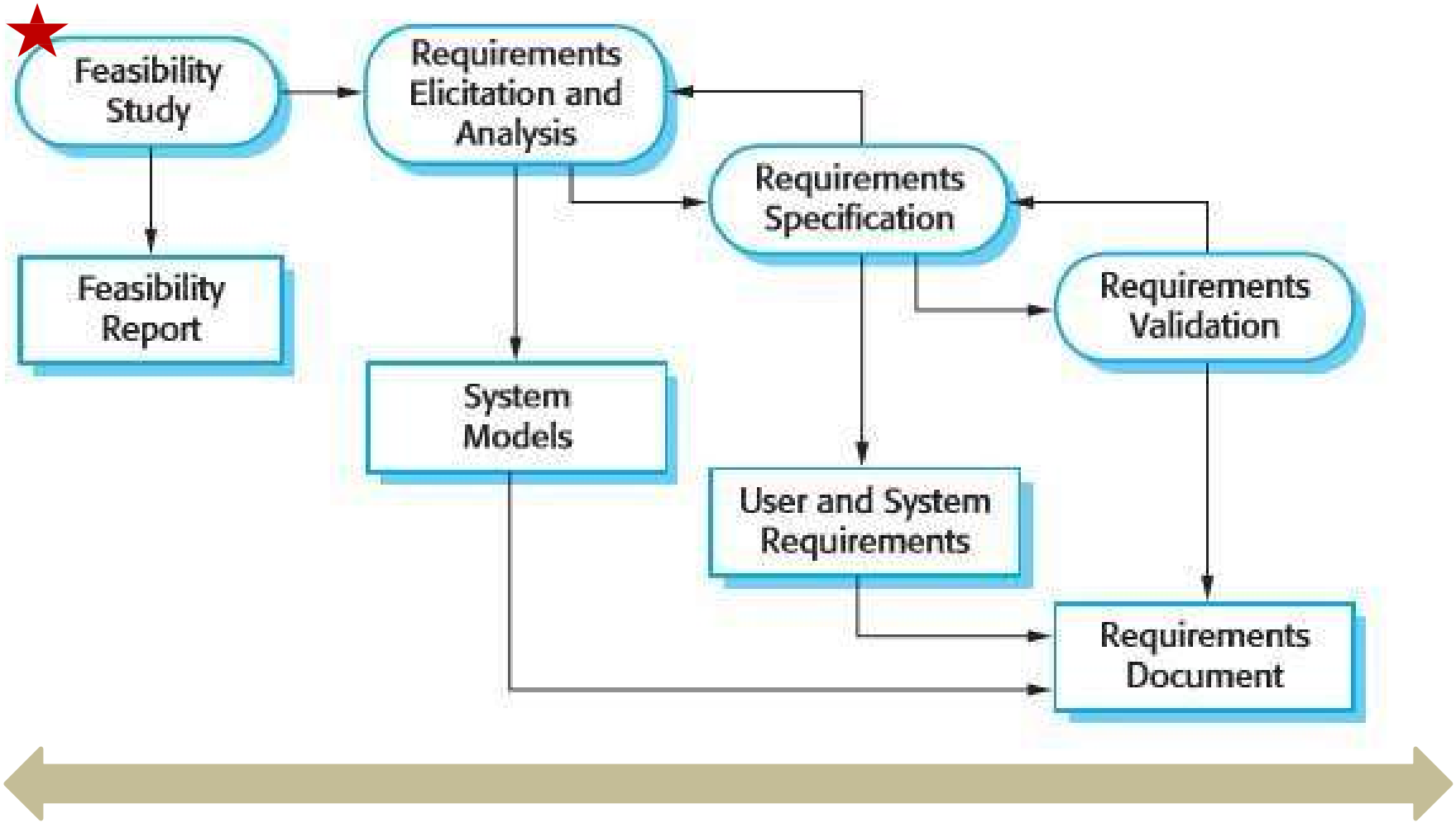
A spiral view of the RE process

- In practice, RE is an iterative activity in which these processes are interleaved

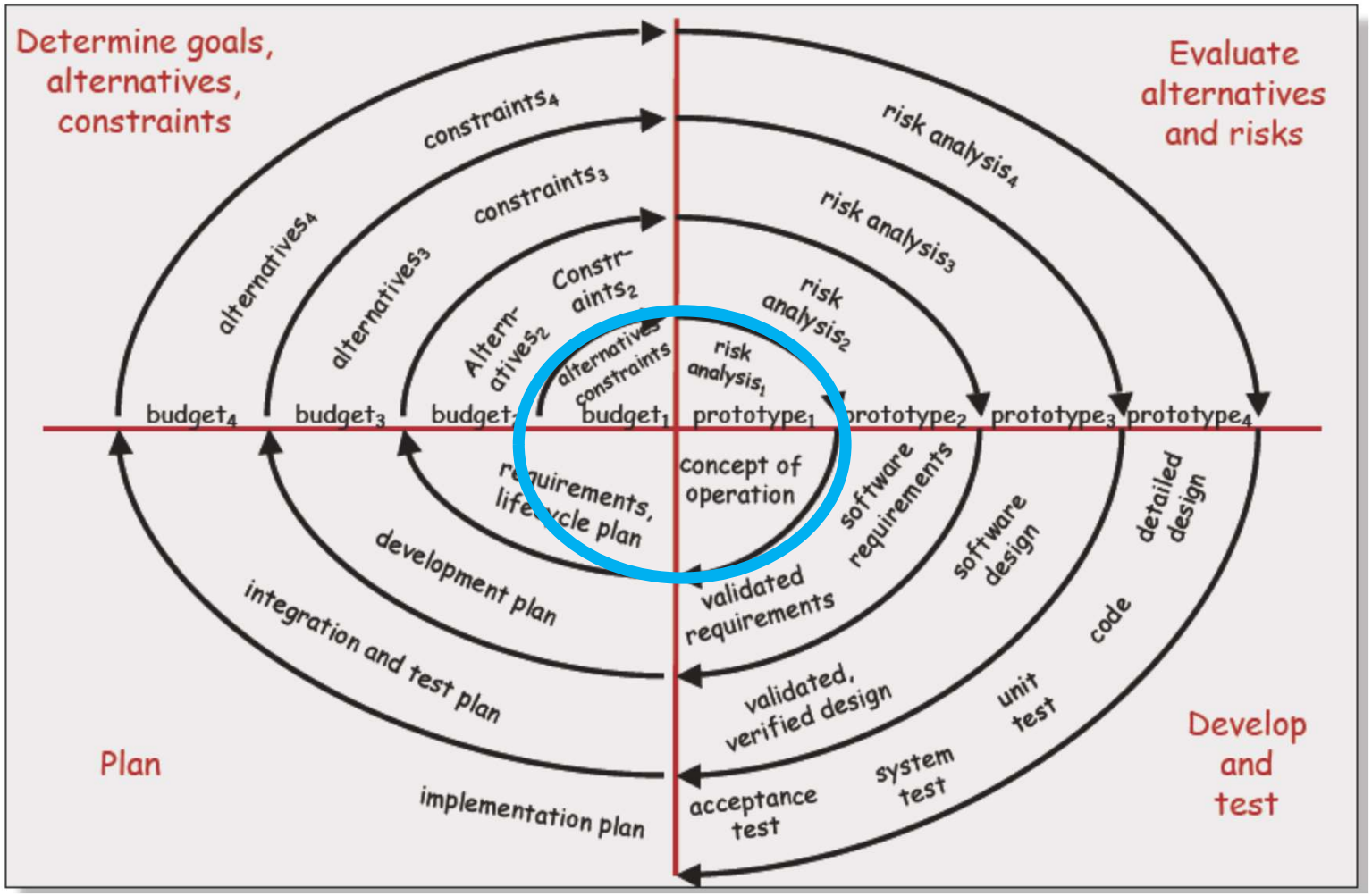


3. Feasibility Study

Requirements Engineering Process



Feasibility Study Phase in The Spiral Model



Why a Feasibility Study?

제안서 작성 단계

Master plan 수립단계

Benchmarking 단계

- **Objectives:**
  - **To find out if** a system development project can be done:
    - "Is it possible?"
    - "Is it justified?"
  - To suggest possible alternative solutions.
  - To provide **management** with enough information to know:
    - Whether the project can be done
    - Whether the final product will benefit its intended users
    - What the alternatives are
    - Whether there is a preferred alternative

- **A management-oriented activity:**
  - After a feasibility study, management makes a "***go/stop***" decision.
  - Need to examine the problem in the context of broader business strategy

Content of Feasibility Study

- **Things to be studied in the feasibility study:**
  - The present (**existing**) organizational system
    - Stakeholders, users, policies, functions, objectives
  - **Problems** with the present system
    - inconsistencies, inadequacies in functionality, performance
  - **Goals** and **other requirements** for the new system
    - Which problems need to be solved?
    - What would the stakeholders like to achieve?
  - **Constraints**
    - Including nonfunctional requirements on the system
  - Possible **alternatives**
    - "Sticking with the current system" is always an alternative
    - Different business processes for solving the problems
    - Different levels/types of computerization for the solutions
  - **Advantages** and **disadvantages** of the alternatives

- **Things to conclude:**
  - Feasibility of the project (Go / Stop)
  - The preferred alternative

4 Types of Feasibility Study

Technical Feasibility	• ***"Is the project possible with current technology?"***
	• What technical risk is there?
	• Availability of the technology
	  • Is it available locally?
	  • Can it be obtained?
	  • Will it be compatible with other systems?

Economical Feasibility	• ***"Is the project possible, given resource constraints?"***
	• What are the benefits?
	  • Both tangible and intangible
	  • Quantification requires
	• What are the development and operational costs?
	• Are the benefits worth the costs?

Schedule Feasibility	• ***"Is it possible to build a solution in time to be useful?"*** • What are the consequences of delay? • Any constraints on the schedule? • Can these constraints be met?

Operational Feasibility	• ***"If the system is developed, will it be used?"***
	• Human and social issues:
	  • Potential labor objections?
	  • Manager resistance?
	  • Organizational conflicts and policies?
	  • Social acceptability?
	  • Legal aspects and government regulations?

Comparing Alternatives

- **Feasibility Analysis Matrix**
  - Each cells contains the feasibility assessment notes for each candidate.
    - Can be assigned a rank or score for each criterion
  - A final ranking or score is recorded in the last row.

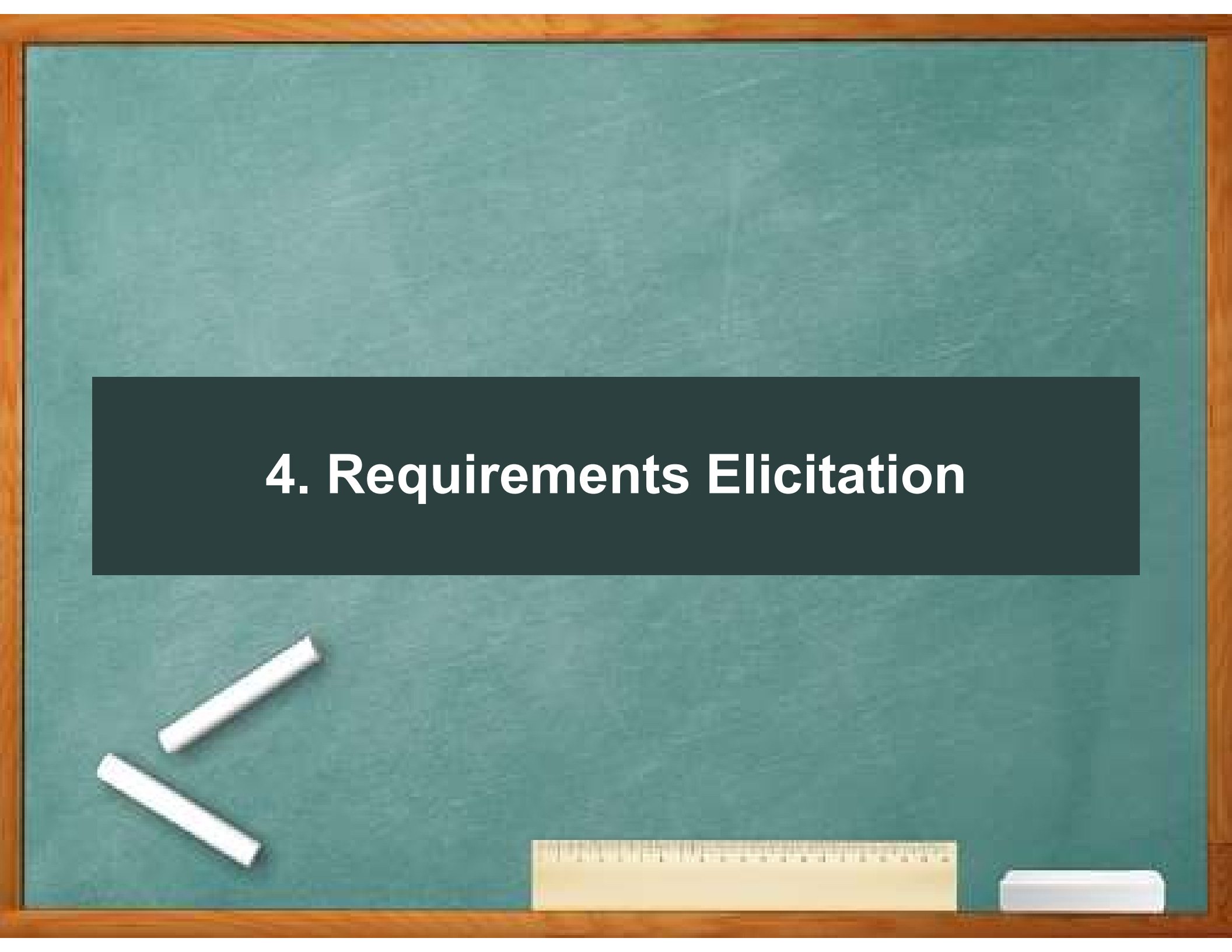
Feasibility Criteria	Wt.	Candidate 1	Candidate 2	Candidate 3	Candidate …
Operational Feasibility	30%	Score: 60	Score: 100	Score: 100	
Technical Feasibility	30%	Score: 50	Score: 95	Score: 100	
Economic Feasibility Cost to develop: Payback period (discounted): Net present value: Detailed calculations:	30%	 Approximately $350,000. Approximately 4.5 years. Approximately $210,000. See Attachment A. Score: 60	 Approximately $418,040. Approximately 3.5 years. Approximately $306,748. See Attachment A. Score: 85	 Approximately $400,000. Approximately 3.3 years. Approximately $325,500. See Attachment A. Score: 90	
Schedule Feasibility An assessment of how long the solution will take to design and implement.	10%	Less than 3 months. Score: 95	9-12 months Score: 80	9 months Score: 85	
Ranking	100%	60.5	92	95.5	

More Tangible Case of Feasibility Analysis

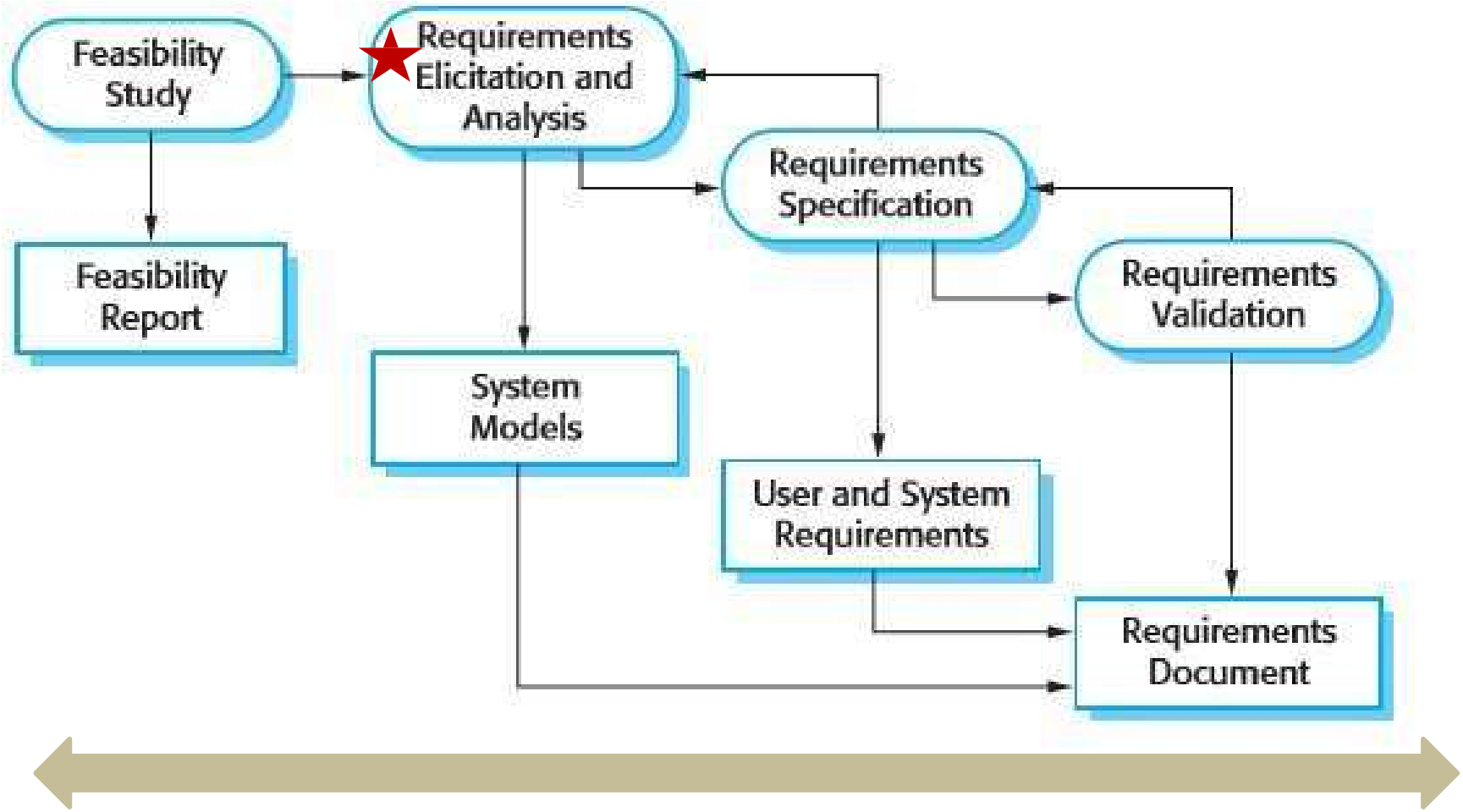
- **Objective: 24시간, 365일 banking service 제공이 가능한 새로운 solution 개발**
  - Alternative 1: 전체 비니지스 모델은 그대로 유지하면서, 각 행정구역별로 24시간 365일 사용 가능한 지점을 운영(언제나 current system을 그대로 유지하는 것은 대안에 포함시킴)
  - Alternative 2: 24시간 365일 banking service를 제공하는 무인 시스템(ATM) 개발
  - Alternative 3: Internet 기반의 banking system 개발
  - Alternative 4: Mobile 기반의 banking system 개발
  - Alternative 5: 기존 금융권과 상관없는 open banking service 개발

- 이 경우, alternative들은 동시에 비교 대상으로 고려될 수는 없었던 것으로, feasibility analysis 결과 채택되는alternative가 변화되어 옴
  - **1995**: *Alternative 2*는 *Alternative1*보다 economical feasibility 측면에서 우위에 있어 채택
  - **2006**: *Alternative 4*는 mobile phone 출현(2007) 이전으로 technical feasibility 측면에서 고려 대상이 될 수 없었음
  - **2015**: *Alternative 5*는 operational(political/legal) feasibility 측면에서 실현 불가능함에 따라 구현 불가능

4. Requirements Elicitation



Requirements Engineering Process



The Requirements Elicitation and Analysis Activities

1. Requirements Discovery

- Interacting with stakeholders to discover their requirements
- Domain requirements are also discovered at this stage.

2. Requirements Classification and Organization

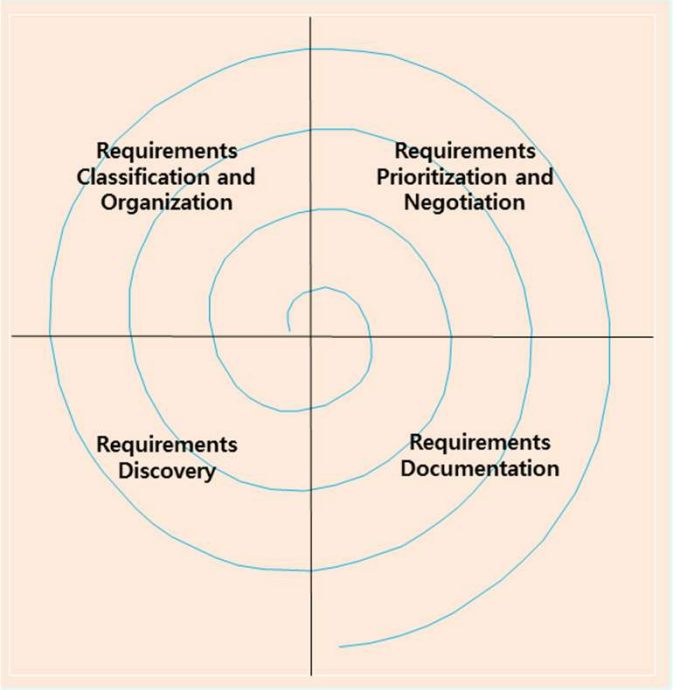
- Groups related requirements and organizes them into coherent clusters

3. Prioritization and Negotiation

- Prioritizing requirements and resolving requirements conflicts

4. Requirements Documentation

- Document requirements
- Input it into the next round of the spiral



Requirements Elicitation

- There should be a "***problem***" that needs solving.
  - Dissatisfaction with the current state of affairs
  - New business opportunity
  - Potential saving of cost, time, resource usage, etc.

- **Collect enough information to** Identify the "***problem***" and "***opportunity***"
  - Which problem needs to be solved? (identify problem Boundaries)
  - Where is the problem? (understand the Context/Problem Domain)
  - Whose problem is it? (identify Stakeholders)
  - Why does it need solving? (identify the stakeholders' Goals)
  - How might a software system help? (collect some Scenarios)
  - When does it need solving? (identify Development Constraints: ex. Schedule)
  - What might prevent us solving it? (identify Feasibility and Risk)

Problems of Requirements Elicitation

- **Vague problem stated by the customer (stakeholders)**
  - Stakeholders **don't know what they really want.**
  - Stakeholders express requirements in their own terms.
  - Different stakeholders may have conflicting requirements.
  - New stakeholders may emerge and the business environment changes.

- **Organizational** and **political factors** influence the system requirements.
  - Key man을 Product Champion으로 !!

    "특정 프로세스 또는 제품이 완전히 개발되고 판매되는 것을 보는 데 과도한 관심 (inordinate interest)을 갖는 사람" - American Marketing Association-

- The requirements **keep changing** during the analysis process.

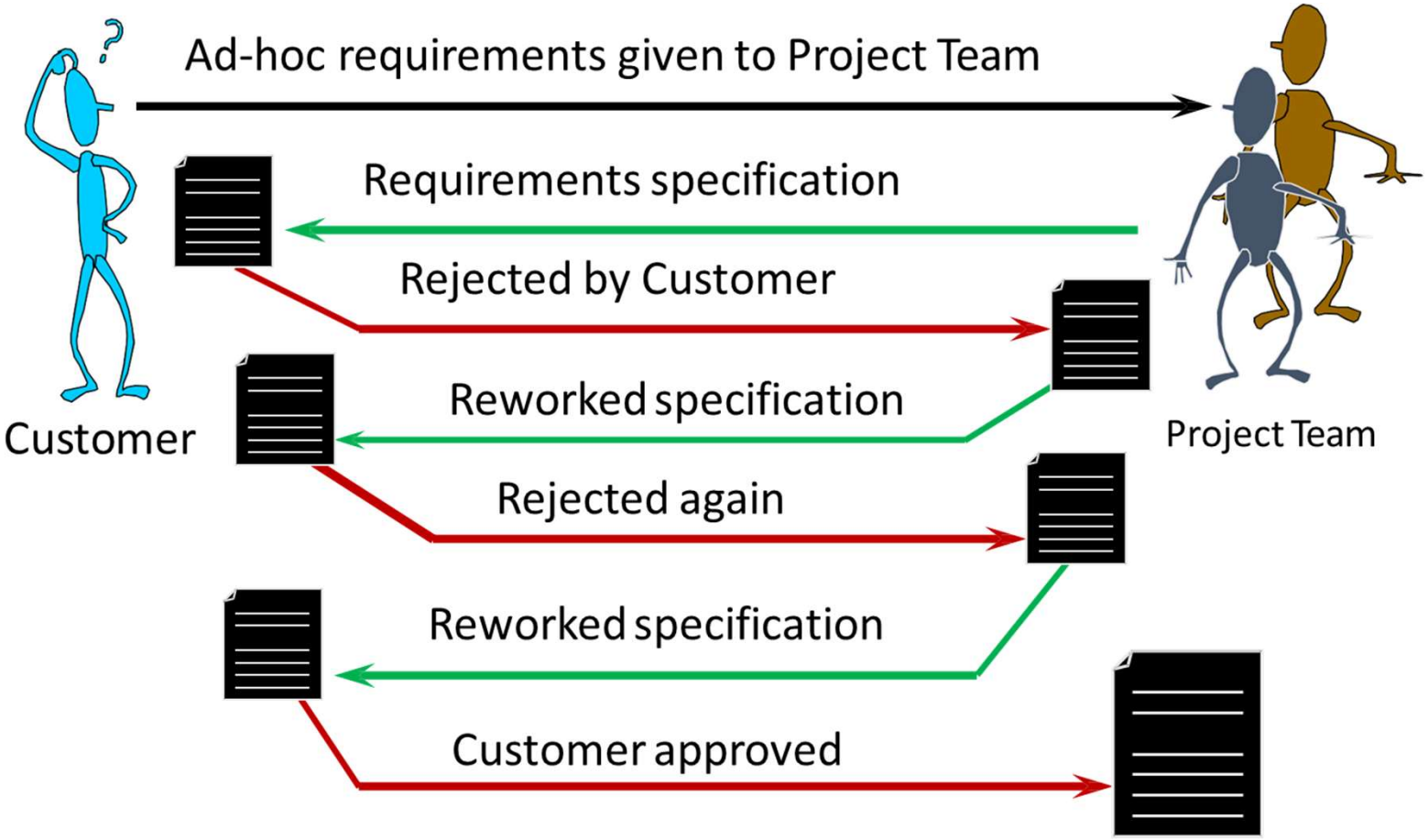
Stakeholders

- **Stakeholder analysis**
  - **Identify all the people** who must be consulted during information acquisition

- **Typical stakeholders**

User(사용자)	Concerned with the features and functionality of the new system
Designer	Want to build a perfect system, or reuse existing code
System Analyst	Want to “get the requirements right”
Training and User Support	Want to make sure the new system is usable and manageable
Business Analyst	Want to make sure “we are doing better than the competition”
Technical Author	Will prepare user manuals and other documentation for the new system
Project Manager	Wants to complete the project on time, within budget, with all objectives met.
Customer(발주자)	Wants to get best value for money invested

A Typical Situation Encountered



Things to Remember When Eliciting Requirements

1. Don’t Lose Sight of the Goal
2. Think Who’s Smart
3. A Single Stakeholder Can’t Speak for All
4. Use Appropriate Elicitation Methods
5. Accept Requirements Changes
6. Manage Elicited Requirements

1. Don't Lose Sight of the Goal

- Establish the system's vision(top-level requirements) and scope to reduce the risk of building the wrong system

- Try to obtain early commitment from stakeholders
  - commitment vs. involvements

2. Think Who's Smart

- Don't try to convince stakeholders that YOU are smart.
- Instead, take everybody to show you think the STAKEHOLDER is smart

- Contrast these 2 cases:

1. My Elevators Are Too Slow!

2-1. I See. Tell Me Why You Feel They Are Too Slow.

2-2. I Don't Think So. I Think You Have an Elevator Throughput Problem, not a Speed Problem.

3. A Single Stakeholder Can't Speak for All

Stakeholder	Role
User	• Users of the system and the results of the system • ALWAYS included • Often many classes - make sure all are represented
Customer	• People with decision making authority • ALWAYS included; no project otherwise! • Often many classes - make sure all are represented • Closely aligned with the marketing function
Marketing	• ESSENTIAL; The experts in the "market" • Too easy for development to dismiss them • In a commercial setting, they know the pulse of customers
Subject Matter Experts (SME) **업무전문가**	• Helpful to learn foundation requirements • Helpful to alleviate disagreements among stakeholders
Developer	• Helpful to learn system implications • Helpful to learn evolution / maintenance requirements
Development Managers	• Knows the development capability and resources
Tester	• Useful a bit later in project • Knows which requirements are testable
Loser Users	• People who loses power as a result of the project • Useful if a system has "loser users"
Technical Writers **Trainers / Customer Support**	• Can also help • Experts in making the system easy to use/teach/explain

4. Use Appropriate Elicitation Methods

- **Methods** for requirements elicitation:
  1. **Requirements workshop**
  2. **Brainstorming**
  3. **Storyboards**
  4. **Interviews**
  5. **Questionnaires**
  6. **Role playing**
  7. **Prototypes**
  8. **Review customer requirement specification**

- **A single method may not be sufficient**.
  - Consider requirements' size, complexity, etc., and select several ones.

Requirements Workshop *

- **Gather all stakeholders together** for an intensive and focused period
  - Create consensus on the scope, risks and key features of the software system
  - Results immediately available
  - Outputs:
    - Problem statement , Key features , Initial business object model, Use-case diagram , Prioritized risk list, etc.

- Provide a **framework** for applying other elicitation techniques such as
  - Brainstorming, use-case workshops, storyboarding, etc.

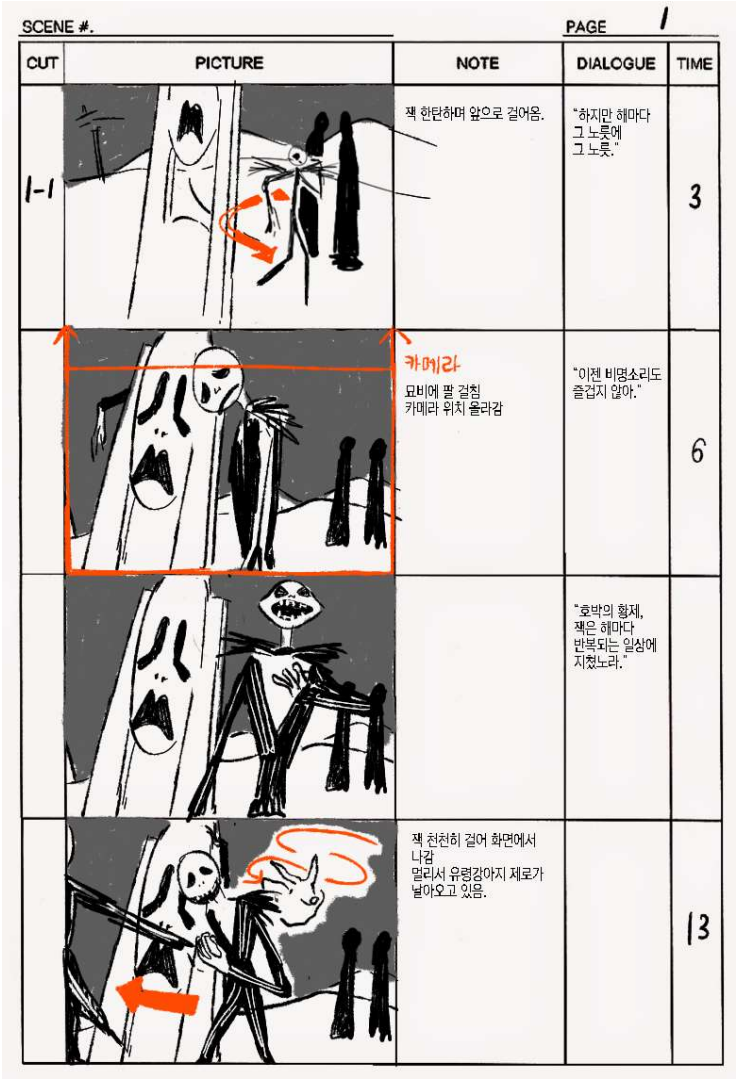
Brainstorming *

- **Rules for Brainstorming**
  - Clearly state the objective of the session
  - Generate as many ideas as possible
  - Let your imagination soar
  - Do not allow criticism or debate
  - Mutate and combine ideas
- **Generate as many ideas as possible**
  - Even the impractical, absurd ideas should not be neglected
  - Merge the various ideas to create new ideas
- **Express freely**
  - Do not explain or specify the ideas
  - Do not evaluate or argue about the ideas
  - Do not put names on the ideas
  - Encourage the unexpected and imaginative
- **Put up ideas openly**
  - Ideas should be put up on a whiteboard where all can see
  - Participants themselves may put up ideas on the board
  - Put tabs of Post-Its on the center table

Storyboards: Text 기반 자료보다는 시청각 자료를 활용

- Visually tell and show:
  - Who/what the players are (actors)
  - What happens to them
  - When it happens

- Benefits
  - Help gather and refine customer requirements
  - Encourage creative and innovative solutions
  - Encourage team review
  - Prevent features that no one wants
  - Ensure that features are implemented in an accessible and intuitive way
  - Ease the interviewing process
  - Help to avoid blank-page syndrome



Interviews

- Provide **a simple and direct technique** to gain understanding of problems and solutions

- Types of interviews
  - **Open interview**
    - No pre-set agenda
    - Irrelevant data can be gathered
    - Needs time and training
  - **Closed interview**
    - Fairly open-questions agenda
    - Needs extended preparation
    - Prevents biases

- Interview tips
  - Avoid asking people to describe things they don't usually describe
    - Example: Describe how to tie your shoes
  - Avoid "Why…?" questions
  - Ask open-ended (context-free) questions: interview 자체에 대한 feedback 관련 질문들
    - High-level abstract questions

Questionnaires

- Give access to a wide audience
  - Apply to broad markets where questions are well-defined

- **Statistical analysis** is applicable.
  - Powerful, but not a substitute for an interview
  - 틈새시장을 공략하고자 하는 상품일 경우엔 더더욱

- Assumptions:
  - Relevant questions can be decided in advance
  - Questions phrased, so reader hears as intended

고 객 명		소 속		조사일자	20 년 월 일
■ 설문조사내용					
구 분	평 가 내 용		평 가		점 수
품질	1. 고객께서 알고 있는 회사명의 이미지는 어떠하십니까?		A : 좋 다 B : 보 통 C : 나쁘다		
	2. 고객께서는 회사명으로 전화를 하셨을 때 친절하게 응대를 하였습니까?		A : 좋 다 B : 보 통 C : 나쁘다		
	3. 고객께서는 회사명의 제품에 만족 하십니까?		A : 만 족 B : 보 통 C : 불만족		
	4. 당사의 제품에 이상 발생 시 고객께서 요구했을 때 신속한 대응이 있었습니까?		A : 그렇다 B : 보 통 C : 아니다		
	5. 품질관련해서 동종업계화의 수준에 있어 어느 정도의 수준이라고 생각하십니까?		A : 상 위 B : 중 위 C : 하 위		
	평 가 점 수	A : 20 B : 15 C : 10		소 계	
생산기술	1. 회사명을 과거에 방문하신 적이 있으면 어떤 느낌을 받았습니까?		A : 좋 다 B : 보 통 C : 나쁘다		
	2. 회사명의 생산기술에 있어 동종 업계와의 기술수준은 어느 정도라고 생각하십니까?		A : 상 위 B : 중 위 C : 하 위		
	3. 제품에 대한 상호간 정보교환은 잘 이루어지고 있습니까?		A : 그렇다 B : 보 통 C : 아니다		
	4. 신규 제품 개발 시 경쟁업체와 비교했을 때 개발일정 및 진행이 잘 되고 있습니까?		A : 그렇다 B : 보 통 C : 아니다		
	평 가 점 수	A : 25 B : 20 C : 15		소 계	
영업	1. 당사의 납입 준수율은 어느 정도라고 생각하십니까?		A : 80%이상 B : 50%이상 C : 50%미만		
	2. 당사의 납품수량은 정확하다고 생각하십니까?		A : 정 확 B : 보 통 C : 미 흡		
	3. 회사명의 서비스(친절함) 수준은 어느 정도라고 생각하십니까?		A : 좋 다 B : 보 통 C : 나쁘다		
	4. 포장 및 제품에 청결함과 더불어 사용하는데 불편함이 있었습니까?		A : 그렇다 B : 보 통 C : 아니다		
	평 가 점 수	A : 25 B : 20 C : 15		소 계	
고객의견				평 점	

Role Playing *

- Perform requirements elicitation from the viewpoint of the roles
  - Learns and performs user's job
  - Performs a scripted walkthrough

- Advantages
  - Gain real insights into the problem domain
  - Understand problems that users may face

Prototypes

- Demonstrate some or all of the externally observable behaviors of a system through building prototypes quickly

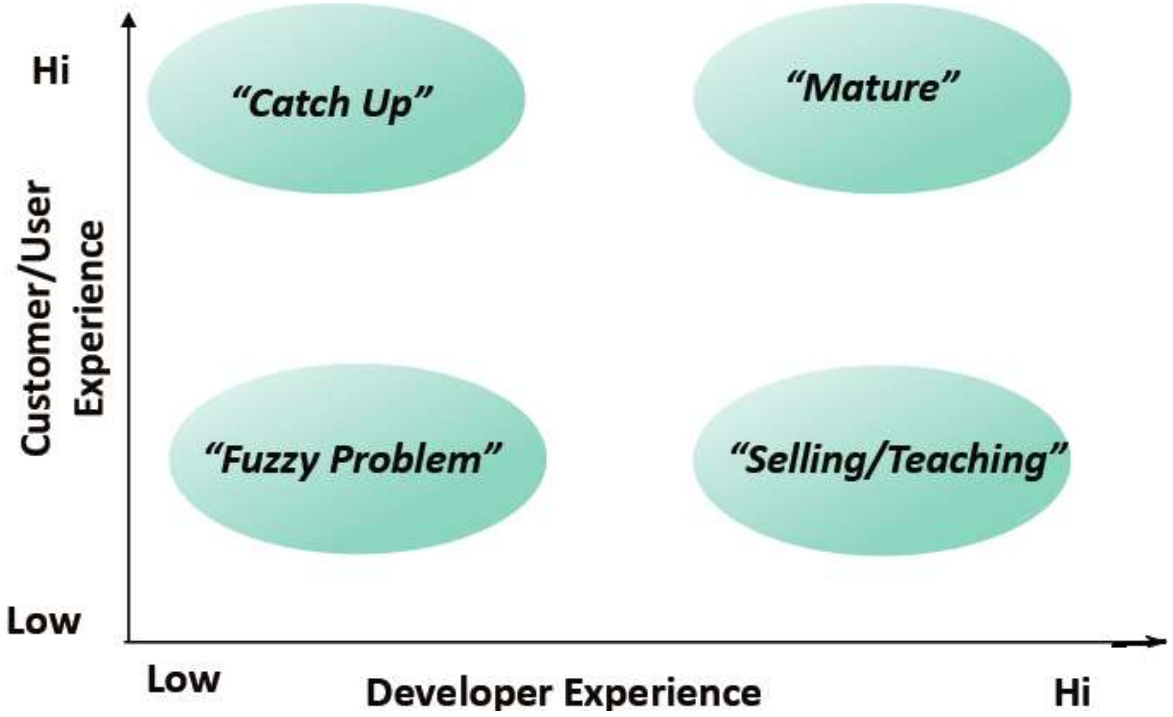
- Used to:
  - Demonstrate understanding of the problem domain
  - Gain feedback on proposed solution
  - Validate known requirements
  - Discover unknown requirements
  - Create simulations
  - Elicit and understand requirements
  - Prove and understand technology
  - Reduce risk
  - Enhance shared understanding
  - Improve
    - Cost and schedule estimates
    - Feature definition

Review Customer Requirement Specifications

- **Customer Requirements Review**
  - Recognize and label
    - Application behaviors
    - Behavioral attributes
    - Issues and assumptions
  - **Ask customers directly**

Which Techniques to Use?

- Appropriate method should be chosen based on:
  - The size and complexity of requirements
  - Problem domain
  - The number of involved stakeholders



- **Catch Up**
  - Role Playing
  - Interview
  - Requirements Review
- **Fuzzy Problem**
  - Requirements Workshops
  - Brainstorming
  - Storyboards
- **Selling / Teaching**
  - Use Case
  - Business Modeling
- **Mature**
  - Questionnaires
  - Prototyping

5. Accept Requirements Changes

- **Requirements change is inevitable.**
  - Clients have right to change requirements.
  - The more features the product has, the more customers want.

- Don't ever ask:
  - "Okay, is that your final requirement ?"

- Change is not a threat, it's an opportunity.

  → 프로젝트 초기에 clients가 requirements change를 빈번하게 요청하는 것은 좋은 징조일 수도 있음

6. Manage Elicited Requirements

- Record the **rationale** of each requirement
  - Reason why requirement is necessary
  - Assumptions on the requirement

- Managing the rationale with **annotated requirements lists**
  - Do not simply rewrite the requirement
  - Make it unique for each requirement
  - Keep it simple

- Example:
  - Requirements : "*The truck shall have a height of no more than 14 feet*."
  - Rationale : "*99% of all U.S. Interstate highway overpasses have a 14-foot or greater clearance.*"

What to Do with Elicited Requirements?

- **Maintain requirements in lists**
  - Maintaining **a list of requirements** can support all activities of requirements.

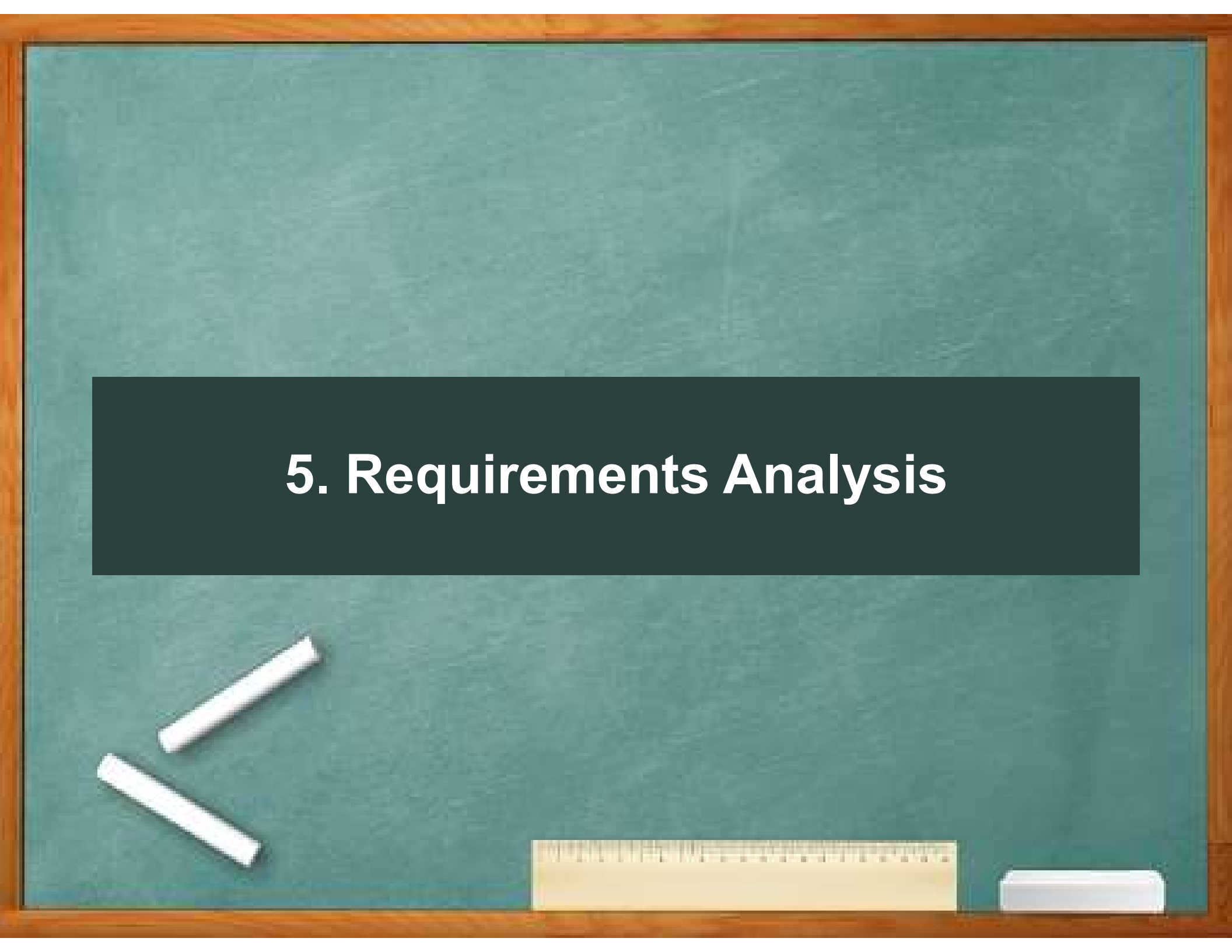
- Enables you to answer questions such as:
  - How many requirements do you have?
  - How many high priority requirements do you have?
  - What percentage of the candidate requirements have you chose to satisfy in your next release?
  - What percentage of the requirements deemed high priority by customer X are you satisfying with?

Example of Elicitation Results

- **A list of annotated requirements**

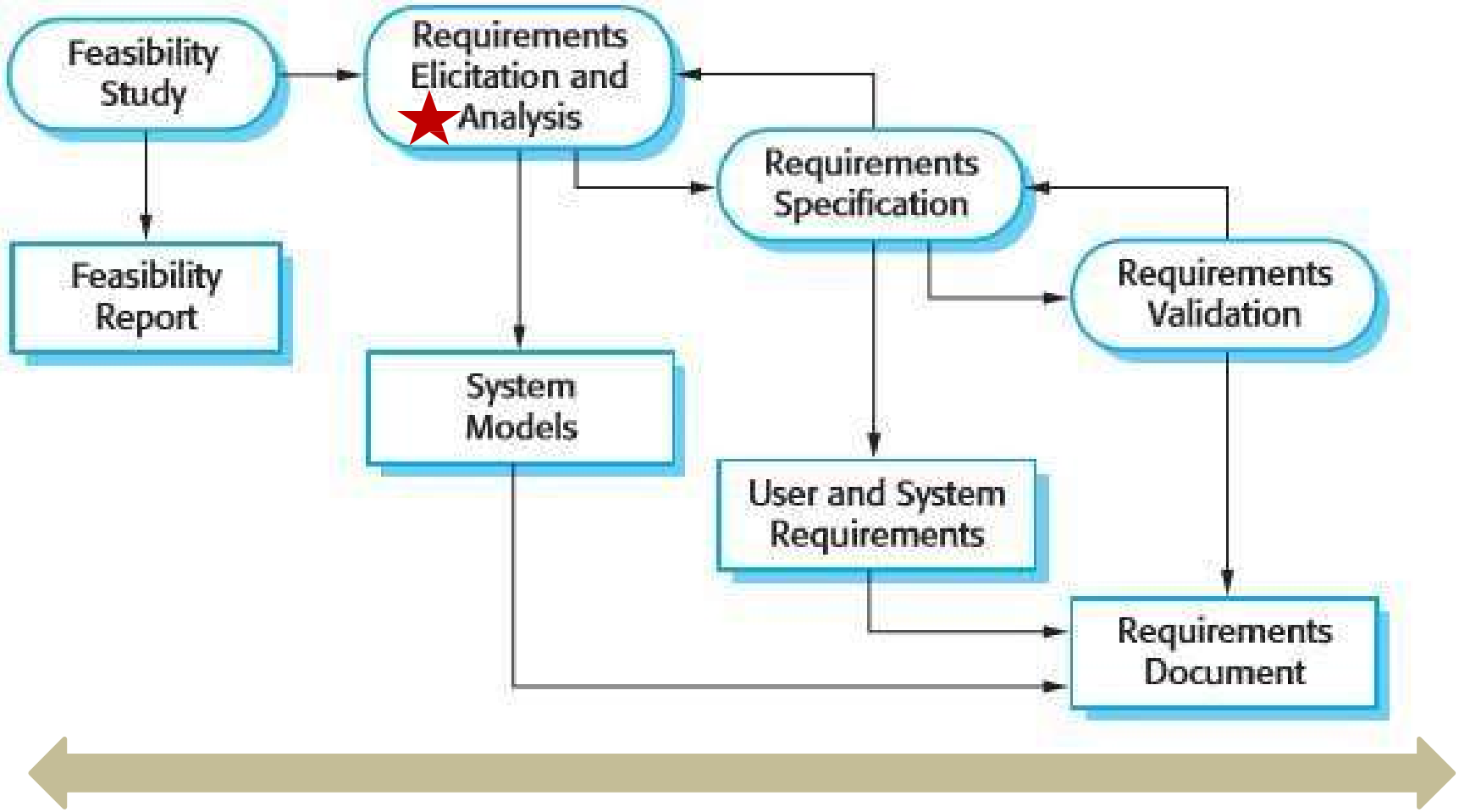
ID	Requirement Text
961	No formal training shall be required to operate the RLM.
955	Any new releases or versions of the software shall be sold as new products. Users must p...
954	User software will not be modified or upgraded.
512	The RLM shall return to the refuel location or dump area to within 10 cm of the user-define...
432	Pressing the screen in an area without a command shall make no sound nor shall it be int...
415	The screen shall be capable of displaying alphanumeric data in blocked, uppercase char...
500	The RLM shall accept lawn and obstacle programming from the user. During programming, th...
321	The RLM shall initiate communications with the GPS through external interface EL-GPS
300	The RLM shall interface with two different external systems, The GPS and the Electronically S...
310	External interfaces include the receipt of location data from GPS and detection of obstacles...
511	The RLM shall not overcut or undercut the border and user defined obstacles by more tha...
510	The RLM shall cut the lawn only within the area defined by the user during the programming.
550	Border programming shall be required to be completed by the user prior to accepting the oth...
411	The Screen shall be 16.25 mm (high) by 105 mm (wide) and capable of displaying two row...
446	Serious errors (for example, blade fouling, Requirement 179) shall not have a button on th...
553	Programming border data shall be terminated by a user request, or when the RLM returns t...
554	After the termination, the RLM shall be ready to receive another command.
418	The screen shall be used to display information from the RLM to the user and accept dire...
561	User shall guide the RLM to the obstacle and indicated that the boundary of obstacle will ...
562	RLM shall record sufficient data (e.g. from GPS) to meet the accuracy requirements stated...
552	RLM shall record sufficient data (e.g. from GPS) to meet accuracy requirements stated in ...
551	User shall guide the RLM to the border of the lawn and indicate that the boundary will be ...

5. Requirements Analysis



Requirements Engineering Process

- **Requirements analysis through requirements models**
- **Requirements prioritization**
- **Requirements selection (Triage)**



The Requirements Elicitation and Analysis Activities

1. Requirements Discovery

- Interacting with stakeholders to discover their requirements
- Domain requirements are also discovered at this stage.

2. Requirements Classification and Organization

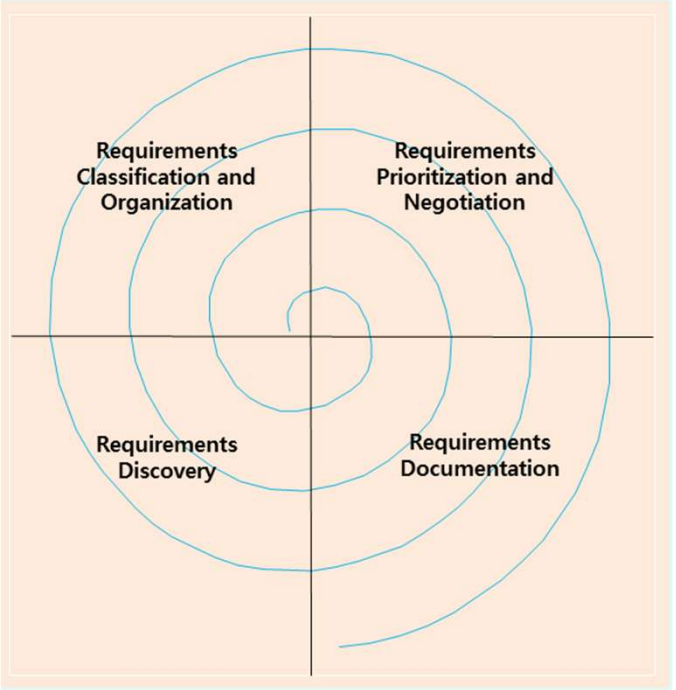
- Groups related requirements and organizes them into coherent clusters

3. Prioritization and Negotiation

- Prioritizing requirements and resolving requirements conflicts

4. Requirements Documentation

- Document requirements
- Input it into the next round of the spiral



Requirements Modeling

- RE uses **requirement models** to understand the requirements well.
  - Help stakeholders to understand the requirements

- Requirements modelling can
  - Guide **elicitation**
    - It can help you figure out what questions to ask
    - It can help to surface hidden requirements
  - Provide **a measure of progress**
    - Completeness of the models → completeness of the elicitation (might be)
  - Help to **uncover problems**
    - Inconsistency in the models can reveal interesting things
      - E.g., conflicting or infeasible requirements
      - E.g., confusion over terminology, scope, etc.
  - Help us check our **understanding**
    - Reason over the model to understand its consequences
    - Animate the model to help us visualize/validate the requirements

Features of Requirements Models

- **Features of good requirements models**
  - **Complete**
    - Modeling guides elicitation
    - Completeness of the model leads to completeness of elicitation
  - **Consistency**
    - Modeling uncovers problems
    - Inconsistency in modeling implies omission, conflict, disagreement and ambiguity
  - **Testability**
    - Modeling checks for expected qualities and predicts end result

- Principles of requirements modeling
  - Early artifacts
  - Cheap to make
  - Easy to visualize and optimize

Requirements Modelling Notations(Formality 측면)

- **Natural language**
  - Extremely expressive and flexible
    - Useful for elicitation, and to annotate models for readability
  - Poor at capturing key relationships

- **Semi-formal notation**
  - Captures structure and some semantics
  - Can perform (some) reasoning, consistency checking, animation, etc.
    - E.g., diagrams, tables, structured English, etc.
    - E.g., **UML**

- **Formal notation**
  - Precise semantics, extensive reasoning possible
    - Underlying mathematical model (e.g., set theory, FSMs, etc.)
  - Very detailed models
    - May be more detailed than we need
    - RE formalisms are for conceptual modelling, hence differ from most computer science formalisms

The State-of-the-Art Requirements Modeling Methods

1. Structured analysis

- Data Flow Diagram (**DFD**)
- State Transition Diagram (**STD**)
- Entity-Relation Diagram (**ERD**)

2. Use Case analysis

- Use Case Modeling (**UC**)

3. Architectural trade-off analysis

- Quality Attribute Scenario(**QAS**)

Scenario-based Requirements Model

4. Goal Analysis

The State-of-the-Art Requirements Modeling Methods

1. Structured analysis

- Data Flow Diagram (**DFD**)
- State Transition Diagram (**STD**)
- Entity-Relation Diagram (**ERD**)

2. Use Case analysis

- Use Case Modeling (**UC**)

Functional Requirements

3. Architectural trade-off analysis

- Quality Attribute Scenario(**QAS**)

Non-Functional Requirements

4. Goal Analysis

Functional/ Non-Functional Requirements

The State-of-the-Art Requirements Modeling Methods

1. Structured analysis

- Data Flow Diagram (**DFD**)
- State Transition Diagram (**STD**)
- Entity-Relation Diagram (**ERD**)

2. Use Case analysis

- Use Case Modeling (**UC**)

Functional Requirements

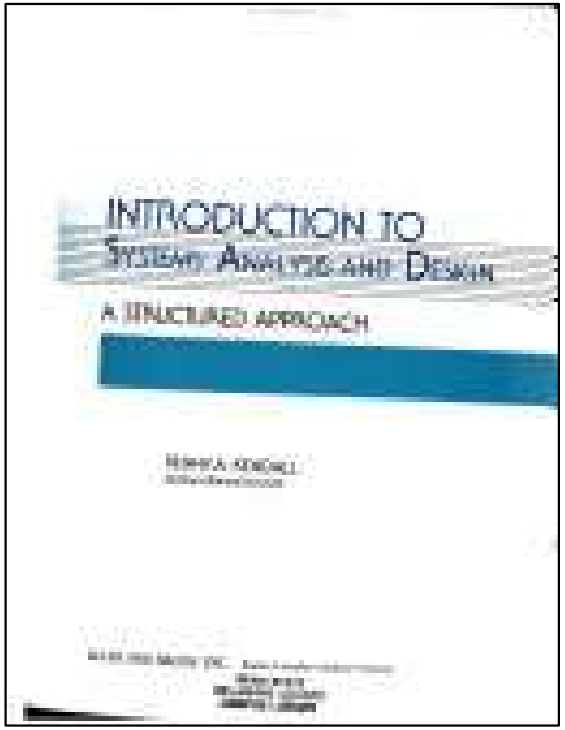
3. Architectural trade-off analysis

- Quality Attribute Scenario(**QAS**)

4. Goal Analysis

1. Structured Analysis

- **Structured analysis** [Kendall 1987]
  - A set of **techniques** and **graphical** tools
    - Allowing the analysts to develop a new kind of system specification that are easily understandable to the users.
  - Data/Functional modeling: Data Flow Diagram(**DFD**), Entity Relationship Diagram(**ERD**),
  - State-oriented modeling: State Transition Diagram(**STD**)

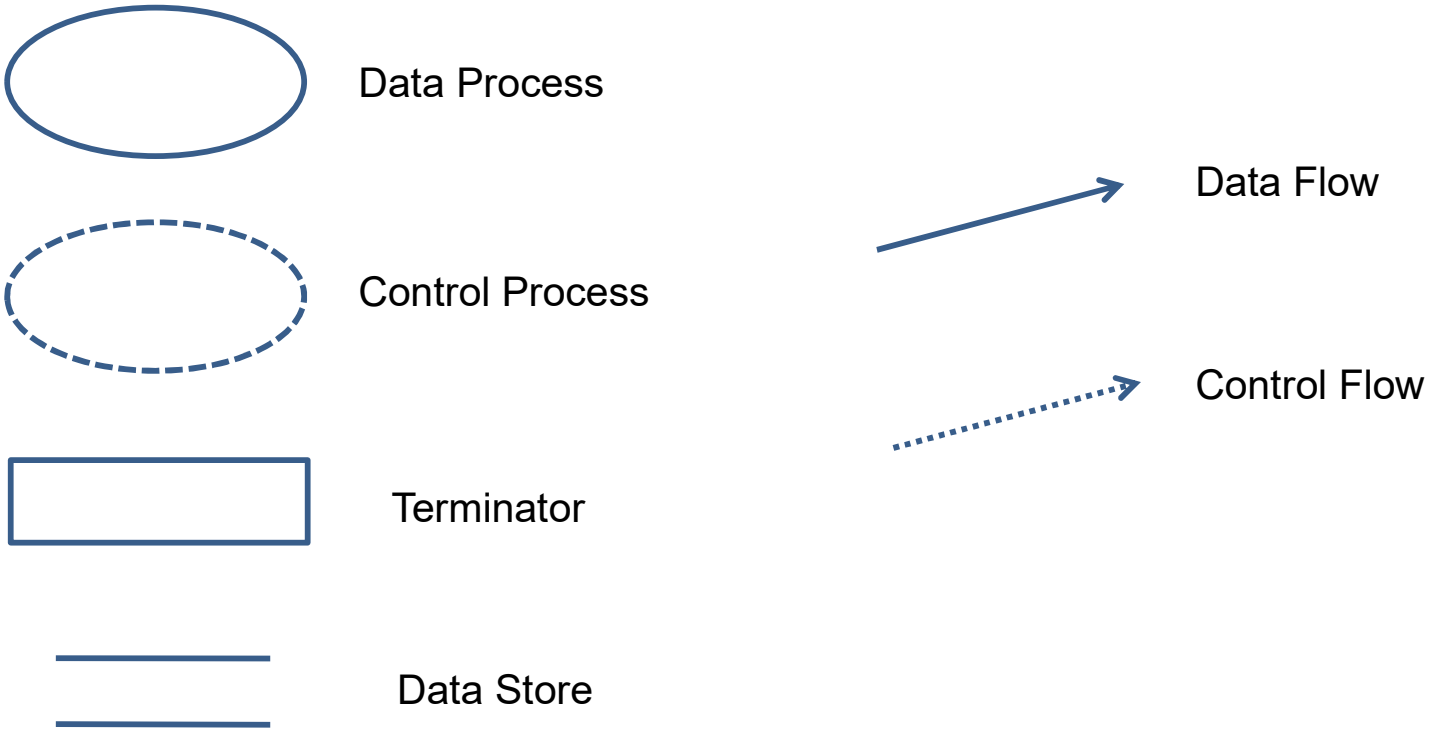


- Analysts attempt to divide large, complex problems into smaller, more easily handled ones.
  - **Top-Down Divide and Conquer** approach

Data Flow Diagram (DFD)

- Provides a means for **functional decomposition**
  - Composed of hierarchies (levels) of DFDs

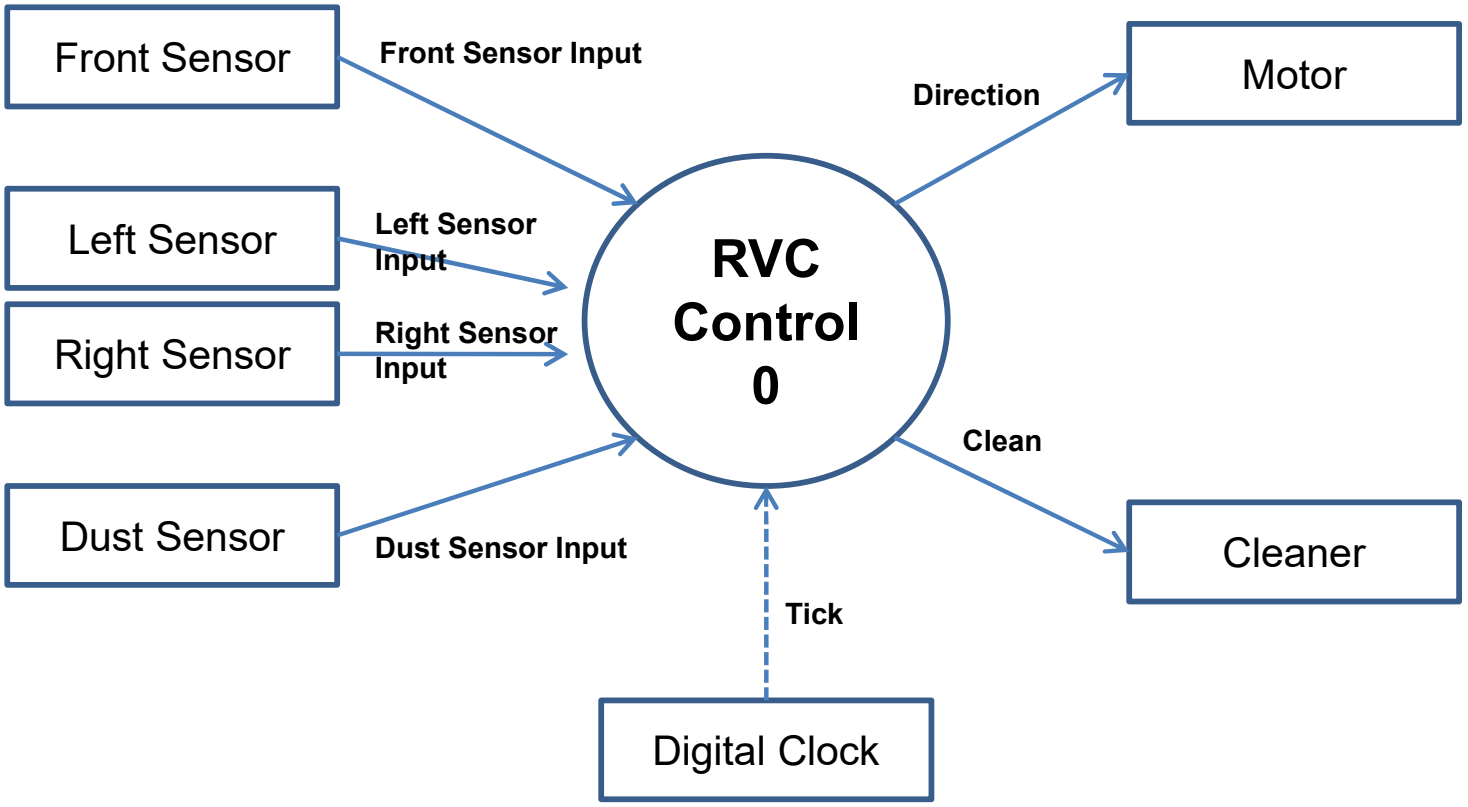
- Model Elements



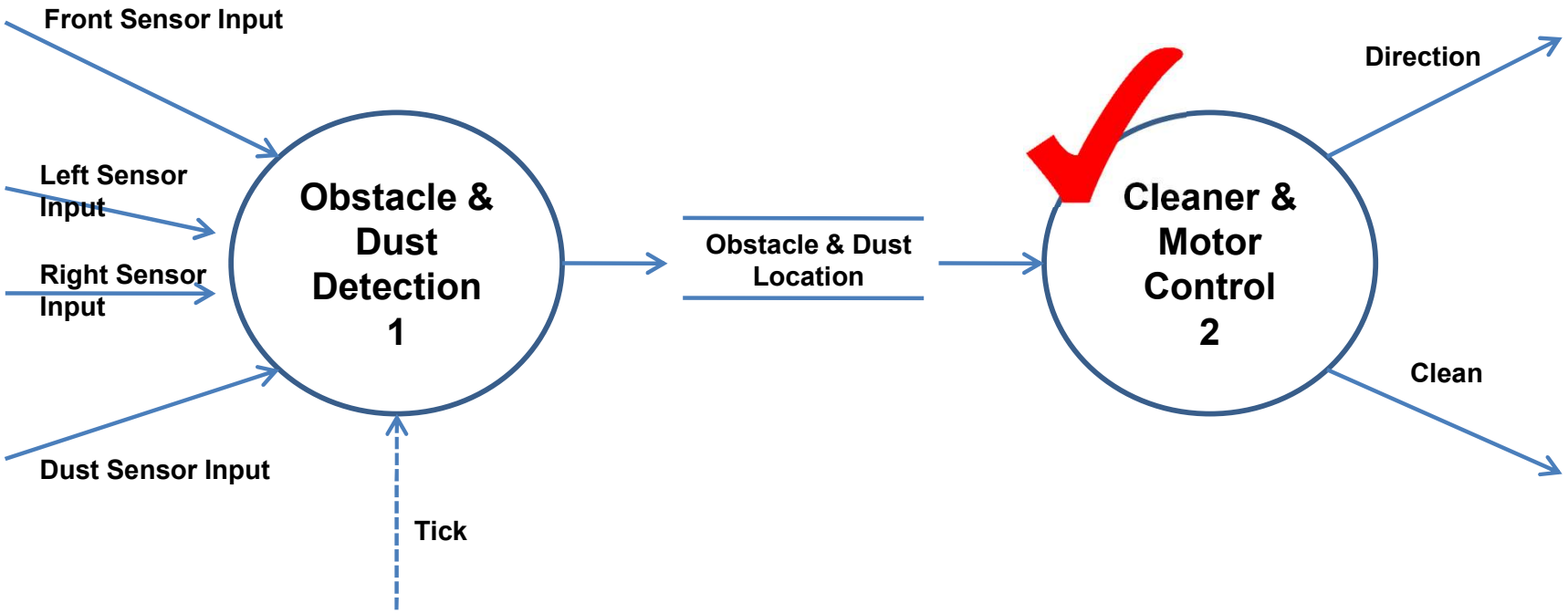
DFD Level 0 - RVC Example

- System context diagram

Robot Vacuum Cleaner

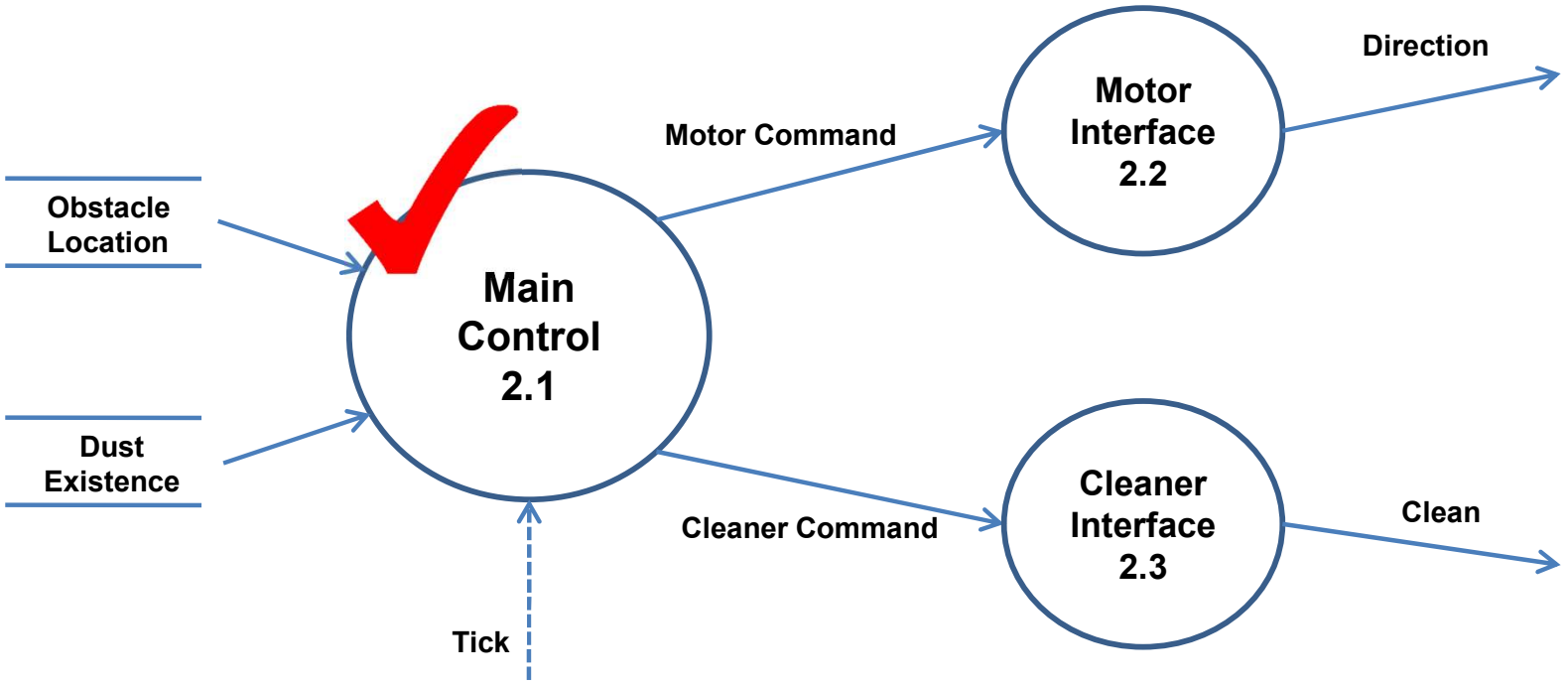


DFD Level 1 - RVC Example

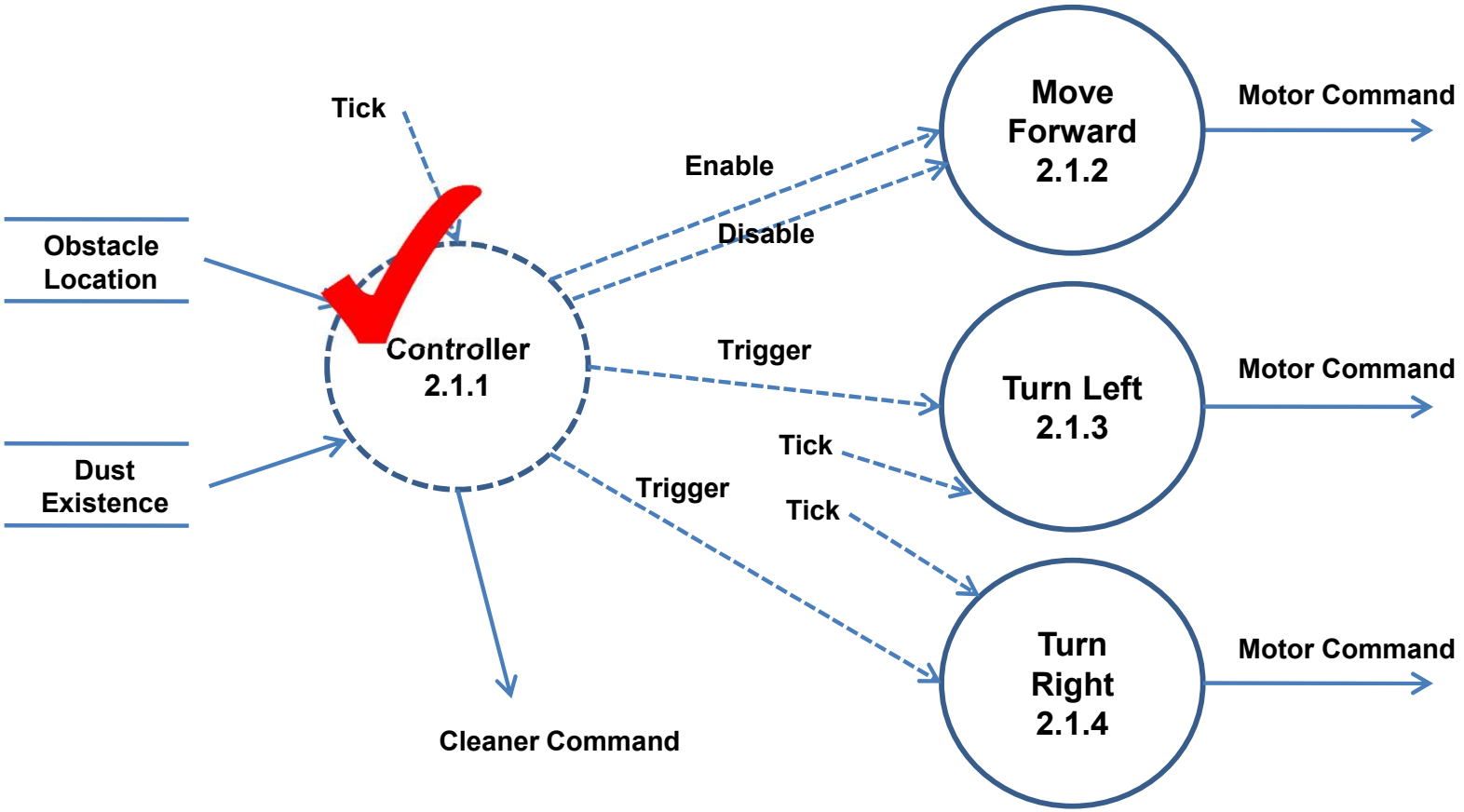


Top-Down Divide and Conquer approach

DFD Level 2 - RVC Example



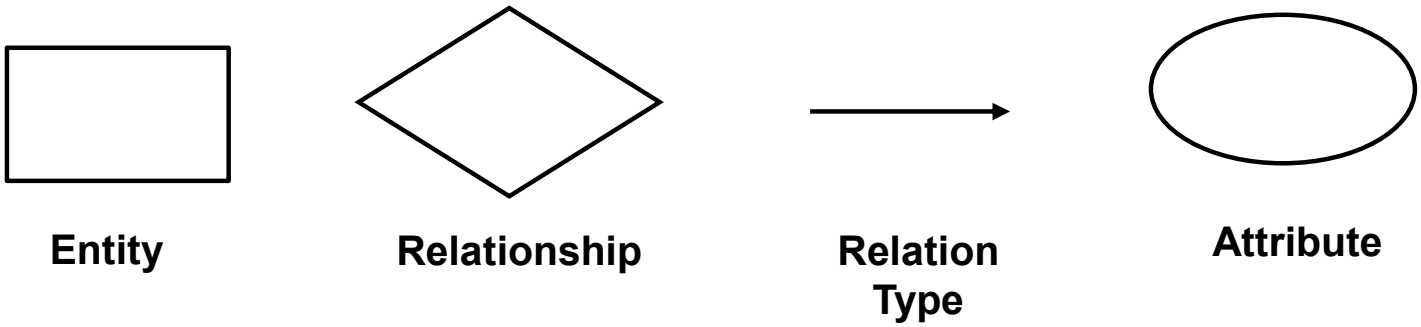
DFD Level 3 - RVC Example



Entity-Relation Diagram(ERD)

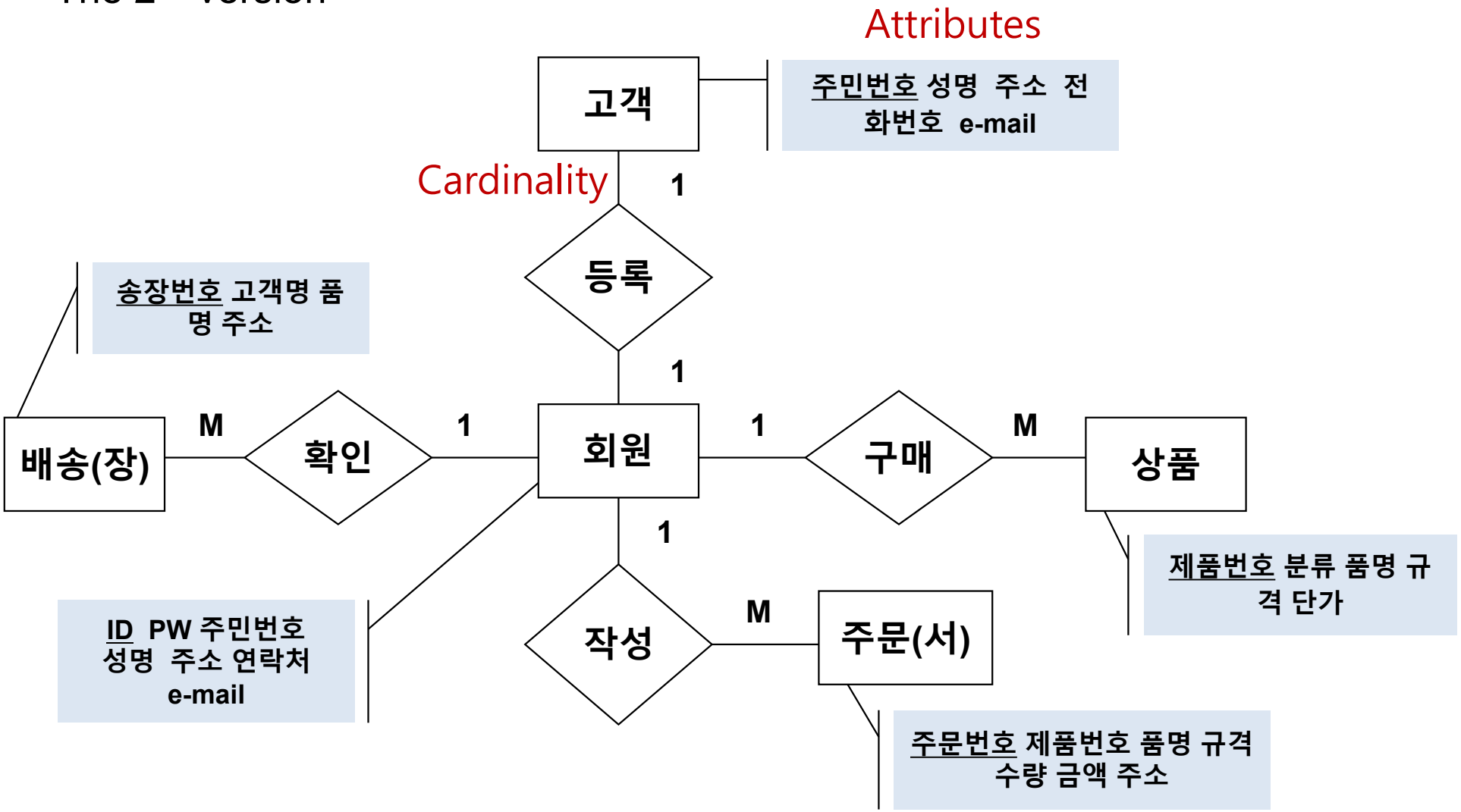
- A graphical representation of the **data layout** of a system at a high level of abstraction
  - Defines data elements(Entity) and their inter-relationships in the system.
  - Similar with the class diagram in UML.

- Model Elements



E-R Modeling

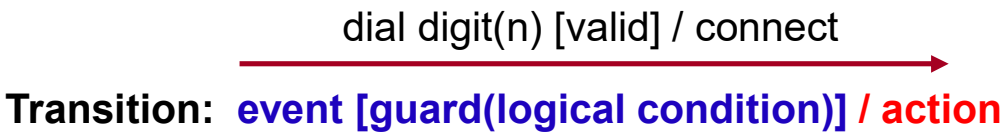
- The 2nd version

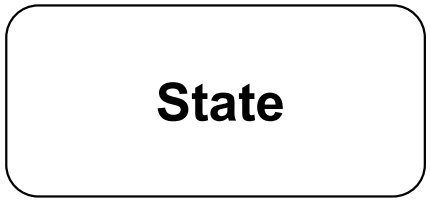


State Transition Diagram (STD)

- **State Transition diagram** is used
  - to model the possible states of a system or controller
  - to show how state transitions occur as a consequence of events
  - to show what behavior the system exhibits in each state

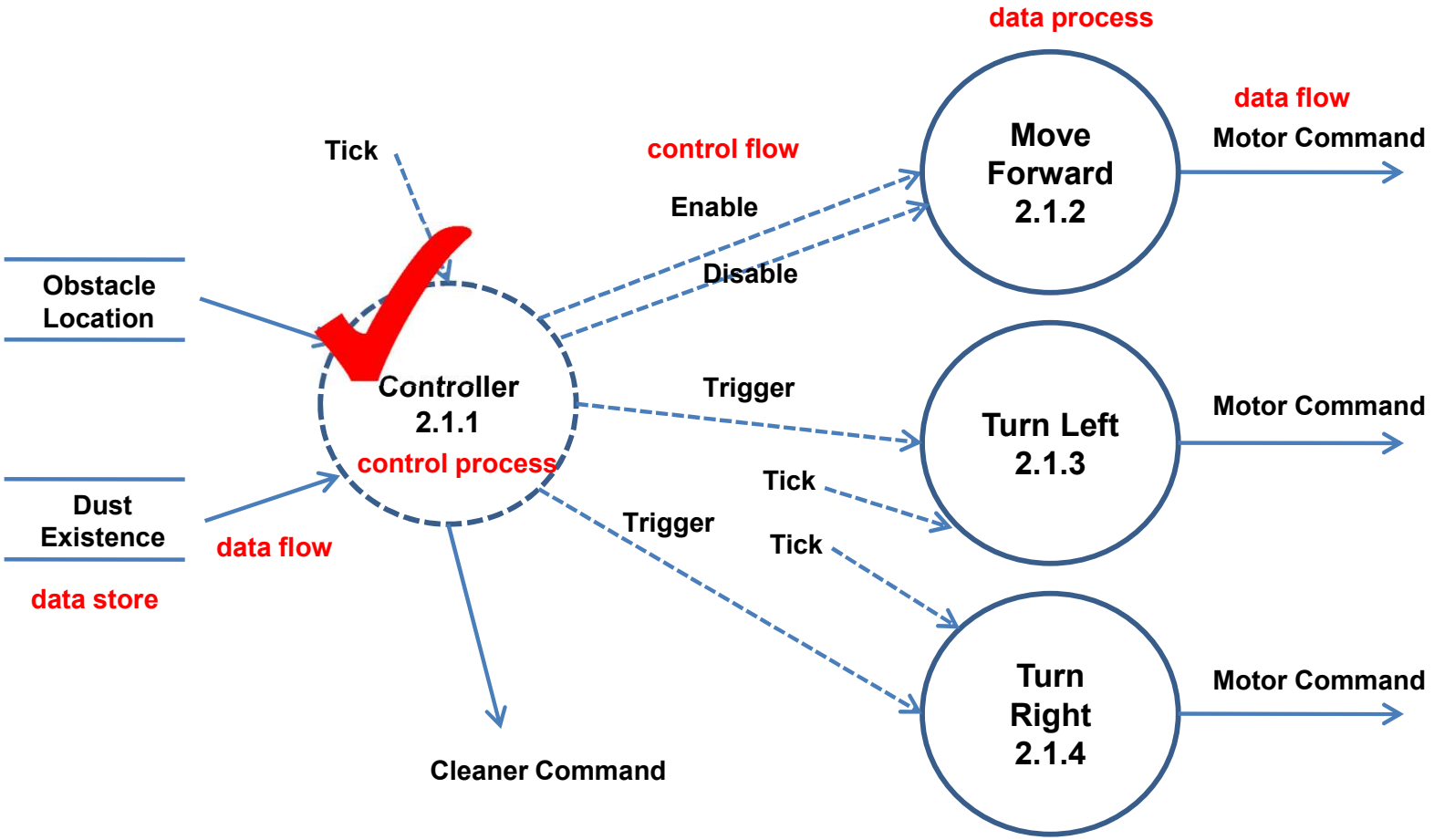
- Model Elements



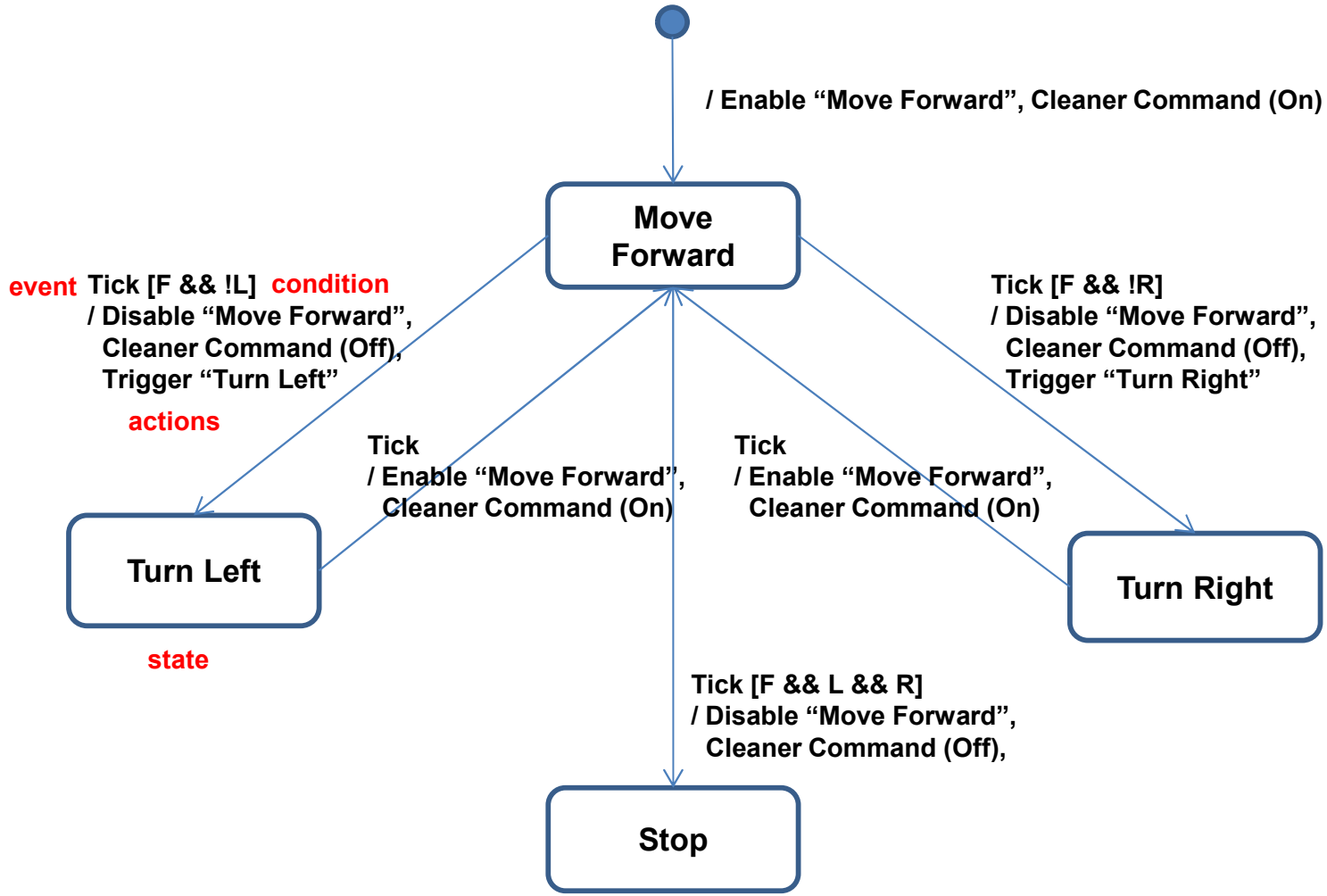


State: State Name

Revisited: DFD Level 3 - RVC Example



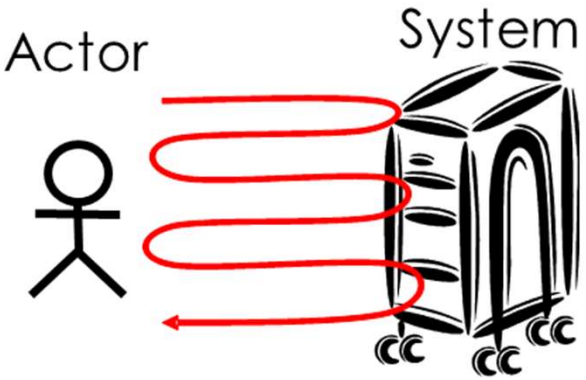
- STD (State Transition Diagram) for RVC Controller 2.1.1



2. Use Case Analysis

- **Use cases** are **text stories** of some actors using a system to meet goals.
  - A mechanism to capture (analyzes) requirements
  - Formats : brief → casual → fully dressed
  - Use case is not a diagram, but a text.

- **Use case diagram** illustrates the name of use cases and actors, and the relationships between them.
  - System context diagram
  - A summary of all use cases

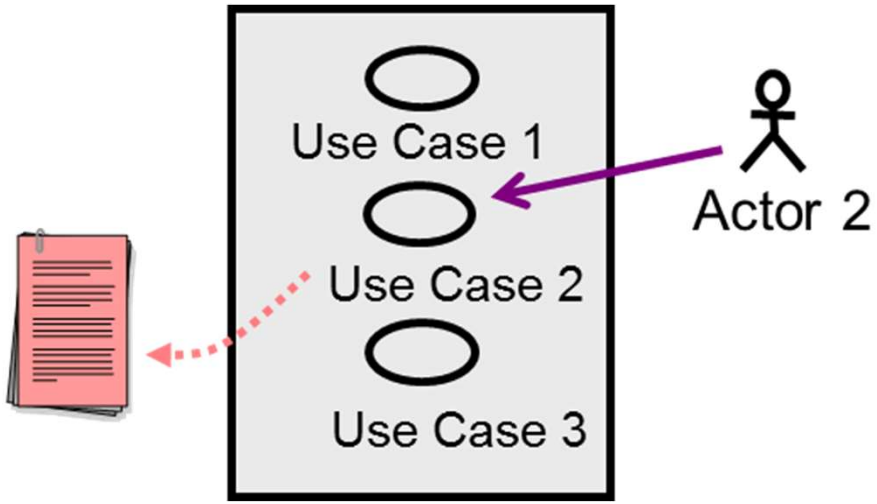


What is Use Case Modeling?

- A means for capturing the desired behavior for the system under development
  - A way to communicate the system's behavior with various stakeholders
  - A way to verify all requirements are captured

- Identifies
  1. Who or what interacts with the system → Actor
  2. What the system should do → Use Case

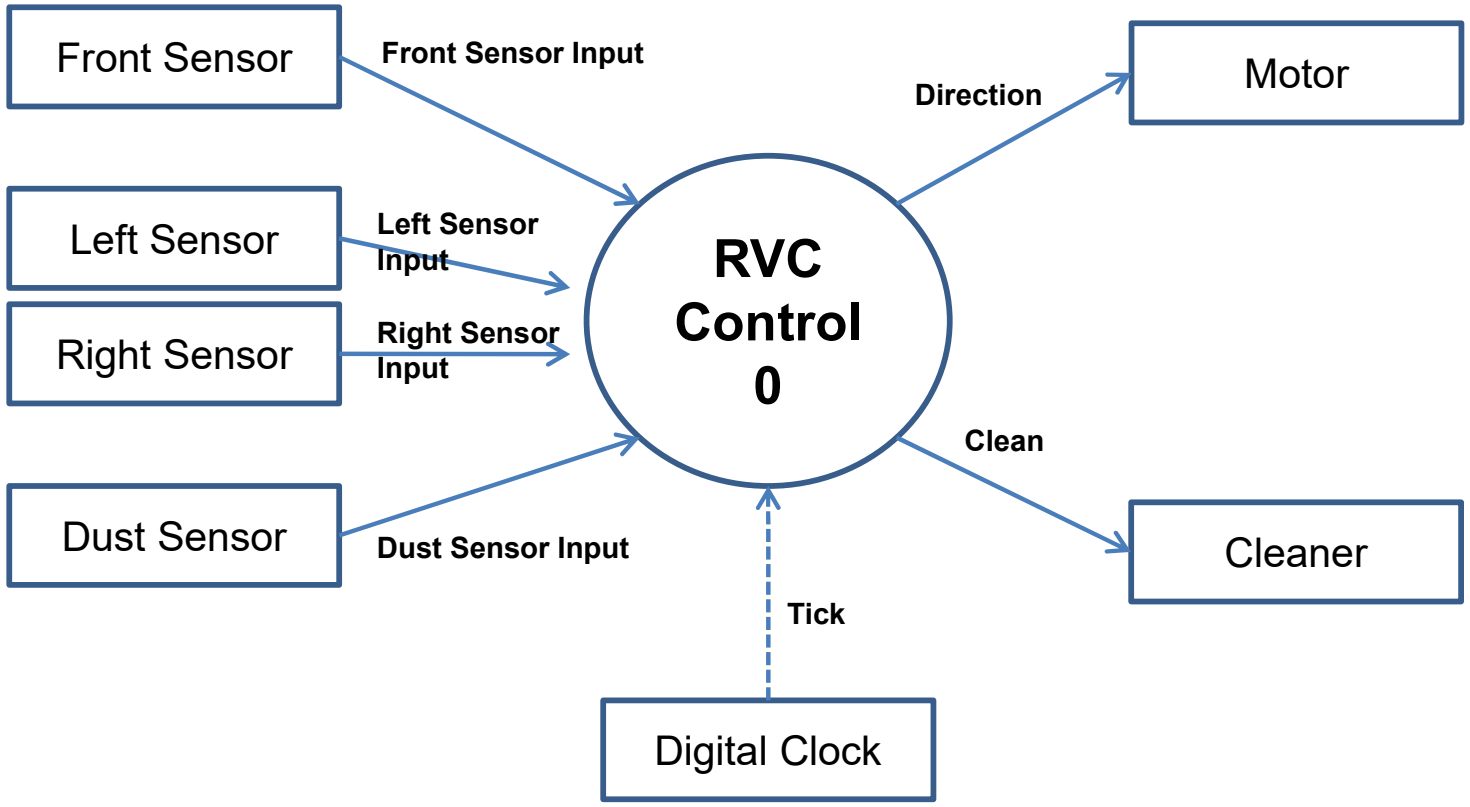
- A planning instrument



Revisited: DFD Level 0 - RVC Example

Robot Vacuum Cleaner

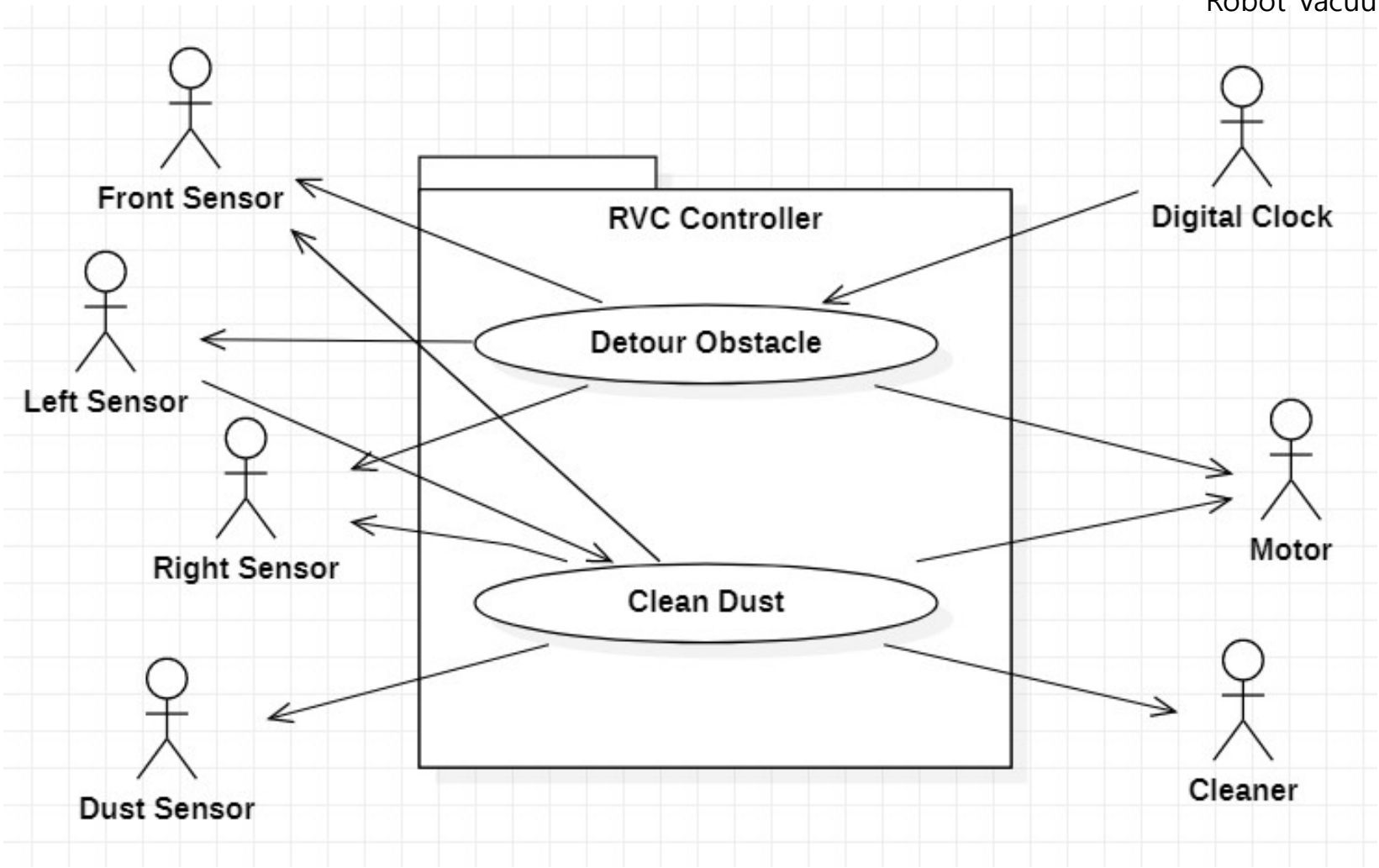
- System context diagram



Use Case Diagram vs. DFD

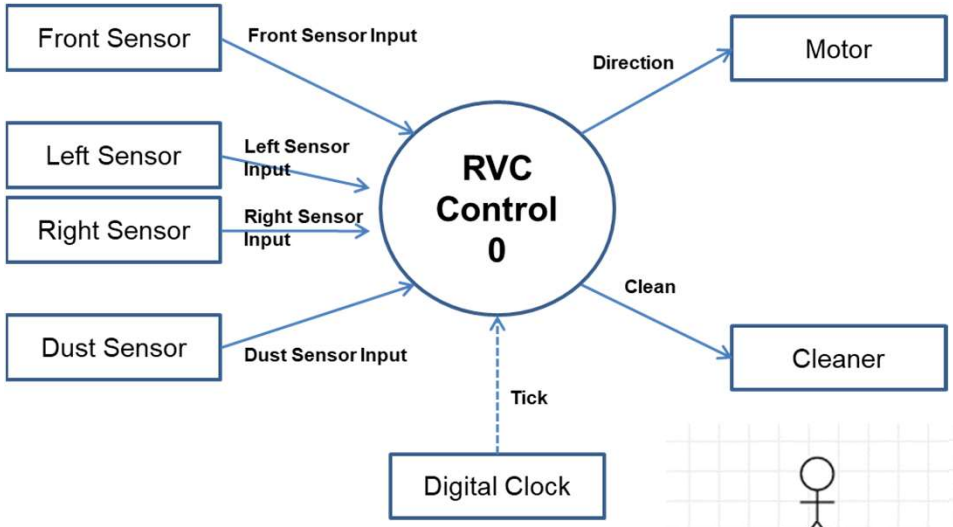
Robot Vacuum Cleaner

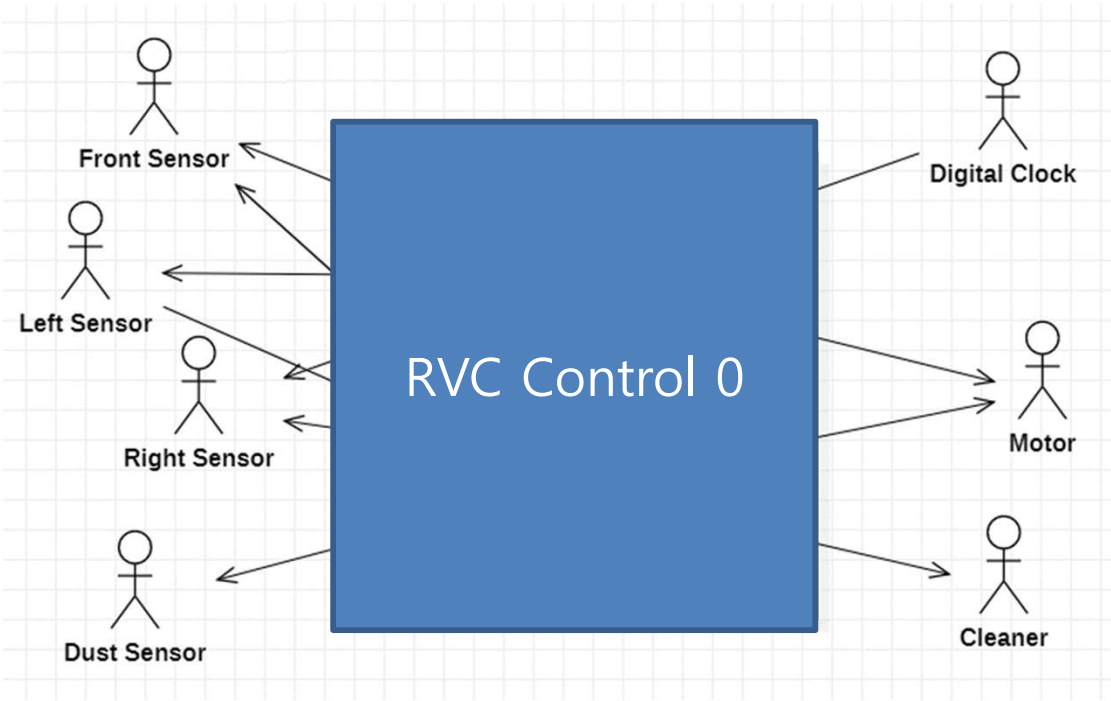
- Use Case Diagram



Use Case Diagram vs. DFD

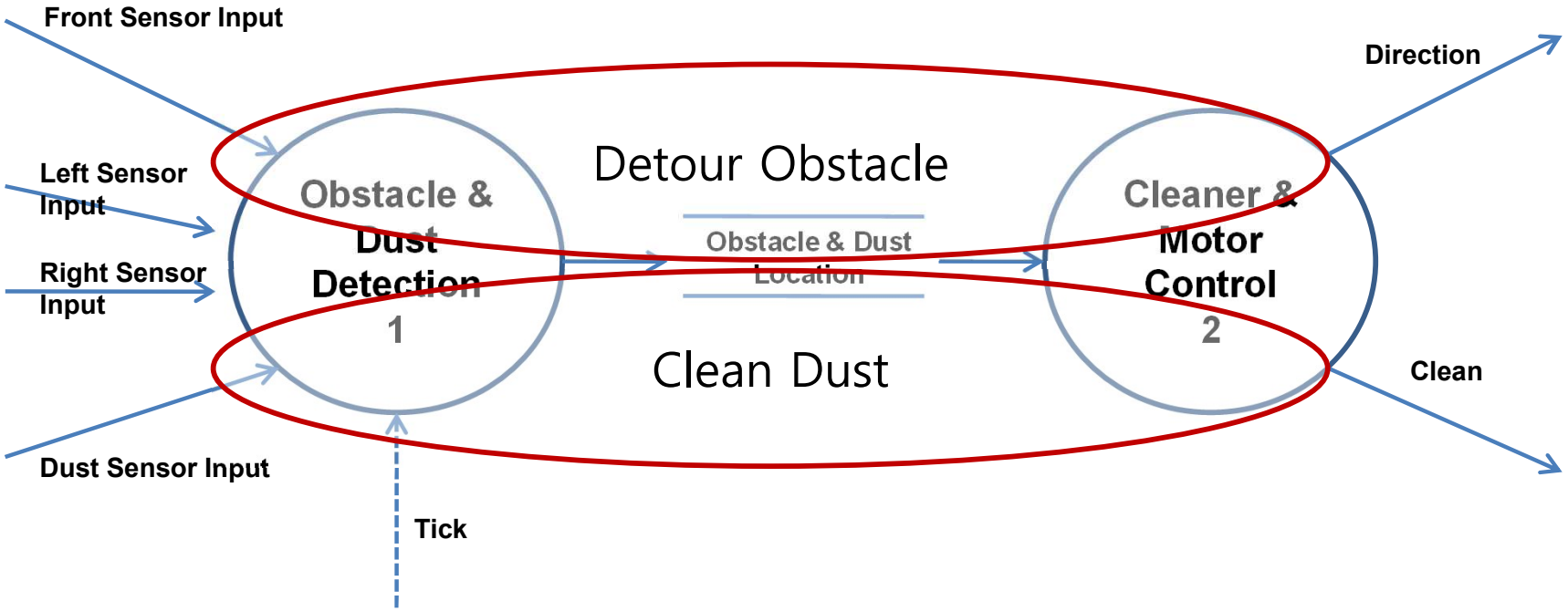
Robot Vacuum Cleaner





Use Case Diagram vs. DFD

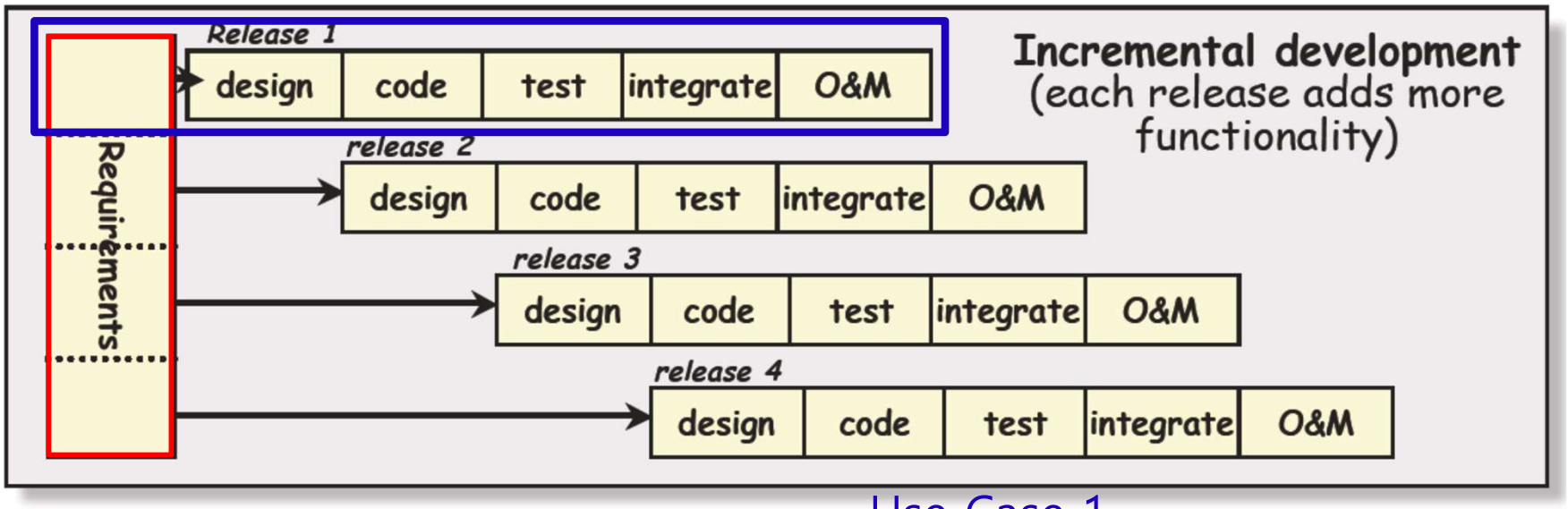
Use Case Model: Scenario-based approach



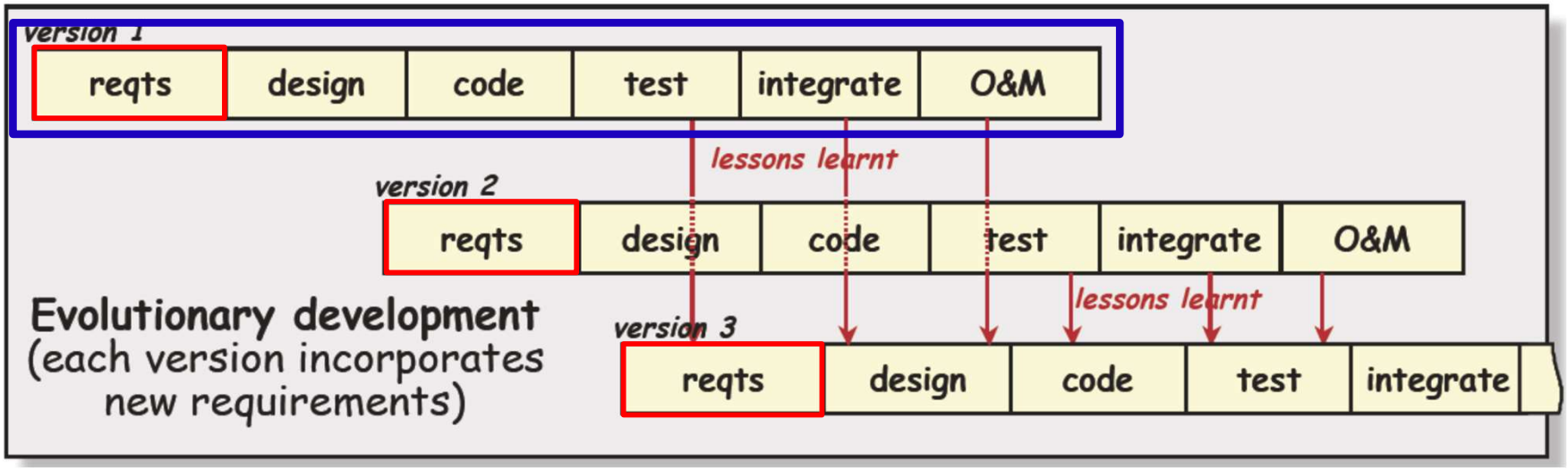
DFD: Top-Down Divide and Conquer approach

Revisited: Phased Lifecycle Models

Use Case 1



Use Case 1



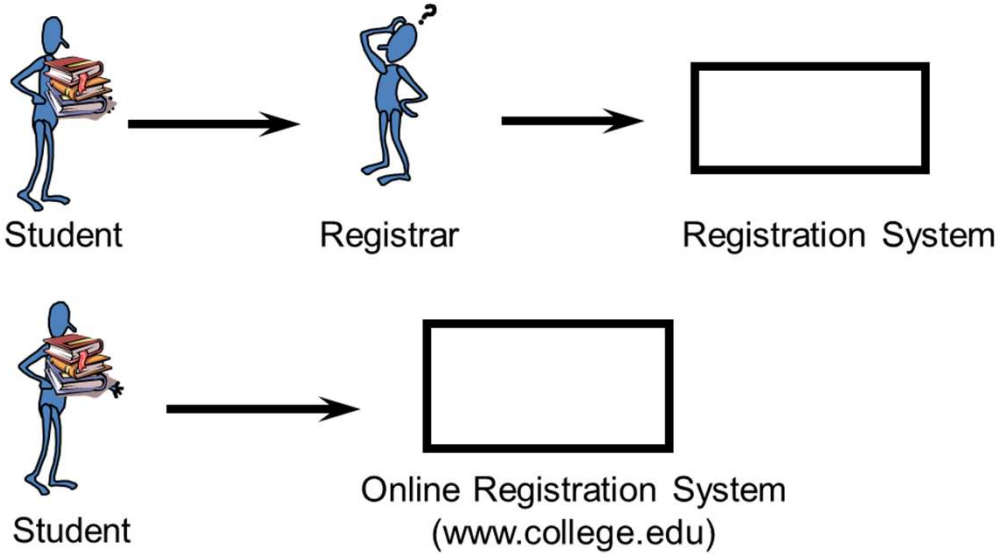
Exercise 1. Use Case Diagram 완성하기

- 주어진 요구사항을 만족하도록 Use Case Diagram을 완성하세요.
- ex1_음료수 자판기 초기요구사항 명세서.doc

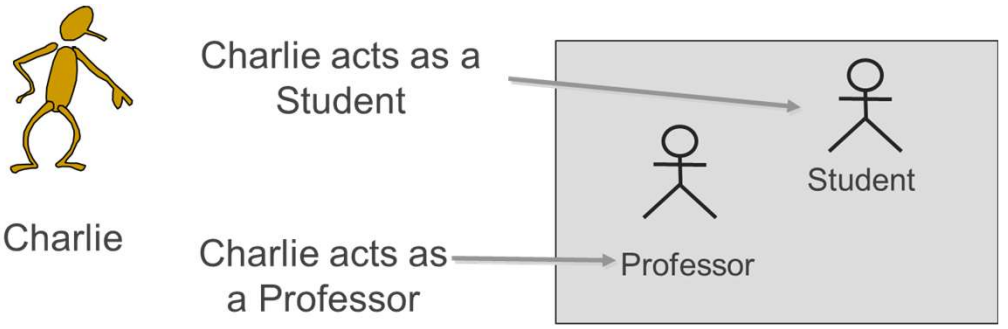
Actors and Use Cases

- Actor
  - Someone/something outside the system that interacts with the system

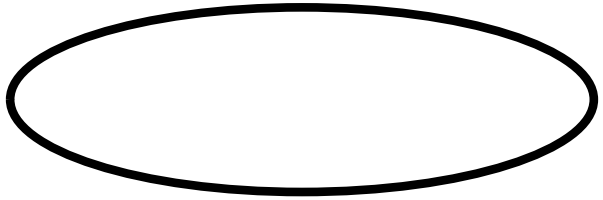
- Who is pressing the keys (interacting with the system)?



- A user can act as several actors



Actors and Use Cases



Use Case Name

A use case defines **a sequence of actions**

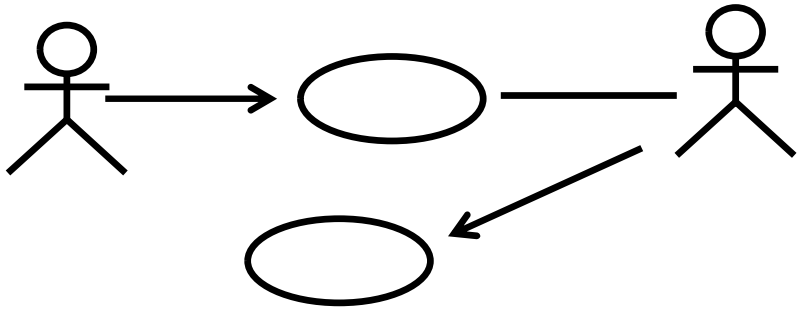
performed **by a system**

that yields an **observable** result of value

to **an actor.**

Use Cases Contain Software Requirements

- Each Use Case
  - Describes actions the system takes to deliver something of value to the actor
  - Shows the system functionality an actor uses
  - Models a dialog between the system and actors
  - Shows a complete and meaningful flow of events from the perspective of a particular actor

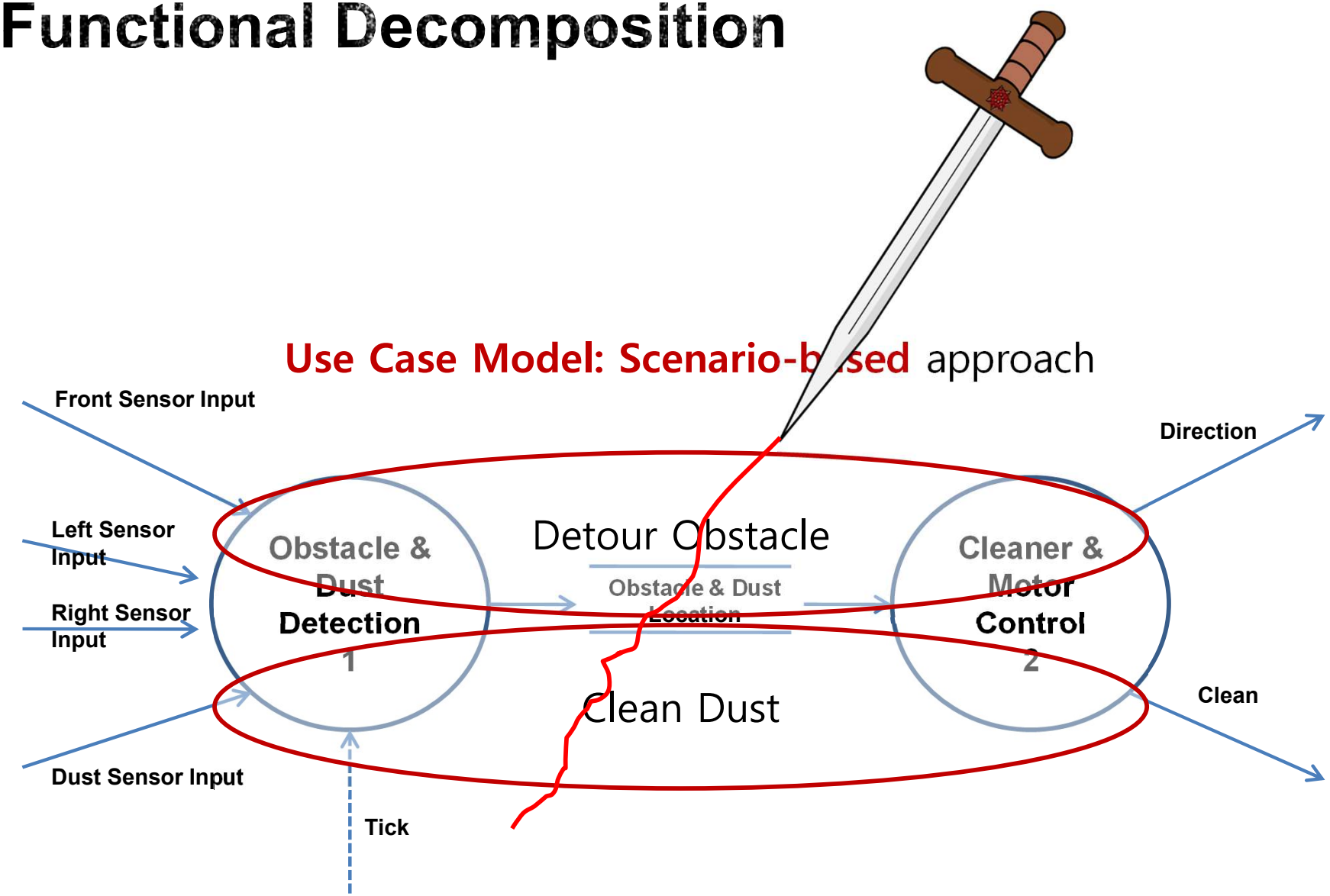


Avoid Functional Decomposition

- **Functional Decomposition**
  - Breakdown of a problem into small isolated parts
  - The parts:
    - Works together to provide the functionality of the system
    - Often do not make sense in isolation

- **Use Cases:**
  - **Are NOT functional decomposition**
  - Keep the functionality together to describe a complete use of the system
  - Provide **context**

Functional Decomposition



Functionally decompose Use Case 사례

Avoid Functional Decomposition

- **Symptoms**
  - Order-relationship among use cases
  - Very small use cases
  - Too many use cases
  - Uses cases with no result of value
  - Names with low-level operations
    - "Operation" + "object"
    - "Function" + "data"
    - Example: "Insert Card"
  - Difficulty on understanding the overall model
- **Corrective Actions**
  - Search for larger context
    - "Why are you building this system?"
  - Put yourself in user's role
    - "What does the user want to achieve?"
    - "Whose goal does this use case satisfy?"
    - "What value does this use case add?"
    - "What is the story behind this use case?"

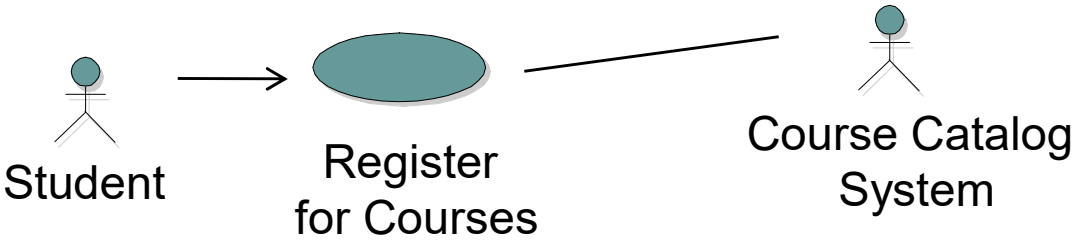
Checkpoint for Use Cases

- The use-case model clearly presents the behavior of the system; it is easy to understand what the system does by reviewing the model.
- All use cases have been identified; the use cases collectively account for all required behavior.
- All functional requirements are mapped to at least one use case.
- The use-case model contains no superfluous behavior; all use cases can be justified by tracing them back to a functional requirement.→ 모든 use case는 초기 functional requirements로의 backward tracing이 가능해야 함
- Do the use cases have unique, intuitive and explanatory names so that they cannot be mixed up at a later stage? If not, change their names.
- Do customers and users understand the names and descriptions of the use cases?
- Does the brief description give a true picture of the use case?
- Is each use case involved with at least one actor?
- Do any use cases have very similar behaviors or flows of events?

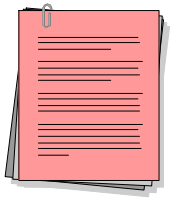
Checkpoint for Actors

- Have you found all the actors? That is, have you accounted for and modeled all roles in the system's environment?
- Is each actor involved with at least one use case?
- Can you name at least two people who would be able to perform as a particular actor?
- Do any actors play similar roles in relation to the system? If so, you should merge them into a single actor.

Diagram → Outline → Detail

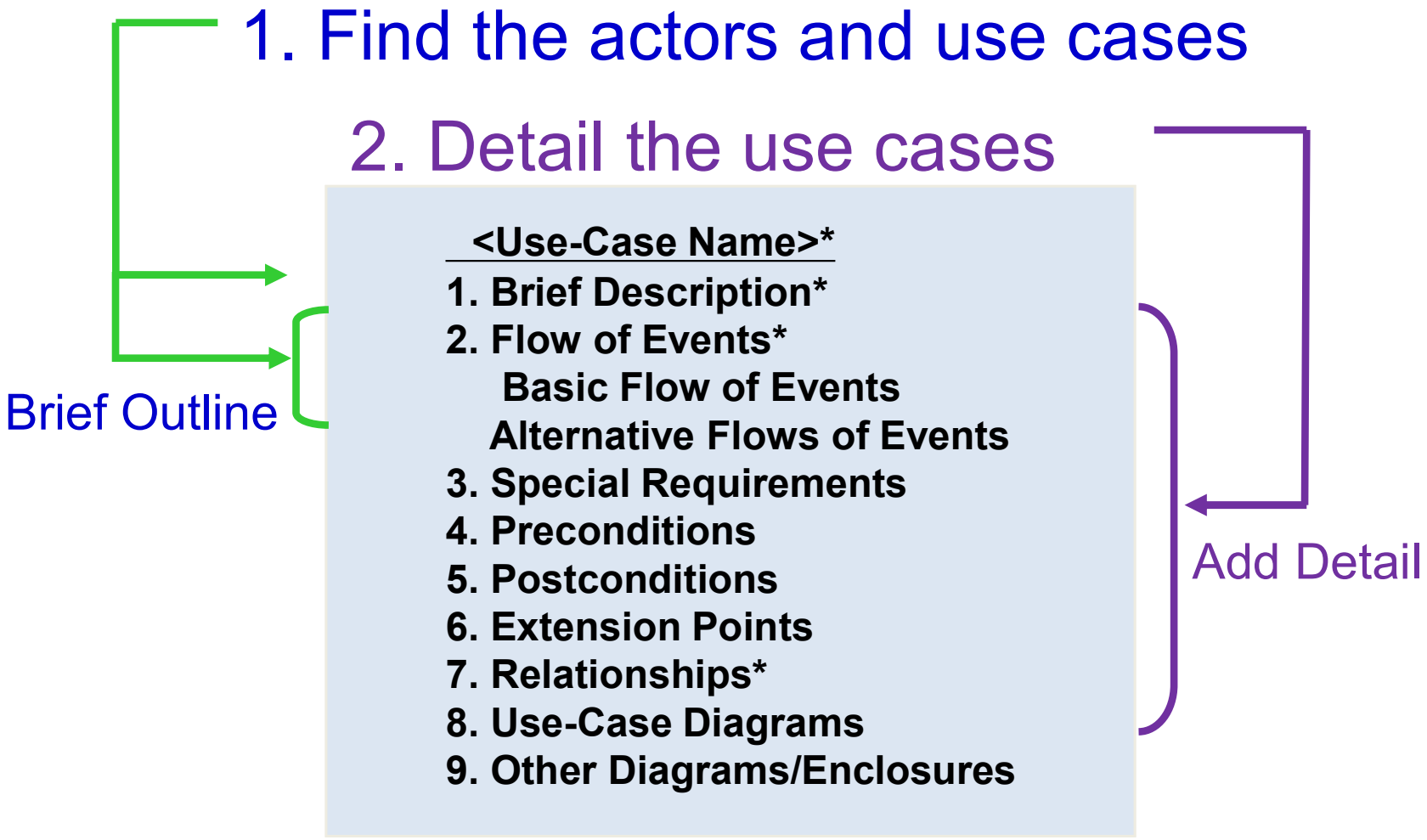


Register for Courses : Outline
- Brief description
- Flow of events
- Step-by-Step



Register for Courses :
Use-Case Specification
- Brief description
- Flow of events
- Special Requirements
- Pre/Post Conditions

Detail a Use Case



Detail the Basic Flow of Events

Select courses to teach

Basic Flow

The use case starts when a Professor wishes to sign up to teach some course offerings for the upcoming semester.

1. **Retrieve and display the list of course offerings**
   The system retrieves and displays the list of course offerings the professor is eligible to teach for the current semester. The system also retrieves and displays the list of courses the professor has previously selected to teach.
2. **Professor selects course offerings**
   The professor selects and/or de-selects the course offerings that he/she wishes to teach for the upcoming semester.
3. **Remove the professor**
   The system removes the professor from teaching the de-selected course offerings….

Structure the flow into steps

Number and title each step

Describe steps (1-3 sentences)

Make each step a roundtrip of events

Detail Specific Alternative Flows

Register for courses

Alternative Flows

A1. No Schedule Found

If, in the **Submit Schedule** sub-flow, the system determines that the Student has not satisfied the necessary prerequisites, or that the selected course offering is full, or that there are schedule conflicts, an error message is displayed. The Student can either select a different course offering and the use case continues, save the schedule, as is (see **Save a Schedule** subflow), or cancel the operation, at which point the **Basic Flow** is re-started at the beginning.

Describe what happens

Start

Condition

Actions

Resume

Alternative Flow: Example

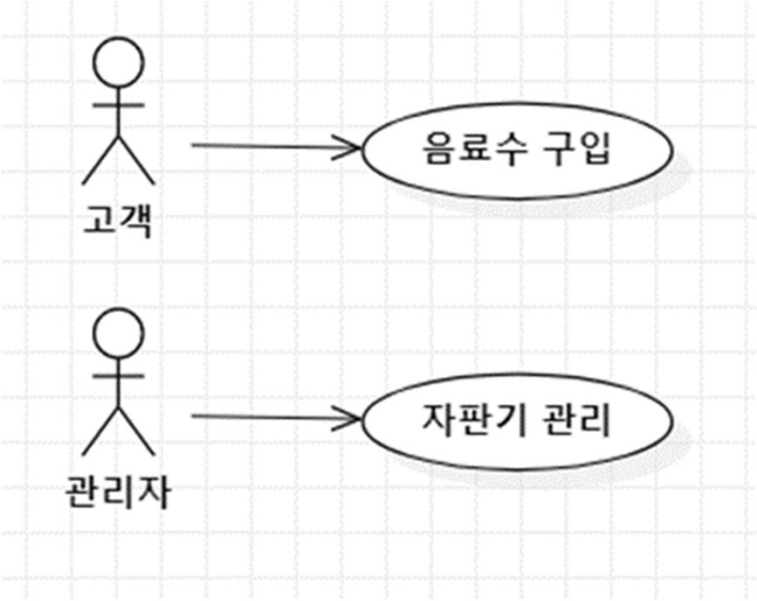
A5. 환율 엑셀파일 Loading선택의 경우 → Condition

분기점 : 대체흐름(A1) 7 → **Start**

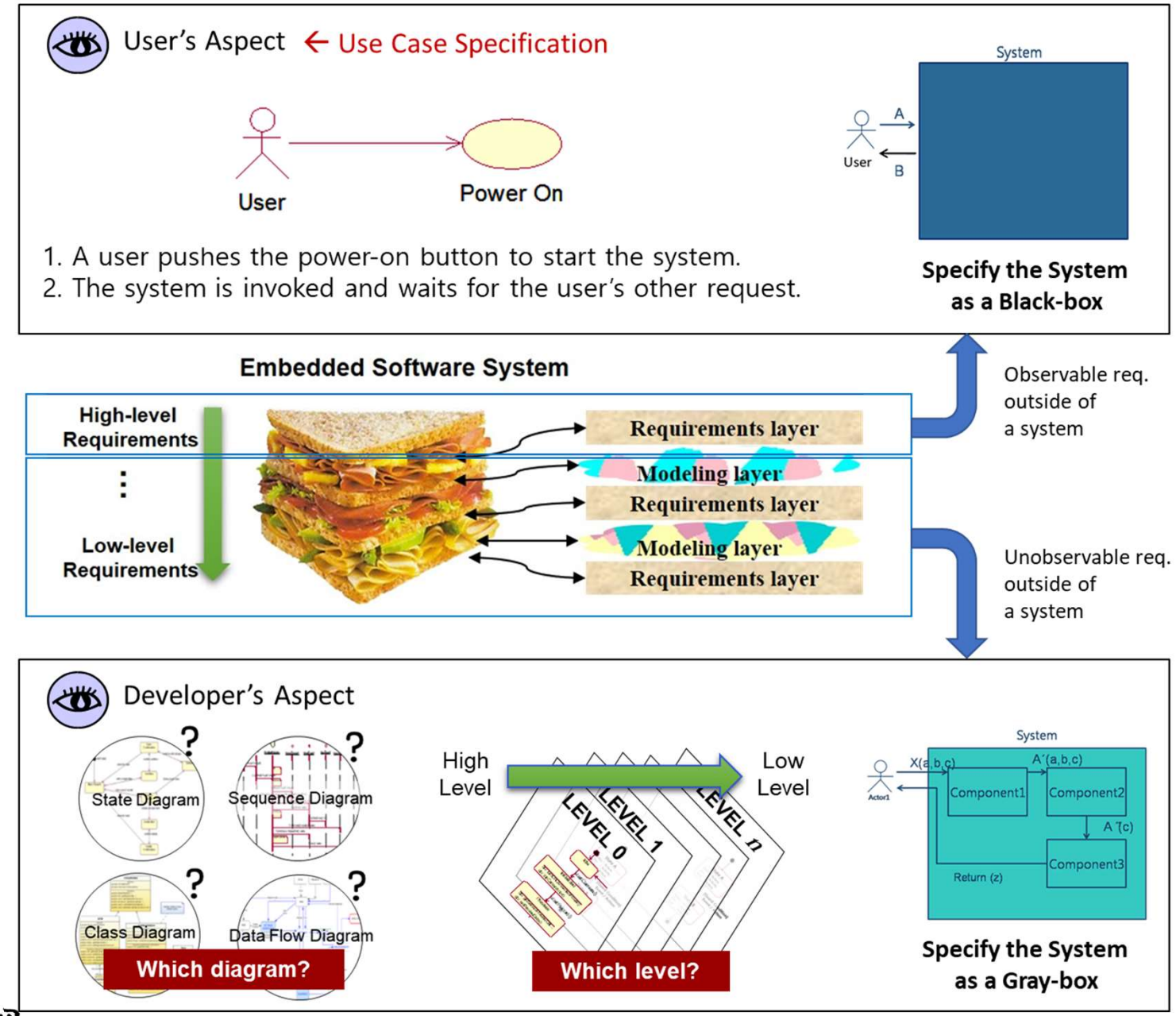
1. 시스템은 Telerate, Reuter로부터 받은 환율자료에 대한 전환자료가 존재하는 해당위치의 엑셀파일을 Loading해서 화면의 각 항목에 display한다.
2. 엑셀파일 자동Loading 실패인 경우, E4수행
3. 본점Teller는 화면의 내용을 검토한 후 수정할 내용에 대한 수정작업을 수행한 후 저장버튼을 선택한다.
4. 대체흐름(A1) 8로 돌아감 → **Resume**

Exercise 2. Use Case Specification 작성하기

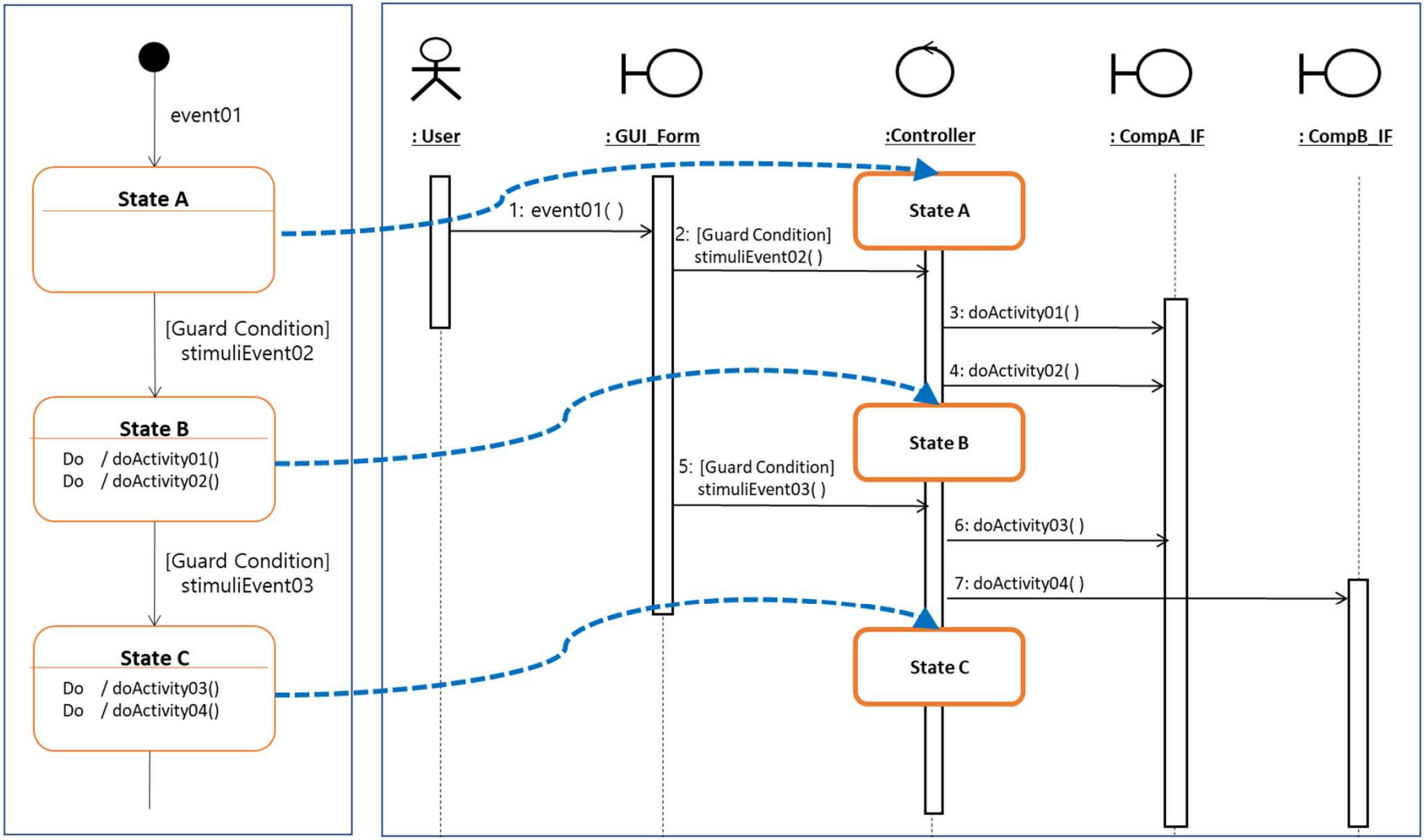
- Exercise 1과 동일한 초기 요구사항을 만족시키도록 Use Case Spec을 작성하세요.
- 단, 이번에는 Use Case 모델은 아래 다이어그램과 같다는 가정 하에, 각 Use Case Specification을 완성하세요.(앞의 Ex1의 Use Case구성과 다릅니다!! 체크하고 시작해 주세요~)
- 음료수 자판기 시스템 요구사항 명세_Template.docx



Difficulties in Specifying Functional Requirements of Embedded Systems



A Recommendation: UC Model + α



(a)State diagram of a controller

(b)Sequence diagram showing interactions among internal components of an embedded system

The State-of-the-Art Requirements Modeling Methods

1. **Structured analysis**
   - Data Flow Diagram (**DFD**)
   - State Transition Diagram (**STD**)
   - Entity-Relation Diagram (**ERD**)

2. **Use Case analysis**
   - Use Case Modeling (**UC**)

3. **Architectural trade-off analysis**
   - Quality Attribute Scenario(**QAS**)

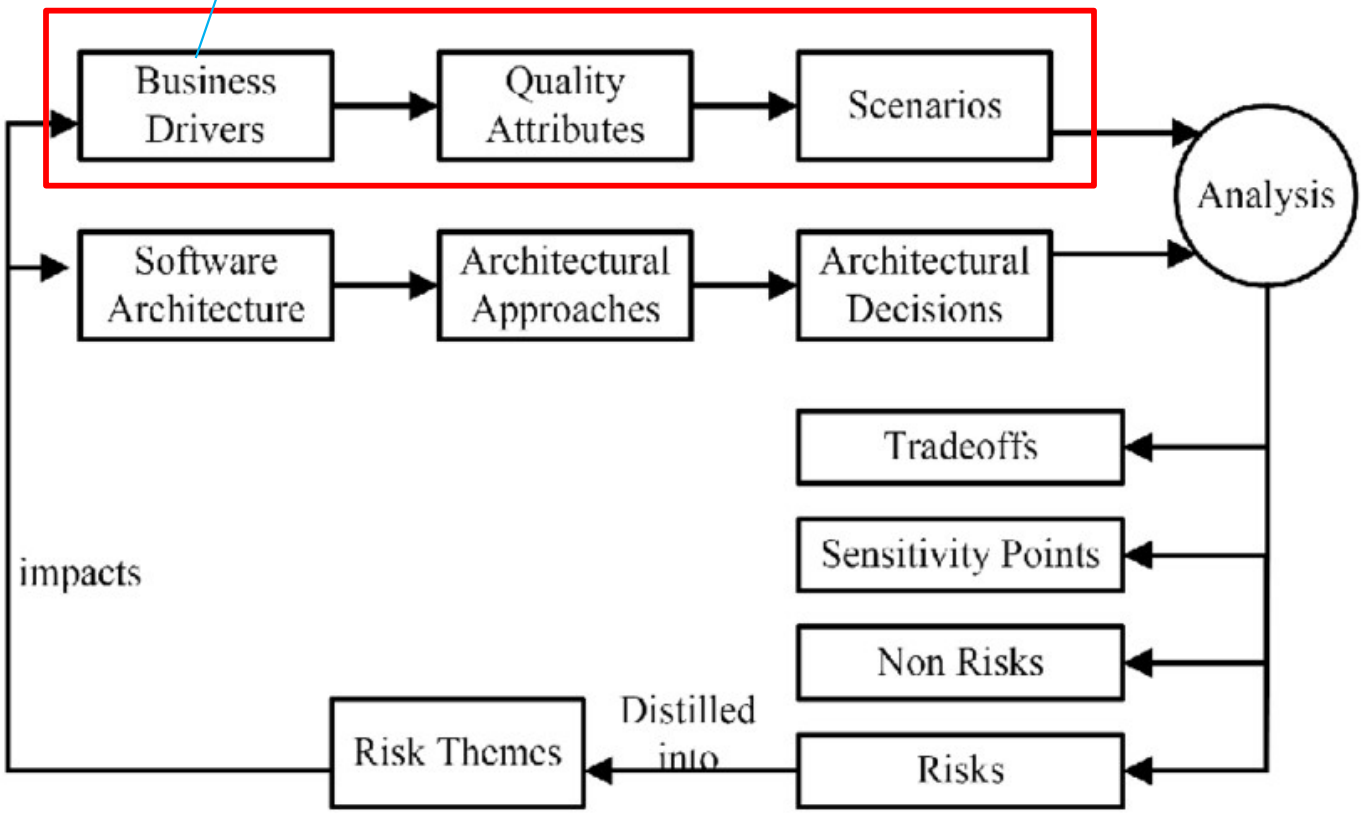
Non-Functional Requirements

4. **Goal Analysis**

3. Architectural Trade-off Analysis (Method)

- ATAM was developed by the Software Engineering Institute(SEI) at the Carnegie Mellon University. Its purpose is to help choose a suitable architecture for a software system by discovering trade-offs and sensitivity points.

system functionality, goals, constraints, desired non-functional properties



Quality Attributes

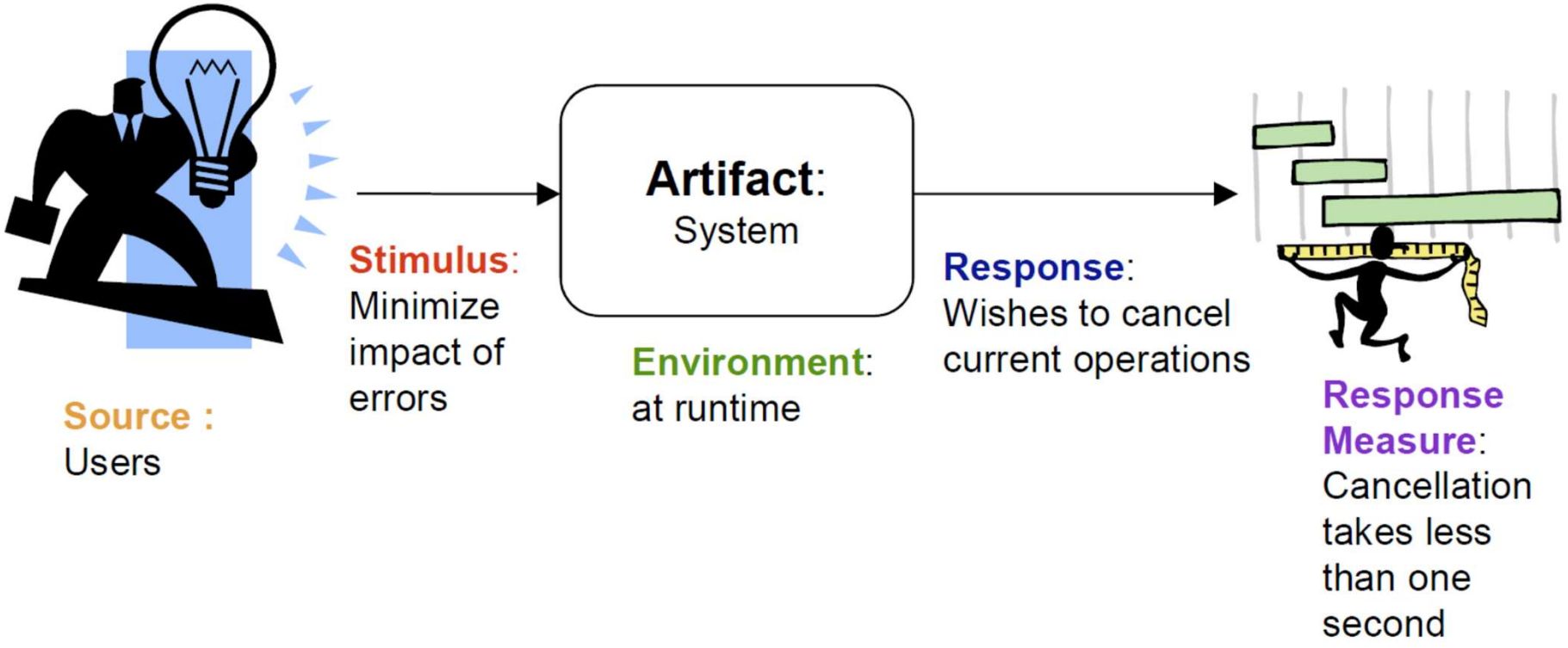
- **Measurable** or **testable properties** of a system
  - Used to indicate how well the system satisfies the needs of its stakeholders
    - Availability, configurability, modifiability, performance, reliability, reusability, security, portability, maintainability, efficiency, usability
  - Emergent properties : not a measure of software in isolation
    - Measures the relationship **between software and its application domain**
    - Cannot measure this until you place the software into its environment
      - Quality will be different in different environments

소프트웨어 자체만을 보고 quality를 평가하는 것이 아니라 소프트웨어와 그것이 적용되는 도메인 간의 관계를 measure하는 새로운 평가 방식

- Software quality is all about **fitness to purpose** of **stakeholders**.
  - *"Does it do what is needed?"*
  - *"Does it do it in the way that its users need it to?"*
  - *"Does it do it reliably enough? fast enough? safely enough? securely enough?"*
  - *"Will it be affordable? will it be ready when its users need it?"*
  - *"Can it be changed as the needs change?"*

Quality Attribute Scenario Example for Usability

- "A user wanting to minimize the impact of an error, wishes to cancel a system operation at runtime; cancellation takes place in less than one second."



Exercise 2. 좀 더 생각해 보기

- 음료수 자판기 컨트롤러의 Quality Attribute로는 어떤 속성이 정의되어야 할까요?

Q1. 음료수 자판기 컨트롤러의 고객은 누구인가요? 음료수를 사는 고객? 자판기 주인?

자판기 주인

Q2. 그렇다면 자판기 주인이 요구하는 음료수 자판기 컨트롤러(혹은 자판기 자체)의 Quality Attribute는 무엇??

1. Availability: 항상 음료수는 판매 가능한 상태인가?
2. Security: 자판기에 자판기 관리자 이외의 사람에 의한 관리자 모드 접근으로부터 얼마나 protect가능한가?
3. Performance: 자판기 시스템은 판매하는 음료의 적정 온도를 얼마나 잘 유지시켜 주는가?

The State-of-the-Art Requirements Modeling Methods

1. Structured analysis

- Data Flow Diagram (**DFD**)
- State Transition Diagram (**STD**)
- Entity-Relation Diagram (**ERD**)

2. Use Case analysis

- Use Case Modeling (**UC**)

3. Architectural trade-off analysis

- Quality Attribute Scenario(**QAS**)

4. Goal Analysis

Functional/ Non-Functional Requirements

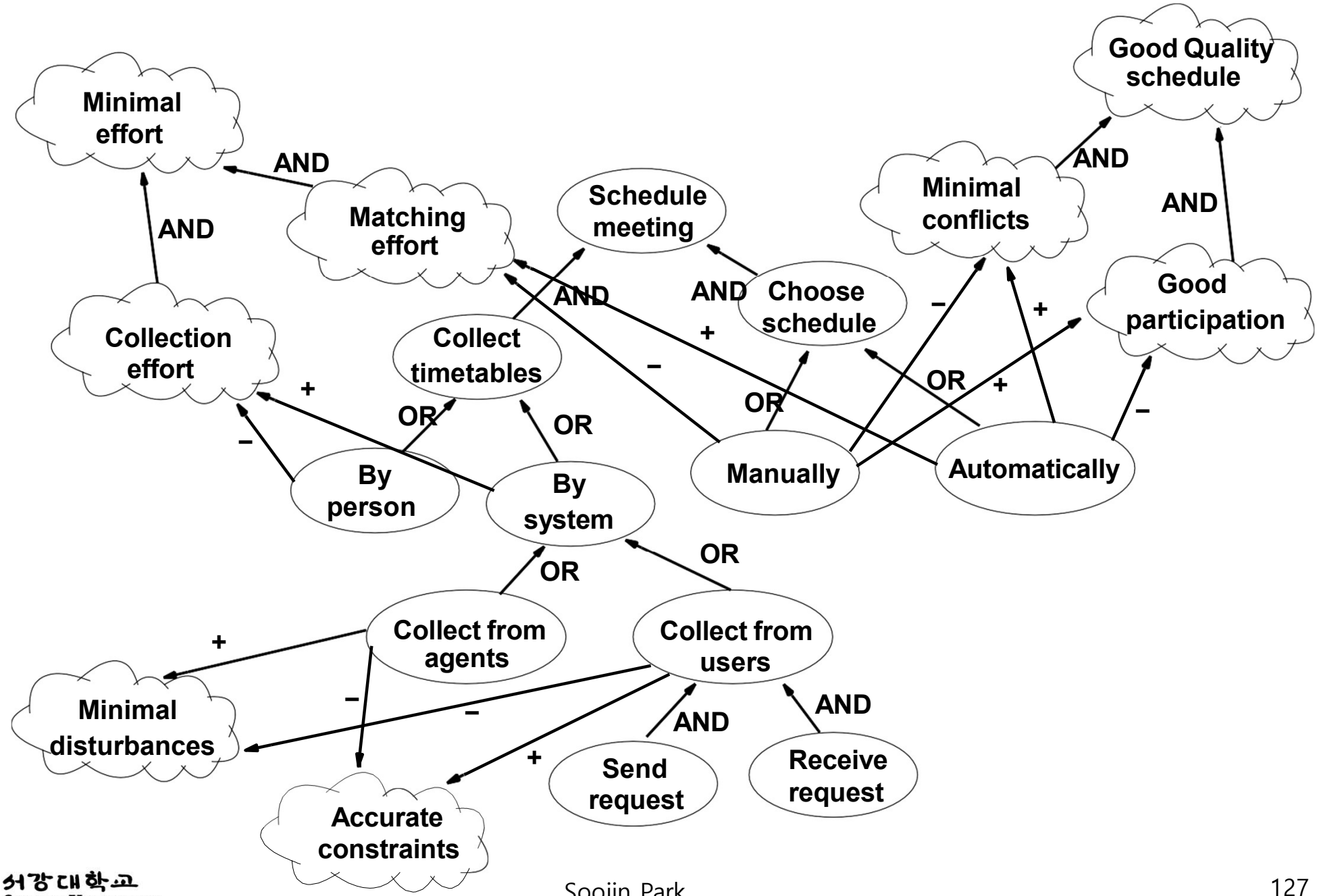
4. Goal Analysis

- **Goal Analysis**
  - Focus on *why* a system is required
    - Express the 'why' as a set of stakeholder goals
  - Goal refinement to arrive at specific requirements
    - Document, organize and classify goals
  - Goal evolution
    - Refine, elaborate, and operationalize goals
  - Goal hierarchies show refinements and alternatives
  - **Goal model** visualizes goal analysis

- Pros
  - Reasonably intuitive
    - Explicit declaration of goals provides sound basis for conflict resolution

- Cons
  - Captures a static picture - what if goals change over time?
    - Can regress forever up (or down) the goal hierarchy

Example: Meeting Scheduler



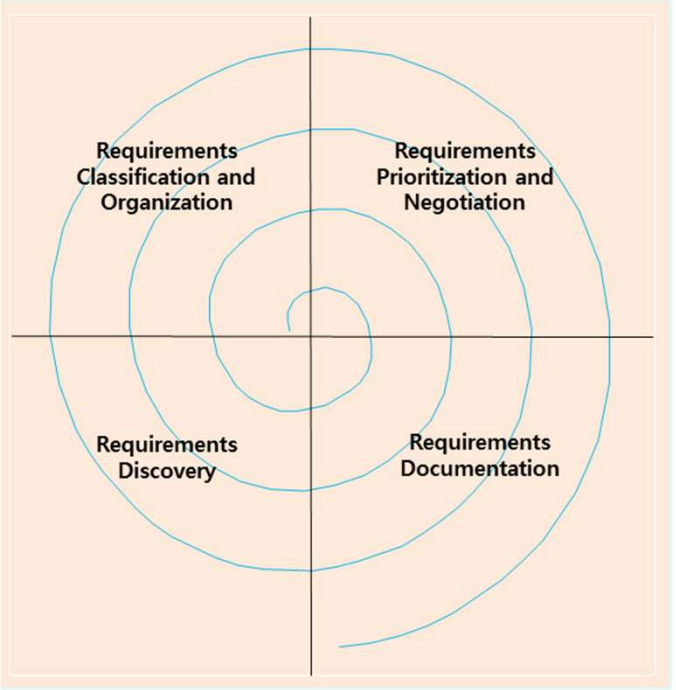
The Requirements Elicitation and Analysis Activities

1. **Requirements Discovery**
   - Interacting with stakeholders to discover their requirements
   - Domain requirements are also discovered at this stage.

2. **Requirements Classification and Organization with a Requirements Model**
   - Groups related requirements and organizes them into coherent clusters

3. **Prioritization and Negotiation**
   - Prioritizing requirements and resolving requirements conflicts

4. **Requirements Documentation**
   - Document requirements
   - Input it into the next round of the spiral



Requirements Prioritization

- **Need to select what to implement, after analyzing requirements**
  - Customers (usually) ask for too much
  - Balance time-to-market with amount of functionality
  - Decide which features go into the next release

- For each requirement/feature, ask:
  - How important is this to the customer?
  - How much will it cost to implement?
  - How risky will it be to attempt to build it?

- **Perform Triage:**
  - Some requirements must be included
  - But, some requirements should definitely be excluded

Considerations in Requirements Prioritization

- **Find factors that affects priority: Return on Investment(ROI) Analysis**
  - How much does the customer want it?
  - How much cost to develop?
  - How much time to deliver?
  - How technologically difficult?
  - How organizationally difficult?
  - How much will the business benefit?

- **Not all factors apply to all projects**
  - Each factor's importance varies from project to project
  - '*Relative*' importance is different to everyone

- **Include all major stakeholders:**
  - We need to prioritize the requirements in collaboration with the customers and developers
  - We must decide on a subset of requirements to be first implemented among various stakeholder interests
  - We need to remember that more influence is exercised by a particular group of stakeholders

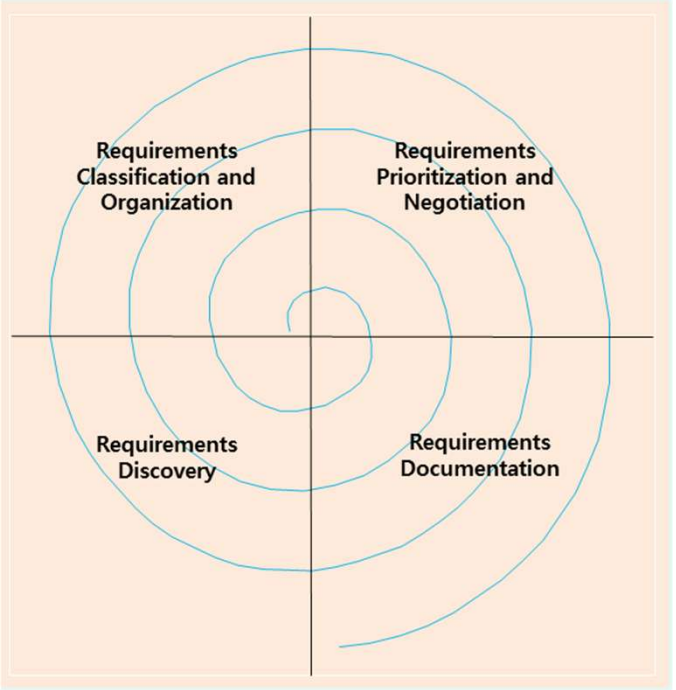
The Requirements Elicitation and Analysis Activities

1. **Requirements Discovery**
   - Interacting with stakeholders to discover their requirements
   - Domain requirements are also discovered at this stage.

2. **Requirements Classification and Organization with a Requirements Model**
   - Groups related requirements and organizes them into coherent clusters

3. **Prioritization and Negotiation**
   - Prioritizing requirements and resolving requirements conflicts

4. **Requirements Documentation**
   - Document requirements
   - Input it into the next round of the spiral



cf. Example of Elicitation Results(Revisited)

- **A list of annotated requirements**

ID	Requirement Text
961	No formal training shall be required to operate the RLM.
955	Any new releases or versions of the software shall be sold as new products. Users must p...
954	User software will not be modified or upgraded.
512	The RLM shall return to the refuel location or dump area to within 10 cm of the user-define...
432	Pressing the screen in an area without a command shall make no sound nor shall it be int...
415	The screen shall be capable of displaying alphanumeric data in blocked, uppercase char...
500	The RLM shall accept lawn and obstacle programming from the user. During programming, th...
321	The RLM shall initiate communications with the GPS through external interface EL-GPS
300	The RLM shall interface with two different external systems, The GPS and the Electronically S...
310	External interfaces include the receipt of location data from GPS and detection of obstacles...
511	The RLM shall not overcut or undercut the border and user defined obstacles by more tha...
510	The RLM shall cut the lawn only within the area defined by the user during the programming.
550	Border programming shall be required to be completed by the user prior to accepting the oth...
411	The Screen shall be 16.25 mm (high) by 105 mm (wide) and capable of displaying two row...
446	Serious errors (for example, blade fouling, Requirement 179) shall not have a button on th...
553	Programming border data shall be terminated by a user request, or when the RLM returns t...
554	After the termination, the RLM shall be ready to receive another command.
418	The screen shall be used to display information from the RLM to the user and accept dire...
561	User shall guide the RLM to the obstacle and indicated that the boundary of obstacle will ...
562	RLM shall record sufficient data (e.g. from GPS) to meet the accuracy requirements stated...
552	RLM shall record sufficient data (e.g. from GPS) to meet accuracy requirements stated in ...
551	User shall guide the RLM to the border of the lawn and indicate that the boundary will be ...

Annotated Requirements Lists after Prioritization and Triage Steps

- **Find relevant importance to stakeholders**
  - What should we annotate?
    - Effect and cost
    - In which release?
    - Duration (optional)
    - Technical risk (optional)

- **Requirements should be in a database.**
  - Access, Excel, RequisitePro, **CaliberRM**, RTM, DOORS, etc.

Annotate Requirements Example

We've ANNOTATED the features.

	ID	Requirement Text	Estim... Devel...	Tec... Risk	Priority	Rel...	Relates To	Comments	Child of	Level
...	961	No formal training shall be required to operate the RLM.	0.00	1	High	1.5			960	3
...	955	Any new releases or versions of the software shall be sold as new products. Users must p...	0.00	1	High	1.5			950	3
...	954	User software will not be modified or upgraded.	0.00	1	High	1.5			950	3
...	512	The RLM shall return to the refuel location or dump area to within 10 cm of the user-define...	10.00	6	Medi...	TBD			510	3
...	432	Pressing the screen in an area without a command shall make no sound nor shall it be int...	12.00	4	Medi...	TBD			430	3
...	415	The screen shall be capable of displaying alphanumeric data in blocked, uppercase char...	1.00	1	High	TBD			410	3
...	500	The RLM shall accept lawn and obstacle programming from the user. During programming, th...	35.00	7	High	TBD				1
...	321	The RLM shall initiate communications with the GPS through external interface EL-GPS	22.00	5	High	TBD			320	3
...	300	The RLM shall interface with two different external systems, The GPS and the Electronically S...	120.00	9	High	TBD				1
...	310	External interfaces include the receipt of location data from GPS and detection of obstacles...	22.00	9	High	TBD			300	2
...	511	The RLM shall not overcut or undercut the border and user defined obstacles by more tha...	10.00	4	High	TBD			510	3
...	510	The RLM shall cut the lawn only within the area defined by the user during the programming.	22.00	5	High	TBD			500	2
...	550	Border programming shall be required to be completed by the user prior to accepting the oth...	4.00	3	High	TBD			500	2
...	411	The Screen shall be 16.25 mm (high) by 105 mm (wide) and capable of displaying two row...	3.00	1	High	TBD			410	3
...	446	Serious errors (for example, blade fouling, Requirement 179) shall not have a button on the screen.	00	1	Medi...	TBD	179	Unclear	440	3
...	553	Programming border data shall be terminated by a user request, or when the RLM returns t...	4.00	3	High	TBD			550	3
...	554	After the termination, the RLM shall be ready to receive another command.	4.00	3	High	TBD			550	3
...	418	The screen shall be used to display information from the RLM to the user and accept dire...	5.00	1	High	TBD			410	3
...	561	User shall guide the RLM to the obstacle and indicated that the boundary of obstacle will ...	11.00	6	High	TBD			560	3
...	562	RLM shall record sufficient data (e.g. from GPS) to meet the accuracy requirements stated...	11.00	6	High	TBD	510, 5...		560	3
...	552	RLM shall record sufficient data (e.g. from GPS) to meet accuracy requirements stated in ...	4.00	3	High	TBD	510, 5...		550	3
...	551	User shall guide the RLM to the border of the lawn and indicate that the boundary will be ...	3.00	3	High	TBD			550	3
...	570	Programming refuel location shall be invoked by user during the initial state of programming ...	11.00	5	Medi...	TBD			500	2
...	571	User shall guide the RLM to the refuel location and indicate that the location of RLM is th...	13.00	5	Medi...	TBD			570	3
...	572	RLM shall record sufficient data to meet the accuracy requirements 510, 511, 512, and 5...	5.00	5	Medi...	TBD	510, 5...		570	3
...	573	Programming refuel location shall be terminated after RLM records its location.	6.00	4	Medi...	TBD		Unclear	570	3
...	574	After the termination, the RLM shall be ready to receive another command.	8.00	5	Medi...	TBD			570	3
...	447	In these cases, the RLM must be shut off and the error corrected by the user.	2.00	1	Medi...	TBD			440	3
...	581	User shall guide the RLM to the dump area and indicate that the boundary data of the du...	3.00	5	Medi...	TBD			580	3

6. Quality Attribute Workshop

Quality Attribute Workshop (QAW)

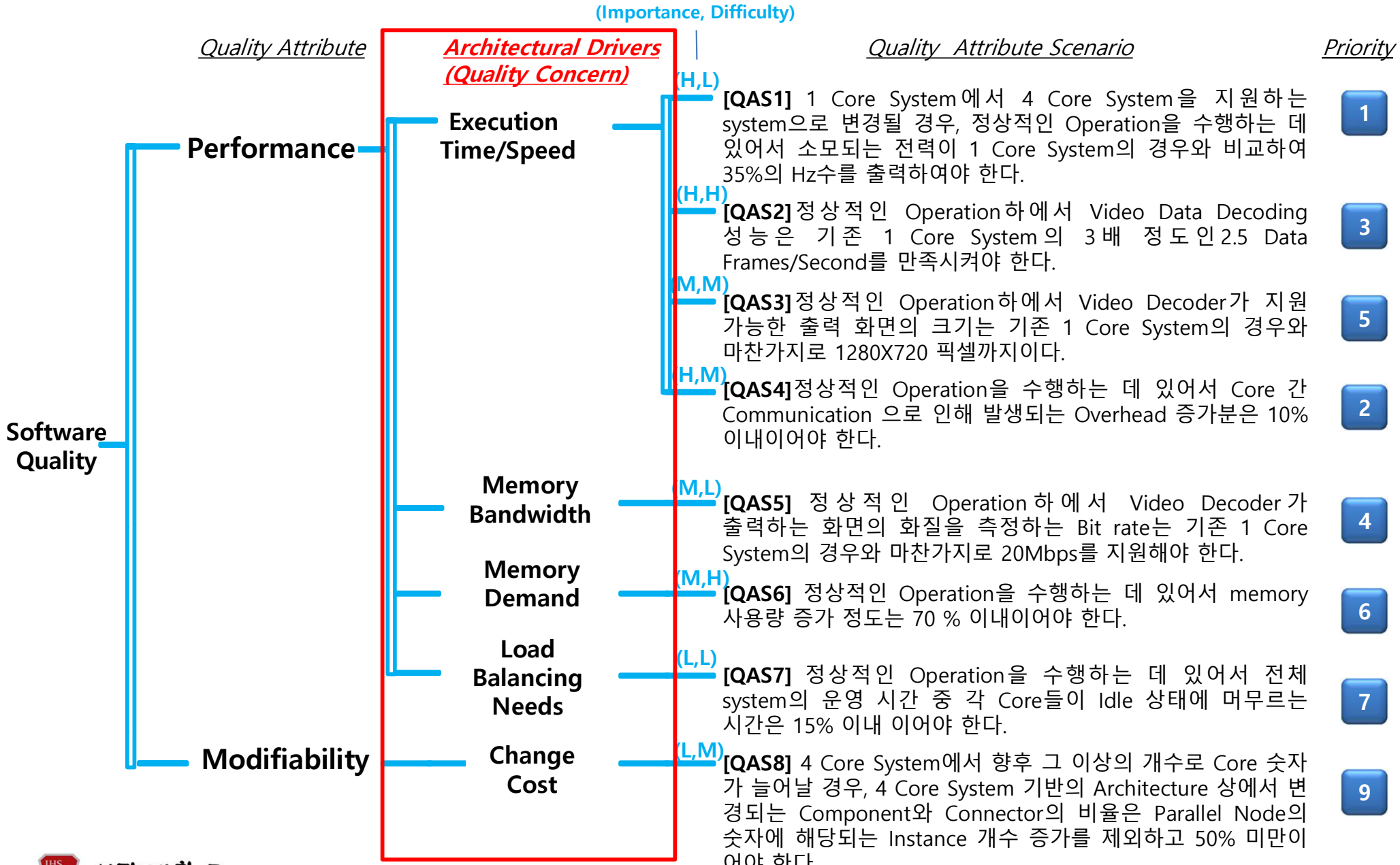
- **Quality Attribute Workshop (QAW)**
  - Facilitated method
    - System-centric
    - Used before the software architecture has been created
  - Engages system stakeholders early in the life-cycle
  - Reveals the driving quality attribute requirements of a software-intensive system
    - Scenario-based

- Outputs of a QAW
  - **Quality attribute requirements** for the system, documented as refined and prioritized **QAS**.
  - The quality attribute scenarios can then be used as the basis for designing the software architecture for the system.

The Traditional QAW Steps

1. QAW Introduction
2. Business/Mission Presentation
3. Architectural Plan Presentation
4. Identification of Architectural Drivers
5. Scenario Brainstorming
6. Scenario Consolidation
7. Scenario Prioritization
8. Scenario Refinement

Construction of a Quality Attribute Tree



The Traditional QAW → Mini-QAW

1. **QAW Introduction** → *Keep it short*

SKIP

5. **Scenario Brainstorming**
6. **Scenario Consolidation**
7. **Scenario Prioritization**

} *Modify*

8. **Scenario Refinement** → *Homework*

Mini-QAW

1. Mini-QAW Introduction
2. Introduction to Quality Attributes, Quality Attributes Taxonomy
3. Scenario Brainstorming
   - "Walk the System Properties Web" activity
4. Raw Scenario Prioritization
   - Dot voting
5. Scenario Refinement
   - While time remains, remainder is homework
6. Review Results with Stakeholders

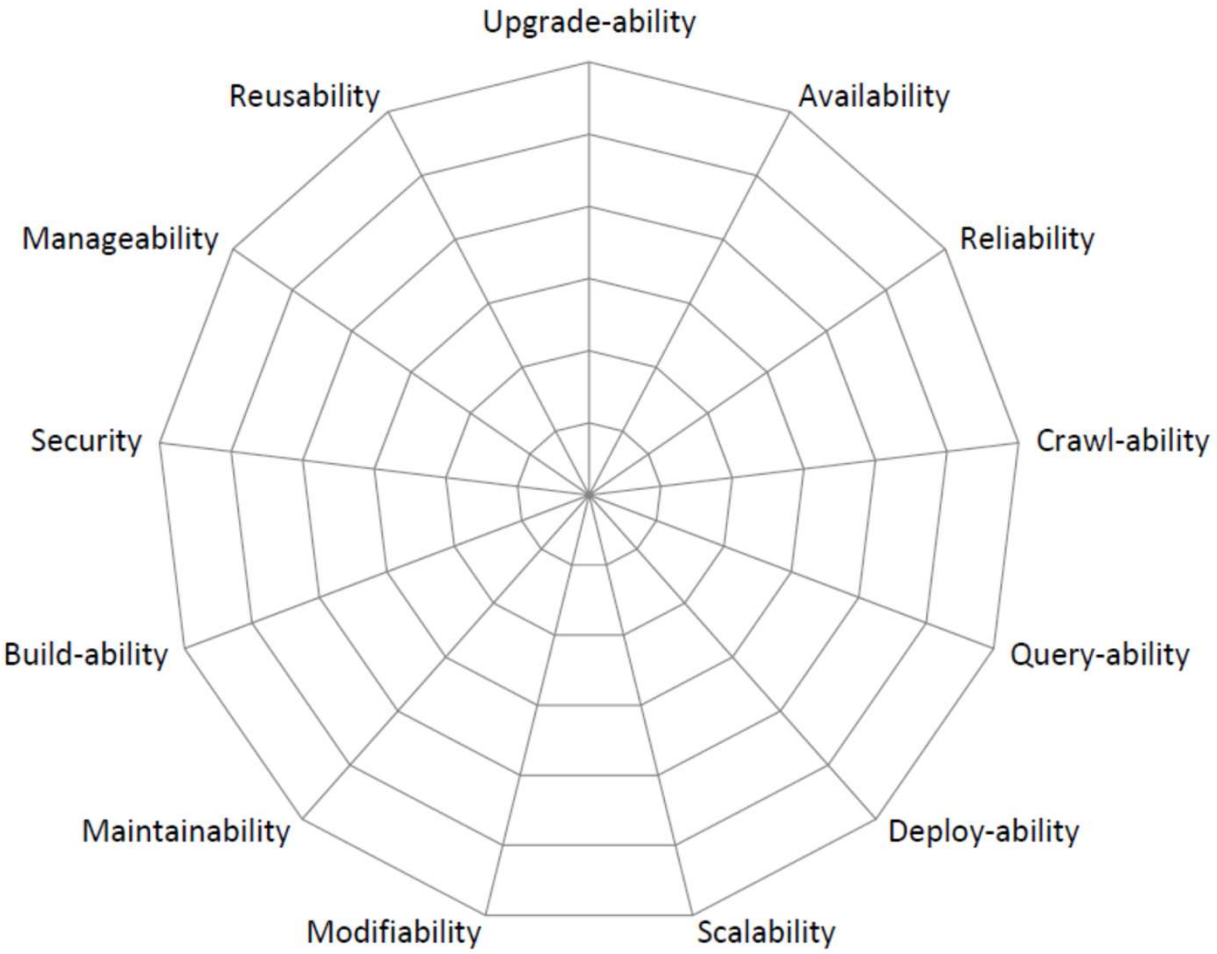
1. Mini-QAW Introduction

- First, sketch **a rough architecture** and **major objectives/functions**
- Take into account specific roles of all stakeholders.
  - For example, “Accessary Service Framework” may have 6~8 different stakeholders and goals.

Stakeholders	주요 역할	희망사항 - Goal
안드로이드 OS 관리자	안드로이드 OS가 안정되게 동작하도록 관리한다.	• 안드로이드 Interface를 확장하지 않았으면 좋겠다. • Interface에 adapter나 wrapper를 붙이지 않았으면 좋겠다. • (가능하면 자세하게)

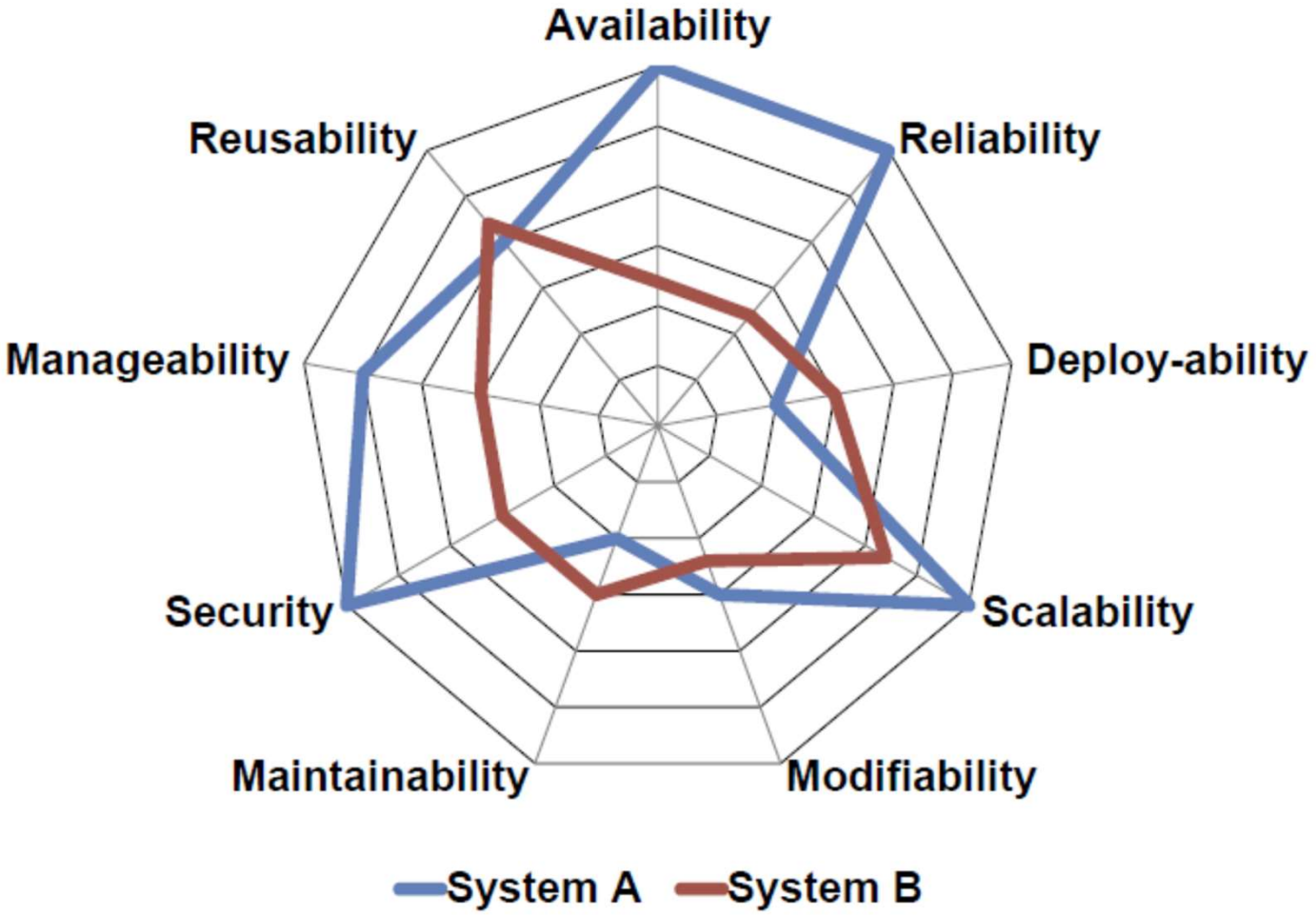
2. Introduction to Quality Attributes, Quality Attributes Taxonomy

- Define your own **system properties web**
  - Select appropriate quality factors for your system under consideration.



System Properties Web

- Different for different systems



3. Scenario Brainstorming

- Identify raw quality attribute scenarios
  - Timing: 30 minutes to 2-3 hours

- Steps:
  1. Start with a Scenario on the web, ask “Is this Quality attribute relevant to your system?”
  2. If Yes, spend 5 minutes brainstorming scenarios / concerns on that scenario.
  3. Write raw scenarios on stickies and put on web
  4. After 5 minutes, move to next scenario

Raw Quality Attribute Scenario

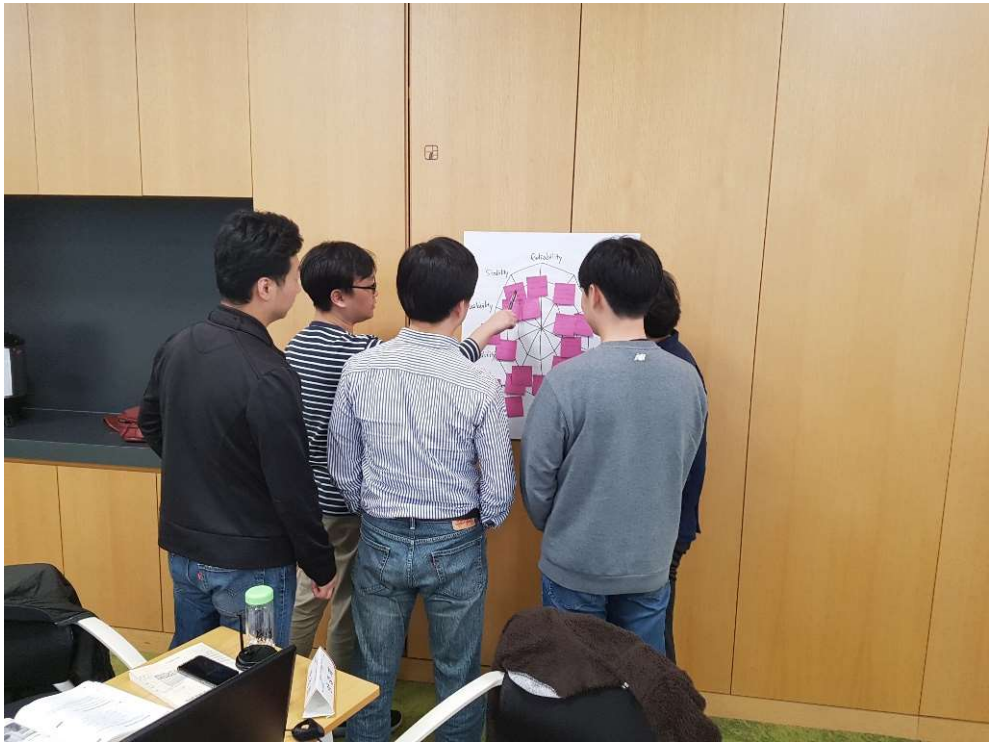
- Informally describes a stakeholder's concern and concrete instances of quality attributes

We need uptime during peak business hours

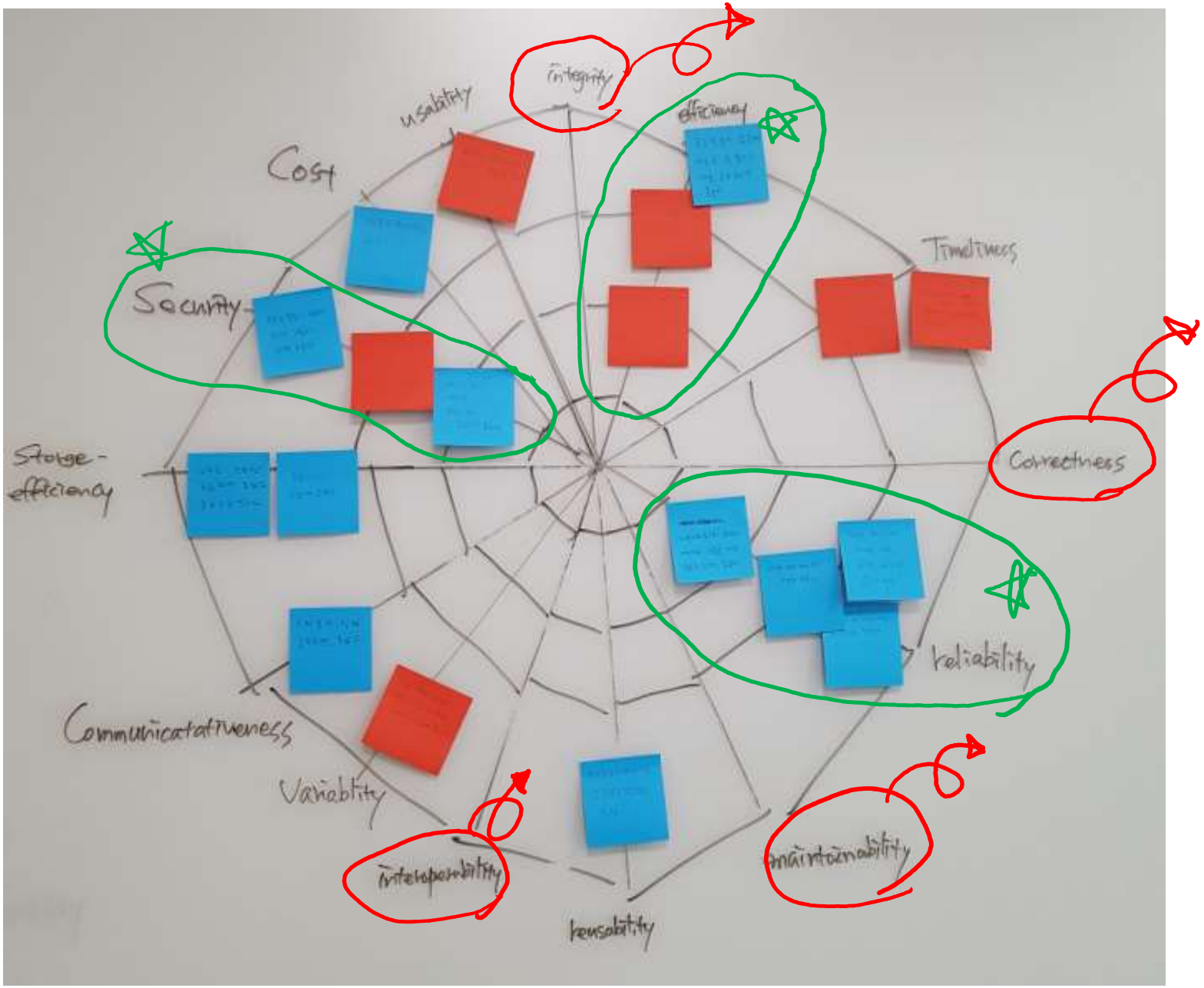
System responds even when parts of the system fail

Peak load is 150 requests per second

“Walk the System Properties Web” Activity



"Walk the System Properties Web" Activity



4. Raw Scenario Prioritization

- Identify Highest Priority Scenarios using dot voting
  - Timing : 5 minutes

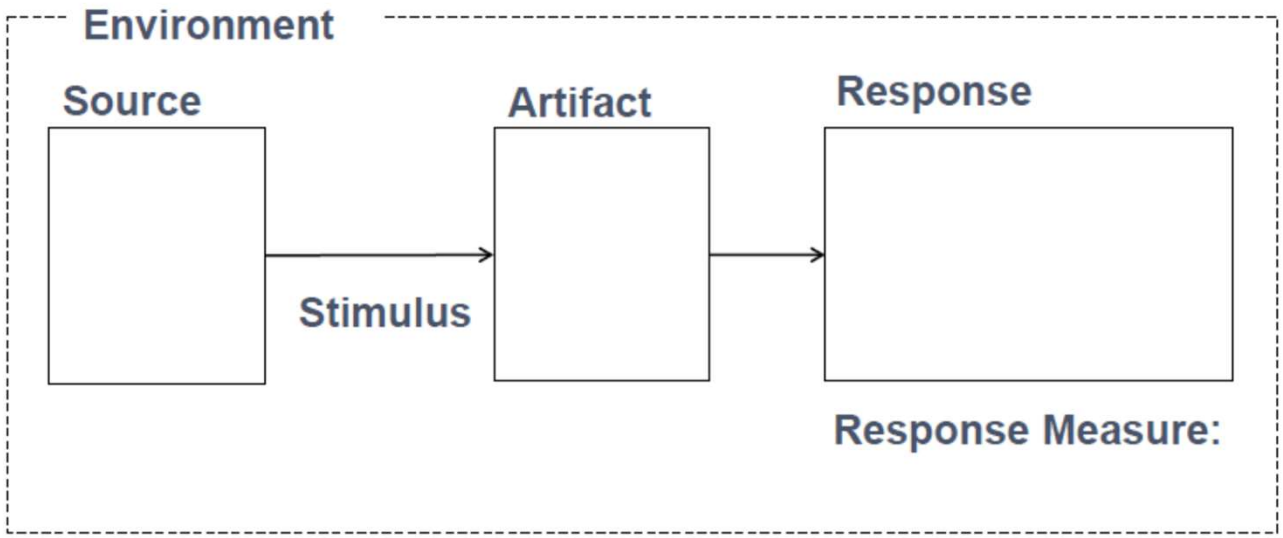
- Steps:
  - Dot Voting:
    - Each stakeholder gets (# scenarios / 3 + 1) dots for scenarios
    - 2 votes to choose “top quality attribute”


Scailability
Modifiability
Performance
Interopera bility
Usability
efficiency

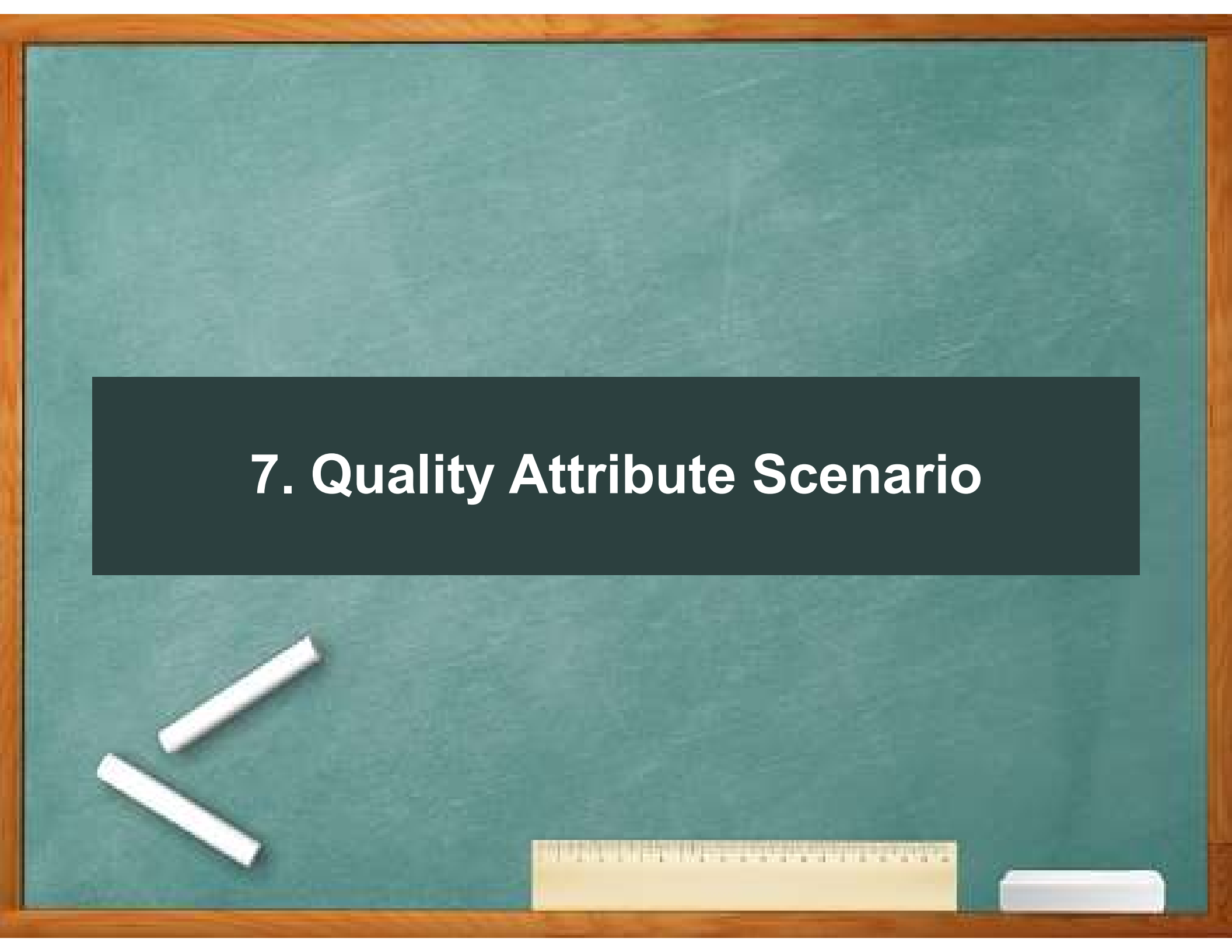
5. Scenario Refinement

- Generate Quality Attribute Scenarios based on raw notes
  - Timing : 30 - 60 minutes

- Steps :
  1. Start with high priority scenario
  2. Fill out the worksheet, identifying the components of a quality attribute scenario
  3. Complete and present to stakeholders

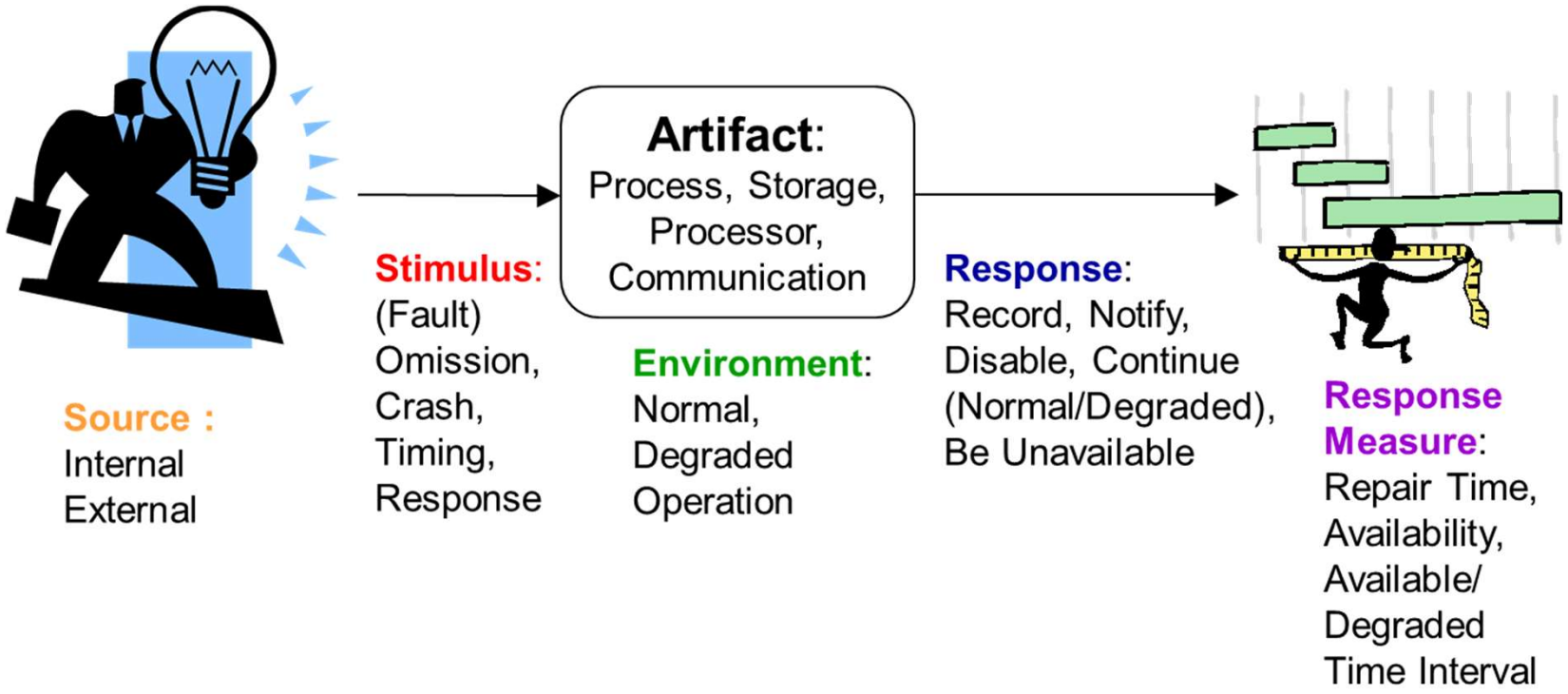


7. Quality Attribute Scenario



Quality Attribute Scenarios

- **QAS (Quality Attribute Scenario)** is an effective way of identifying and specifying quality-attribute-specific requirements.
  - Specific to the particular system under considerations
  - Instantiated from the attribute characterizations of general scenarios



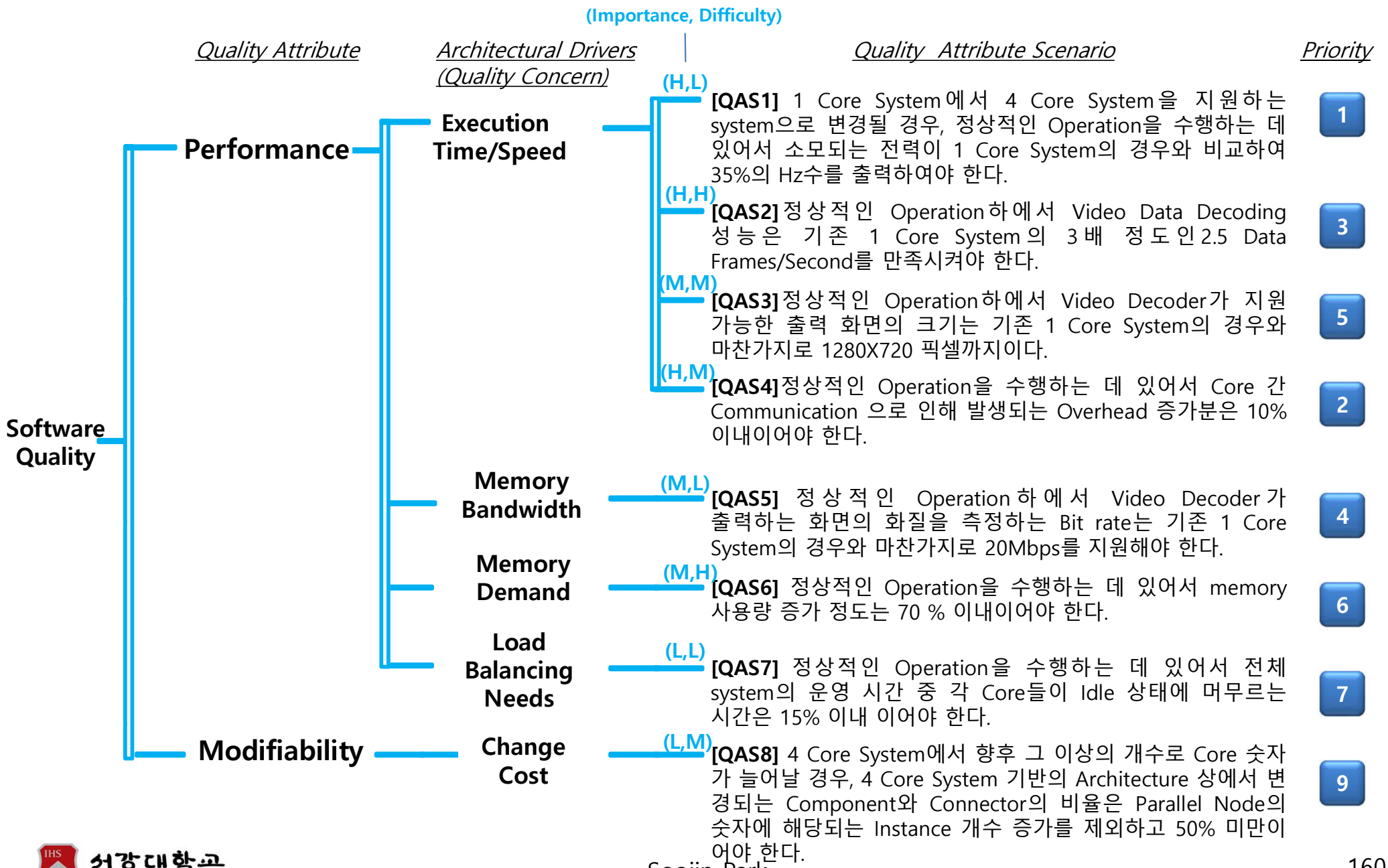
QAS Template

Requirement-ID	QA_###
Category	관련된 Quality Attribute가 무엇인지(ex. Performance, Reliability, Security 등등)
Source	Stimulus를 발생시키는 주체가 무엇인지 기술
Stimulus	시스템에 입력되는 내외부 자극이 무엇인지 기술
Environment	해당 Stimulus 발생시 시스템의 환경(운영모드일 수도 있고, 그 외 다양한 상황 가능)
Artifacts	Stimulus의 영향을 받는 시스템의 내부 모듈, 컴포넌트, 혹은 시스템 전체
Response	기술된 Environment에서 Artifact이 Stimulus를 받아들인 후 취하는 Action이 무엇인지
Response Measure	위의 Response의 정도를 측정하는 단위가 무엇인지(ex. 초당 데이터 처리량, 반응시간 (시간일 경우 Hr인지, Min인지, Sec인지 등등))
Priority	Quality Attribute Tree 상에서의 우선순위
Description	위에 기술된 Source부터 Response Measure까지의 내용을 하나의 문장으로 요약해서 기술. 본 항목이 실제 Quality Attribute Scenario에 해당되며, Quality Attribute Tree의 Leaf Node에 나열됨

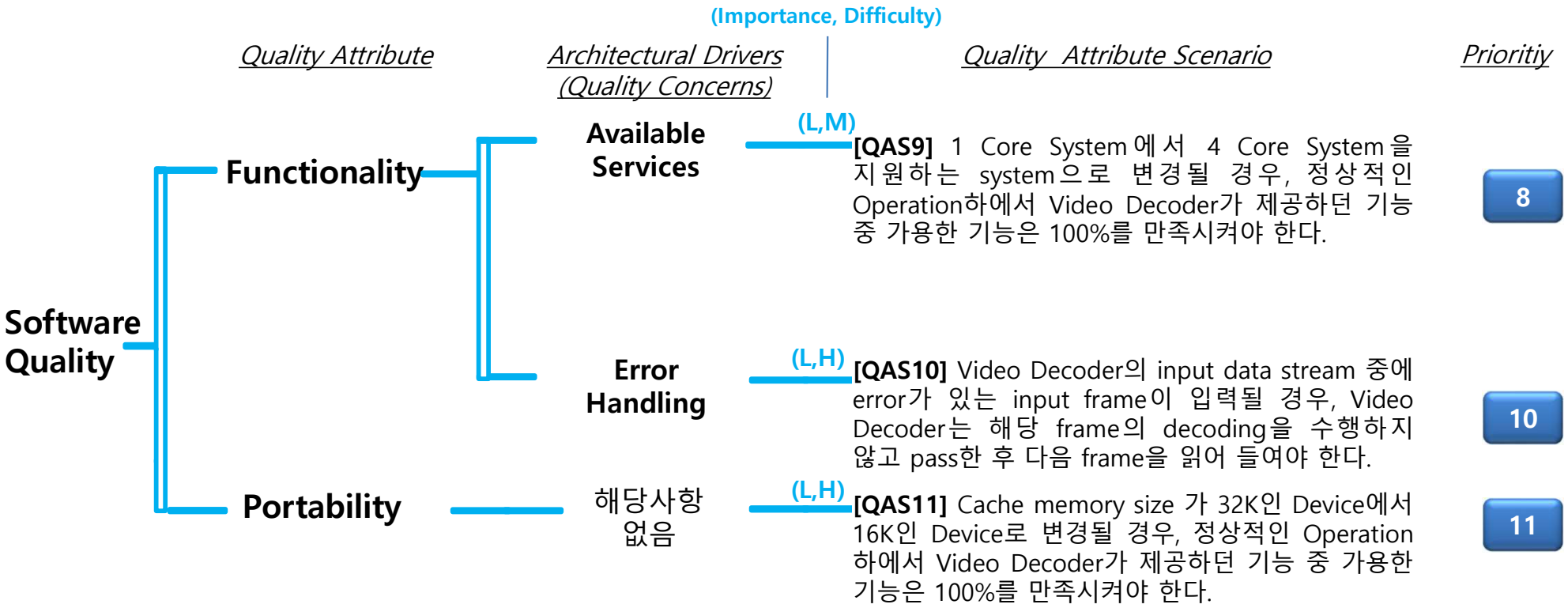
QAS Example

- 로봇청소기 QAS 작성 사례

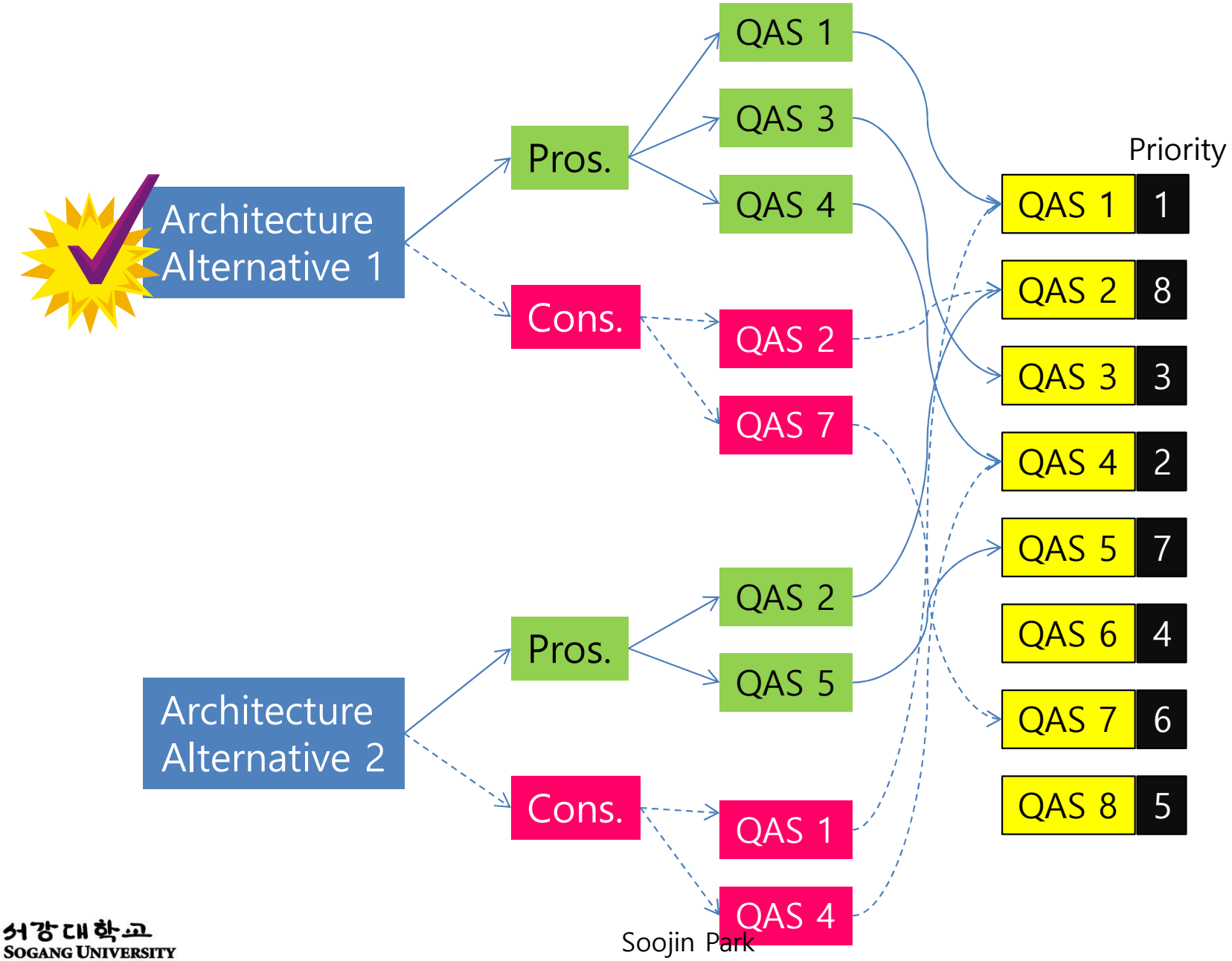
Revisited: Construction of a Quality Attribute Tree (1/2)



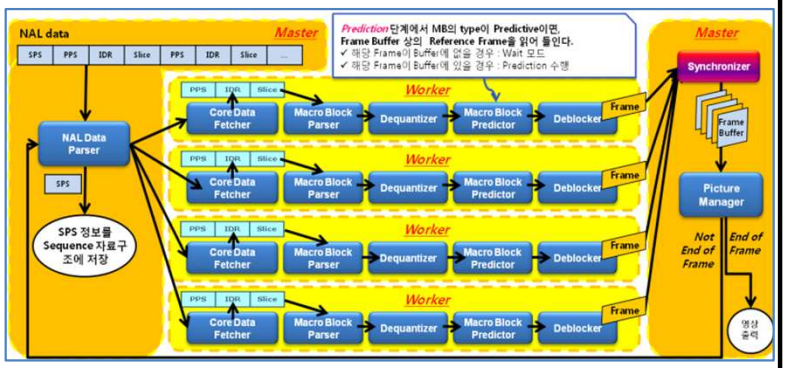
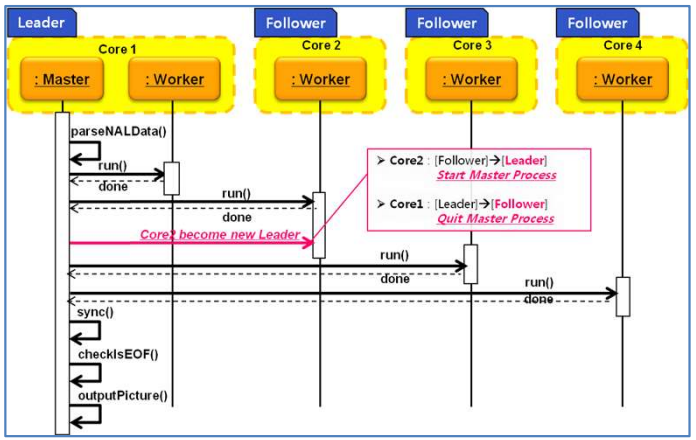
Revisited: Construction of a Quality Attribute Tree (1/2)



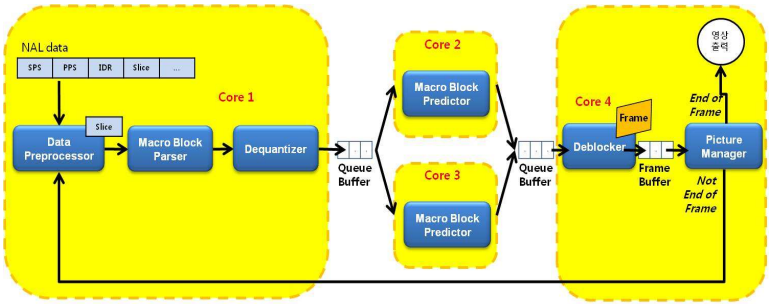
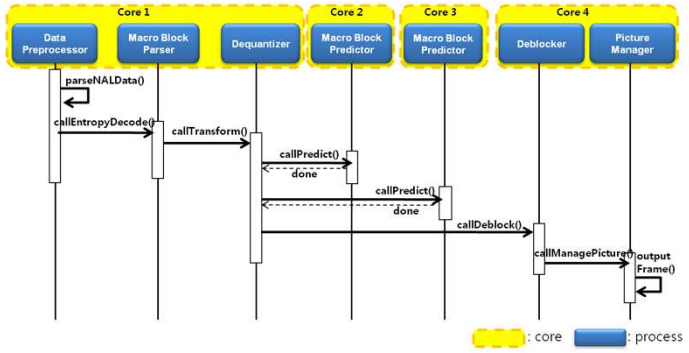
Analyze the Architectural Approaches



Selection of the Best by Trade-off Analysis(ATAM) : Qualitative Evaluation(1/3)

아키텍처 접근법 분석서(Analysis of Architecture Approach)	
Architectural Approach 1	***Dynamic Master & Worker***
Architecture Diagram	●Static View  ●Dynamic View 
Pros.	● 각 Core의 Idle Time을 최소화한 구조 (++QAS1) ● 4개의 독립된 Core에 Decoding Thread를 각각 하나씩 할당함으로써, 싱글 코어와 비교하여 일정한 수준 이 상의 데이터 처리 속도 증가를 보장 (++QAS1, ++QAS3, +QAS4)
Cons.	●Dynamic한 Leader&Worker 운영으로 인해 모든 Core상에 거의 대부분의 소스코드가 탑재되어야 하므로, Memory Size가 커짐(- - QAS6) ●각 Core에서 처리된 Frame을 출력 Buffer에 쓰기 전에 별도의 Synchronization 작업이 필요함 (- QAS1) ●Master와 Worker 프로세스간의 Data 전송량이 적어도 4Frame 이상 발생함(--QAS6)

Selection of the Best by Trade-off Analysis(ATAM): Qualitative Evaluation (2/3)

<table>
<tr><th colspan="3">아키텍처 접근법 분석서(Analysis of Architecture Approach)</th></tr>
<tr><td>Architectural Approach 2</td><td colspan="2">Balanced Pipe & Filter</td></tr>
<tr><td>Architecture Diagram</td><td>●Static View
 
</td><td>● Dynamic View
 
</td></tr>
<tr><td>Pros.</td><td colspan="2">● Load balancing 지원: 과부하가 예상되는 특정 process를 두 개의 core가 담당하도록 할당하여 concurrency를 향상시킴 (++QAS7)
●Pipeline 을 따라 흐르는 데이터 전송량은 특정 시점에서 1Frame에 해당하는 량으로 Core간 Data 전송량 감소 효과, Queue Buffer Size 감소효과, 각 Core 상에 싱글코어에서 동작하던 Filter 들이 부분적으로 산재되어 있으므로, 하나의 Core에 탑재되는 소스코드 Size 감소 (++QAS6)</td></tr>
<tr><td>Cons.</td><td colspan="2">●특정 Core의 Computation Time에 전체 시스템의 Execution Time이 의존적일 수 있으며, 특정 Core에서 과부하가 발생할 경우 싱글코어와 비교하여 데이터 처리 속도향상을 기대할 수 없음 (--QAS1)
●Core간의 Communication 횟수가 Approach1에 비해 많음(-QAS4)</td></tr>
</table>

Selection of the Best by Trade-off Analysis(ATAM): Qualitative Evaluation (3/3)

아키텍처 접근서 분석서(Analysis of Architecture Approach)

Conclusion

Approach 1과 Approach2의 장단점과 품질 속성 시나리오, 그리고 각 품질속성에 부여된 개발 우선순위와의 관계를 도식화하면 아래와 같다.

Quality Attribute Scenario No.	Architecture Approach1	Architecture Approach2	Priority
QAS1	+++	--	1
QAS3	++	/	5
QAS4	+	-	2
QAS6	----	++	6
QAS7	/	++	7

1. Approach1은 각 Core의 Idle Time을 최소화한 구조로, QAS1, QAS3, QAS4를 지원하는 아키텍처이다. 반면 우선순위 6이 부여된 QAS6에 대해서는 품질을 상당히 저하시키는 경향이 있다.

2. Approach2는 우선순위 6이 부여된 가지는 QAS6은 비교적 잘 지원하고 있으나 상대적으로 높은 우선순위를 가지는 QAS1과 AQS4의 품질은 감소시키는 접근법이다.

따라서, 두 가지 아키텍처 접근방법 중에서 우리는 Approach 1안을 채택하기로 한다.

Quantitative Evaluation??

- No specific guideline is provided
- Prototyping
  - A tool for quantitative evaluation of the constructed architecture

Quality Attribute - Availability

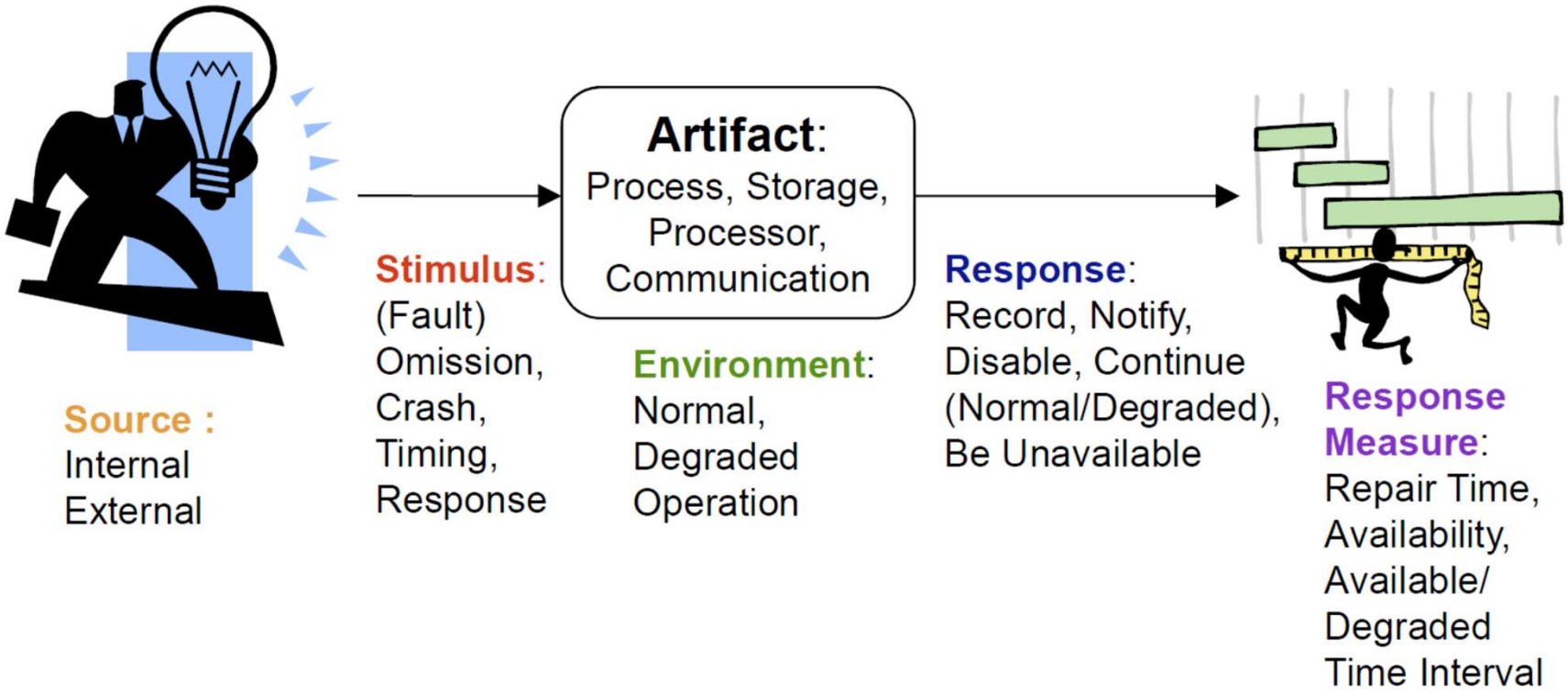
- Ability of a system to mask or repair faults such that the cumulative service outage period does not exceed a required value over a specified time interval

- The availability of a system can be calculated as the probability that it will provide the specified services within required bounds over a specified time interval.

- When referring to hardware, there is a well-known expression used to derive steady-state availability:
  - MTBF : the mean time between failures
  - MTTR : the mean time to repair

QAS Example for Availability

- "An **unanticipated external message** is received by a process during **normal operation**. **The process informs the operator of the receipt of the message and continues to operated with no downtime.**"



Quality Attribute Scenario - Availability

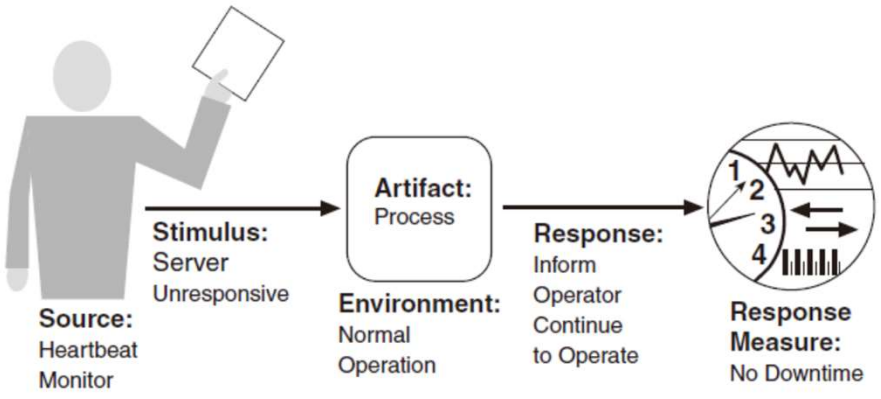


FIGURE 5.3 Sample concrete availability scenario

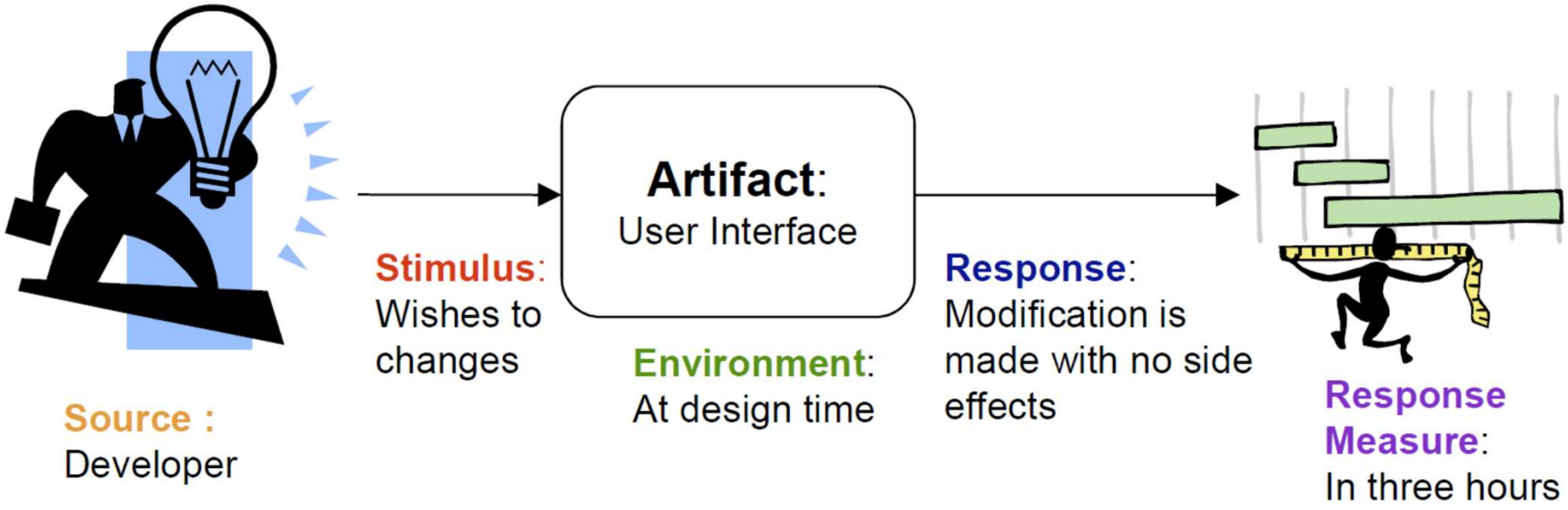
Portion of Scenario	Possible Values
Source	Internal/external: people, hardware, software, physical infrastructure, physical environment
Stimulus	Fault: omission, crash, incorrect timing, incorrect response
Artifact	Processors, communication channels, persistent storage, processes
Environment	Normal operation, startup, shutdown, repair mode, degraded operation, overloaded operation
Response	Prevent the fault from becoming a failure Detect the fault: ▪ Log the fault ▪ Notify appropriate entities (people or systems) Recover from the fault: ▪ Disable source of events causing the fault ▪ Be temporarily unavailable while repair is being effected ▪ Fix or mask the fault/failure or contain the damage it causes ▪ Operate in a degraded mode while repair is being effected
Response Measure	Time or time interval when the system must be available Availability percentage (e.g., 99.999%) Time to detect the fault Time to repair the fault Time or time interval in which system can be in degraded mode Proportion (e.g., 99%) or rate (e.g., up to 100 per second) of a certain class of faults that the system prevents, or handles without failing

Quality Attribute - Modifiability

- The ability to quickly make changes to a system at a higher performance-to-price ratio
  - Often based on some specific changes and is measured by examining the costs of these changes.
    - What can change?
    - What is the likelihood of the change?
    - When is the change made and who makes it?
    - What is the cost of the change?

QAS Example for Modifiability (Adaptability)

- "A developer wishes to change the user interface to make a screen's background color blue. This change will be made to the code at design time. It will take less than three hours to make and test the change and no side effect changes will occur in the behavior."



Example: 4 Core System에서 향후 그 이상의 개수로 Core 숫자가 늘어날 경우, 4 Core System 기반의 Architecture 상에서 변경되는 Component와 Connector의 비율은 Parallel Node의 숫자에 해당되는 Instance 개수 증가를 제외하고 50% 미만이어야 한다.

Quality Attribute Scenario - Modifiability

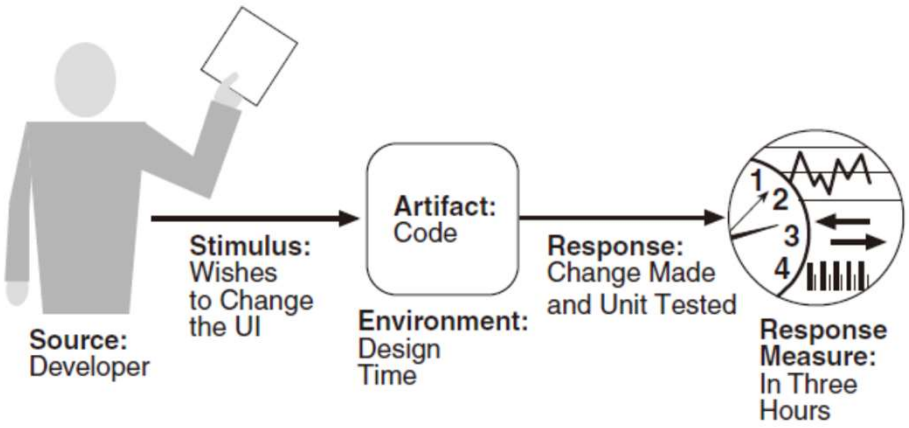


FIGURE 7.1 Sample concrete modifiability scenario

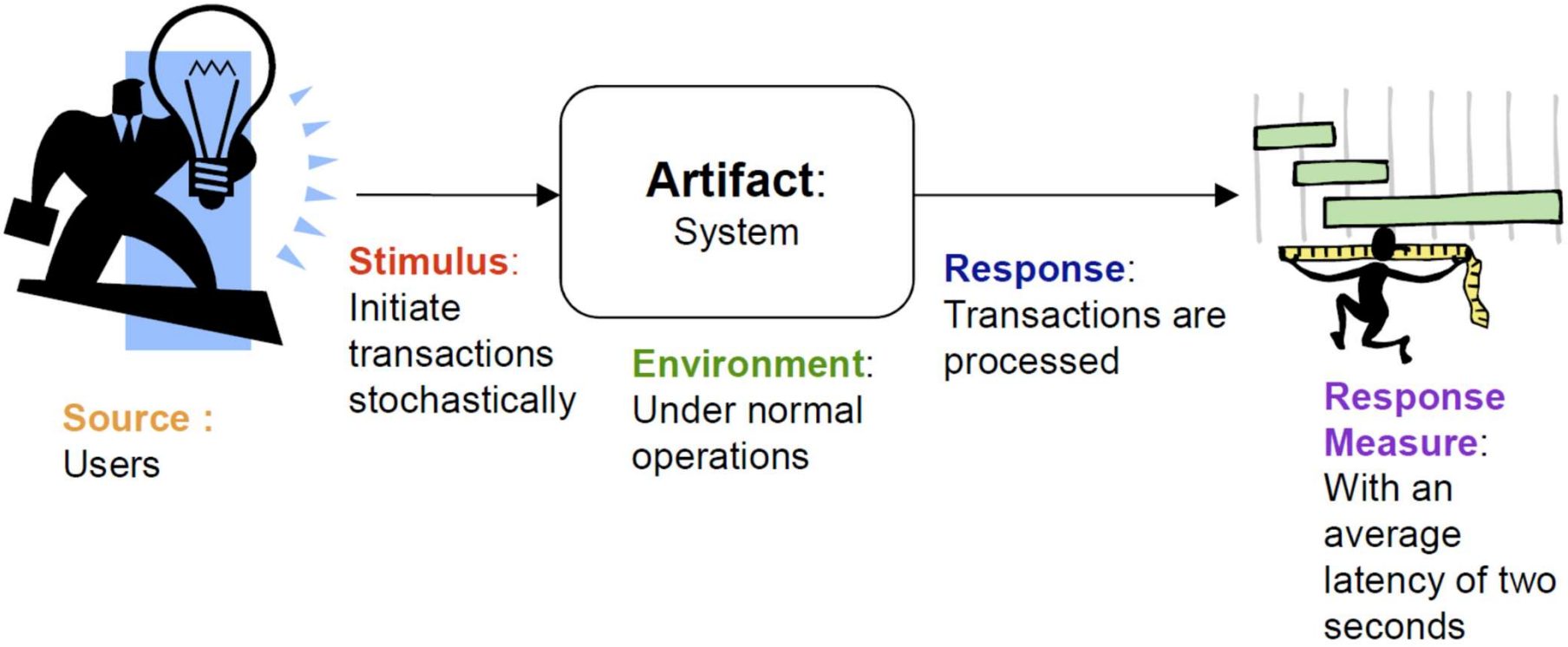
Portion of Scenario	Possible Values
Source	End user, developer, system administrator
Stimulus	A directive to add/delete/modify functionality, or change a quality attribute, capacity, or technology
Artifacts	Code, data, interfaces, components, resources, configurations, . . .
Environment	Runtime, compile time, build time, initiation time, design time
Response	One or more of the following: ▪ Make modification ▪ Test modification ▪ Deploy modification
Response Measure	Cost in terms of the following: ▪ Number, size, complexity of affected artifacts ▪ Effort ▪ Calendar time ▪ Money (direct outlay or opportunity cost) ▪ Extent to which this modification affects other functions or quality attributes ▪ New defects introduced

Quality Attribute - Performance

- About time and the software system's ability to meet timing requirements
  - When events occur, the system or some element of the system must respond to them in time.
    - interrupts, messages, requests from users or other systems, or clock events marking the passage of time
  - Characterizing the events that can occur (and when they can occur) and the system or element's time-based response to those events is the essence is discussing performance.

QAS Example for Performance

- "Users initiate 1,000 transactions per minute stochastically under normal operations, and these transactions are processed with an average latency of two seconds."



Quality Attribute Scenario - Performance

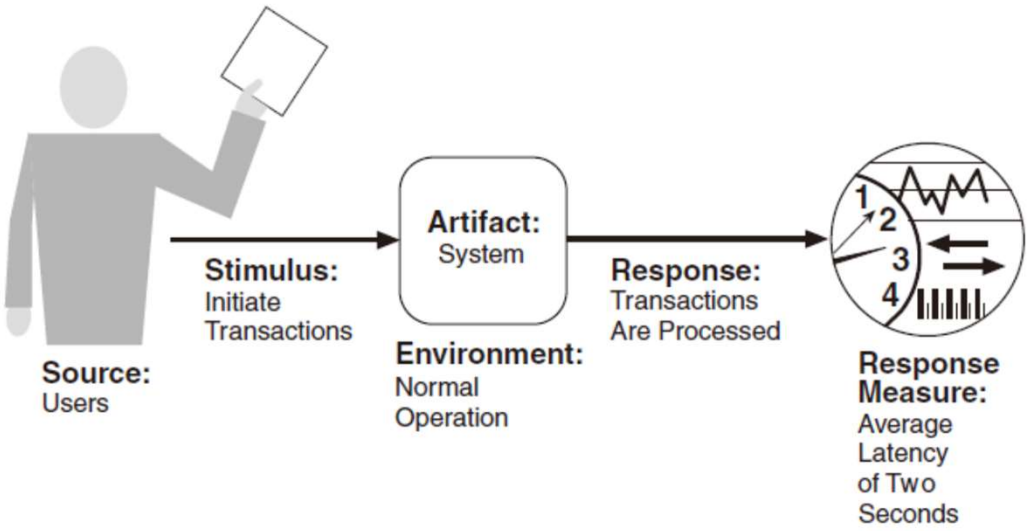


FIGURE 8.1 Sample concrete performance scenario

Portion of Scenario	Possible Values
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	Process events, change level of service
Response Measure	Latency, deadline, throughput, jitter, miss rate

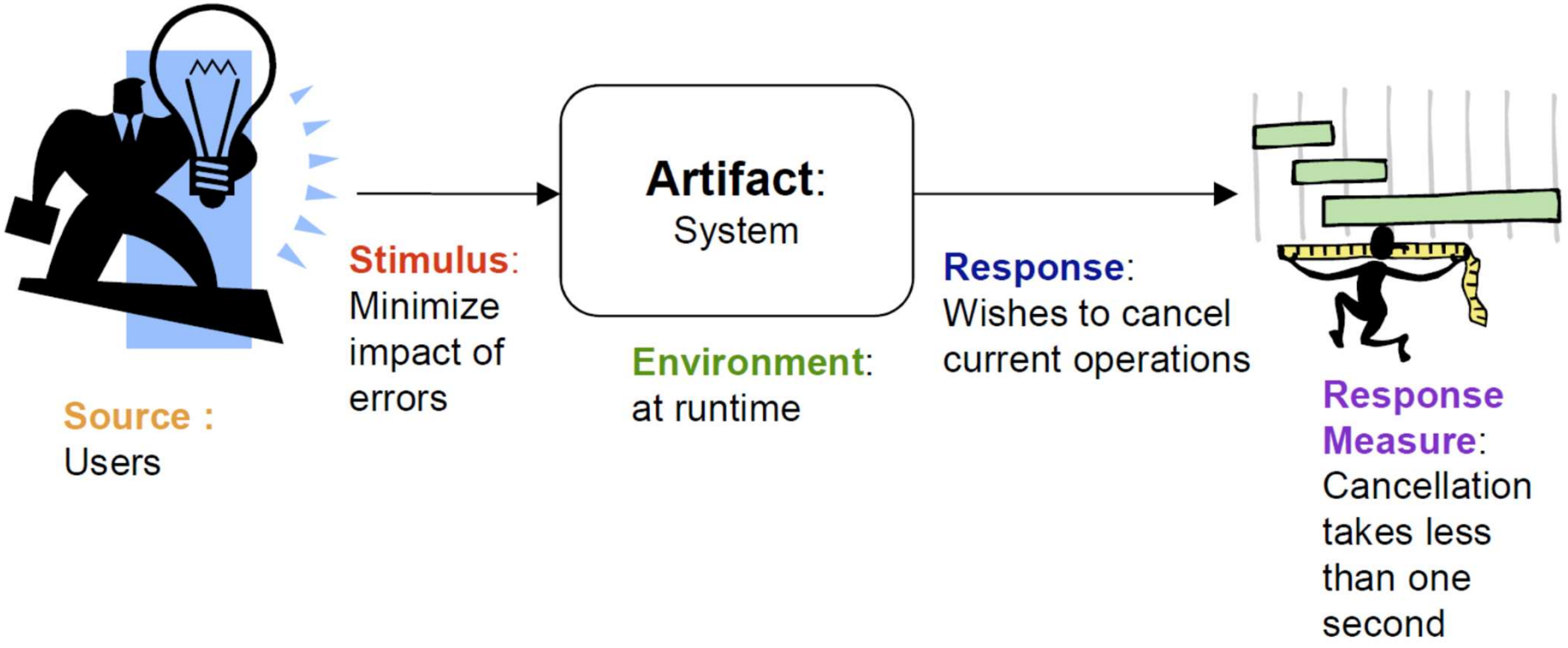
Quality Attribute - Usability

- Concerned with how easy it is for the user to accomplish a desired task and the kind of user support the system provides

- Usability comprises the following areas:
  - Learning system features.
  - Using a system efficiently.
  - Minimizing the impact of errors.
  - Adapting the system to user needs.
  - Increasing confidence and satisfaction.

QAS Example for Usability

- "A user wanting to minimize the impact of an error, wishes to cancel a system operation at runtime; cancellation takes place in less than one second."



Quality Attribute Scenario - Usability

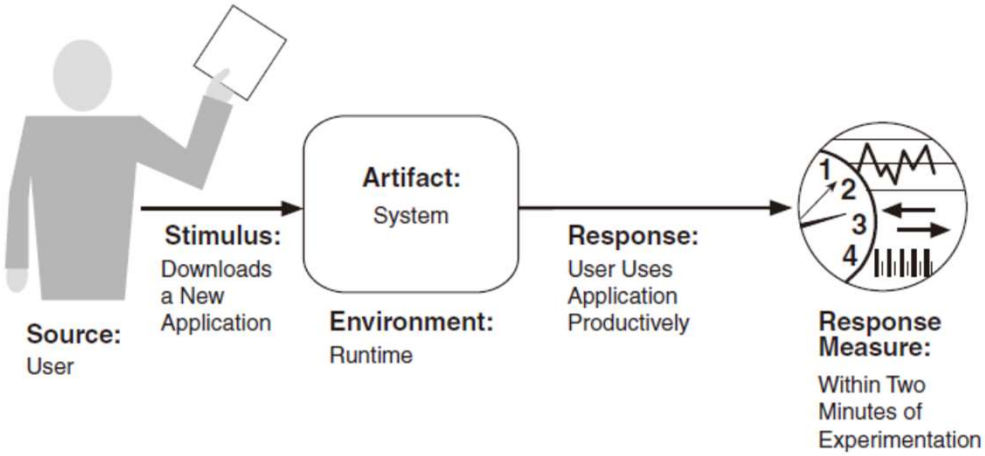


FIGURE 11.1 Sample concrete usability scenario

Portion of Scenario	Possible Values
Source	End user, possibly in a specialized role
Stimulus	End user tries to use a system efficiently, learn to use the system, minimize the impact of errors, adapt the system, or configure the system.
Environment	Runtime or configuration time
Artifacts	System or the specific portion of the system with which the user is interacting
Response	The system should either provide the user with the features needed or anticipate the user's needs.
Response Measure	One or more of the following: task time, number of errors, number of tasks accomplished, user satisfaction, gain of user knowledge, ratio of successful operations to total operations, or amount of time or data lost when an error occurs

Quality Attribute - Interoperability

- The degree to which two or more systems can usefully exchange meaningful information via interfaces in a particular context

- The definition includes
  - Syntactic interoperability: The ability to exchange data
  - Semantic interoperability: The ability to correctly interpret the data being exchanged

Quality Attribute Scenario - Interoperability

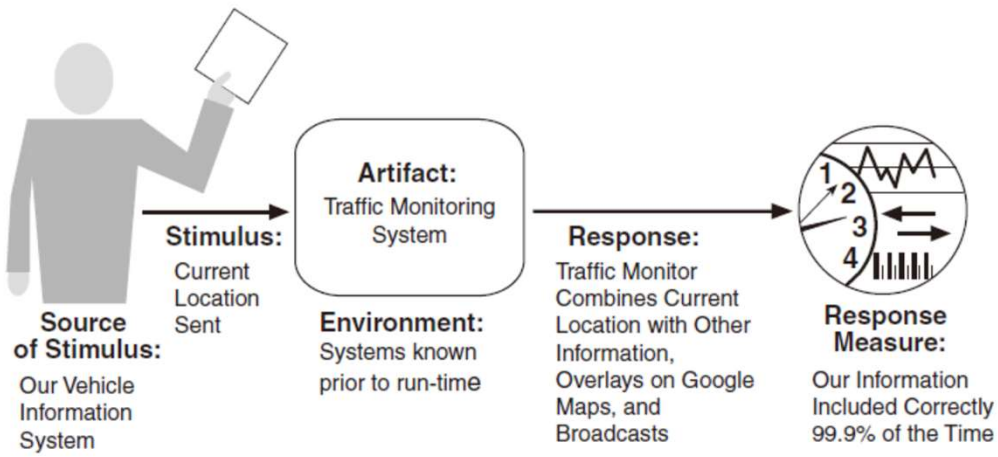


FIGURE 6.1 Sample concrete interoperability scenario

Portion of Scenario	Possible Values
Source	A system initiates a request to interoperate with another system.
Stimulus	A request to exchange information among system(s).
Artifact	The systems that wish to interoperate.
Environment	System(s) wishing to interoperate are discovered at runtime or known prior to runtime.
Response	One or more of the following: ▪ The request is (appropriately) rejected and appropriate entities (people or systems) are notified. ▪ The request is (appropriately) accepted and information is exchanged successfully. ▪ The request is logged by one or more of the involved systems.
Response Measure	One or more of the following: ▪ Percentage of information exchanges correctly processed ▪ Percentage of information exchanges correctly rejected

Quality Attribute - Security

- A measure of the system's ability to protect data and information from unauthorized access while still providing access to people and systems that are authorized

- The simplest approach to characterizing security has 3 characteristics **(CIA)**:
  - **Confidentiality:** 기밀보장
  - **Integrity:** 데이터무결성 보장
  - **Availability:** 승인된 사용자에게는 언제나 가용한 상태여야 함

- Other characteristics that are used to support CIA are these:
  - Authentication(인증)
  - Non-repudiation(부인방지): 특정데이터에 해당하는 기존 서명을 생성할 수 있었던 signature key pair의 소유자가 데이터에 서명한 것이라는 사실을 부인할 수 없음을 보증
  - Authorization(허가)

Quality Attribute Scenario - Security

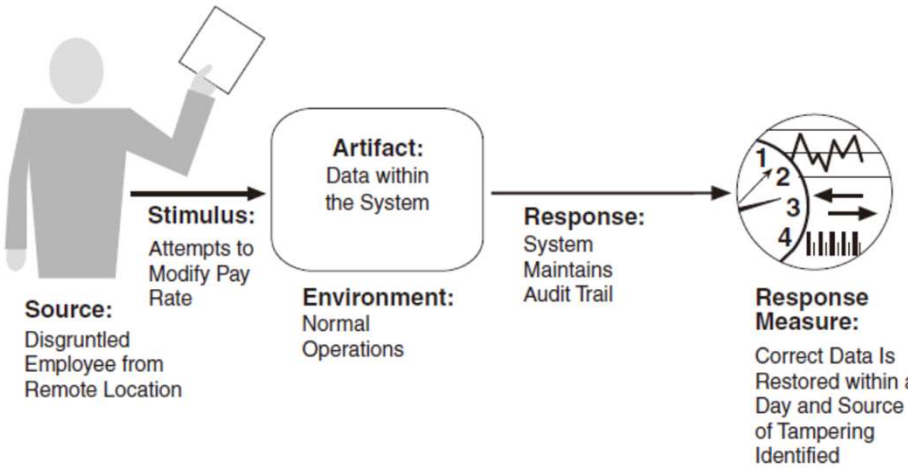


FIGURE 9.1 Sample concrete security scenario

Portion of Scenario	Possible Values
Source	Human or another system which may have been previously identified (either correctly or incorrectly) or may be currently unknown. A human attacker may be from outside the organization or from inside the organization.
Stimulus	Unauthorized attempt is made to display data, change or delete data, access system services, change the system's behavior, or reduce availability.
Artifact	System services, data within the system, a component or resources of the system, data produced or consumed by the system
Environment	The system is either online or offline; either connected to or disconnected from a network; either behind a firewall or open to a network; fully operational, partially operational, or not operational.
Response	Transactions are carried out in a fashion such that ▪ Data or services are protected from unauthorized access. ▪ Data or services are not being manipulated without authorization. ▪ Parties to a transaction are identified with assurance. ▪ The parties to the transaction cannot repudiate their involvements. ▪ The data, resources, and system services will be available for legitimate use. The system tracks activities within it by ▪ Recording access or modification ▪ Recording attempts to access data, resources, or services ▪ Notifying appropriate entities (people or systems) when an apparent attack is occurring
Response Measure	One or more of the following: ▪ How much of a system is compromised when a particular component or data value is compromised ▪ How much time passed before an attack was detected ▪ How many attacks were resisted ▪ How long does it take to recover from a successful attack ▪ How much data is vulnerable to a particular attack

Quality Attribute - Testability

- The ease with which software can be made to demonstrate its faults through (typically execution-based) testing
  - Specifically, testability refers to the probability, assuming that the software has at least one fault, that it will fail on its next test execution.
  - Intuitively, a system is testable if it “gives up” its faults easily.

Quality Attribute Scenario - Testability

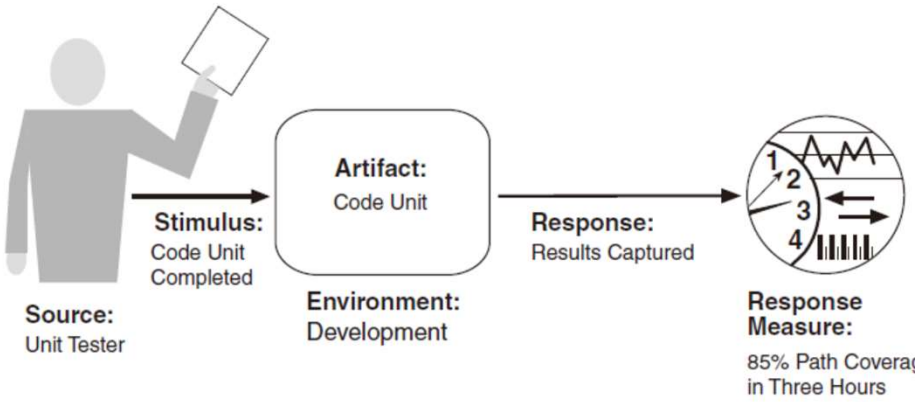


FIGURE 10. 2 Sample concrete testability scenario

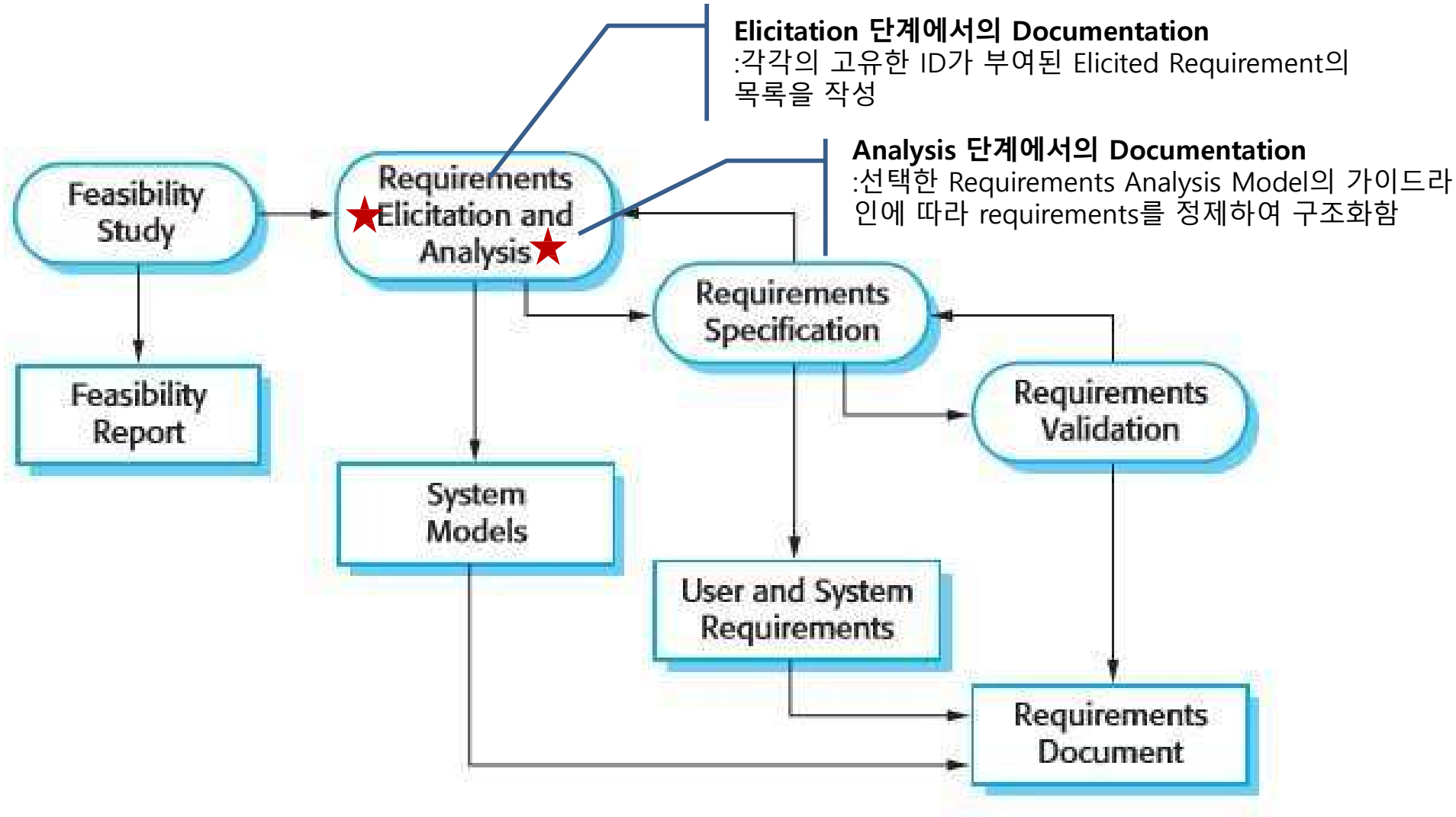
Portion of Scenario	Possible Values
Source	Unit testers, integration testers, system testers, acceptance testers, end users, either running tests manually or using automated testing tools
Stimulus	A set of tests is executed due to the completion of a coding increment such as a class layer or service, the completed integration of a subsystem, the complete implementation of the whole system, or the delivery of the system to the customer.
Environment	Design time, development time, compile time, integration time, deployment time, run time
Artifacts	The portion of the system being tested
Response	One or more of the following: execute test suite and capture results, capture activity that resulted in the fault, control and monitor the state of the system
Response Measure	One or more of the following: effort to find a fault or class of faults, effort to achieve a given percentage of state space coverage, probability of fault being revealed by the next test, time to perform tests, effort to detect faults, length of longest dependency chain in test, length of time to prepare test environment, reduction in risk exposure (size(loss) × prob(loss))

Other Quality Attributes

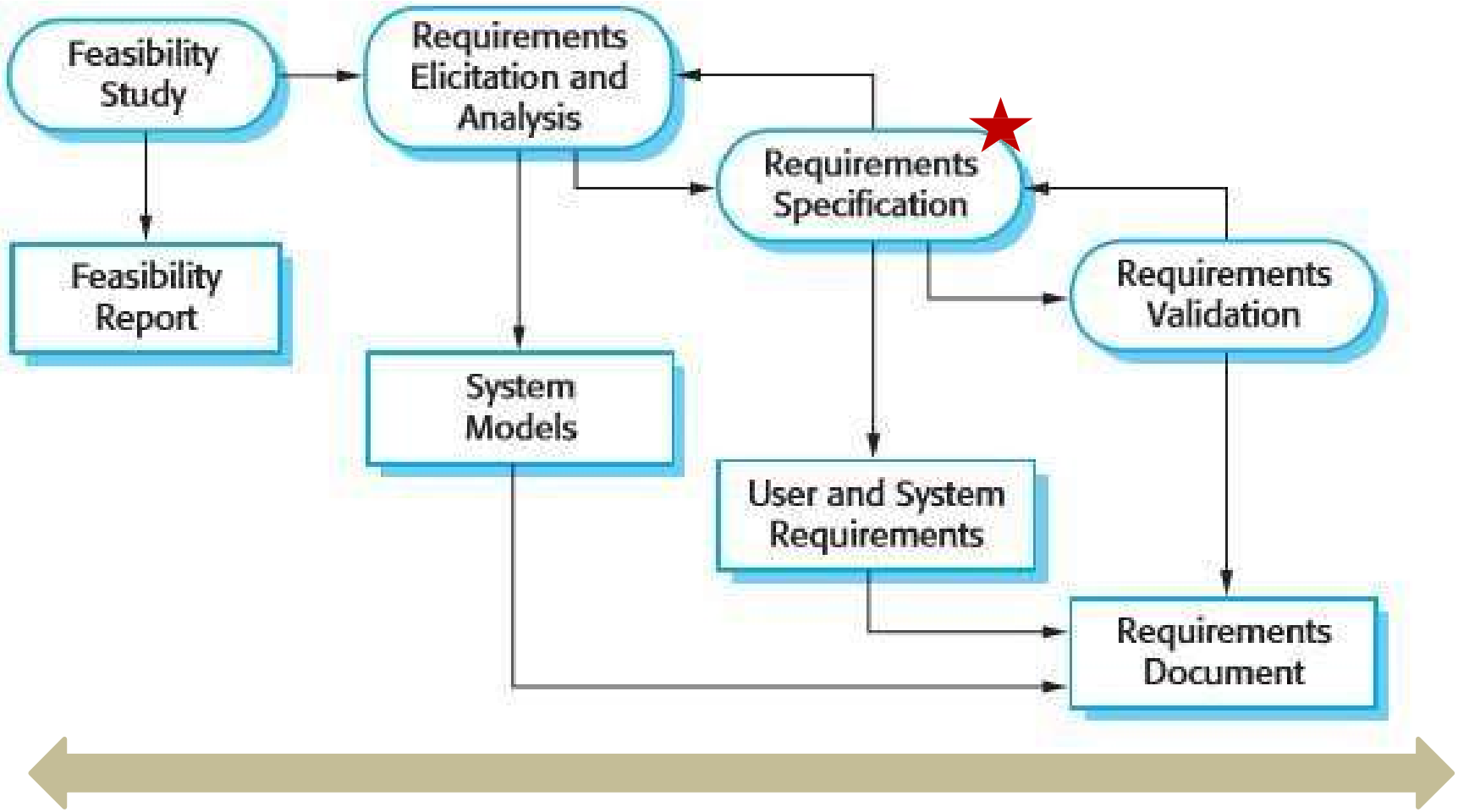
- **Other important quality attributes**
  - Variability
  - Portability
  - Development Distributability
  - Scalability
  - Deployability
  - Mobility
  - Monitorability
  - Safety

8. Requirements Specification

Different Documentation Activities in Requirements Engineering Process



Requirements Engineering Process



Software Requirements Specification(SRS)

- The software requirements document is the **official statement** of **what is required of the system developers**.
  - Should include both a definition of user requirements and a specification of the system requirements
  - NOT a design document.
  - As far as possible, it should set of WHAT the system should do rather than HOW it should do it.

- The goal of requirements engineering:
  - "***Not to write the perfect requirements specification, but create the best possible product at the right time***"

Purposes of SRS

- How do we communicate the Requirements to others?
  - It is common practice to capture them in an SRS
    - But, an SRS doesn't need to be a single paper document

- Purpose
  - Communication
    - Explains the application domain and the system to be developed
  - Contractual
    - May be legally binding!
    - Expresses agreement and a commitment
  - Baseline for evaluating the software
    - Supporting testing, V&V
    - "Enough information to verify whether delivered system meets requirements"
  - Baseline for change control

Features for Good Specifications

Features	Considerations
Valid (Correct)	- Expresses the real needs of the stakeholders (customers, users,…) - Does not contain anything that is not “required”
Unambiguous	- Every statement can be read in exactly one way
Complete	- All the things the system must do and all the things it must not do! - Conceptual Completeness • E.g., responses to all classes of input - Structural Completeness • E.g., no TBDs!!!
Understandable (Clear)	- E.g. by non-computer specialists
Consistent	- Doesn’t contradict itself - Uses all terms consistently
Ranked	- Indicates relative importance / stability of each requirement
Verifiable	- A process exists to test satisfaction of each requirement
Modifiable	- Can be changed without difficulty • Good structure and cross-referencing
Traceable	- Origin of each requirement is clear - Labels each requirement for future referencing

SRS Contents

- **Software Requirements Specification** should address:
  - **Functionality**
    - What is the software supposed to do?
  - **External interfaces**
    - How does the software interact with people, the system's hardware, other hardware, and other software?
    - What assumptions can be made about these external entities?
  - **Required performance**
    - What is the speed, availability, response time, recovery time of various software functions, and so on?
  - **Quality attributes**
    - What are the portability, correctness, maintainability, security, and other considerations?
  - **Design constraints imposed on an implementation**
    - Are there any required standards in effect, implementation language, policies for database integrity, resource limits, operating environment(s) and so on?

SRS Should Not Include

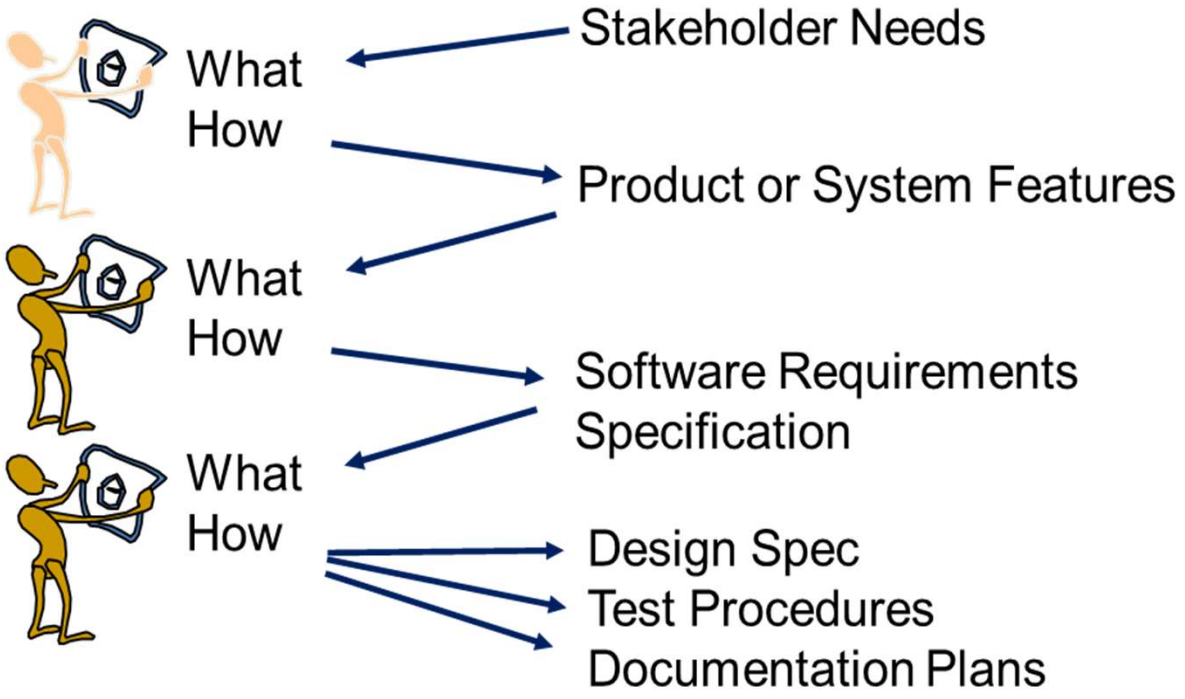
- **Designs**
  - Requirements and designs have different audiences
  - Analysis and design are different areas of expertise
  - Except where application domain constrains the design
    - E.g., limited communication between different subsystems for security reasons

Requirements and Design are Inseparable

- In principle,
  - **Requirements** should state what the system should do.
  - **Design** should describe how it does this.

- In practice, requirements and design are inseparable.

"One man's ceiling is another man's floor."



Davis, 1993

Requirements Document Commonality

- **Requirements documents standards** have been designed.
  - E.g., **IEEE standards**
  - Mostly applicable to the requirements for large systems engineering projects

Requirements Document Variability

- Natural languages
- Structured specifications
- Form-based specifications
- Tabular specifications

Natural Languages

- **Lack of clarity**
  - Precision is difficult without making the document difficult to read.
- **Requirements confusion**
  - Functional and non-functional requirements tend to be mixed-up.
- **Requirements amalgamation(**뭉뚱그려진 requirements 문장들**)**
  - Several different requirements may be expressed together.

3.2 The system shall measure the blood sugar and deliver insuli n, if required, every 10 minutes. *(Changes in blood sugar are rel atively slow so more frequent measurement is unnecessary; less frequent measurement could lead to unnecessarily high sugar l evels.)*

3.6 The system shall run a self-test routine every minute with t he conditions to be tested and the associated actions defined i n Table 1. *(A self-test routine can discover hardware and softwa re problems and alert the user to the fact the normal operation may be impossible.)*

Example : insulin pump software system

Structured Specifications

- An approach to writing requirements where
  - The freedom of the requirements writer is limited and
  - Requirements are written in a standard way.

- This works well for some types of requirements such as embedded control system.
  - But, sometimes too rigid for writing business system requirements

Form-based Specifications

- Specification includes information **in a form format** :
  - Definition of the function or entity
  - Description of inputs, source, outputs, and destination
  - Description of the action to be taken
  - Pre and post conditions (if appropriate)
  - The side effects (if any) of the function

- Requirements for the insulin pump software system

Insulin Pump/Control Software/SRS/3.3.2
Function Compute insulin dose: safe sugar level.
Description
Computes the dose of insulin to be delivered when the current measured sugar level is in the safe zone between 3 and 7 units.
Inputs Current sugar reading (r2); the previous two readings (r0 and r1).
Source Current sugar reading from sensor. Other readings from memory.
Outputs CompDose—the dose in insulin to be delivered.
Destination Main control loop.

Action
CompDose is zero if the sugar level is stable or falling or if the level is increasing but the rate of increase is decreasing. If the level is increasing and the rate of increase is increasing, then CompDose is computed by dividing the difference between the current sugar level and the previous level by 4 and rounding the result. If the result, is rounded to zero then CompDose is set to the minimum dose that can be delivered.
Requirements
Two previous readings so that the rate of change of sugar level can be computed.
Pre-condition
The insulin reservoir contains at least the maximum allowed single dose of insulin.
Post-condition r0 is replaced by r1 then r1 is replaced by r2.
Side-effects None.

Tabular Specification

- Particularly useful when you have to define a number of possible alternative courses of action
  - For example, the insulin pump systems bases its computations on the rate of change of blood sugar level, and the tabular specification explains how to calculate the insulin requirement for different scenarios.

- Requirements for the insulin pump software system

Condition	Action
Sugar level falling (r2 < r1)	CompDose = 0
Sugar level stable (r2 = r1)	CompDose = 0
Sugar level increasing and rate of increase decreasing ((r2 – r1) < (r1 – r0))	CompDose = 0
Sugar level increasing and rate of increase stable or increasing ((r2 – r1) ≥ (r1 – r0))	CompDose = round ((r2 – r1)/4) If rounded result = 0 then CompDose = MinimumDose

Variation exampled in Requirements Specification Style

- Business Application Domain: 최대한 자세히 줄글 형태로 풀어서 작성하는 스타일을 선호함
  - Example1: 은행 환율등록 시스템(특이사항: CBD방법 적용으로 인한 Variation 포함)
  - Example2: 렌터카 서비스_시스템 유스케이스 명세서(표 형태의 Template 사용)

- Embedded Application Domain: 표 형태의 template+개조식 기술방식을 선호함
  - Example3: DVD Player Controller(table style)
  - Example2: H.264 표준을 따르는 영상 데이터 디코딩(table style + suedo code에 가까운 depth)

Example1: 은행 환율등록 시스템(특이사항: CBD방법 적용으로 인한 Variation 포함)

1.1 개략적인 설명

본점 Teller 가 환율고시작업을 위하여 TELERATE 또는 REUTER 를 통해서 입수된 기준환율(USD)을 이용하여 각 통화별 대미환산율(CROSS 율)을 입력하는 환율등록작업을 수행한다. 이 외에도 등록완료작업, 등록완료해제작업을 수행하거나, 각종환율기본자료(Spread, 기간별 고시/비고시 Libor)등록작업 등의 외환업무를 수행할 수 있도록 한다.

2. 이벤트의 흐름

2.1 기본 흐름

본 유스케이스는 본점 Teller 가 환율등록(완료, 완료해제)및 기간별 고시/비고시 LIBOR 등록, Spread 등록작업을 하기 위해 해당되는 작업을 선택하면서 시작된다.

1. 환율등록선택

   본점 Teller 는 메뉴에서 환율등록을 선택한다.(A1)

   시스템은 환율등록 화면을 display 한다.

2. 취급점번 체크

   시스템은 취급점번을 체크해서 취급점번이 국제팀(1955), 외환업무팀(4033), 법인영업부(9724)가 아닌경우, E1 수행한다.

3. 기간별 고시/비고시 Libor 등록여부 확인

   시스템은 기간별 고시/비고시 Libor 등록여부를 확인하여 해당일자의 Libor 가 등록되지 않은 경우, A1 수행한다.

4. 선처리 작업을 위해 "선처리 " 유스케이스를 포함한다.

5. 업무개시일자의 익영업일 검색

   시스템은 선처리작업의 결과로 검색된 업무개시일자와 Current Date 를 비교한다.

   5.1 Current Date 가 업무개시일자보다 큰 경우(업무개시전):

   업무개시일자의 익영업일을 검색한다.

   등록일자 = 업무개시일자의 익영업일

   사용 Component : 캘린더

   input : 업무개시일자

   output : 익영업일자

   5.2 Current Date 와 업무개시일자가 동일한 경우(업무개시후):

Example2: 렌터카 서비스_시스템 유스케이스 명세서(표 형태의 Template 사용)

System Use Case Description – 차량 예약 신청서 작성(1)

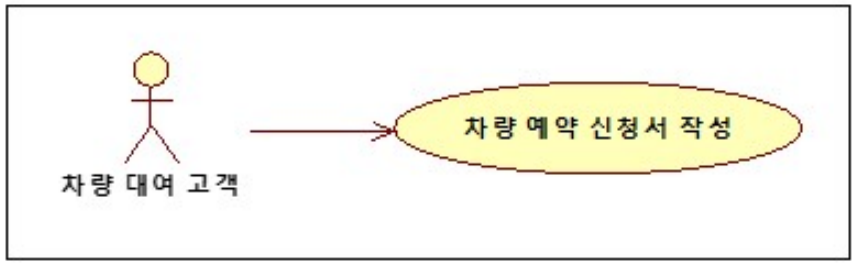
Name	차량 예약 신청서 작성
Brief Description	차량 대여 고객이 차량의 종류, 대여 기간, 보험 종류, 기타 옵션 사항들을 선택한다.
Relationship	차량 대여 고객
Precondition	차량 대여 고객은 본인 인증을 완료한 상태여야 한다.
Postcondition	예약 신청서서가 시스템에 저장되어 있어야 한다.
Special Requirement	차량 대여 고객이 예약 신청서 작성을 중단하고 싶은 경우 언제든지 신청서 작성을 취소 할 수 있다. 차량 대여 고객이 예약 신청서 작성을 중간에 저장하고 싶은 경우 언제든지 저장할 수 있다.

Basic Flow

본 유스케이스는 차량 대여 고객이 차량을 예약하고 싶은 경우 시작된다..

1. 차량 대여 지점 선택
   - 1.1 차량 대여 고객이 차량을 대여 하고 싶은 지역을 선택한다.
   - 1.2 시스템은 해당 지역의 지점을 차량 대여 고객에게 출력한다.
   - 1.3 차량 대여 고객은 원하는 지점을 선택 한다. (A1)
2. 차량 사용 기간 선택
   - 2.1 시스템은 차량 대여 고객에게 대여 기간을 선택할 수 있는 화면을 출력한다.
   - 2.2 차량 대여 고객은 대여 시작일과 종료일을 선택한다.
3. 차량 종류 선택
   - 3.1 시스템은 예약 가능한 차량을 출력한다.
   - 3.2 차량 대여 고객은 예약하고 싶은 차량을 선택한다. (A2)
4. 차량 보험 선택
   - 4.1 시스템은 보험 선택 화면을 출력한다.
   - 4.2 차량 대여 고객은 보험을 선택한다.
5. 부가 장치 선택
   - 5.1 시스템은 차량 대여 고객이 선택할 수 있는 부가 장치를 출력한다.
   - 5.2 차량 대여 고객은 원하는 부가 장치를 선택한다.
6. 예약 사항 확인
   - 6.1 차량 대여 고객이 예약 신청서 작성 완료를 선택한다.
   - 6.2 시스템은 차량 대여 고객이 선택한 사항을 요약하여 화면에 출력한다.
   - 6.3 차량 대여 고객이 신청한 내용에 차이가 없는지 확인한다. (A3)

System Use Case Description – 차량 예약 신청서 작성(2)

Alternate Flow	
A1. 선택할 지점이 없는 경우 분기점 : Basic Flow 1.3 1. 차량 대여 고객이 다른 지역을 선택하고자 하는 경우 Basic Flow 1로 돌아간다. A2. 선택할 수 있는 차량이 없는 경우 분기점 : Basic Flow 3.2 1. 차량 대여 고객이 다른 지역을 선택하고자 하는 경우 Basic Flow 1로 돌아간다. A3. 신청한 내용과 차이가 있는 경우 분기점 : Basic Flow 6.3 1. 수정하고자 하는 해당 Basic Flow로 돌아간다.	
Use-Case Diagram	

Example3: DVD Player Controller(table style)

Feature Group ID	SETUP.AUDIO		
그룹설명	Audio 출력에 관한 설정을 처리하는 module		
고려사항			
세부사양			
Requirement ID	SRS_FEAT_4		시스템 적용여부
Feature ID	SETUP.AUDIO.DIGITAL-OUTPUT		
설명	Digital Audio 출력 방식을 설정한다.		
Pre-condition	1.Stop 상태여야 한다.		
Post-condition	Disc play 시 설정된 방식의 Digital Audio 가 출력된다.		
개발 우선순위	A		
기본동작	BF1	Triggering Event	
	BF1.1	사용자에 의해서 Menu/Setup/Audio/Digital Output 항목을 선택한다.	Y
	BF2	사용자가 원하는 출력 방식을 방향키를 이용해서 선택한다.	Y
	BF3	다음 중 한 Step 을 수행한다.	
	BF3.1	사용자가 PCM 을 선택하고 Enter 를 누를 경우 [AF1]	Y
	BF3.2	사용자가 Bitstream 을 선택하고 Enter 를 누를 경우 [AF2]	Y
	BF4	Digital output 설정 값을 EEPROM 에 저장한다.	Y
대체동작	AF1	사용자가 PCM 을 선택하고 Enter 를 누를 경우 Digital output 설정 값을 PCM 으로 변경한다.	Y
	AF2	사용자가 Bitstream 을 선택하고 Enter 를 누를 누를 경우 Digital output 설정 값을 Bitstream 으로 변경한다.	Y
예외동작			
품질요구사항	QR1	변경 후 재생 시 Noise 가 발생되지 말아야 한다.	Y

Example2: [H.264 표준을 따르는 영상 데이터 디코딩(table style + suedo code에 가까운 depth)]

Requirement ID	H.264 표준을 따르는 영상 데이터 디코딩		Parallelism
설명	H.264 규약에 따라 동작하는 Video Decoder 가 입력되는 Video Data 의 압축을 풀어 화면상에 출력하기까지 시스템의 동작을 기술한다.		
Pre-condition	입력 Video Data 가 DRAM 상에 load 되어 있는 상태이다.		
Post-condition	입력된 Video Data 가 출력 Buffer 상에 모두 출력된 상태이다.		
개발우선순위	1		
기본동작	BF1	비디오 디코더에게 디코딩 메시지가 도착한다.	N
	BF2	**Input Data Preprocessing 을 시작한다.**	N
	BF2.1	Macro Block(MB) 정보를 저장하는 자료구조 버퍼를 초기화한다.	N
	BF2.2	NAL 데이터들을 입력 받아, FileBuffer[]에 저장한다.	N
	BF2.3	FileBuffer[]에서 한 개의 NAL 데이터를 작업 Buffer 로 복사한다.	N
	BF2.4	해당 NAL 데이터의 Header type 값을 parsing 한다. 만약에 입력된 NAL 데이터 자체에 문제가 있거나, 해당 데이터의 Header Type 이 유효하지 않으면, **EF1 로 분기한다.**	N
	BF2.4.1	[NAL Header Type == PPS]	N
	BF2.4.1.1	Picture 정보를 picture_parmeter_set 버퍼에 저장한다.	N
	BF2.4.2	[NAL Header Type == IDR]	N
	BF2.4.2.1	IDR Picture 정보를 해당 버퍼에 저장한다.	N
	BF2.4.3	[NAL Header Type == PPS]	N
	BF2.4.3.1	sequence 정보를 seq_parmeter_set 버퍼에 저장한다.	N
	BF2.4.4	[NAL Header Type == Slice]	N
	BF2.4.4.1	Slice 를 구성하는 MB 에 대한 영상 복원을 시작한다.	N
	BF3	**Entropy Decoding 을 시작한다.**	**FP_01**
	BF3.1	처리하고자 하는 MB 의 정보를 SPS, PPS, IDR 버퍼에서 읽어온다.	DP_01
	BF3.2	MB 의 picture type 을 parsing 한다.	DP_01
	BF3.2.1	[MB type == Intra-Picture]	DP_01
	BF3.2.1.1	해당 MB 정보를 ipredmode[] 버퍼에 복사한다.	DP_01

사용자 시나리오 기술 예제

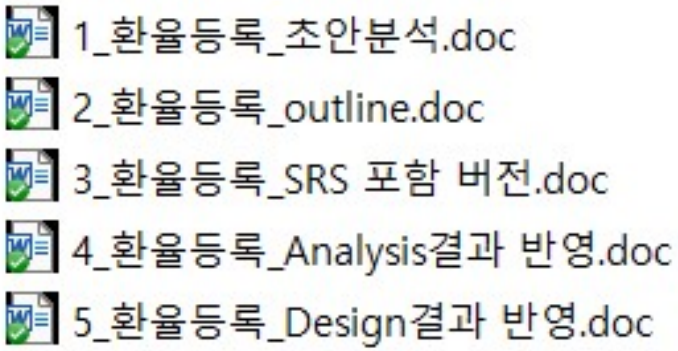
- **장면 1: 포도밭 센서 네트워크 상태보기**

  사무실에 들어온 재배자는 본인의 사무용 PC를 켜 인터넷 브라우저를 연다. 포도밭(포도밭)에 설치된 장비들이 정상적으로 동작하는지를 보기 위해, 재배자는 URL을 입력하고 나타난 로그인 화면에 아이디와 비밀번호를 입력하여 접속한다. 화면엔 재배자의 포도밭이 위성사진(또는 맵)으로 표시된다. 표시된 포도밭 위에는 설치되어 있는 측정 센서장비와 제어장치들이 아이콘으로 표현되어 나타나자, 재배자는 각각의 센서장비들 간에 연결이 정상적인지를 확인한다. 화면에서 각 센서들의 아이콘 위에 마우스를 가져다 대면, 센서에서 측정된 값들이 요약되어 실시간으로 보여진다. 화면에서 토양센서 모양의 센서장비 옆에는 측정된 토양수분과 온도가 표시되고, 온/습도 모양의 센서에서는 온도와 습도가 표시된다. 대기모양의 아이콘을 갖는 대기 센서에서는 풍향, 풍속 및 일조량이 표시된다.

  화면상단에는 재배자가 관리하는 다른 포도밭들도 나열되어 있다. 재배자는 화면을 클릭하여 나타난 포도밭 리스트에서 'GOOD'를 선택한다. 화면은 GOOD 포도밭으로 이동하고 마찬가지로 포도밭 위의 설치된 센서들이 나타난다.

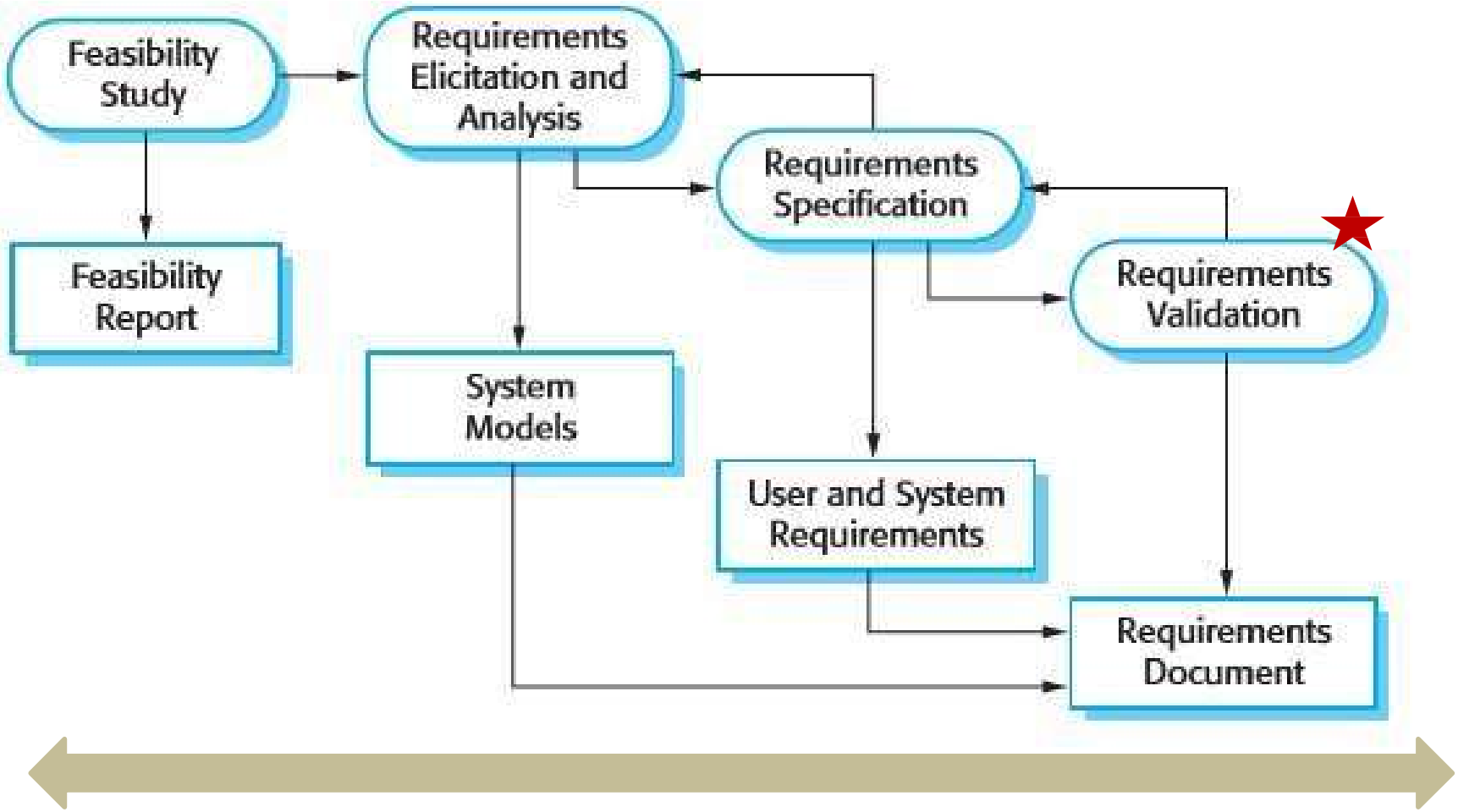
Different Versions of SRS for a System

- SRS 1.0 버전은 Requirements Analysis 단계 종료 시점에 완성되지만,
- 1.0 버전 완성 이후에도 Software Lifecycle Development Cycle인 각 단계들을 거치면서 지속적으로 update → System의 current feature를 반영하는 문서로서 역할을 하기 위함



9. Requirements Validation

Requirements Engineering Process



Requirements Validation

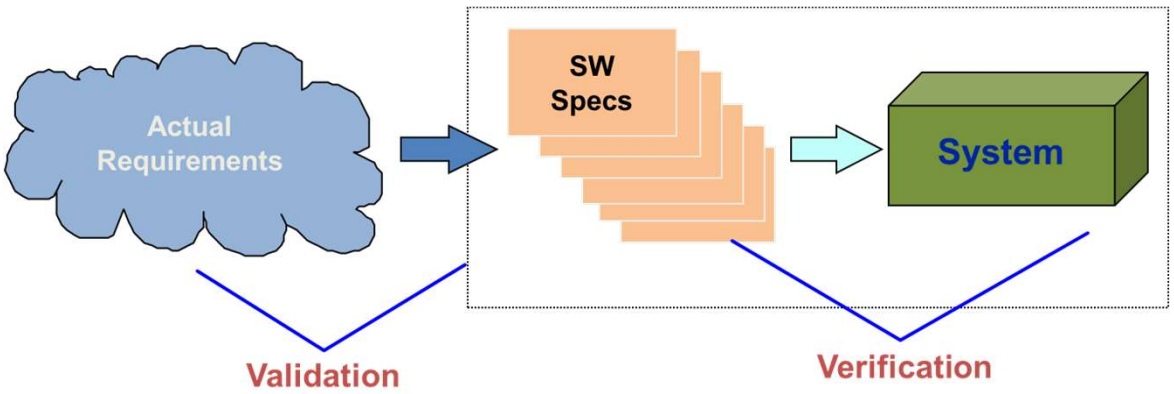
- Demonstrate whether the requirements we defined are **what the customer really wants**

- **Requirements validation checks:**
  - Validity: *Does the system provide the functions which support the customer's needs well?*
  - Correctness: *Are these the right requirements to achieve goal X*
  - Consistency: *Are there any requirements conflicts?*
  - Completeness: *Are all functions required by the customer included?*
  - Feasibility: *Can the requirements be implemented with available budget and technology?*
  - Verifiability: *Can the requirements be checked?*

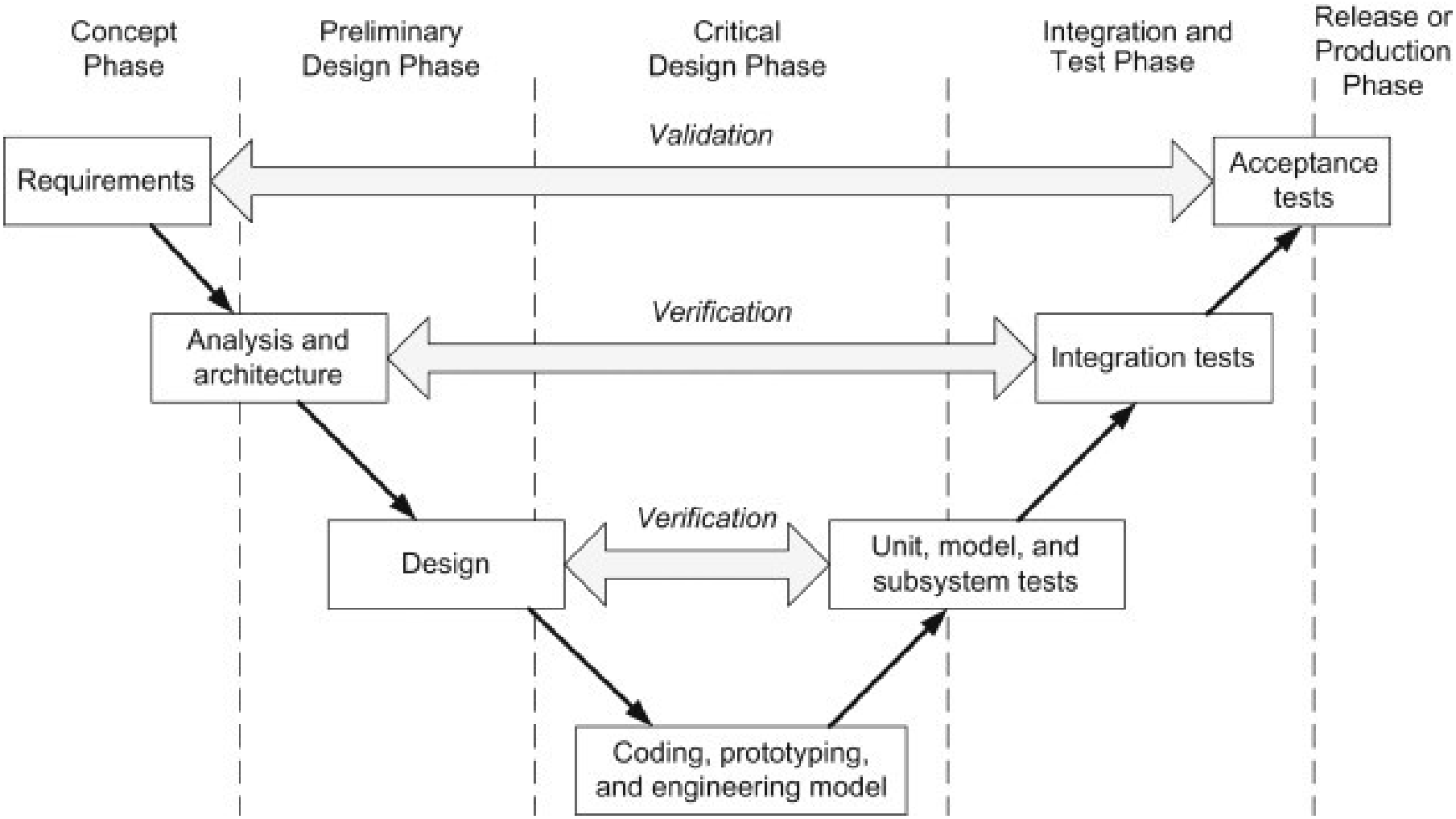
Verification and Validation in SDLC

- **Validation**: "Does the software system **meets the user's real needs**?"
  - Are we building the right software?
  - Does our design meet the spec?
  - Does our implementation meet the spec?
  - Does the delivered system do what we said it would do?
  - Are our requirements models consistent with one another?

- **Verification**: "Does the software system **meets the requirements specifications**?"
  - Are we building the software right?
  - Does our problem statement accurately capture the real problem?
  - Did we account for the needs of all the stakeholders?



V-Model of V&V Activities in SDLC



Requirements Validation Techniques

- Requirements reviews
  - Systematic manual analysis of the requirements.
- Prototyping
  - Using an executable model of the system to check requirements.
- Test-case generation
  - Developing tests for requirements to check testability.

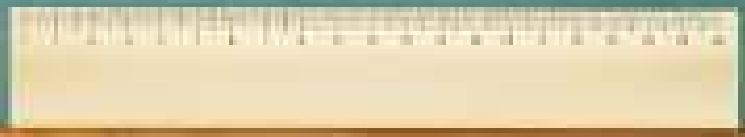
Requirements Reviews

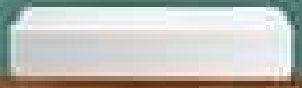
- Regular reviews should be held while the requirements definition is being formulated.
- Both client and contractor staff should be involved in reviews.
- Reviews may be formal (with completed documents) or informal. Good communications between developers, customers and users can resolve problems at an early stage.

Requirements Reviews

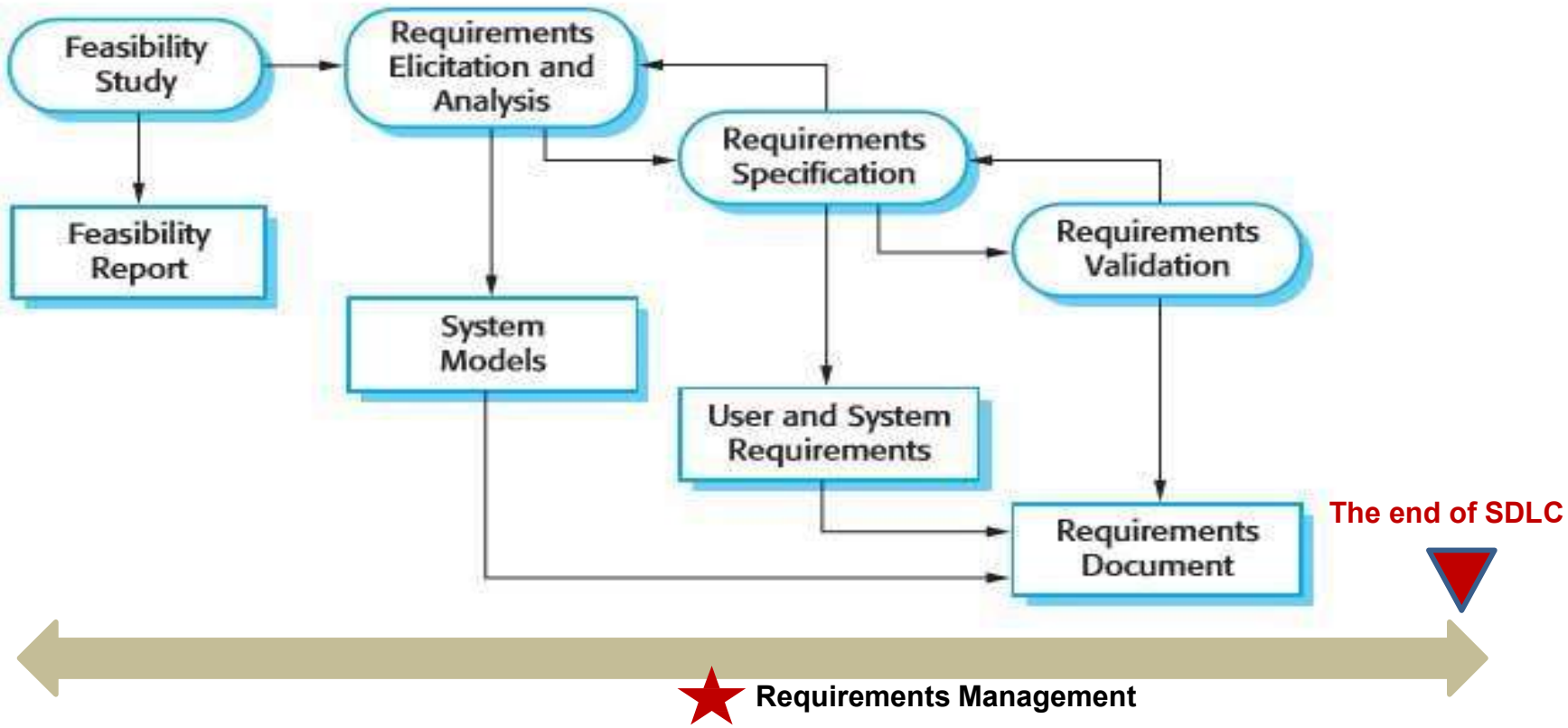
	Walkthrough	Inspection	Formal Review
목 적	• 초기에 에러 발견 • 기술, 표준, 정보로부터 이탈 발견	• 다른 조직과 관점 공유 • 승인, 수정, 기각 결정	• 결과의 완전성과 일치성을 보장
참여자	• Project Members	• Moderator, Recorder, Author and Inspectors	• PM, Process owners, Analysts
절 차	• 결과 개요 제공 • 제시된 문서 및 다른 참여자 의견 검토 • 발견된 에러 문서화	• Moderator : meeting 진행 • Recorders : 모든 issue 기록 • Inspectors : 에러 발견, 논의	• 문서 상태 검토 • 프로젝트의 결과 검토 • 다음 단계 진행 인증

10. Requirements Management





Requirements Engineering Process



Requirements Change Management

- The process of **managing requirements change** during the RE process and the overall system development
  - Requirements are inevitably incomplete and inconsistent.
  - New requirements emerge during the process, as business needs change and a better understanding of the system is developed.

- **Traceability** is the heart of requirements management.
  - Source ↔ Requirements ↔ Design ↔ Code

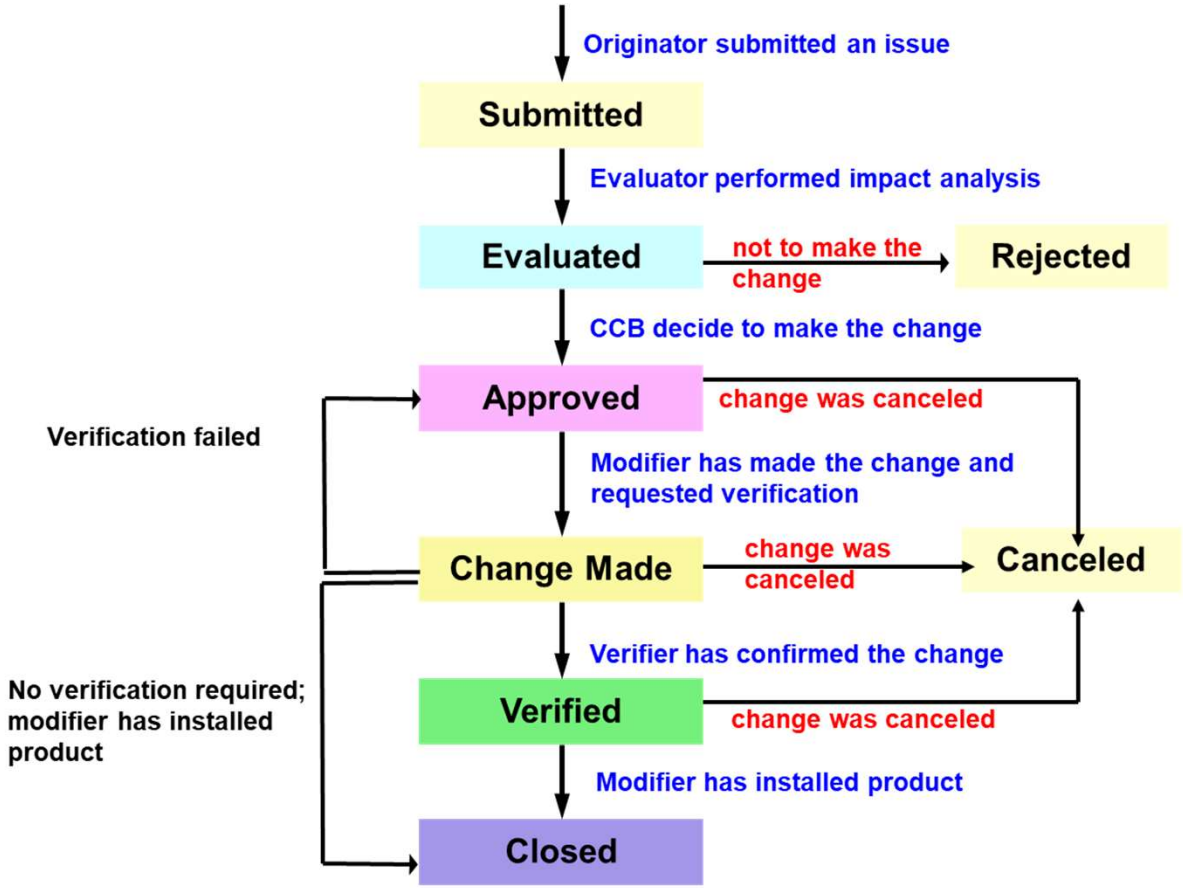
Change Management

- **Configuration Management**
  - Each distinct product is a Configuration Item (CI)
  - Each configuration item is placed under version control
  - Control which version of each CI belongs to which build of the system

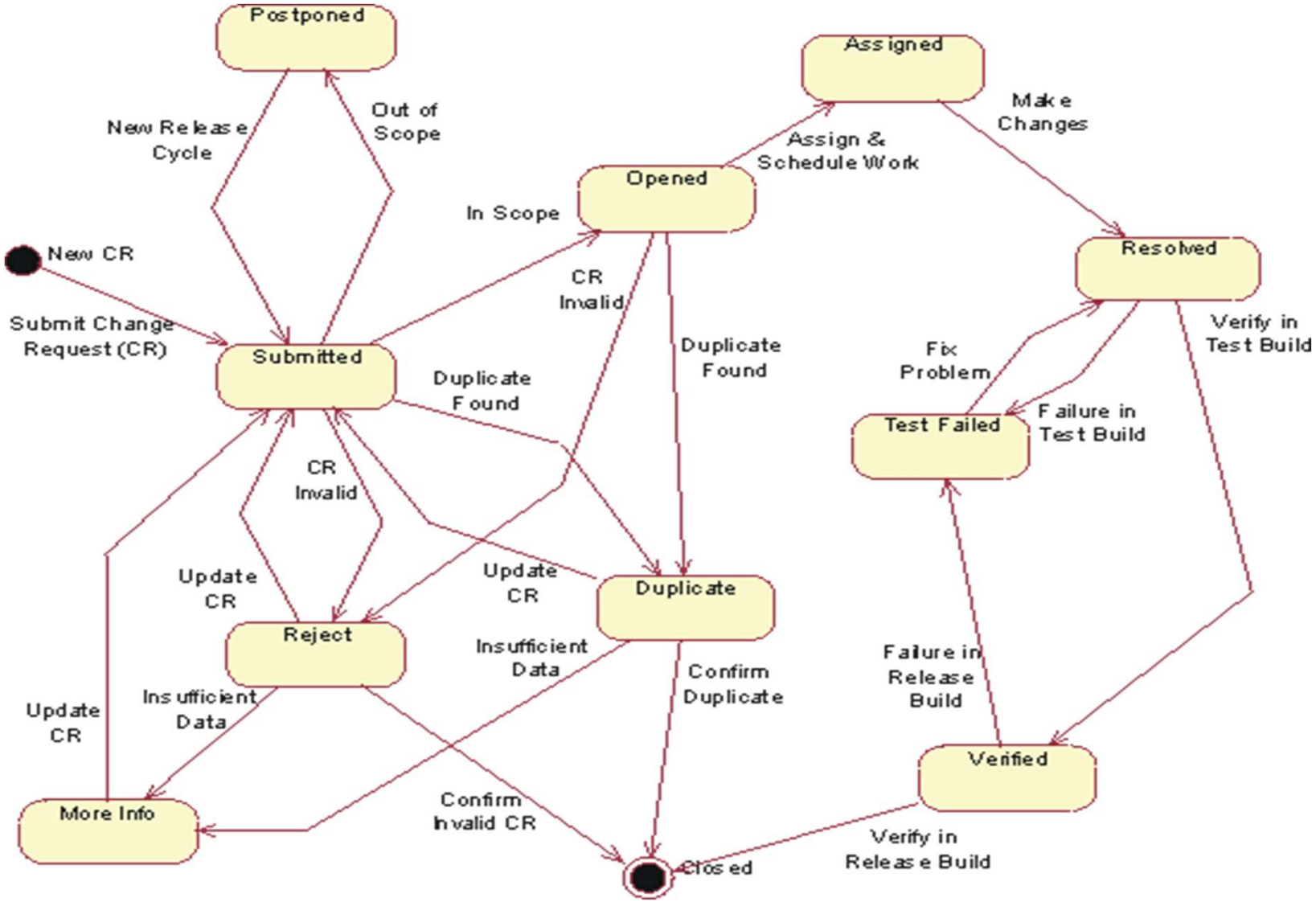
- **Baseline**
  - A stable version of a document or system
    - Safe to share among the team
  - Formal approval process for changes should be incorporated into the next baseline

Change Management Process

- **Change Management Process**
  - All proposed changes are submitted formally as change requests
  - A review board reviews these periodically and decides which to accept

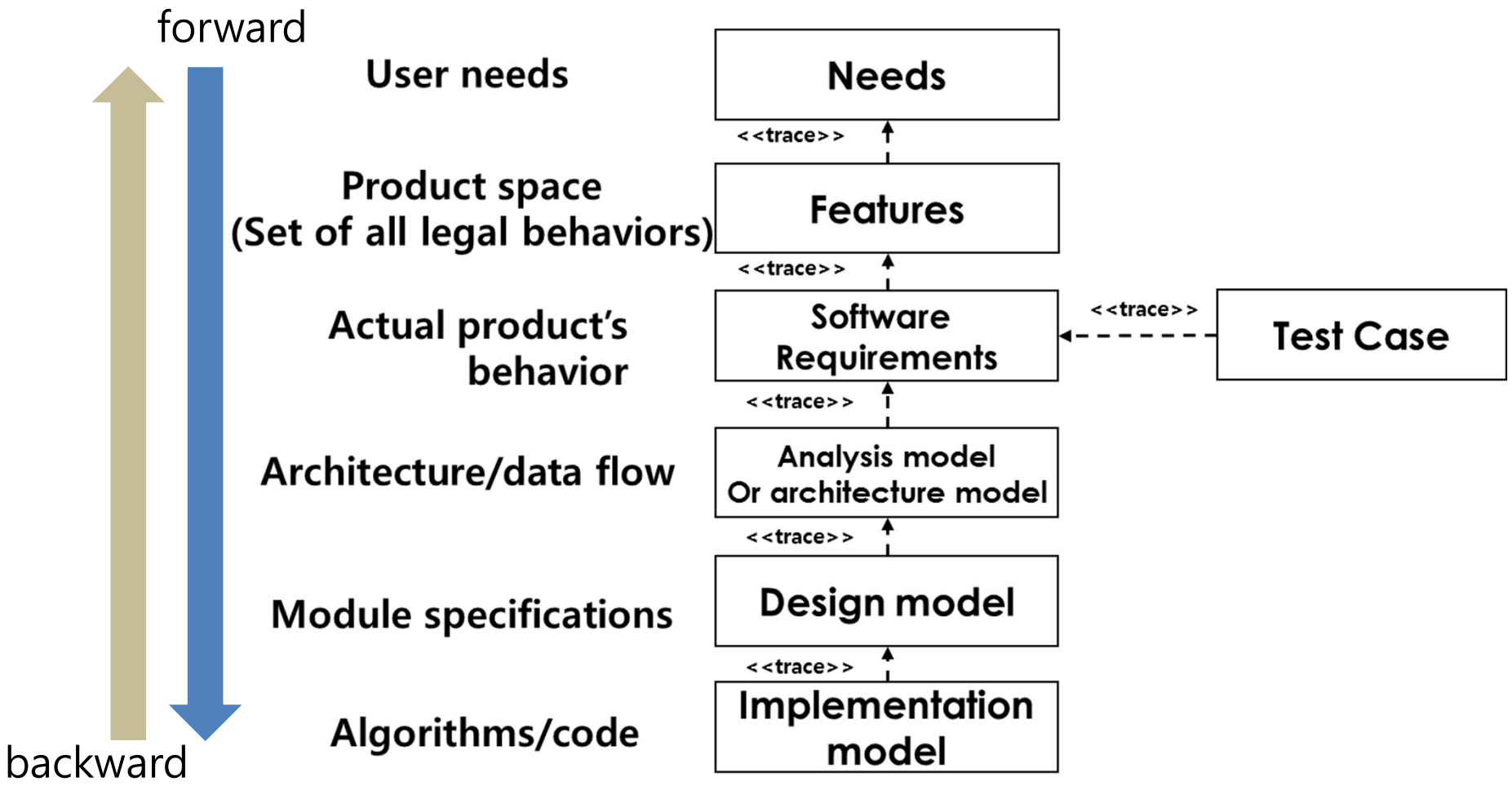


Example: Change Request Management State Descriptions



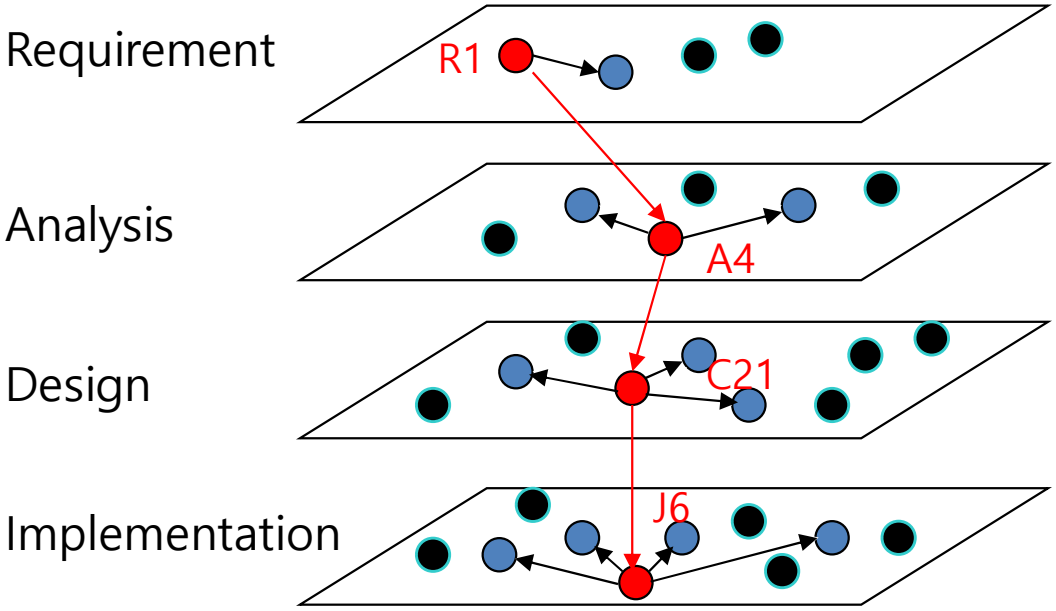
Requirements Traceability

- When a high level artifact derives a refined artifact, **Traceability link** should be generated between two artifacts.

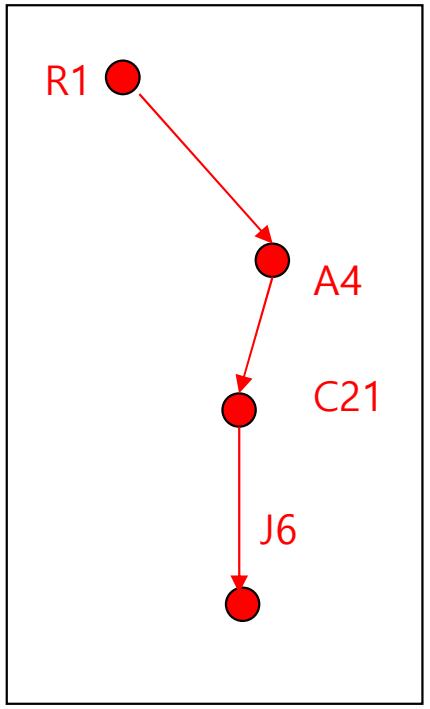


Traceability Policy

- Horizontal Traceability Link & Vertical Traceability Link

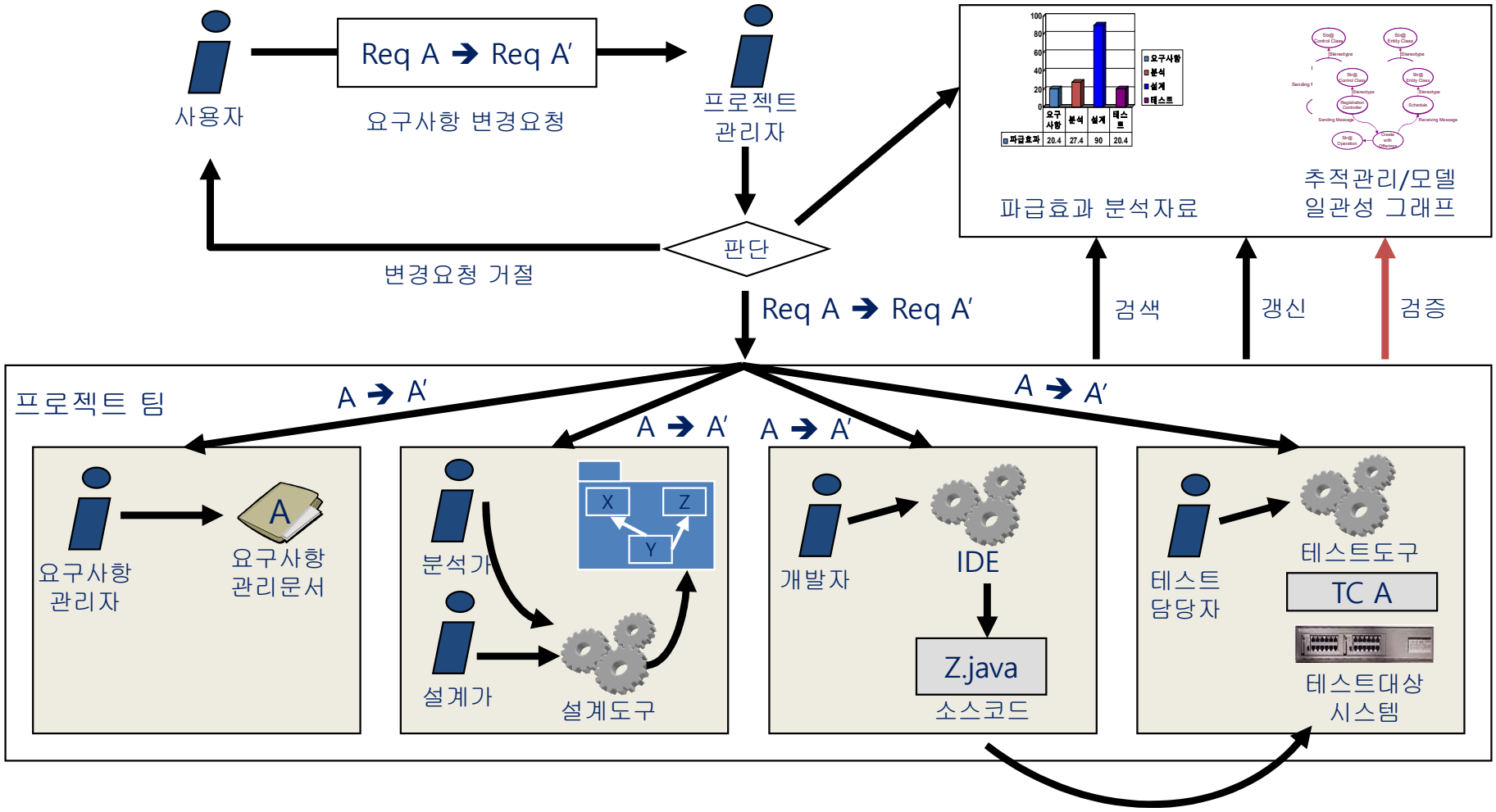


(a) A Trace Model with Vertical Traceability Link and Horizontal Traceability Link



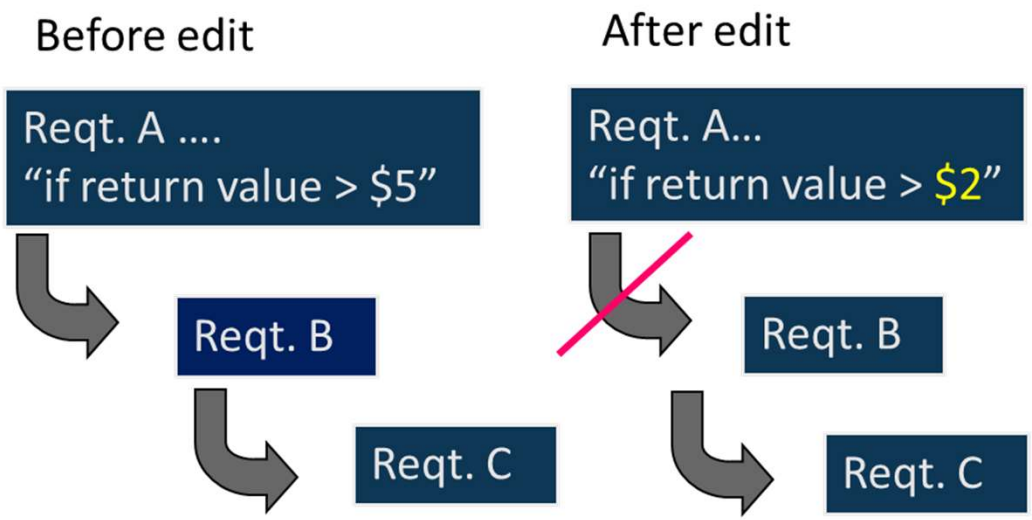
(b) Focusing Vertical Traceability

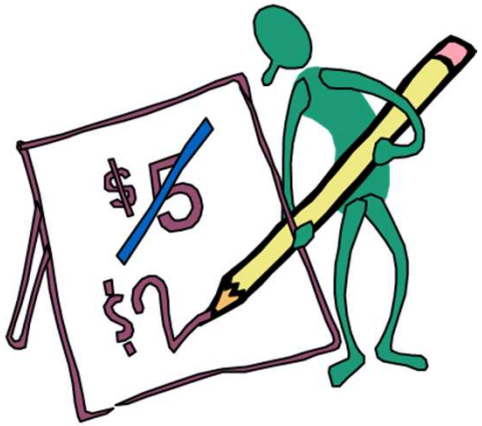
Change Impact Analysis



Change Impact Analysis Using Traceability Links in Automated Tools

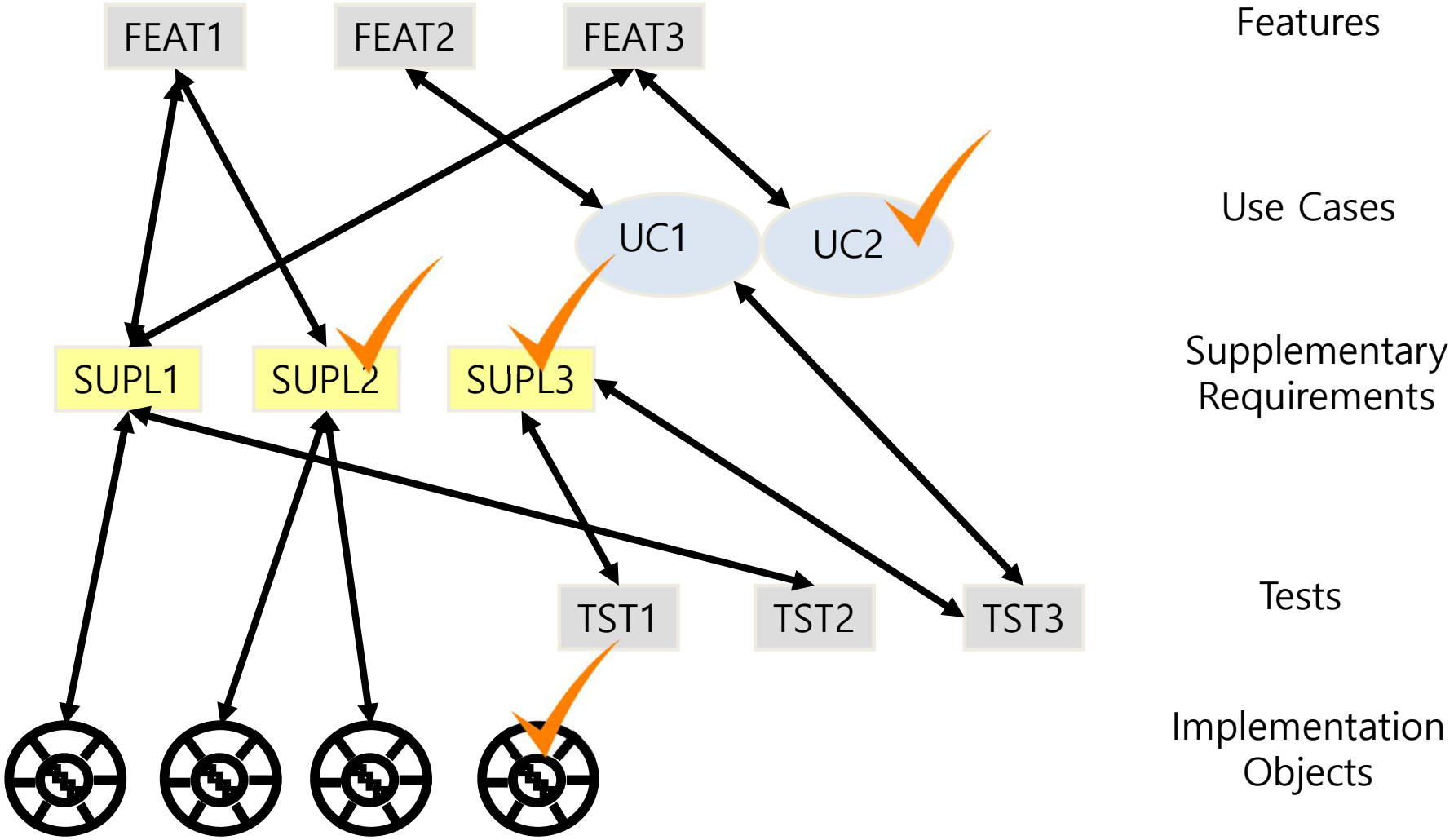
- Links to requirements that changed are marked “suspect”
- Suspect links must be resolved by the user





Requirements Traceability Verification

- What Is Wrong with This requirements traceability model?

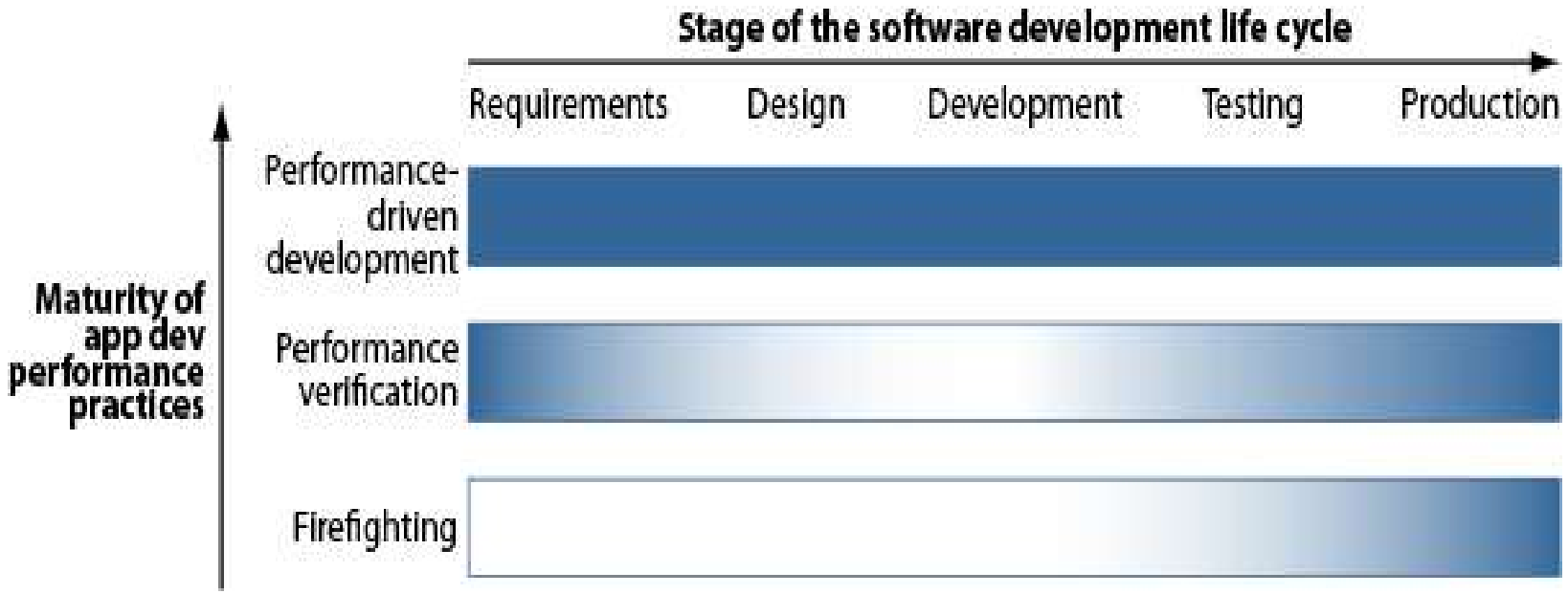


Traceability Difficulties

- **Cost**
  - Very little automated support
  - Full traceability is very expensive and time-consuming
- **Delayed gratification**
  - The people defining traceability links are not the people who benefit from it
    - Development vs. V&V
  - Much of the benefit comes late in the lifecycle
    - Testing, integration, maintenance
- **Size and diversity**
  - Huge range of different document types, tools, decisions and responsibilities
  - No common schema exists for classifying and cataloging these
  - In practice, traceability concentrates only on baselined requirements

Traceability Difficulties

- Just Enough Process!

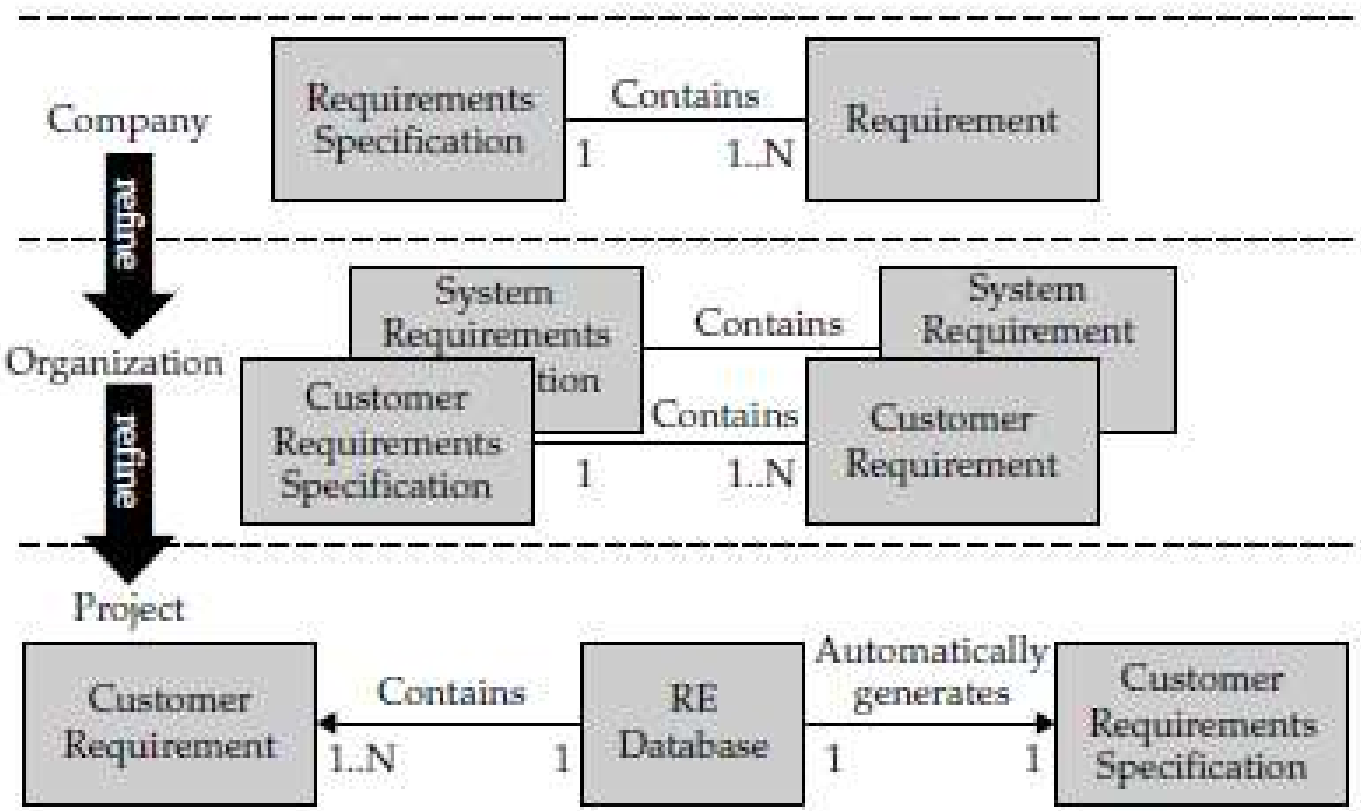


Forrester, February 2006, Best Practices "Performance-Driven Software Development"

RE Process Tailoring

- Define a Glossary of Terms for your project or product.
- Create an RE Taxonomy while keeping in mind what tools will be used to maintain it and how it will be communicated to the project team (e.g., publish to a project web site).
- Develop an RE Artifact Model specific to your project.
- Communicate project roles to all team members and the artifacts they are responsible for as defined in the RE Artifact Model.
- Use templates to define RE artifacts.
- For scaling projects, provide tailoring information in the RE Artifact Model; e.g., a specific artifact may be mandatory, optional, or not used, depending on the project size.
- Tailor the RE Artifact Model for a specific project from any corporate-level models, if they exist.
- Create a system life cycle.

Organizational Tailoring of an artifact model



Customer Requirement = User Requirement

RE Process Tailoring 사례

- 삼성전자 DAV
  - 요구사항 프로세스 Tailoring Proposal
  - 요구사항관리 프로세스 지침서_삼성전자_DAV
  - SW 요구사양서(SRS) 작성지침_삼성전자_DAV

- 삼성전자 첨기연: Single Core에서 Multi Core로의 Architecture 전환 프로세스 set-up
  - SW 요구사양서(SRS) 작성지침

Tips for RE Tailoring

빈 양식(Template)만으로는 아무것도 가이드할 수 없음. 해당 도메인의 대표적인 요구사항 타입을 정의하고 이에 대한 STEREOTYPED EXAMPLE을 제시해야 함