

Chapter 6

요구 명세

목 차

Chapter 6. 요구명세

- 요구 명세의 수준
- 소프트웨어 요구문서 작성
- 요구문서 구성 전략
- 요구명세의 품질요소
- 엄밀한 요구명세 방법
- 요구검증 기법

1. 요구 명세 개요

명세의 완전성, 일관성

- ❑ 원칙적으로 요구문서는 완전하고, 일관성이 있어야 한다.
- ❑ 완전성 (completeness)
 - 명세는 모든 필요로 하는 요소를 포함하고 있어야 한다
- ❑ 일관성 (consistency)
 - 명세에 포함된 요소들 간에는 모순(**contradictions**)이나 충돌(**conflicts**)이 없어야 한다.
- ❑ 실제로 일관성과 완전성 문제를 완전히 해결한 요구 문서를 확보하는 것은 어려운 일이다

1. 요구 명세 개요

엄밀도(rigor)의 수준

Informal Specification

- 자연어 기반의 서술, 작업흐름도 등의 그림 중심 작성
- 작성 및 이해 용이, 사용자와 개발 팀간의 의사전달이 용이
- 불충분한 명세, 일관성 결여, 내용의 모호성, 완전성 검증 곤란

Semi-Formal Specification

- 기능중심(Function-Oriented) 명세 : FSM, State Charts, Petri Nets
- 상태중심(State-Oriented) 명세 : Decision tables/trees, PDL, SDL, RLP
- 객체중심(Object-Oriented) 명세 : UML

Formal (Mathematical) Specification

- 모델기반 언어(Z, VDM), 대수처리 기반 언어(CSP, CCS , LOTOS)
- 명세의 정확성, 불완전성, 불일치 검토 입증, 모호성을 방지
- Not everybody can read formal specs

2. 소프트웨어 요구문서 작성

자연어 명세

- ❑ Lack of clarity
- ❑ Requirements confusion
- ❑ Requirements amalgamation
- ❑ Requirements for the insulin pump software system

3.2 The system shall measure the blood sugar and deliver insulin, if required, every 10 minutes. (Changes in blood sugar are relatively slow so more frequent measurement is unnecessary; less frequent measurement could lead to unnecessarily high sugar levels.)

3.6 The system shall run a self-test routine every minute with the conditions to be tested and the associated actions defined in Table 1

2. 소프트웨어 요구문서 작성

양식 기반 명세 (Form-based Specifications)

- ❑ Definition of the function or entity
- ❑ Description of inputs and where they come from
- ❑ Description of outputs and where they go to
- ❑ Information about the information needed for the computation and other entities used
- ❑ Description of the action to be taken.
- ❑ Pre- and post- conditions (if appropriate)
- ❑ The side effects (if any) of the function

2. 소프트웨어 요구문서 작성

양식 기반 명세의 예

- ❑ 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.

2. 소프트웨어 요구문서 작성

양식 기반 명세의 예 [계속]

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.

2. 소프트웨어 요구문서 작성

표형식 명세 (Tabular specifications)

- ❑ Used to supplement natural language.
- ❑ 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.

2. 소프트웨어 요구문서 작성

표형식 명세의 예

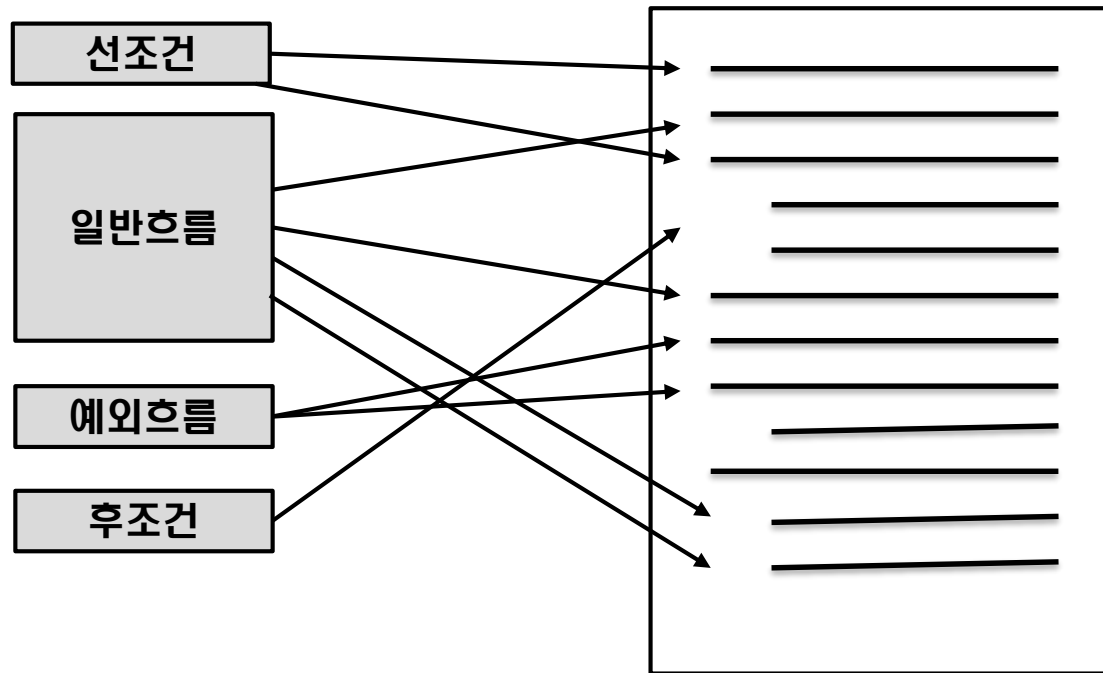
- ❑ Requirements for the insulin pump software system

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 ($(r2 - r1) \geq (r1 - r0)$)	CompDose = round $((r2 - r1)/4)$ If rounded result = 0 then CompDose = MinimumDose

2. 소프트웨어 요구문서 작성

Use case와 SRS

- ❑ Usecase와 SRS는 구조와 목적이 다름
- ❑ 유스케이스로부터 나온 기능요구사항은 계층적으로 구조화되어 SRS 곳곳에 분산됨




2. 소프트웨어 요구문서 작성

Use case 예: Checkout

Flow of event:

- 1 The use case begins when the Customer selects “Checkout”.
- 2 The system asks the Inventory actor to provisionally reserve the items in the shopping basket.
- 3 For each item that is out of stock
 - 3.1 The system informs the Customer that the item is currently unavailable and it is removed from the order.
- 4 The system presents the final order to the Customer. The order includes an order line for each product that shows the product identifier, the product name, the quantity, the unit price, the total price for that quantity. The order also includes the shipping address and credit card details of the Customer and the total cost of the order including tax and postage and packing.
- 5 The system asks the Customer to accept or decline the order
- 6 The Customer accepts the order.
- 7 include(AcceptPaymentByCard)

SRS:



R3	The ECP shall display detailed product descriptions consisting of name, photograph, price and description on demand.	<ul style="list-style-type: none">• Products• Functional	MustHave
		
R5	The ECP shall accept all major credit cards.	<ul style="list-style-type: none">• Payment• Functional	MustHave
R6	The ECP shall validate payment with the credit card processing company.	<ul style="list-style-type: none">• Payment• Functional	MustHave
R7	The ECP shall automatically calculate and add a shipping charge to the order.	<ul style="list-style-type: none">• Payment• Functional	Should-Have

3. 요구사항 명세 전략

고객관점의 유지

고객 관점에서 명세를 한다.

- 요구사항 작성자/독자가 사용하는 표현에 의존

고객과 개발자가 이해하기 쉽게 작성한다.

- 최신 유행을 피한다 - 전문가의 자기중심적 특정 기법 지양
- 이해하기 쉬운 표현 기법을 사용 : Model과 Picture를 포함

읽고 이해하기 쉬운 표현을 한다.

- 긍정적인 표현 / 능동적인 표현 / 충분한 정보를 제공 : 생략의 문제, 수치 범위에서 경계 값

명확한 용어를 사용한다.

- 모호한, 다중 의미를 가지는 용어 지양
- 일관성 있는 용어를 사용

3. 요구사항 명세 전략

수동표현, 능동표현

수동적 표현: 많은경우 기능 요구는 수동 표현이다. 이는 동작이 갖는 엔티티를 명백히 식별하는 장점이 있지만, 가급적 능동 표현으로 바꾼다

출력상태가 변할 때 이벤트는 로그에 기록된다

능동표현의 예

출력상태가 변할 때, **시스템은** 이벤트 로그에 새로운 상태와 상태변경 시간을 기록해야 한다

3. 요구사항 명세 전략

부모자식 요구사항

계층적이 방식의 요구 서술은 부모 요구사항과 하나 이상의 자식 요구사항을 갖는다. 다음 예에서는 3.4와 3.4.2가 충돌한다

3.4 요청자는 각 주문한 화학 제품의 관리 번호를 입력한다

3.4.1 해당시스템은 마스터 관리번호 목록에 대해 관리번호를 확인해야 한다.
관리번호가 타당치 않으면 시스템은 요청자에게 알려주고 주문을 받지 않아야 한다.

3.4.2 관리번호는 주문에서 개별 품목이 아닌 전체주문에 적용한다.

개선

3.4 관리번호

3.4.1 요청자는 각 주문한 화학 제품의 관리 번호를 입력한다

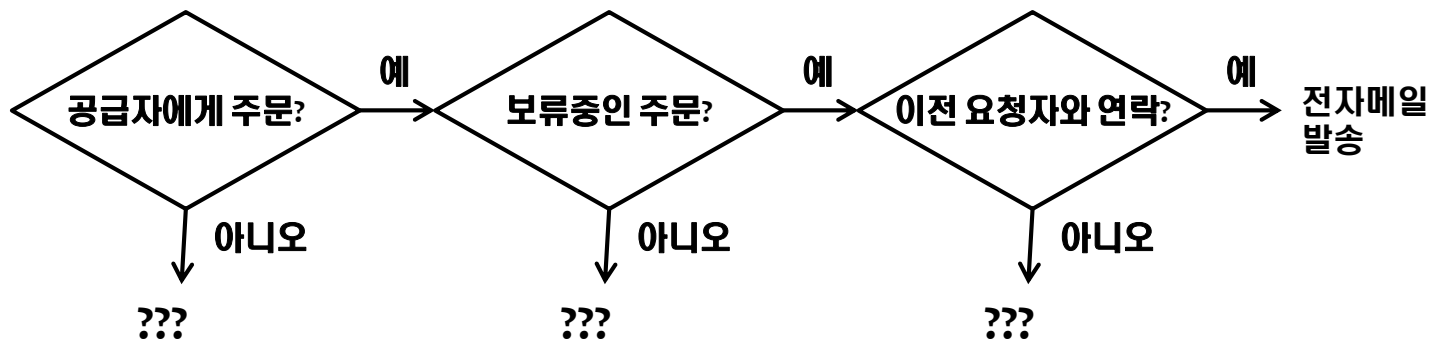
3.4.2 해당시스템은 마스터 관리번호 목록에 대해 관리번호를 확인해야 한다.
관리번호가 타당치 않으면 시스템은 요청자에게 알려주고 주문을 받지 않아야 한다.

3. 요구사항 명세 전략

복잡한 요구사항

- ❑ 복잡한 논리는 요구사항을 모호하게 만들 위험이 있다.

공급자에게 화학제품 주문이 들어가면, 시스템은 그 화학제품에 대한 보류 중인 다른 주문이 있는지 확인한다. 만일 있다면 시스템은 공급자 이름, 공급자 카탈로그 번호, 각각의 주문자 이름을 표시한다. 사용자가 이전 주문자와 연락하길 원하면, 시스템은 그에게 전자메일을 보낼 수 있게 한다.



- ❑ 이 경우 괄호나 분리를 사용한다.

3. 요구사항 명세 전략

부정적 요구사항

- ❑ 부정적, 혹은 반대 요구사항은 혼란의 원인이 된다.
- ❑ 이 경우 긍정적인 표현으로 바꾸어 전달하는 의미를 명확히 한다

Before	After
세 개 이상의 계정을 가진 사용자는 이동할 수 없다	시스템은 사용자가 가진 계정이 세개보다 작은 경우만 이동한다
등록된 언어는 영어가 기본이 될 것이고, 나라 와 언어가 선택되지 않을 때 까지 지역화된 기능을 나타내지 않을 것이다	등록된 언어는 영어가 기본이다.사용자가 나라 와 언어를 선택한 후, 등록프로세스는 지역화 된 기능을 나타낸다
도메인 이름은 등록유예 기간 동안 다른 등록 자에게 전달될 수 없다	도메인 관리자는 등록유예 기간 이후에만 다른 등록자에게 전달해도 좋다
일반사용자는 사용자를 변경할 능력을 가지고 있지 않을 것이다	시스템 관리자만 사용자를 변경할 수 있다

4. 요구문서의 품질요소

명세 품질을 위해 고려해야 할 요소

- ❑ Correct
- ❑ Complete
- ❑ Consistent
- ❑ Unambiguous
- ❑ Verifiable
- ❑ Modifiable
- ❑ Traceable
- ❑ Design Independent
- ❑ Feasible
- ❑ Organized

4. 요구문서의 품질요소

정확성 (Correctness)

Bad Requirement

All screens must appear on the monitor **quickly**.

→ Is the term 'quickly' what user really needed ?

Good Requirement

When the user accesses any screen, it must appear on the monitor **within 2 seconds**.

4. 요구문서의 품질요소

완전성 (Completeness)

Bad Requirement

On loss of power, the battery backup must support normal operations.

→ missing information: For how long?

Good Requirement

On loss of power, the battery backup must support normal operations **for 20 minutes**.

4. 요구문서의 품질요소

완전성 (Completeness)

Bad Requirement

3.2.5 The system shall process two new fields (provides production count balancing info to states) at the end-of-state record.

→ What are the two fields, 'info' should be spelled out

Good Requirement

3.2.5 The system shall process **the following fields** (provides production count balancing **information** to states) at the end-of-state record.

- a. **SDATE, and**
- b. **YR-TO-DATE-COUNT**

4. 요구문서의 품질요소

일관성 (Consistency)

Bad Requirement

RQ3: The electronic batch records shall be **Part 11** compliant.

RQ 27: An on-going training program for **21 CFR Part 11** needs to be established at the sites.

→ Do these refer to the same regulation or different ones?

Good Requirement

RQ3: The electronic batch records shall be **21 CFR Part 11** compliant.

RQ 27: An on-going training program for **21 CFR Part 11** needs to be established at the sites.

4. 요구문서의 품질요소

일관성 (Consistency)

Bad Requirement

3.3.2.1 The system shall have no single point failure.

→ This is an ambiguous requirement. Needs identification of “no single point failure”

Good Requirement

3.3.2.1 **The following system components** shall have no single point failure:

- a. **Host servers**
- b. **Networks**
- c. **Network routers**
- d. **Access servers**
- e. **Hubs**

4. 요구문서의 품질요소

확인가능성 (Verifiability)

Bad Requirement

The system must be user friendly.

→ How should we measure user friendliness??

Good Requirement

The user interface shall be **menu driven**. It takes normally **1 hour to learn** the usage of the system.

4. 요구문서의 품질요소

추적성 (Traceability)

Bad Requirement

The system must generate a batch end report **and** a discrepancy report when a batch is aborted.

→ How is this uniquely identified? If the requirement is changed later so that it does not require a discrepancy report, how will you trace it back?

Good Requirement

RQ 6v1: The system must generate a batch end report when a batch is aborted.

RQ 7v2: The system must generate a discrepancy report when a batch is completed or aborted.

4. 요구문서의 품질요소

타당성 (Feasibility)

Bad Requirement

The replacement control system shall be installed with no disruption to production.

→ This is an unrealistic expectation.

Good Requirement

The replacement control system shall be installed causing **no more than 2 days of production disruption**.

4. 요구문서의 품질요소

설계 독립성 (Design Independence)

Bad Requirement

After 3 unsuccessful attempts to log on, **a Java Script routine** must run and lock the user out of the system.

→ Specifying a JavaScript routine concerns how the requirement will be implemented.

Good Requirement

After 3 unsuccessful attempts to log on, **the user must be locked out of the system.**

5. 엄밀한 요구의 명세

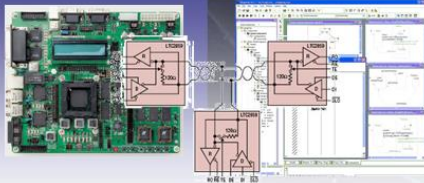
명세 기법의 비교

- ❑ 개발자는 적절한 수준의 명세 기법을 선택하여야 한다
- ❑ 정형 기법
 - 가장 강력한 표현 기법이다
 - 습득과 활용에 많은 시간이 걸린다
- ❑ 준 정형기법
 - 보통 수준의 엄밀성을 갖는다
 - 쉽게 배울 수 있고 활용이 용이하다
 - 다양한 형태의 도구가 지원된다
- ❑ 비정형기법
 - 가장 일반적이며 친숙하다
 - 명세 언어로 가장 바람직하지 못한 형태이다
- ❑ Trade-off
 - 사용성 vs. 표현력,
 - 비용 vs. 효과

5. 엄밀한 요구의 명세

Safety-critical, mission-critical systems

SOC



Highly-Integrated Circuit

Security



Security Protocols / Smart Card

Robotics



Planning / Ontology / Hybrid Control System

Railway



Railway Control System

Aerospace



Space Shuttle / Aircraft Guidance / Navigation / Control System

Nuclear Power Plant



Nuclear Power Plant Control System

Medical



Safety-Critical System In Medical Areas

Automotive



ECU (Electronic Control Unit) / Cruise Control System

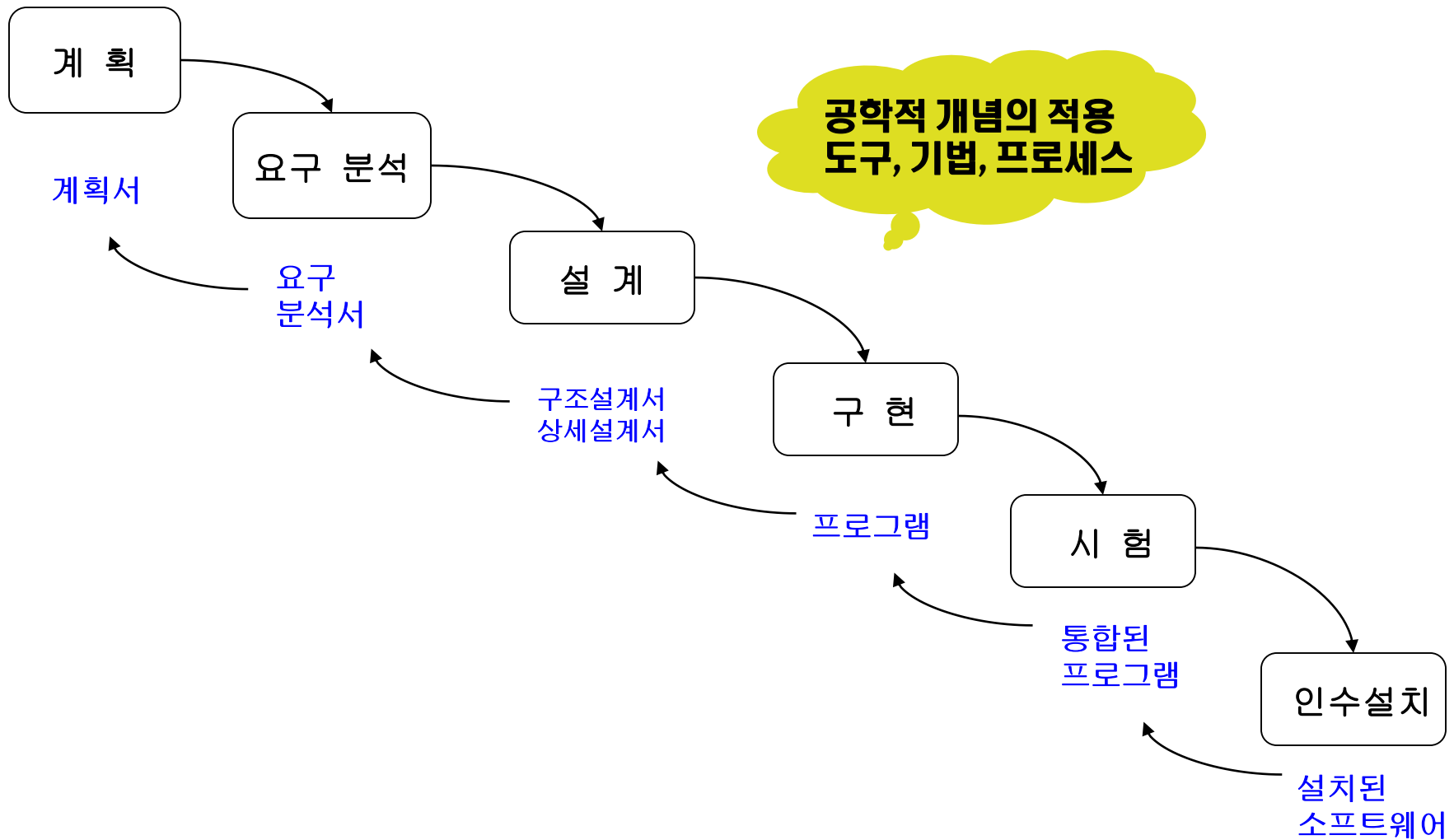
5. 엄밀한 요구의 명세

Formal Specification

- ❑ 정형기법(formal methods)은 정확한 요구 명세 획득을 위한 수학적 명세 및 검증 기술이다
- ❑ 실시간 응용 및 통신, 보안, 항공 및 국방 등 임무 중요(mission-critical) 혹은 안전 중요(safety-critical) 영역에 있어 고객의 품질 요구를 만족시키기 위해 활용된다
- ❑ 비용과 활용 난이도로 인해 성공적으로 적용될 수 있는 분야는 한정되어 있다
 - 보안, 반도체 등 관련분야 제품의 수출을 위해 인증이 필요한 경우
 - 원자로 제어, 우주/항공기 제어, 철도 제어 등 시스템 실패시 생명 및 재산상의 큰 손해와 직결된 분야
 - 기타 정확한 작동이 요구되는 임베디드 시스템 등의 응용분야

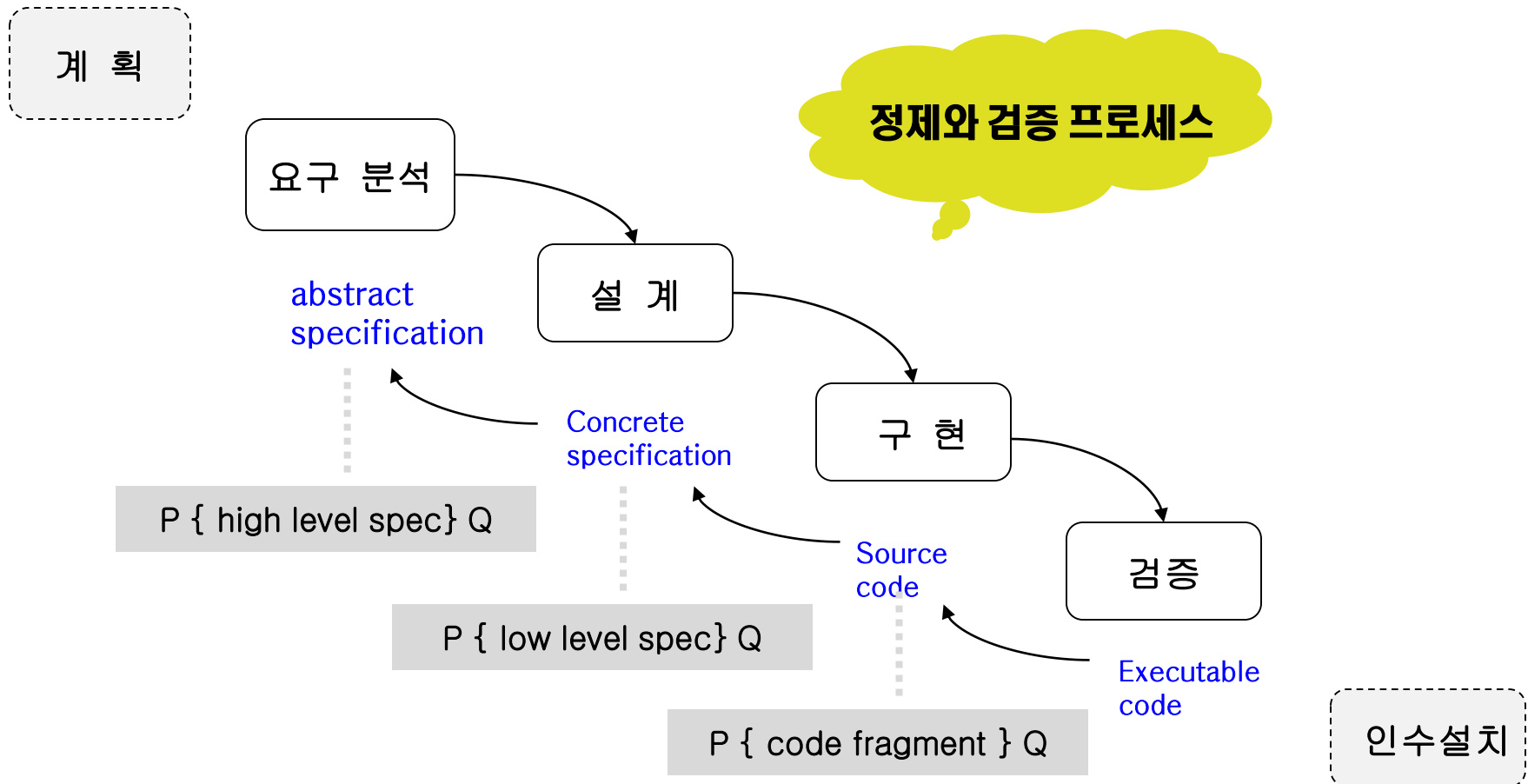
5. 엄밀한 요구의 명세

기존 방법에서의 명세 활용



5. 엄밀한 요구의 명세

Specification refinement process



6. 정형 명세

Pre-, Post-condition 명세: 몫과 나머지 예

- 임의의 수 x 를 y 로 나눈 몫을 q , 나머지를 r 이라 할 때 몫 q , 나머지 r 을 구하는 함수의 기능을 표현함
- 변수들 사이의 관계: $x == y * q + r$

pre-condition : $\longrightarrow 0 \leq x \wedge 0 < y$

```
findQandR(x,y){  
    r=x; q=0;           // {0 ≤ r ∧ 0 < y ∧ x=y*q+r }  
    while (r >= y) {     // {0 ≤ r ∧ 0 < y ≤ r ∧ x=y*q+r }  
        r= r-y; q=q+1;   // {0 ≤ r ∧ 0 < y ∧ x=y*q+r }  
    }
```

post-condition : $\longrightarrow 0 \leq r < y \wedge x=y*q+r$

6. 정형 명세

Z 명세의 예

BANK를 명세하기 위한 조건들

- The signature of the state schema **BANK** declares a function **balance** (positive or negative) held in any account in use.
- The balances are in pence, so they are integers.
- a set **active** which consists of the account numbers which are currently in use;
- a set **overdrawn** which consists of all account numbers of accounts that are overdrawn;
- a set **deposit** which consists of the account numbers of deposit accounts,
- a set **current** which consists of the account numbers of current accounts.

6. 정형 명세

스키마를 활용한 Z 명세

BANK

balance: ACC \rightarrow Z

active, overdrawn, deposit, current: P ACC

active = dom balance

active = current \cup deposit

current \cap deposit = \emptyset

overdrawn = dom (balance \triangleright { n:Z | n < 0 })

overdrawn \cap deposit = \emptyset

자연어 설명

Line 1: The active accounts are those for which there is a balance recorded

Line 2: The active accounts are the current accounts and the deposit accounts taken together

Line 3: No account can be both current account and a deposit account

Line 4: The overdrawn accounts are active accounts for which the balance is negative

Line 5: No deposit account can be overdrawn

6. 정형 명세

오퍼레이션 명세의 예

OpenDepAccount
Δ BANK amount? :Z accno!: ACC r!: seq CHAR
accno! \notin active amount? > 0 current' = current deposit' = deposit \cup {accno!} balance' = balance \cup {accno! \rightarrow amount?} r! = "account opened"

자연어 설명

- Opening a deposit account: the input is an integer being the sum to be deposited and the outputs are an account number and a response message "account opened".
- Closing an account: the input is an account number and the outputs are an integer, being the sum of money in the account, and a response message "account closed"; there should be a precondition in this operation that the account is not overdrawn.

6. 정형 명세

오퍼레이션 명세의 예

CloseAccount
Δ BANK
amount! :Z
accno?: ACC
r!: seq CHAR
accno? \in active
accno? \notin overdrawn
amount! = balance (accno?)
deposit' = deposit \setminus {accno?}
current' = current \setminus {accno?}
balance' = {accno?} \triangleleft balance
r! = "account closed"

자연어 설명

- Opening a deposit account: the input is an integer being the sum to be deposited and the outputs are an account number and a response message "account opened".
- Closing an account: the input is an account number and the outputs are an integer, being the sum of money in the account, and a response message "account closed"; there should be a precondition in this operation that the account is not overdrawn.

Workshop #6 – Functional & NF requirements

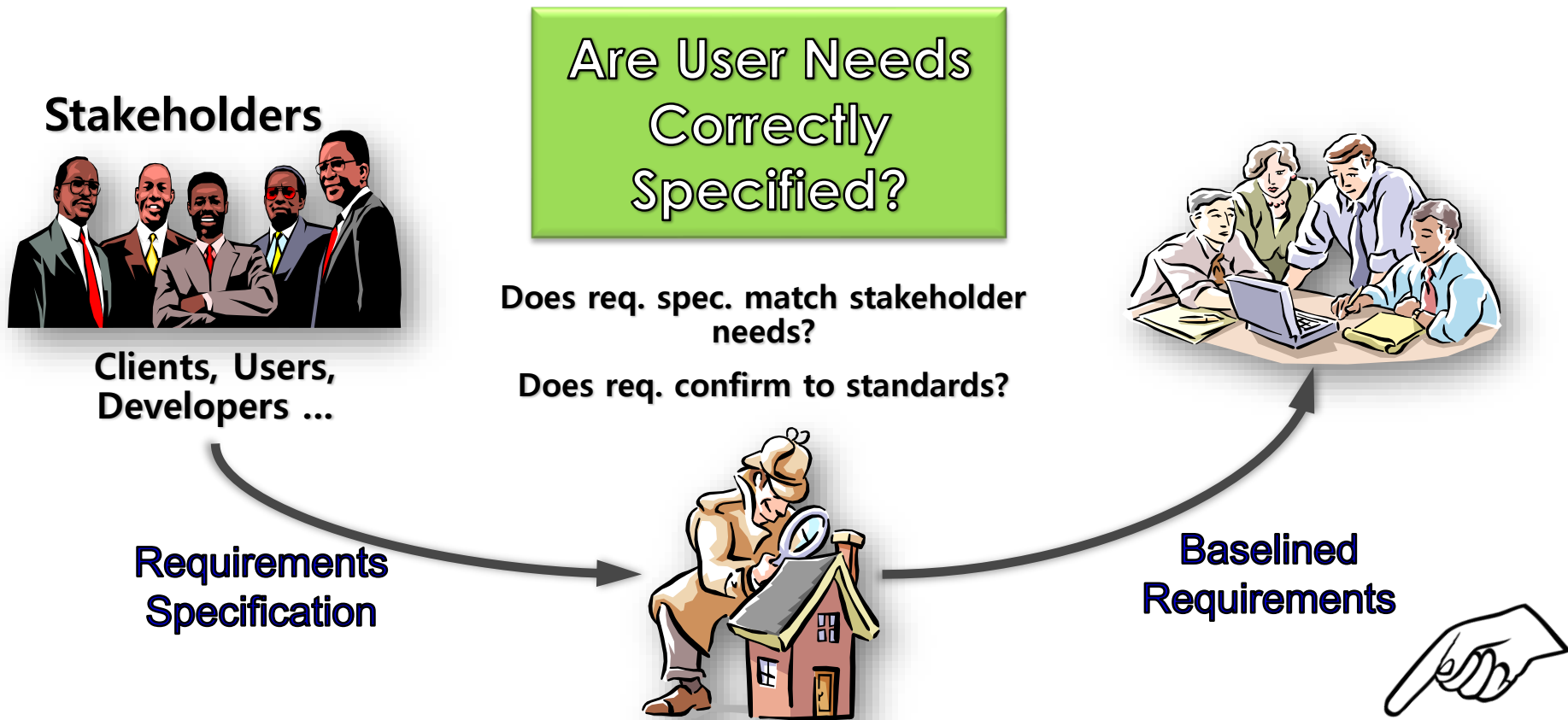
- ❑ 요구모델링 워크샵의 내용을 바탕으로 시스템 요구사항을 도출합니다.
- ❑ 요구사항의 필요한 부분은 부록의 Use case를 참고로 도출합니다.
- ❑ 기능요구와 비기능 요구를 포함한 팀별 20개+ 의 요구를 도출합니다
- ❑ 예시:

ECP System Requirements Specification

ID	Details	Type	Priority
R1	The ECP shall display a list of all products offered by Clear View Training Limited.	<ul style="list-style-type: none">• Products• Functional	MustHave
R2	The ECP shall organise the list of products by product category.	<ul style="list-style-type: none">• Products• Functional	MustHave
R3	The ECP shall display detailed product descriptions consisting of name, photograph, price and description on demand.	<ul style="list-style-type: none">• Products• Functional	MustHave
R4	The ECP shall store sales transaction data.	<ul style="list-style-type: none">• Availability• NonFunctional	Should-Have

7. 요구 검증

요구 검증의 목적



The process of checking whether the requirements, as identified, **do not contradict the expectations about the system of various stakeholders, and do not contradict each other.**

7. 요구 검증의 개념

명세에 대한 주요 검증 속성

❑ 정확성(Correctness) 혹은 유효성 (Validity)

- Does the system provide the functions which best support the customer's needs?

❑ 일관성(Consistency)

- Are there any requirements conflicts?

❑ 완전성(Completeness)

- Externally - all desired properties are present?
- Internally – Are there any undefined references?

❑ 추적성(Traceability)

- Should relationship between requirements and their sources be able to be understood in both directions?

❑ 확인가능성(Verifiability)

- Is the requirement realistically testable?



3C + TV

7. 요구 검증의 개념

그 외 검증 속성

▣ 이해성(Comprehensibility)

- Is the requirement properly understood?

▣ 적응성(Adaptability)

- Can the requirement be changed without a large impact on other requirements?

7. 요구 검증의 개념

요구 확인과 검증 (Verification & Validation)

□ 확인 (Verification)

- Am I building the product right?
- 개발하고 있는 시스템이 미리 정의된 사양 (specification)에 부합하고 있는지를 검증하는 것
- 아직 실제 시스템은 없기 때문에 시뮬레이션 결과 등을 통해서 검증하는 과정이 필요함
- 주로 분석가나 개발자에 의해 확인이 수행됨

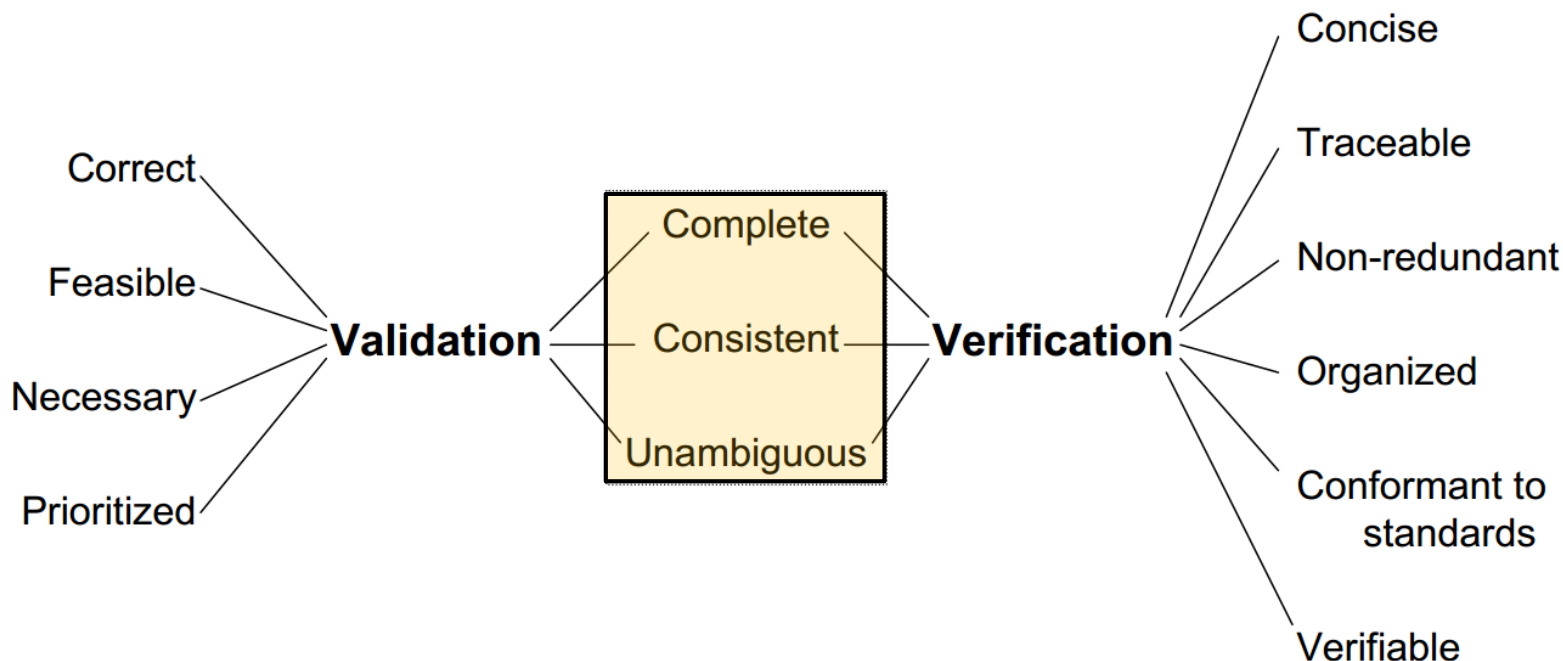
□ 검증 (Validation)

- Am I building the right product?
- 개발 완료된 시스템이 사용자의 요구 사항을 충족하는지 확인하는 것이 목적임
- 실제 시스템이 존재하기 때문에, 직접 타겟 시스템을 이용해 검증함
- 주로 이해당사자가 참여하여 검증이 이루어짐

7. 요구 검증의 개념

요구 확인과 검증 (Verification & Validation)

- ❑ V&V는 미리 정의된 품질 기준에 준해서 오류의 존재 여부를 확인한다는 점에서 공통의 목적을 갖는다
- ❑ 이러한 이유로 확인과 검증을 용어적으로 구분하여 사용하지 않고 검증이라 부르는 경우가 많다.



7. 요구 검증의 개념

검증(V&V)과 테스트

- ❑ 시스템이 미리 정의한 사양 (specification)에 부합하고 있는지를 확인하는 측면에서 검증과 테스트는 동일한 목적을 가짐
- ❑ 테스트는 SDLC 상에서 정의되는 독립적이고 이산적인 단계 (discrete phase)의 활동임에 비해 검증은 각 단계를 통해 수행되는 QA 활동임
- ❑ 그럼에도 불구하고 종종 테스트를 V&V 활동의 일부로 보거나 혹은 일부 V&V 기법(예: code review)을 테스트 (예: static testing)으로 분류하기도 함*
- ❑ 테스트와 검증은 제품 품질 향상을 위해 보완적으로 사용되어야 함

“The first mistake that people make is thinking that the testing team is responsible for assuring quality”

* Static testing is done to test the software work products , requirement specifications, test plan , user manual etc. They are not executed, but tested with the set of some tools and processes.

<http://www.abodeqa.com/2014/03/22/reviewswalkthrough-and-inspection-in-software-testing/>

8. 요구 검증 기법

요구 검증 방법의 분류

▣ Requirements reviews*

- systematic manual analysis of the requirements
- review, formal inspection

▣ Requirement translation

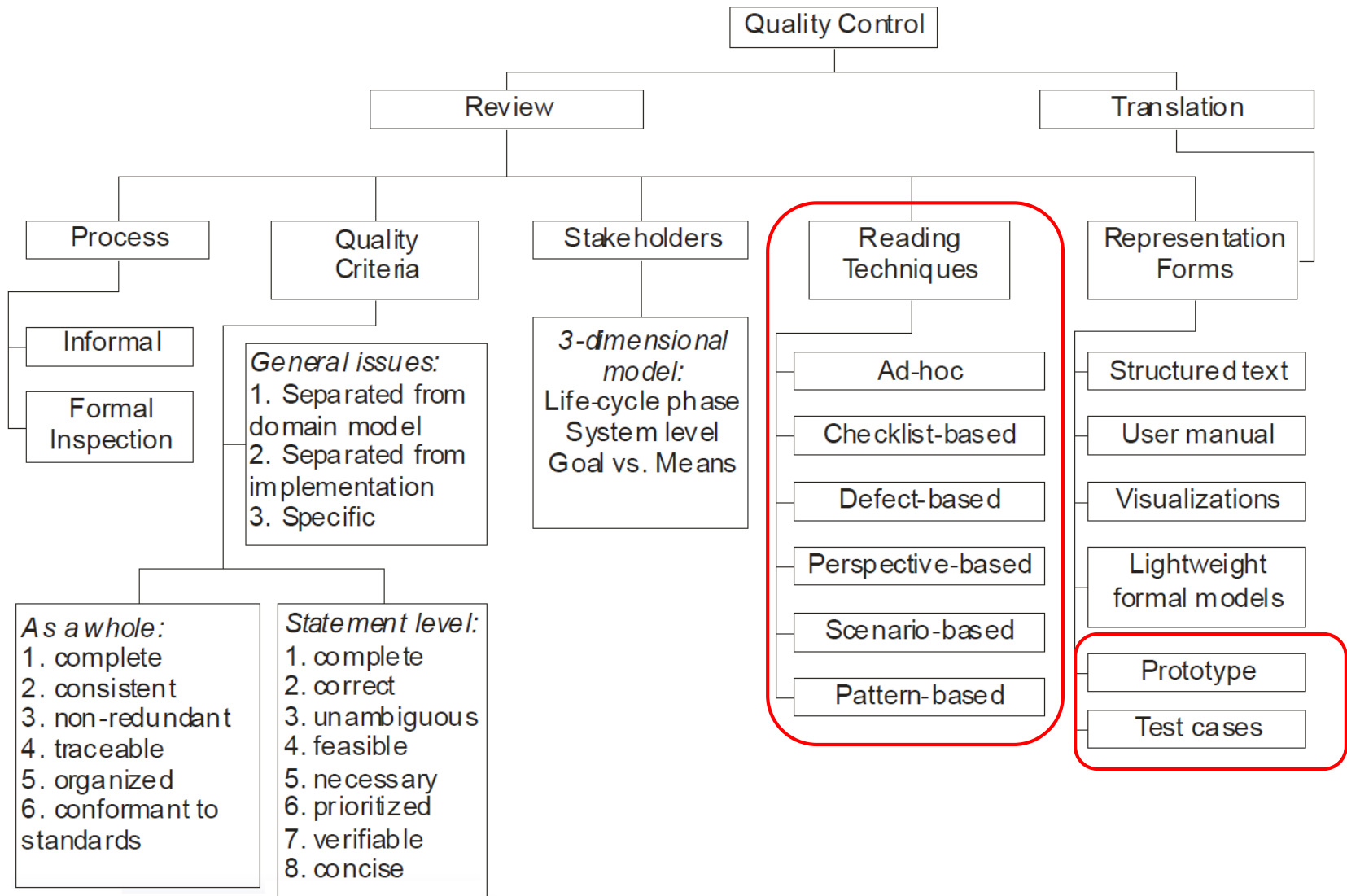
- rewriting requirements to more analytic forms
- requirement visualization, requirement animation
- requirement proof
- prototyping

* Definitions are not widely agreed

– Other terms used: Formal Technical Reviews(FTRs), Formal Inspections

8. 요구 검증 기법

요구 검증 방법의 상세 분류



8. 요구 검증 기법

Fagan Inspection Process

1 Overview Meeting

- communicate and educate about product
- circulate materials
- Rate: 500 SLOC per hour

2 Preparation

- All participants perform individually
- review materials to detect defects
- Rate: 100-125 SLOC per hour

3 Inspection Meeting

- a reader paraphrases the design
- identify and note problems (don't solve them)
- Rate: 130-150 SLOC per hour

8. 요구 검증 기법

Review/Inspection Reading techniques

- ❑ In both formal and informal reviews, an important factor is the level of guidance the reviewers receive for accomplishing their task, or *reading technique*
- ❑ Reading techniques:
 - Ad-hoc
 - Checklist-based
 - Defect-based
 - Perspective-based
 - Scenario-based
 - Pattern-based

8. 요구 검증 기법

Review/Inspection Reading techniques

- ❑ Ad-hoc reading – **no guidance is provided**, reviewers use only their own knowledge and experience to identify defects.
- ❑ Checklist-based reading – **a list of questions is provided specifying what properties of the document must be checked** and what specific problems should be searched for
 - Usually, the only alternative that is discussed in many requirements engineering books.
 - Every relevant requirements quality criterion is rewritten in the form of two or more questions.
 - A checklist works just as a reminder for reviewers of those quality criteria

8. 요구 검증 기법

Reading Techniques: Checklist 예

1. Organization and Completeness

- ☐ Are all internal cross-references to other requirements correct?
- ☐ Are all requirements written at a consistent and appropriate level of detail?
- ☐ Do the requirements provide an adequate basis for design?
- ☐ Is the implementation priority of each requirement included?
- ☐ Are all external hardware, software, and communication interfaces defined?
- ☐ Have algorithms intrinsic to the functional requirements been defined?
- ☐ Does the SRS include all of the known customer or system needs?
- ☐ Is any necessary information missing from a requirement? If so, is it identified as TBD?
- ☐ Is the expected behavior documented for all anticipated error conditions?

현실적인 수준의 명세 완전성 점검 요소를 판단할 수 있음

8. 요구 검증 기법

Reading Techniques: Checklist 예 [계속]

2. Correctness and Consistency

- ☐ Do any requirements conflict with or duplicate other requirements?
- ☐ Is each requirement written in clear, concise, unambiguous language?
- ☐ Is each requirement verifiable by testing, demonstration, review, or analysis?
- ☐ Is each requirement in scope for the project?
- ☐ Is each requirement free from content and grammatical errors?
- ☐ Can all of the requirements be implemented within known constraints?
- ☐ Are any specified error messages unique and meaningful?

3. Quality Attributes

- ☐ Are all performance objectives properly specified?
- ☐ Are all security and safety considerations properly specified?
- ☐ Are other pertinent quality attribute goals explicitly documented and quantified, with the acceptable tradeoffs specified?

8. 요구 검증 기법

Review/Inspection Reading techniques

❑ **Defect-based reading** – each procedure is aiming for detecting a particular type of defects, different procedures are usually assigned to different reviewers.

- Some empirical evidence exists that this may outperform checklist-based and ad-hoc approaches

❑ **Scenario-based reading** - a set of concrete scenarios is provided.

Reviewers walk through the sequence of events in each scenario and examine the requirements document for presence, correctness, etc. of requirement statements that cover those.

- May be usage scenarios, maintenance work scenarios, etc.

8. 요구 검증 기법

Review/Inspection Reading techniques

- ❑ Perspective-based reading (PBR) – each procedure is based on the viewpoint of a particular stakeholder.
- ❑ The goal is to examine a software artifact description from the perspectives of the artifact's stakeholders in order to identify defects
- ❑ Characteristics
 - specific steps of the review process can be defined
 - different reviewers focus on different aspects of the document.
 - PBR is customizable depending on the project and organization.
 - PBR instructs how the perspectives and procedures can be created based on the software artifact at hand.
 - based on experience from previous reviews, the scenarios used can be improved by modifying the questions that are part of the scenarios.

8. 요구 검증 기법

PBR sample : Tester scenario

Introduction: Assume you are reading the requirements from a system tester's perspective. You are interested in knowing whether the requirements are such that test cases can be written based on them. Concentrate on finding vague, ambiguous and unclear requirements.

Instructions: For each requirement, create a test case or a set of test cases that can be used to verify that requirement. Try to create tests based on the equivalence sets of inputs (one test case for each functionally different outcome). Document inputs and expected outputs for each test case.

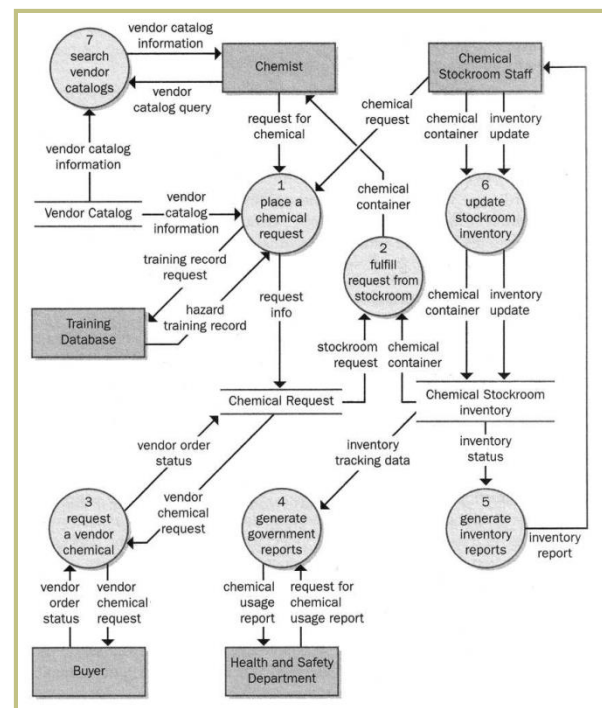
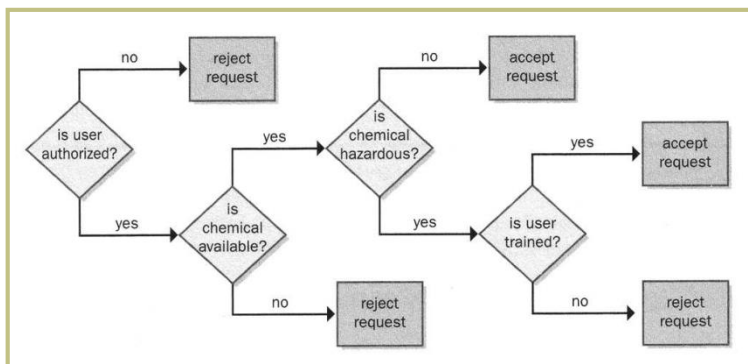
Questions:

1. Is all relevant information available to produce the test cases (inputs)?
2. Is the outcome of the test specified unambiguously?
3. Is there an alternative interpretation of the requirement that would result in a functionally different outcome?

8. 요구 검증 기법

요구 검증 방법 – Translation : Visualization

- Visualization is often seen as a way to help people gain insight from large and complex data sets.
- Graphical presentations link individual statements together and present a coherent picture of a slice of the system.
- Decision trees, data flow diagrams, state transition diagrams, etc.



9. 엄밀한 명세 검증 방법

정확성, 일관성, 완전성 -revisited

□ 정확성(correctness)에 관한 학문적 관점

Correctness by itself is a vague concept. We can consider correctness from at least two different perspectives:

- (1) From a formal point of view, correctness is usually meant to be the combination of consistency and completeness. Consistency refers to situations where a specification contains no internal contradictions, whereas completeness refers to situations where a specification entails everything that is known to be “true” in a certain context. We will be more specific when we refer to correctness in the next section, but for the moment let us emphasize that consistency is an internal property of a certain body of knowledge, whereas completeness is defined with respect to an external body of knowledge.
- (2) From a practical point of view, however, correctness is often more pragmatically defined as satisfaction of certain business goals. This indeed is the kind of correctness which is more relevant to the customer, whose goal in having a new system developed is to meet his overall business needs.

9. 엄밀한 명세 검증 방법

정확성, 일관성, 완전성 -revisited

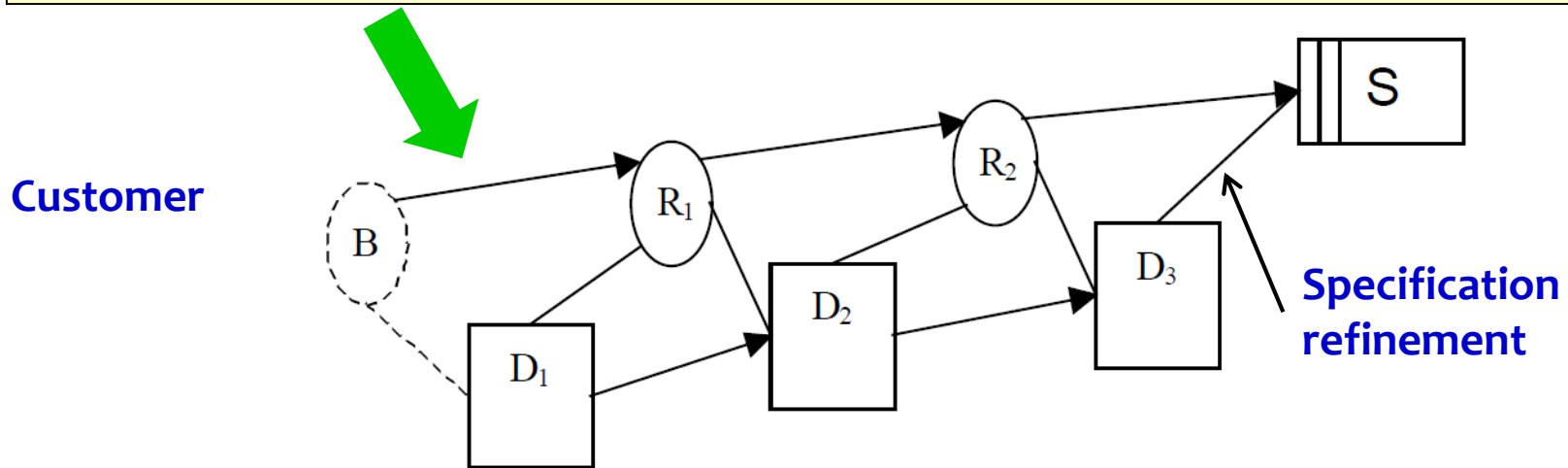
□ 완전성(completeness)에 관한 학문적 관점

Davis states that *completeness* is the most difficult of the specification attributes to define and *incompleteness* of specification is the most difficult violation to detect [4]. According to Boehm [2], to be considered complete, the requirements document must exhibit three fundamental characteristics: (1) No information is left unstated or “to be determined”, (2) The information does not contain any undefined objects or entities, (3) No information is missing from this document. The first two properties imply a closure of the existing information and are typically referred to as *internal completeness*. The third property, however, concerns the *external completeness* of the document [3]. External completeness ensures that all of the information required for problem definition is found within the specification. This definition for external completeness clearly demonstrates why it is impossible to define and measure absolute completeness of specification because how could analysts know with certainty what is missing from the specification when they do not even know what it is that they are looking for in the first place. Clearly one of the available techniques that could assist

9. 엄밀한 명세 검증 방법

정확성, 일관성, 완전성 -revisited

$B=R_0=\{\text{when a customer comes near the entrance, the door shall open}\}$
 $D_1=\{\text{when a person comes near the entrance, a presence sensor gets activated}\}$
 $R_1=\{\text{when the sensor gets activated, the door shall open}\}.$
 $D_2=D_1 \cup \{\text{when a sliding door's motor is turned on, the door opens}\}$
 $R_2=\{\text{when the sensor gets activated, the door's motor shall be turned on}\}.$



$R_i \cup D_i \models R_{i-1}$ (completeness of R_i and D_i w.r.t. R_{i-1}) and
 $R_i \cup D_i \not\models \perp$ (consistency of $R_i \cup D_i$).

9. 엄밀한 명제 검증 방법

정확성, 완전성의 수학적 의미

□ Soundness (\rightarrow Consistency)

- 어떤 명제가 증명가능하다면 그 명제는 반드시 참이다
- $x \vdash y$ means y is provable from x (예: $A \rightarrow B \vdash \sim B \rightarrow \sim A$)
- 시스템이 Sound하지 않다면 이 시스템으로 증명한 명제가 참이 아닐 수도 있다
- Soundness를 갖는 proof system으로 유도 가능한 모든 proof는 참이다.
- 문제점: 하지만 모든 참인 명제를 이 proof system으로 증명할 순 없다.

□ Completeness

- 어떤 명제가 참이라면 그 명제는 반드시 증명가능하다
- $x \models y$ means x semantically entails y (예: $A \rightarrow B \models \sim B \rightarrow \sim A$)
- Completeness를 갖는 proof system으로는 실제로 참인 모든 명제를 증명가능하다. 한 개라도 증명하지 못한다면 Complete하지 않다.



이들 두 가지 모두를 만족하는 proof system은 모든 명제에 대해 참인지 거짓인지를 증명할 수 있다

9. 엄밀한 명세 검증 방법

분석가의 딜레마

□ 완전성의 딜레마

- 일반적으로 완전성을 높이기 위해서는 하나 혹은 그 이상의 요구를 명세에 추가한다
- 만일 완전성을 높이기 위해 명세를 추가할 경우 새로 추가된 요구로 인해 일관성을 잃을 수 있다

□ 일관성의 딜레마

- 일반적으로 일관성을 높이기 위해서는 하나 혹은 그 이상의 명세를 요구에서 제거한다.
- 만일 일관성을 높이기 위해 명세를 제거할 경우 제거된 요구로 인해 완전성을 잃을 수 있다.

Is it impossible to adequately address all three Cs simultaneously?

10. Requirement Engineering using AI Techniques

AI driven requirement management

- ❑ Requirement management with AI can reduce the amount of time to complete project by up to 60%.
- to automate the process, reduces time and increases effectiveness.
- to deal with complexity which helps to increase visibility.
- reduces data redundancy to bring out the correct information.



Reduce the cost of defects by **60 percent**



Reduce the cost of manual reviews by **25 percent**



Reduce development costs by **57 percent**



Accelerate time to market by **20 percent**



Reduce cost of quality by **69 percent**

10. Requirement Engineering using AI Techniques

요구 추출과 AI

- ❑ AI can enhance requirements elicitation by automating some of these techniques, such as **generating surveys, analyzing natural language inputs, and creating prototypes.**
- ❑ AI can also augment human capabilities **by providing suggestions, feedback, and insights based on data analysis, machine learning, and natural language processing.**
- ❑ Examples
 - IBM' s Engineering Requirements Quality Assistant uses Watson for requirements analysis managed on the DOORS Next platform.
 - ScopeMaster has developed a stand-alone requirements analysis tool that plugs into Jira, Azure DevOps

10. Requirement Engineering using AI Techniques

요구 분석/문서화와 AI

- ❑ AI can enhance requirements analysis by automating some of activities, such as validating and verifying requirements, tracing requirements to sources and artifacts, and generating specifications.
- ❑ AI can also augment human capabilities **by providing analysis, visualization, and simulation tools** that can help evaluate the feasibility, impact, and quality of requirements.
- ❑ AI can enhance requirements documentation by automating some of formats and methods, **such as generating natural language descriptions, diagrams, models, use cases, user stories, and scenarios from requirements.**