

The background of the image is a wide-angle photograph of a rural landscape. It features rolling hills covered in green and brown agricultural fields. Several white wind turbines are scattered across the hills, their blades pointing upwards. The sky is a clear, vibrant blue with no clouds. In the bottom right corner, there's a small, isolated blue building with a white roof.

Software Architecture

목 차

1. 여섯 가지 에피소드
2. EA와 TOGAF
3. Quality Attributes
4. View와 Viewpoint
5. IEEE 1471
6. IEEE 1471 – Extension
7. Architecting Process
8. Architecting Process – SEI/ADD

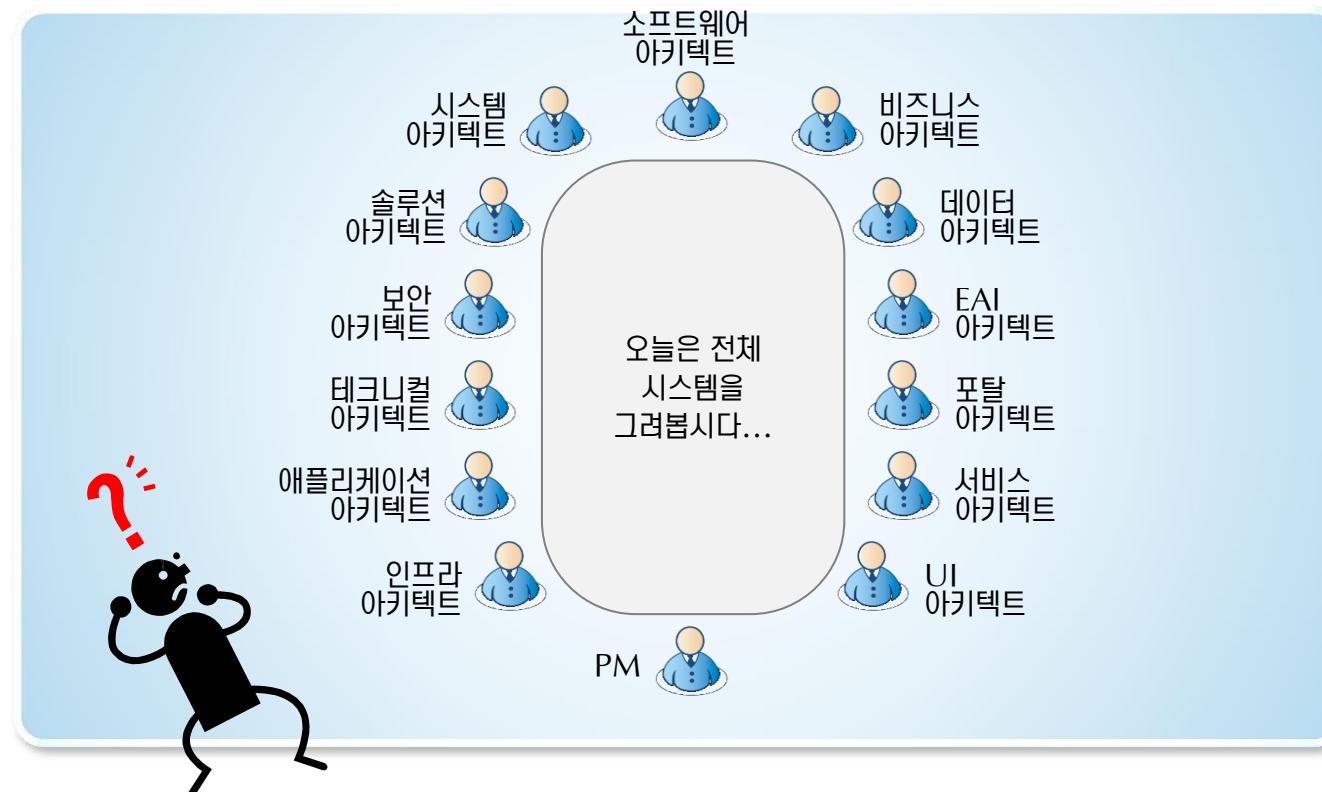


1. 여섯 가지 에피소드

-
- 1. Episode #1 – 13 Architects
 - 2. Episode #2 – Nobody's tasks
 - 3. Episode #3 – Not That Architecture
 - 4. Episode #4 – A system has four architectures
 - 5. Episode #5 – Different but same
 - 6. Episode #6 – Unidentified best job
 - 7. 토의

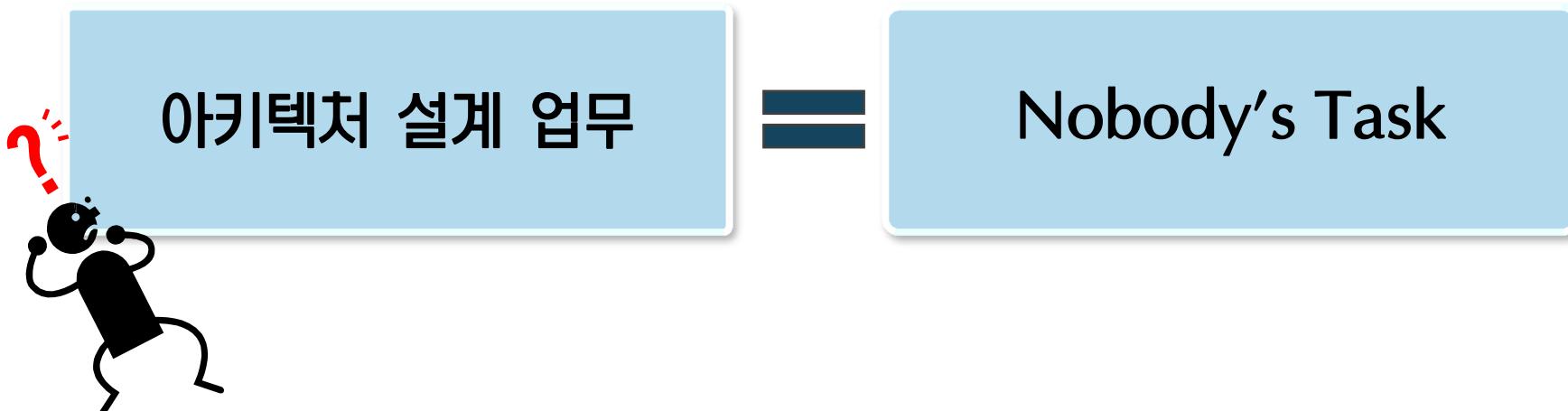
1. Episode #1 – 13 Architects

- ✓ 어느 금융 회사의 차세대 프로젝트에서 아키텍트 회의가 열렸습니다. 때는 11월13일 금요일, 13 인의 아키텍트가 모였습니다. 해가 지도록 시스템 Big picture에 대해서 이야기를 합니다.
- ✓ 밤은 깊어 가고 결론은 나지 않습니다...



2. Episode #2 – Nobody's tasks

- ✓ 업계에서 이름을 대면 알만한 SW 아키텍트 한 분이 초급 개발자 한 명을 데리고 프로젝트에 참여했습니다. PM은 아키텍트가 왔다고 반기면서, [공통]으로 분류된 여러 태스크를 주었습니다. 차츰 [아키텍처팀 == 공통팀] 등식이 성립되었고, 분쟁의 소지가 있는 일이나 책임소재가 불명확한 일은 아키텍처 팀에게 할당되었습니다. 2명이 참여를 했지만, 10명이 해도 다 못할 분량의 일이 생겼습니다. 아키텍트는 이 일을 PM과 진지하게 논의했습니다. PM은 “공통” 일을 한다고 계약했으므로 알아서 끝내어 놓으라고 합니다. 그 SW 아키텍트는 다시는 아키텍트 역할을 하고 싶지 않다고 이야기했습니다.



3. Episode #3 – Not That Architecture

- ✓ 국방 분야 대형 프로젝트가 끝난 후에 사용자들은 시스템 간 연계 부분에 문제가 있다는 사실을 발견했습니다. 시스템 담당자는 현재의 연계방식은 정보를 주고받는 속도도 느리고, 유연성이 부족해 변경 요구를 받아 들이기 힘들다는 이야기를 사용자로부터 들었습니다. 곧바로 시스템 진단을 시작했고, 진단 팀은 연계 방식과 사용 기술, 기술 적용에 문제가 있다는 결론을 내면서, 이것은 전형적인 “아키텍처 설계 부실”이라고 진단을 내렸습니다. 담당자는 이 부분을 개선하기 위한 “아키텍처 개선” 예산을 준비했고, 주변에서 이것은 대규모 시스템의 아키텍처 문제이므로 EA(Enterprise Architecture)로 문제를 해결해야 한다고 조언했습니다. 이듬해 이 EA 프로젝트 공고가 났습니다.

규모가 크거나 정도가 심각한 아키텍처 문제는 모두
Enterprise Architecture로 풀어야 한다...



4. Episode #4 – A system has four architectures

- ✓ 많은 소프트웨어 개발 업체에게 아키텍처 팀을 AA, DA, TA, BA 팀으로 분리하여 운영하고 있습니다. SA 팀을 포함하여 다섯 가지 아키텍처 팀을 운영하는 회사도 있습니다. 아키텍팅 역량 향상은 곧 이 네 가지 영역에서의 역량 향상을 의미합니다. 소프트웨어 개발업체의 주요 활동은 (Software Intensive) 시스템 구축 프로젝트입니다. 시스템 구축에 AA, DA, TA, BA 네 가지 아키텍팅 역량을 동원하고 있습니다. 이전에 존재하던 시스템 분석가, 비즈니스 분석가, 설계자, 소프트웨어 아키텍트 등의 역할이 눈에 띄게 사라지면서 모두 무슨무슨 아키텍트라는 역할로 대체되었습니다.

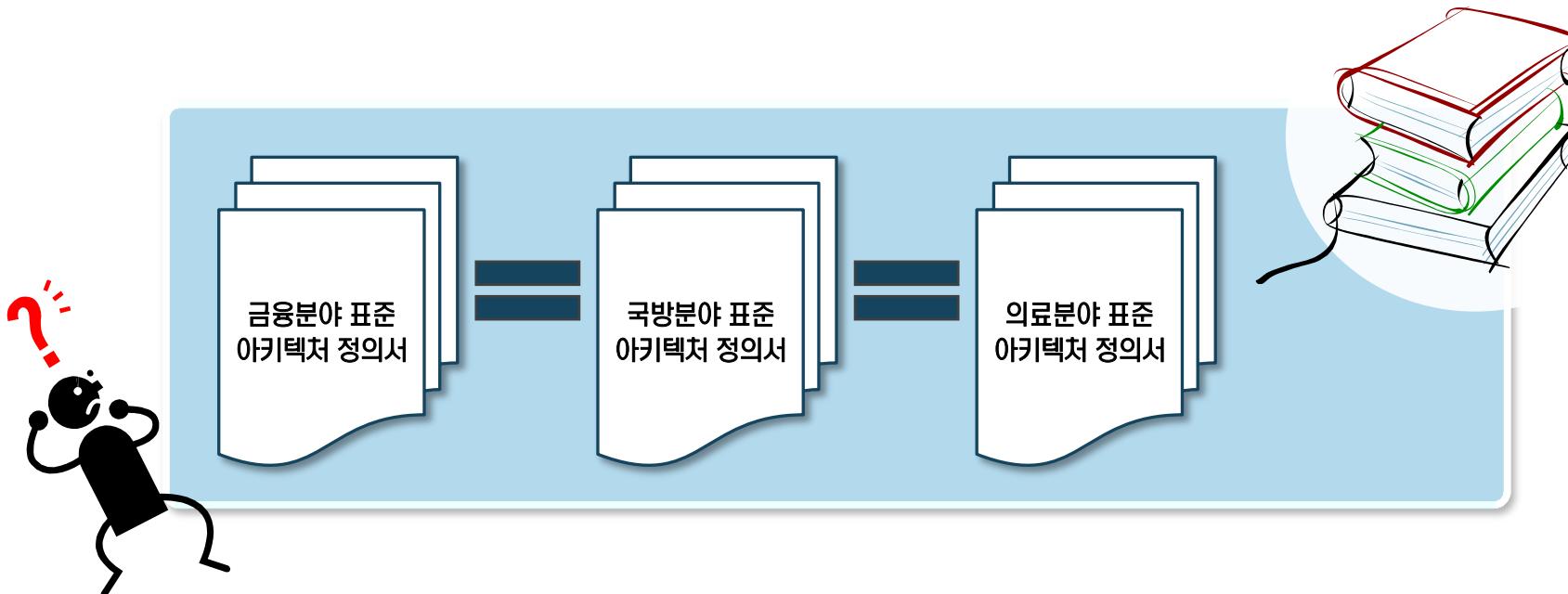


- ❖ AA에서 A가 시스템 구축 영역의 애플리케이션일까요?
- ❖ DA에서 D는 시스템 구축 영역의 데이터일까요?
- ❖ BA에서 B는 비즈니스 프로세스일까요?
- ❖ TA에서 T는 Technical thing일까요?

	AA:Application Architecture
	DA: Data Architecture

5. Episode #5 – Different but same

- ✓ 2000년대 중반, 정통부에서 “산업별 표준 아키텍처 수립” 사업을 발주하였습니다. 사업의 취지는 아키텍처 수립에 비용이 많이 들어가니, 각 산업 분야별로 표준 아키텍처를 수립하고, 그것을 재사용하자는 것이었습니다. 의료, 국방, 교육, 공공 등으로 분류하여 사업자를 선정하였고, 취지대로 각 산업 별로 아키텍처 수립을 한 후 과제를 제출하였습니다. 저마다 두터운 바인더에 분야별 아키텍처를 제출했습니다.
- ✓ 그런데 놀라운 것은 제출한 내용이 대부분 비슷하다는 것이었습니다. 제대로 수행되었다면 5년이 넘게 지난 오늘 그 아키텍처들은 업계의 표준이 되어야 했습니다. 그런데 과제 제출로 모든 것이 끝나버렸습니다.



6. Episode #6 – Unidentified best job

- ✓ 잘 아는 후배가 한 명 찾아왔다. 미국에서 SW 아키텍트가 최고의 Job으로 뽑혔다고 호들갑이다. 그리고는 SW 아키텍트가 되려면 어떤 커리어패스가 필요한지 물어왔다. 회사에서 알아보니 AA, DA, TA, BA라는 것이 있는데, BA는 특정 도메인을 잘 알아야 한다고 하고, DA는 데이터 모델링에 능숙해야 한다고 하고, AA는 프레임워크에 능숙해야 한다고 한다. 도무지 무엇을 어떻게 해야 할지 모른다고 했다.
- ✓ 기업 인사담당자도 마찬가지이다. 소프트웨어 아키텍트가 기업의 경쟁력이라고 한다는데 도대체 그 역할이 어떤 것인지 알수 없다고들 한다.



BEST JOBS IN AMERICA

Money/Payscale.com's list of great careers

2010

Full List

High Pay

Job Growth

Quality of Life

Sectors

1. Software Architect

[Recommend](#) 5K

1 of 100

Next

Top 100 rank: 1

Sector: Information Technology

What they do: Like architects who design buildings, they create the blueprints for software engineers to follow -- and pitch in with programming too. Plus, architects are often called on to work with customers and product managers, and they serve as a link between a company's tech and business staffs.

What's to like: The job is creatively challenging, and engineers with good people skills are liberated from their screens. Salaries are generally higher than for programmers, and a typical day has more variety.

"Some days I'll focus on product strategy, and other days I'll be coding down in the guts of the system," says David Chaiken, 46, of Yahoo in Sunnyvale, Calif., whose current projects include helping the web giant customize content for its 600 million users. Even though programming jobs are moving overseas, the face-to-face aspect of this position helps cement local demand.



PHOTO: DAVID LAURIDSEN

Chaiken, a software engineer for more than two decades, relishes the more collaborative work.

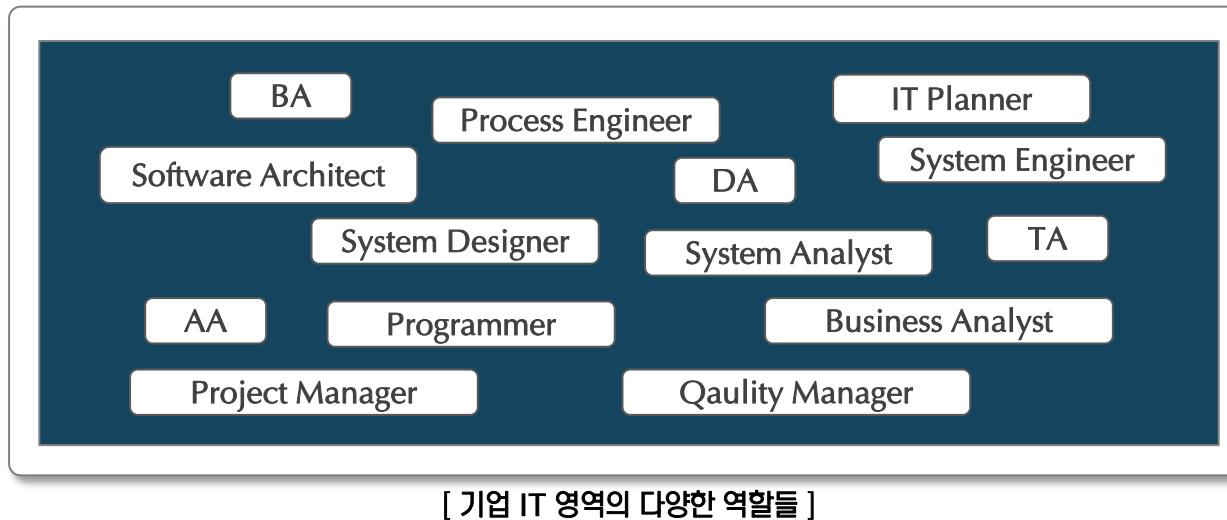


2. EA와 TOGAF

1. Role mixup
2. IT의 네 영역
3. EA
4. EA 논쟁
5. 토의

1. Role mixup

- ✓ 기업 IT 영역에는 다양한 역할들이 존재합니다.
- ✓ 어떤 역할은 IT의 표준에 정의되어 있고, 어떤 역할은 조직에서 정의한 것들입니다.
- ✓ IT 표준에 준하는 역할을 조직에서 적용하는 과정에서 많은 오류와 시행착오가 발생합니다.
- ✓ 역할에 대한 정확한 이해와 적용이 필요합니다.



2. IT의 네 영역

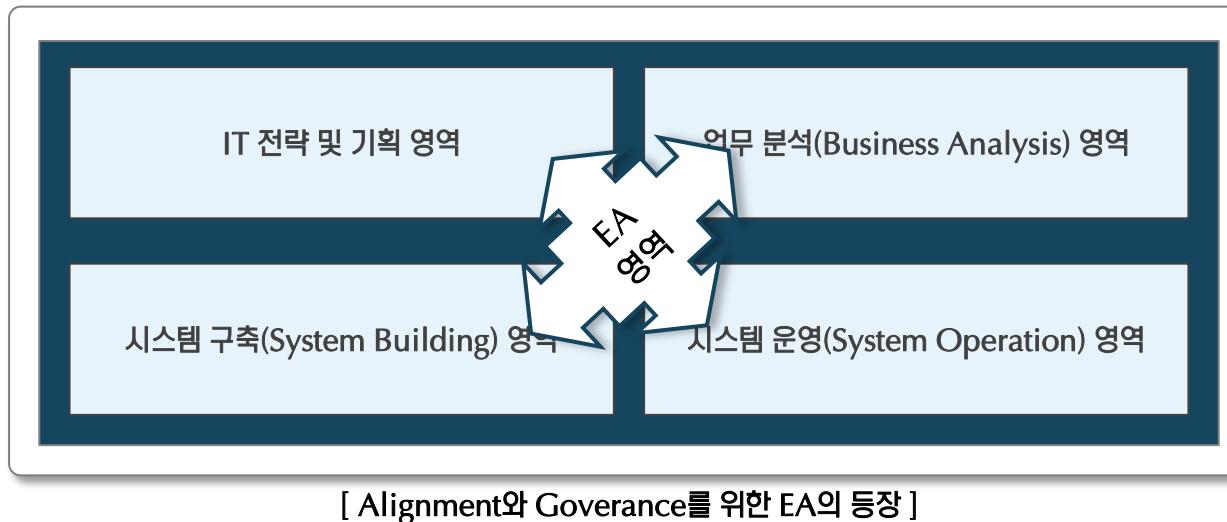
- ✓ 역할 혼란을 줄이기 위해 어떤 분야든 영역을 정의하고 역할을 영역 안에 둡니다.
- ✓ IT 분야는 정보시스템 라이프사이클을 바탕으로 네 가지 영역으로 나눌 수 있습니다.

- IT 전략 및 기획(Strategy & Planning) 영역
- 업무 분석(Business analysis) 영역
- 정보 시스템 구축(Building) 영역
- 정보 시스템 운영(Operation) 영역



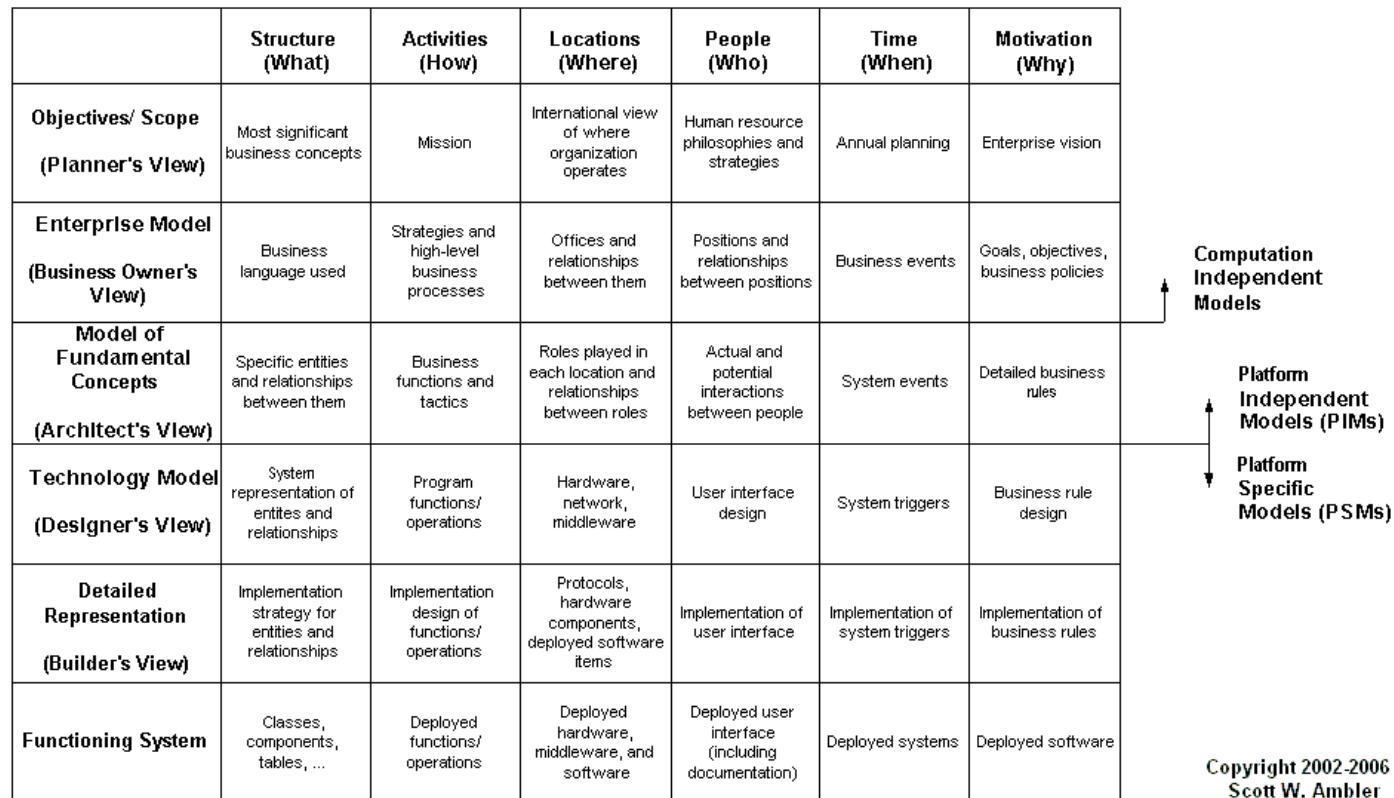
3. EA(1/4) - 등장

- ✓ 기업 경영에서 IT의 비중이 커지면서 표준부재, 중복투자 등으로 인한 비효율성 문제가 드러납니다.
- ✓ 이런 환경에서 조정(alignment)과 거버넌스(governance)를 위해 EA가 등장함
- ✓ 기업의 IT를 높은 수준에서 조망하는 방법을 제시하며 ‘아키텍처’를 유행시킴
- ✓ 관련자료: *Building Bridges between EA and SA(2009, 아키텍트대회, 넥스트리, 송태국)* 참조



3. EA(2/4) - Zachman Framework

- ✓ 기업의 복잡한 IT 영역을 바라보는 방법을 제시할 목적으로 다양한 [Viewing] 프레임워크 등장함
- ✓ 1987년 Zachman은 View와 Perspective를 양축으로 하는 프레임워크를 제시함
- ✓ 기업의 IT를 바라보는 일관성있는 관점을 제공해 주었지만,
- ✓ 서로 다른 대상은 서로 다른 관점으로 바라보아야 함을 고려하지 않았으므로 실효성 문제에 부딪힘



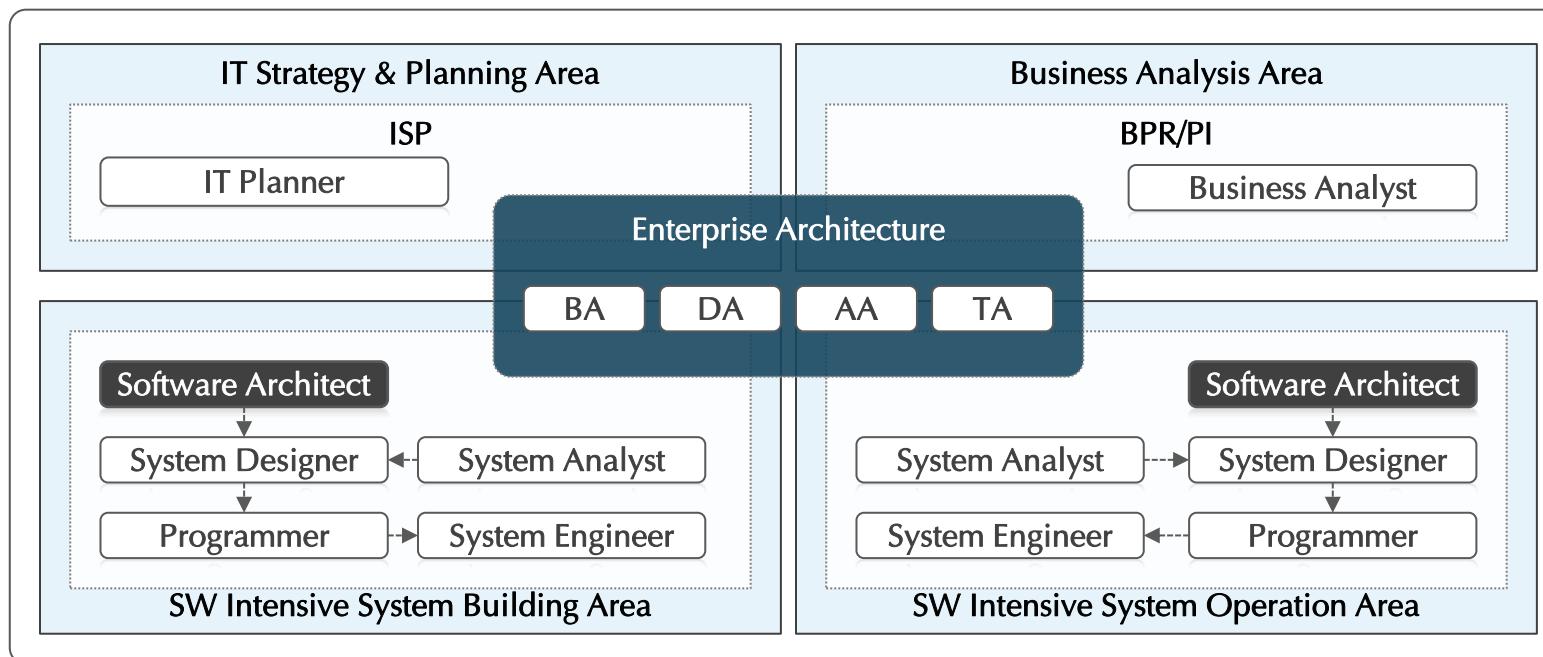
3. EA(3/4) - 범정부 EA 프레임워크

- ✓ 범정부 프레임워크의 BA 부분은 조직, 업무 두 가지를 중심으로 전개함
- ✓ 기술은 인프라로 확장함으로써 SW 아키텍처 설계 영역과 중복이 발생함
- ✓ 보안 아키텍처를 EA 영역으로 끌어올림으로써 SW 아키텍처 영역과 중첩

View Perspective	BA	AA	DA	Technology/Infra Arch.	Security Arch.
CEO	<ul style="list-style-type: none">• 조직구성도/정의서• 업무구성도/정의서	<ul style="list-style-type: none">• 응용 구성도/정의서	<ul style="list-style-type: none">• 데이터 구성도/정의서	<ul style="list-style-type: none">• 기반구조 구성도/정의서	<ul style="list-style-type: none">• 보안정책• 보안구성도/정의서
관리자	<ul style="list-style-type: none">• 업무기능관계도/기술서• 업무기능분할도/기술서	<ul style="list-style-type: none">• 응용 관계도/기술서• 응용기능분할도/기술서	<ul style="list-style-type: none">• 개념데이터 관계도/기술서• 데이터교환 기술서	<ul style="list-style-type: none">• 기반구조 관계도/기술서	<ul style="list-style-type: none">• 보안관계도/기술서
설계자	<ul style="list-style-type: none">• 업무절차 설계도/설계서	<ul style="list-style-type: none">• 응용기능설계도/설계서	<ul style="list-style-type: none">• 논리데이터 모델• 데이터교환 설계서	<ul style="list-style-type: none">• 기반구조설계도/설계서	<ul style="list-style-type: none">• 관리보안설계서• 물리보안설계서• 기술보안설계서
개발자		<ul style="list-style-type: none">• 응용프로그램 목록	<ul style="list-style-type: none">• 물리데이터 모델	<ul style="list-style-type: none">• 제품목록	<ul style="list-style-type: none">• 보안 메뉴얼

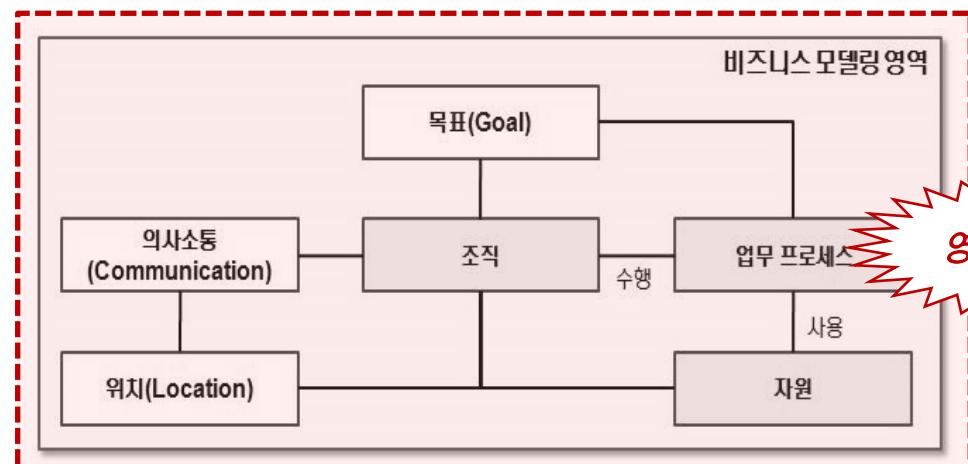
3. EA(4/4) - Coverage

- ✓ Alignment와 Governance를 위해 네 가지 영역에 걸쳐져 있음
- ✓ 각 영역 스스로 잘 조정이 되고 관리가 체계적으로 된다면 EA의 역할이 줄어들 수 있음
- ✓ 따라서, 일부 영역과는 Alignment 보다는 오히려 Collision이 발생함



4. EA 논쟁(1/10) - BA와 AA 충돌

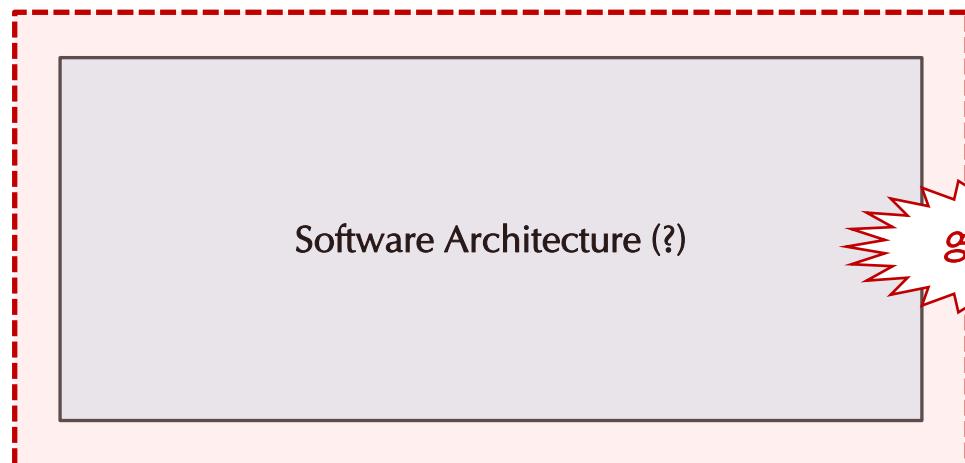
- ✓ 비즈니스 모델링 또는 Business Analysis 영역에서 여섯 가지 분석 또는 모델링 대상이 있음
- ✓ EA의 Business Architecture 정의는 조직과 업무를 중심으로 분석을 수행함
- ✓ 두 영역의 활동 내용에 중복이 발생함
- ✓ EA-BA 활동은 BPR/PI 활동과는 다르다고 하나 활동 내용이 중첩되고 있음



View Perspective	BA	AA
CEO	• 조직구성도/정의서 • 업무구성도/정의서	• 응용 구성도/정의서
관리자	• 업무기능관계도/기술서 • 업무기능분할도/기술서	• 응용 관계도/기술서 • 응용기능분할도/기술서
설계자	• 업무절차 설계도/설계서	• 응용기능설계도/설계서
개발자		• 응용프로그램 목록

4. EA 논쟁(2/10) - AA와 SA 충돌

- ✓ EA-AA와 SA 사이에는 활동의 충돌이라기보다는 역할의 충돌이 발생함
- ✓ 일부 조직이나 프로젝트에서는 SA 역할을 AA 역할로 대체하고 있음
- ✓ 이 경우는 충돌이라는 용어보다는 혼란이라는 용어가 더 적절할 것임
- ✓ 충돌과 혼란을 방지하기 위해 EA를 표준기반으로 정확히 이해할 필요가 있음



View Perspective	BA	AA
CEO	• 조직구성도/정의서 • 업무구성도/정의서	• 응용 구성도/정의서
관리자	• 업무기능관계도/기술서 • 업무기능분할도/기술서	• 응용 관계도/기술서 • 응용기능분할도/기술서
설계자	• 업무절차 설계도/설계서	• 응용기능설계도/설계서
개발자		• 응용프로그램 목록

4. EA 논쟁(3/10) - TOGAF

- ✓ IT 업계 표준 [Enterprise] 아키텍처 프레임워크임
- ✓ 오픈그룹의 아키텍처 포럼에서 지속적으로 발전시키고 있음
- ✓ 현재 Version 9까지 릴리즈하였음
- ✓ www.opengroup.org/togaf



4. EA 논쟁(4/10) - EA-AA at TOGAF

- ✓ TOGAF에서는 Application에 대해서 다음과 같이 말하고 있습니다.
- ✓ 조직에서 정의하고 있는 Application Architecture의 의미와 비교하여 봅니다.
- ✓ 조직에서 정의하고 있는 Application Architect의 역할을 검토해 봅니다.

AA의 목표는 데이터를 처리하고 업무를 지원하는 애플리케이션이 주로 어떤 종류가 있는지 정의하는 것이다. 이러한 활동은 애플리케이션 시스템 설계와 관련이 없다는 사실에 유의하여야 한다. 목표(goal)는 기업과 관련된 애플리케이션이 어떤 것이 있는가를 정의하고, 데이터를 관리하고 직원들이 정보를 사용하도록 하기 위해서 애플리케이션이 필요로 하는 것이 무엇인가를 정의하는 것이다. 애플리케이션은 컴퓨터 시스템이 아니라 DA의 데이터를 관리하고 BA의 업무 기능을 지원하는 역량의 논리적인 그룹이다. 애플리케이션과 애플리케이션의 역량은 특정 기술에 대한 고려 없이도 정의된다.



TOGAF Version 9

4. EA 논쟁(5/10) - EA-DA at TOGAF

- ✓ TOGAF에서는 Data Architecture에 대해서 다음과 같이 이야기 하고 있습니다.
- ✓ 데이터는 우리가 수행했던 논리 데이터 모델, 물리 데이터 모델 등과 관련이 없다고 합니다.
- ✓ DA에서 말하는 데이터는 데이터 엔티티 또는 비즈니스 엔티티, 비즈니스 자원 등을 의미합니다.

10.1 목표

DA의 목표는 비즈니스를 지원하는데 필요한 데이터의 주요 타입과 소스를 다음과 같은 수준으로 정의하는 것이다.

- ❖ 이해관계자가 이해할 수 있어야 함
- ❖ 완전성과 일관성을 갖추어야 함
- ❖ 안정적이어야 함

이러한 활동은 데이터베이스 설계와 관련이 없음을 이해하는 것이 중요하다. 목표는 기업과 관련된 데이터 엔티티를 정의하는 것이지, 논리 또는 물리 데이터 저장 시스템을 설계하는 것이 아니다.



TOGAF Version 9

4. EA 논쟁(6/10) - EA-TA at TOGAF

- ✓ Technology Architecture의 핵심은 AA를 실현할 대상 기술 포트폴리오를 작성하는 것임
- ✓ 보안 아키텍처 설계, 인프라 아키텍처 설계와 같은 [설계 활동]은 TA 범위가 아님
- ✓ “TA는 목표 아키텍처로 가는 로드맵을 상세화하고...”

TA 단계에서는 AA단계에서 정의한 애플리케이션을 기술 컴포넌트 세트로 매핑 할 연결고리를 찾는다. 기술 컴포넌트란 시장에서 얻을 수 있거나 조직의 기술 플랫폼 안으로 들일 수 있는 소프트웨어와 하드웨어 컴포넌트를 말한다.[중략]

TA는 **기술 포트폴리오**에 대한 목표 뷰와 기준선을 정의할 것이며, 목표 아키텍처로 가는 로드맵을 상세화한다.... [중략]

TA는 아키텍처 정보를 아우르며 따라서 특정 이전 시나리오를 위한 비용평가 작업을 지원한다.



TOGAF Version 9

4. EA 논쟁(7/10) - EA-BA at TOGAF

- ✓ Technology Architecture의 핵심은 AA를 실현할 대상 기술 포트폴리오를 작성하는 것임
- ✓ 보안 아키텍처 설계, 인프라 아키텍처 설계와 같은 [설계 활동]은 TA 범위가 아님
- ✓ “TA는 목표 아키텍처로 가는 로드맵을 상세화하고...”

BA의 목표:

...

다음을 포함하여 목표 비즈니스 아키텍처를 서술함;

비즈니스 원칙과 비즈니스 목표에 근거하여 제품과 서비스 전략, 조직과 업무 프로세스, 정보, 그리고 비즈니스 환경의 지리적인 특성 등을 서술한다.

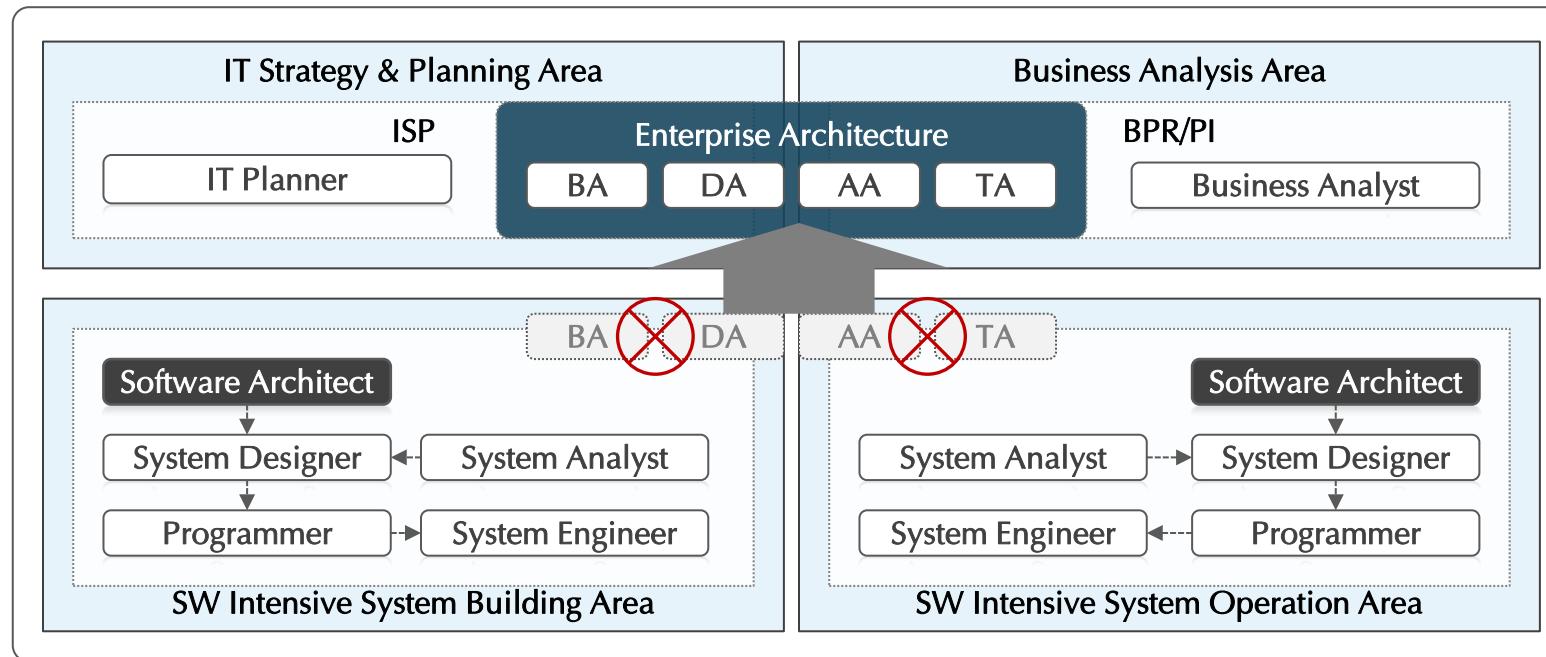
...



TOGAF Version 9

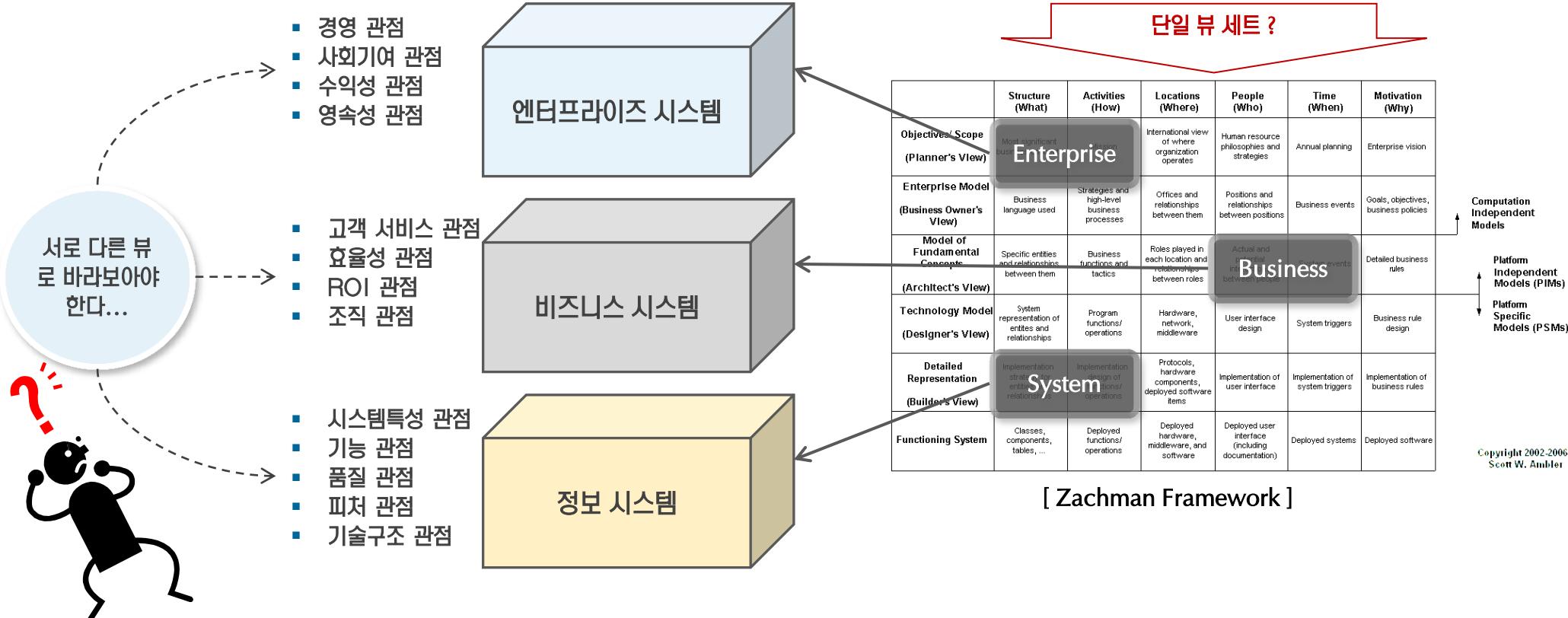
4. EA 논쟁(8/10) - EA 포지션

- ✓ 시스템 구축 영역은 다양한 표준화 활동을 통해서 EA에서 이야기하는 Alignment 대상이 거의 없음
- ✓ 시스템 영역의 소프트웨어 아키텍트는 전사 시스템 뷰를 가지고 설계를 수행함
- ✓ “EA는 도시 설계이고 SA는 건축이다”를 주제로 토의합시다.
- ✓ EA를 도시 설계라고 한다면, ISP는 무엇이고, BPR은 무엇이라고 해야 하는가?



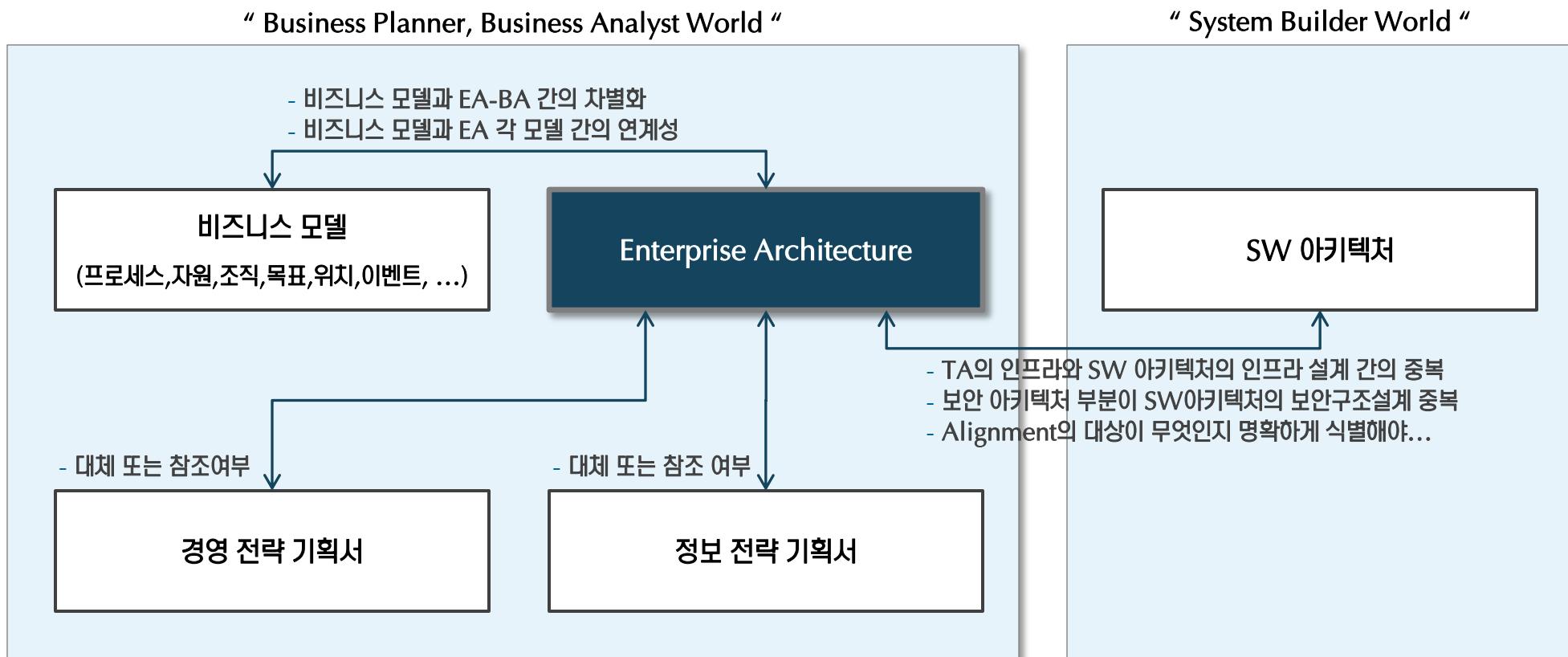
4. EA 논쟁(9/10) - Zachman Framework Review

- ✓ 서로 다른 대상은 서로 다른 뷰포인트를 가지고 바라보아야 함
- ✓ 엔터프라이즈 시스템은 경영, 사회기여, 수익의 관점이 필요하고, 비즈니스 시스템은 효율성 관점이 필요하며, 정보 시스템은 기능과 품질, 시스템 특성(aspect) 관점이 필요함
- ✓ 따라서, 세 가지 시스템을 하나의 틀에 놓고 동일한 관점으로 바라보는 것은 타당성이 부족함



4. EA 논쟁(10/10) - EA Review

- ✓ 기존에 다양하게 수행하던 Planner 관점의 작업들과 관계를 명확히 해야함
 - PI 또는 BPR의 비즈니스 모델, 경영 전략, 정보 전략 등
- ✓ EA는 Planner 뷰를 읽지 말고 Planner 관점의 작업 수행
 - 논리데이터 모델, 물리 데이터 모델, 보안 구조, 인프라 구조 등은 Builder 뷰의 작업임



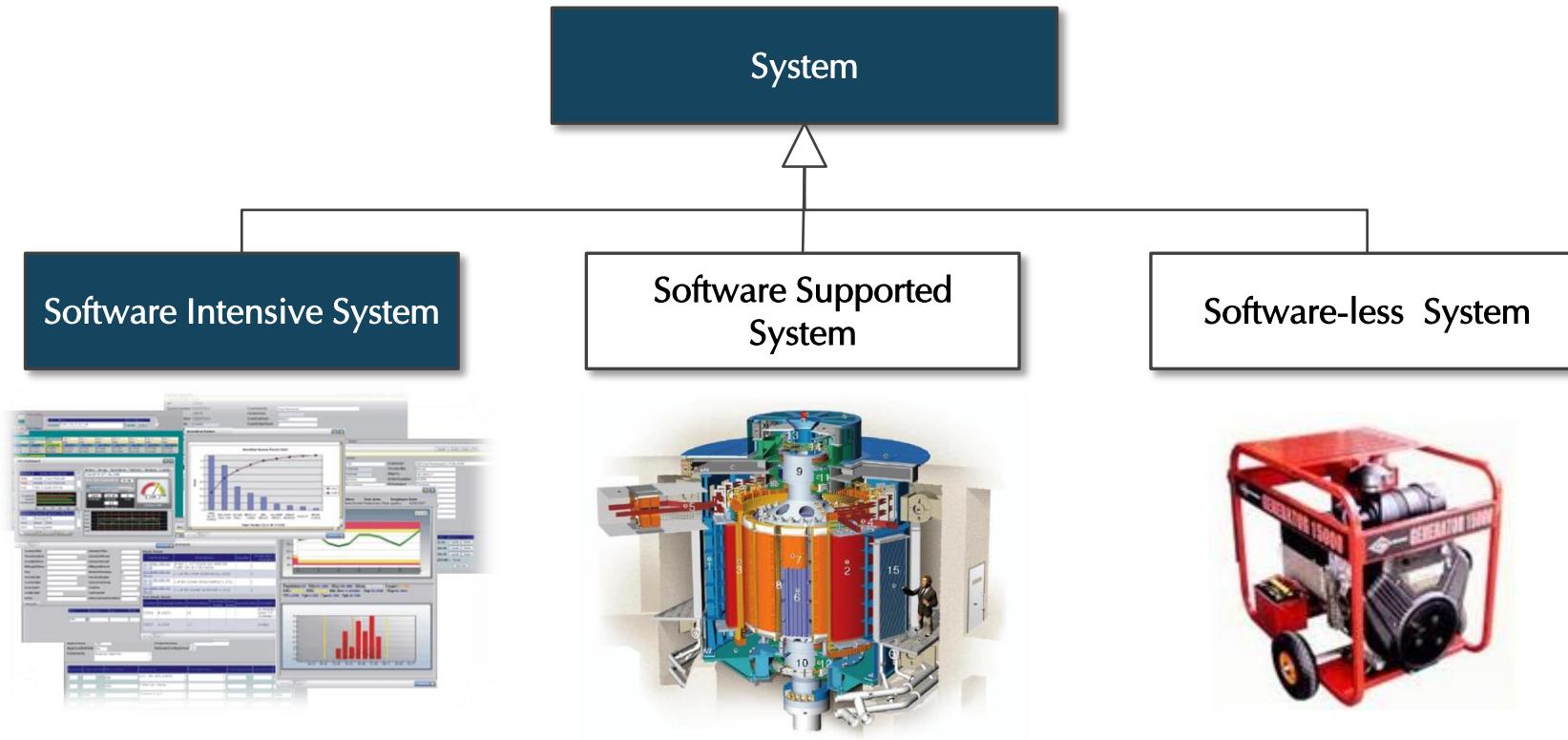


3. Quality Attributes

1. 시스템
2. 소프트웨어 Intensive 시스템
3. 기능, 품질, 피처
4. 기능과 품질 관계
5. 시스템 품질
6. 품질속성 시나리오
7. 품질속성 시나리오 효율성
8. 품질속성과 프로세스
9. 요약
10. 토의

1. 시스템

- ✓ 다양한 유형의 시스템이 있으며, 그 다양성으로 인해 분류체계를 제시하기 어려움
- ✓ 소프트웨어를 중심으로, 전혀 없거나, 또는 지원 역할을 하거나, 주(main)가 되는 시스템으로 분류할 수 있음
- ✓ 소프트웨어 아키텍처 설계 대상이 되는 시스템은 소프트웨어가 주가 되는 시스템임



[돌극형 발전기, 출처:두산중공업홈페이지]

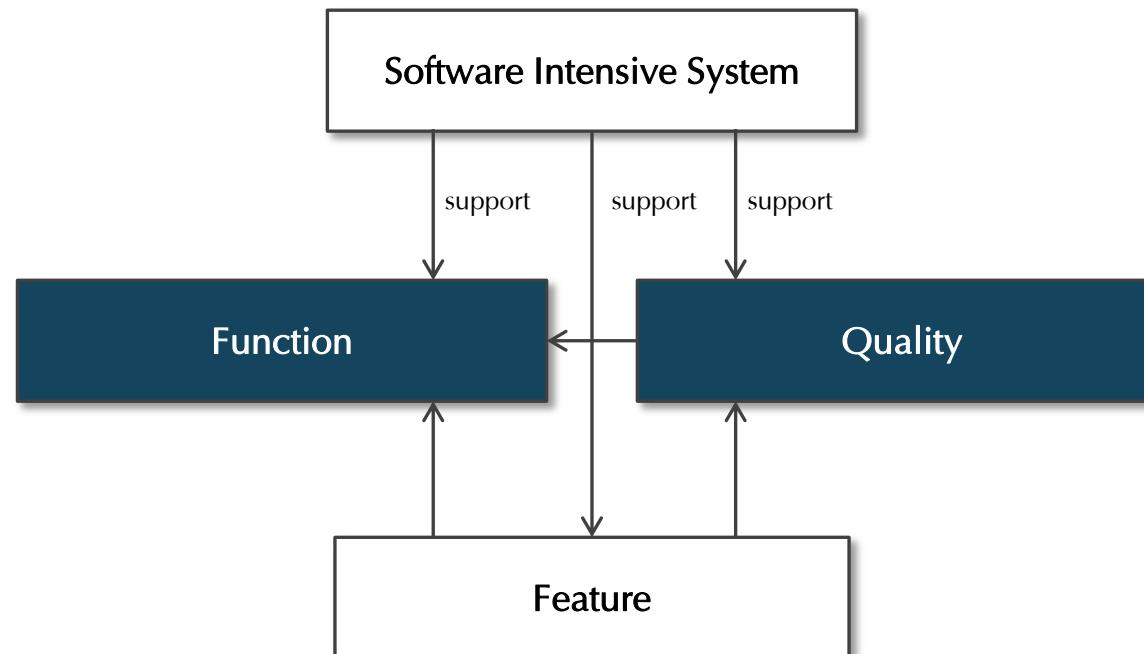
2. 소프트웨어 Intensive 시스템

- ✓ 라이트 형제가 만든 인류 최초 비행기인 Flyer 1은 Software-less 시스템이었음
- ✓ 현재 생산되는 보잉기 원가의 40%는 소프트웨어 비용이며, 기능의 80%는 소프트웨어가 담당하고 있음
- ✓ 시스템에서 소프트웨어가 차지하는 비용과 역할은 점차 증가하고 있음



3. 기능, 품질, 피처

- ✓ 시스템은 사용자에게 기능(function)을 제공하고, 기능은 사용자가 요구하는 수준의 품질 범위 안에서 제공되어야 함
- ✓ 품질은 그 자체로써 의미가 없으며, 항상 기능을 통해서만 가치를 보여줄 수 있음
- ✓ Feature는 시스템을 제공하는, 강조할 만한 것으로 기능과 품질을 모두 포함하는 개념임



4. 기능과 품질 관계

- ✓ 기능과 품질은 직교(orthogonal) 관계임
- ✓ 다리를 하나의 시스템으로 보았을 때, 다리의 상판은 기능을, 교각은 품질을 결정함
- ✓ 교각을 먼저 건설하듯이, 시스템 설계에서 품질에 대한 설계가 우선되어야 함, 물론 기능에 대한 이해가 선행되어야 함



프랑스 남부 미요에 위치한 다리로 산악지대를 흐르는 다른 강 계곡을 가로지름, 2004년에 공식 개통함

5. 시스템 품질 [1/6] – 종류

- ✓ 소프트웨어 Intensive 시스템의 품질에는 어떤 것들이 있는가?
- ✓ 조직마다 서로 용어로 품질을 표현하고 있으며, 각 품질 항목을 시스템의 품질 속성(Quality Attribute)라고 함
- ✓ Embedded 시스템에서는 SEI에서 제시하는 품질 속성을 사용하며, 엔터프라이즈 시스템에서는 TOGAF를 참조함

시스템의 품질 속성(From SEI)

- 가용성(availability)
- 변경 용이성(modifiability)
- 수행성능(performance)
- 보안성(security)
- 테스트 용이성(testability)
- 사용성(usability)
- 상호운영성(interoperability)
- 이식성(Portability)
- 범위성(Scalability)

조직에서
정의한
품질속성은 ?

비즈니스의 품질속성 (From SEI)

- 시장 출하 시기(time to market)
- 비용과 장점(cost and benefit)
- 시스템의 예정 생명주기(projected lifetime of the system)
- 목표 시장(targeted market)
- 최초 공개 일정(rollout schedule)
- 기존 시스템과의 통합(integration with legacy systems)

시스템의 품질 속성(From TOGAF)

Availability

- manageability
- serviceability
- performance
- reliability
- recoverability
- locatability

Assurance:

- security
- integrity
- credibility

Usability

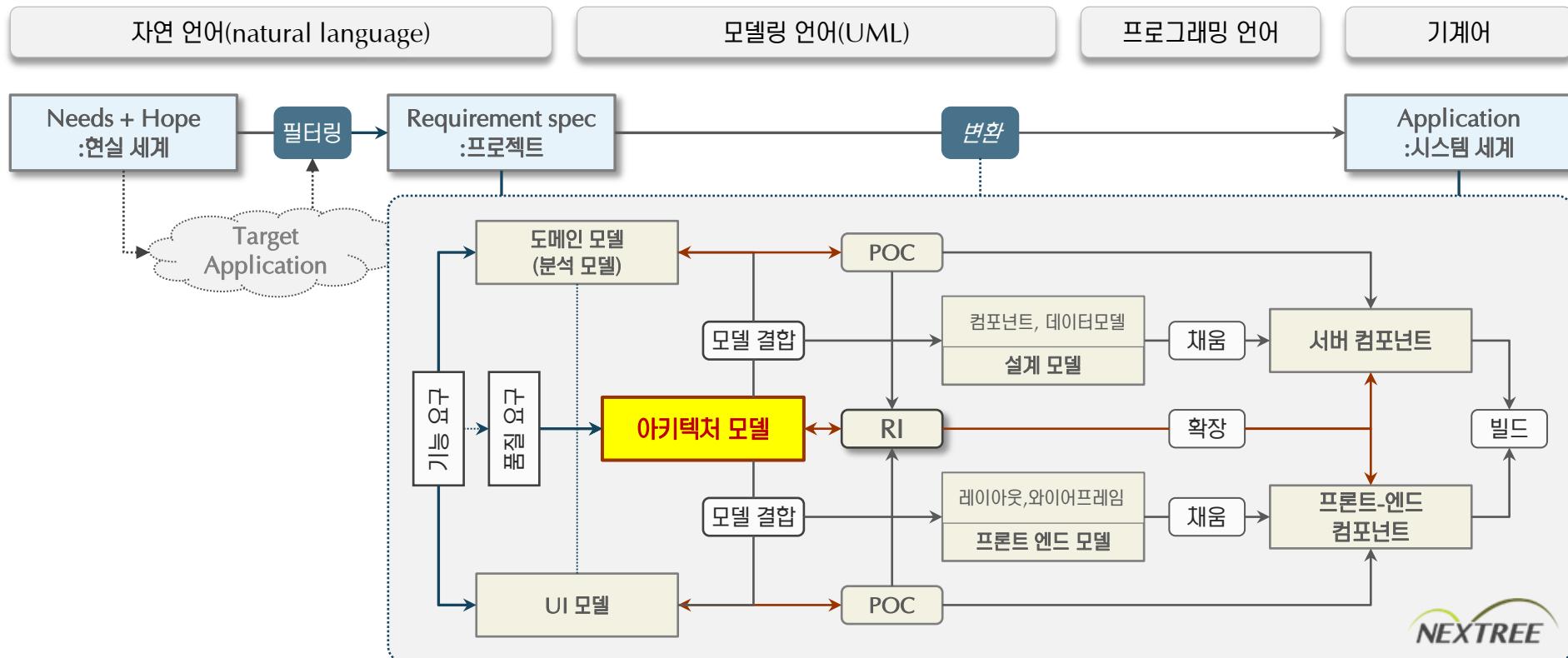
- International operation

Adaptability

- interoperability
- scalability
- portability
- extensibility

5. 시스템 개발 절차 (2/6)

- ✓ 개발 프로세스의 목표는 현실 세계의 필요를 충족하는데 필요한 시스템을 구축하는 것입니다.
- ✓ 개발이란 현실 세계에 존재하는 개념들을 필터링, 변환, 번역 과정을 거쳐서 작은 시스템 무대로 이동시키는 것입니다.
- ✓ 일련의 과정에서 서로 다른 언어(자연어 → 모델링 언어 → 프로그래밍 언어)로의 연속적인 변환이 이루어집니다.
- ✓ 개발 기술과 프로세스는 소프트웨어 개발의 특성을 반영하여야 합니다. ← 소통이 중요한 이유입니다.



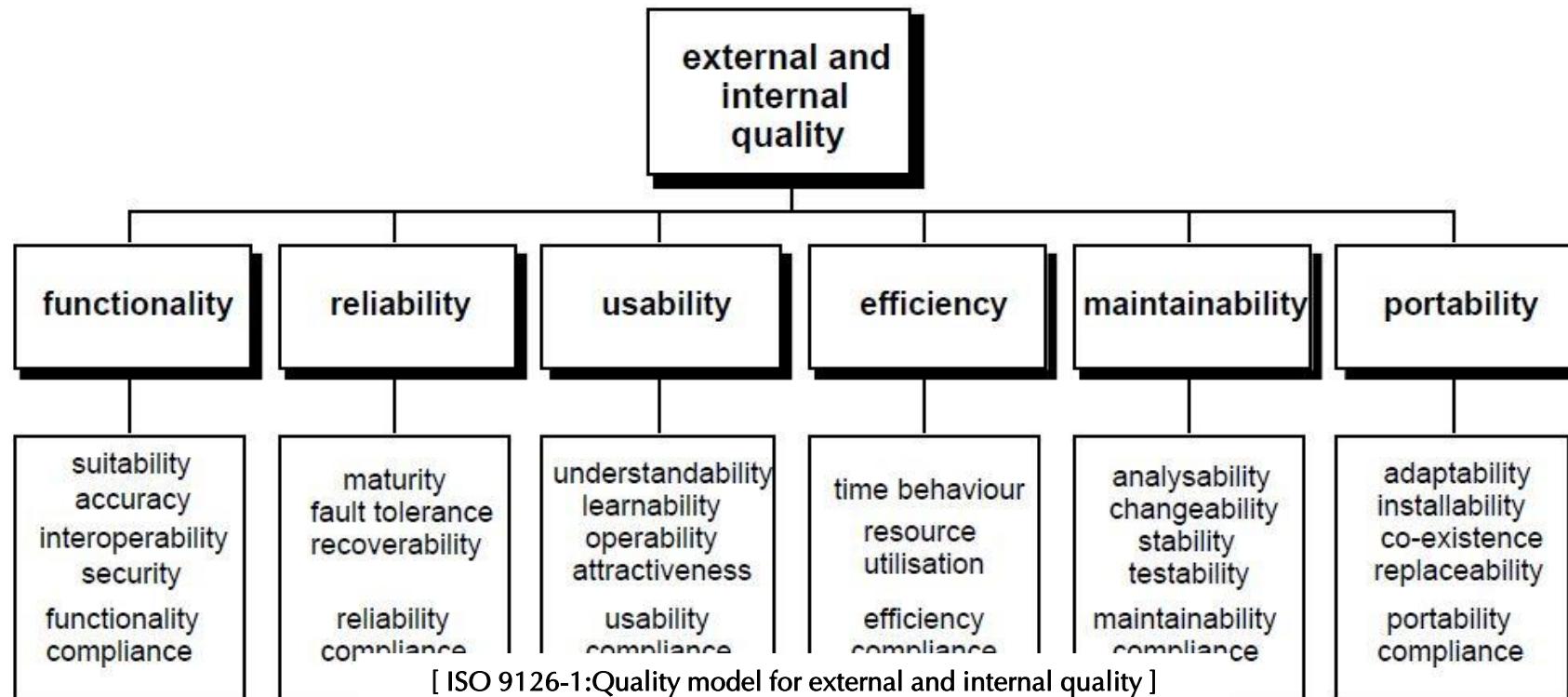
5. 시스템 품질 [3/6] – 표현

- ✓ 품질은 기능을 통해 확인되고 검증될 뿐만 아니라 기능 별로 서로 다른 수준의 품질을 요구할 수 있음
- ✓ 품질수준을 표현하는 단위를 표현하기 위한 ISO-9126의 기준도 있지만, 기업용 소프트웨어에서 적용사례는 찾기 어려움
- ✓ 품질 요구수준은 조직이나 프로젝트 별로 적절한 수준의 표준을 정의한다면 매우 유용할 것임

구분			경매등록	경매물품조회	입찰	낙찰금액지불	경매마감
품질속성	명세내용	단위					
가용성	중요도	상중하	중	상	상	중	상
	허용범위	다운시간/년	30H	10H	20H	30H	1H
성능	중요도	상중하	하	상	상	하	상
	허용범위	초(sec)	3	1	1	5	1
범위성	중요도	상중하	하	상	중	하	하
	허용범위	동시 사용자수	5천	10만명	1만명	5천명	5천명
보안	중요도	상중하	중	하	중	상	하
	허용범위	인증/인가/암호화	인증/인가	인증	인증/인가	인증/인가/암호화	해당사항없음
확장성	중요도	상중하	중	상	중	중	하
	허용범위	작업일	3	3	3	5	3
관리성	중요도		Cross-cutting 이슈				
	허용범위						
사용성	중요도	상중하	중	상	상	중	하
	허용범위	사용자만족도	80%	95%	90%	80%	해당사항없음

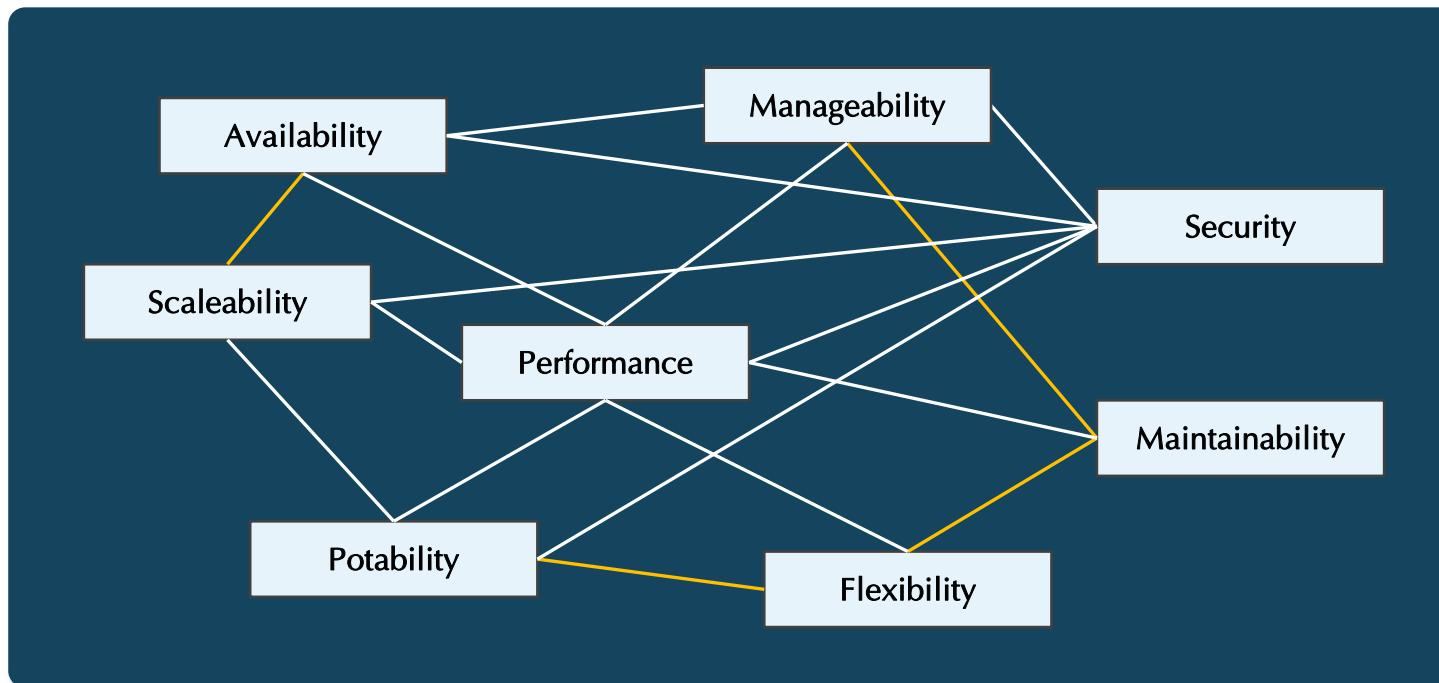
5. 시스템 품질 [4/6] – 표준

- ✓ ISO-9126에서 소프트웨어의 품질 유형과 특성을 정의하였음
- ✓ 소프트웨어의 규모가 작지만 매우 중요한 시스템(mission-critical)에서 ISO-9126을 준수하기가 더 쉬움
- ✓ 표준 기반의 품질 목표를 달성하기 위해서는 많은 시간과 비용이 수반되므로, 시스템에 맞는 기준을 찾아서 적용해야 함



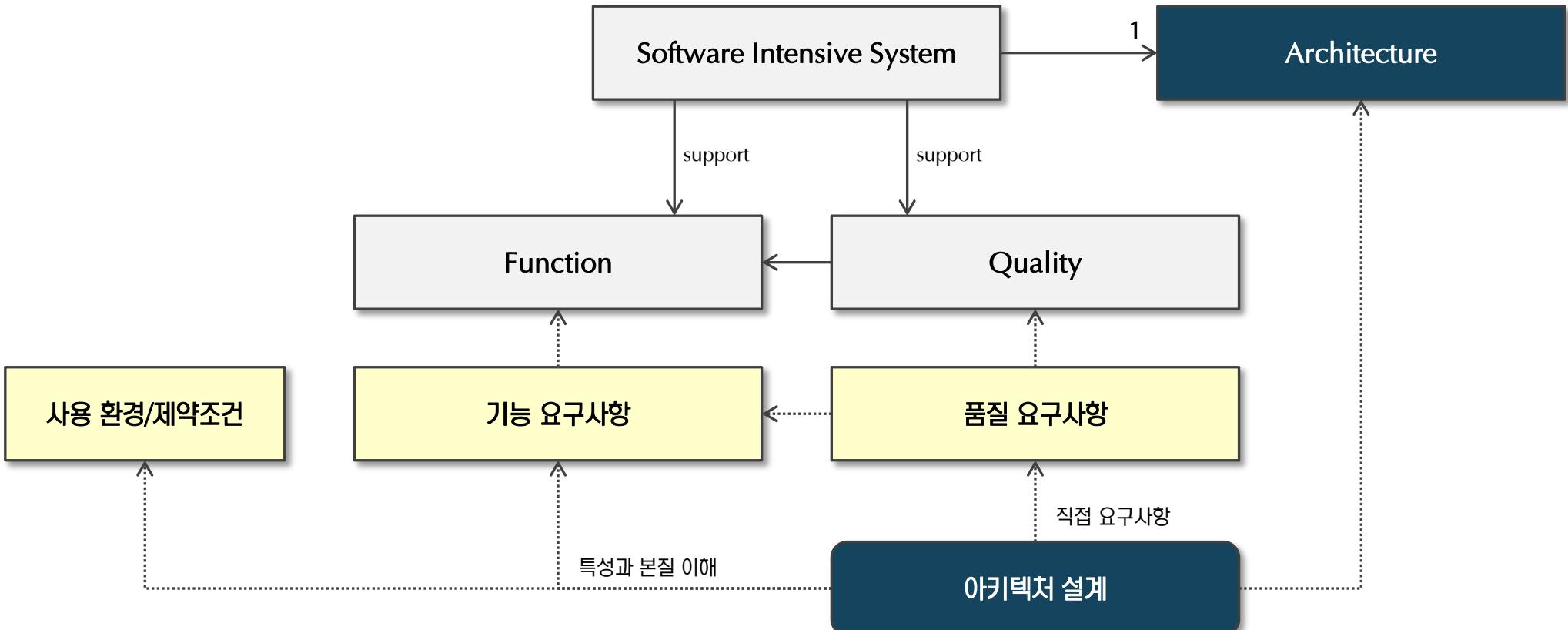
5. 시스템 품질 [5/6] – 상관관계

- ✓ 시스템의 품질 속성 상호 간에는 대체로 부정(negative) 관계가 형성됨, 노란색은 긍정(positive)관계를 표현함
- ✓ 예를 들면, 가용성(availability)을 높이기 위해 서버를 늘리면, 성능(performance)과 보안(security)에 문제가 발생함
- ✓ 품질 수준을 동시에 모두 높이려 하면 시간과 비용의 상승을 피할 수 없음



5. 시스템 품질 [6/6] – 아키텍처

- ✓ 제약조건이 있는 환경에서 원활한 운영을 위해서 시스템은 품질목표를 가짐
- ✓ 시스템의 품질 목표 달성을 위해 아키텍처 설계를 하고, 아키텍처 설계는 다양한 제약조건을 극복하여야 함
- ✓ 품질은 기능을 통해 표현되므로, 아키텍처 설계 시 기능요구사항과 사용환경에 대한 이해가 선행되어야 함



6. 품질속성 시나리오 (1/5) – 개요

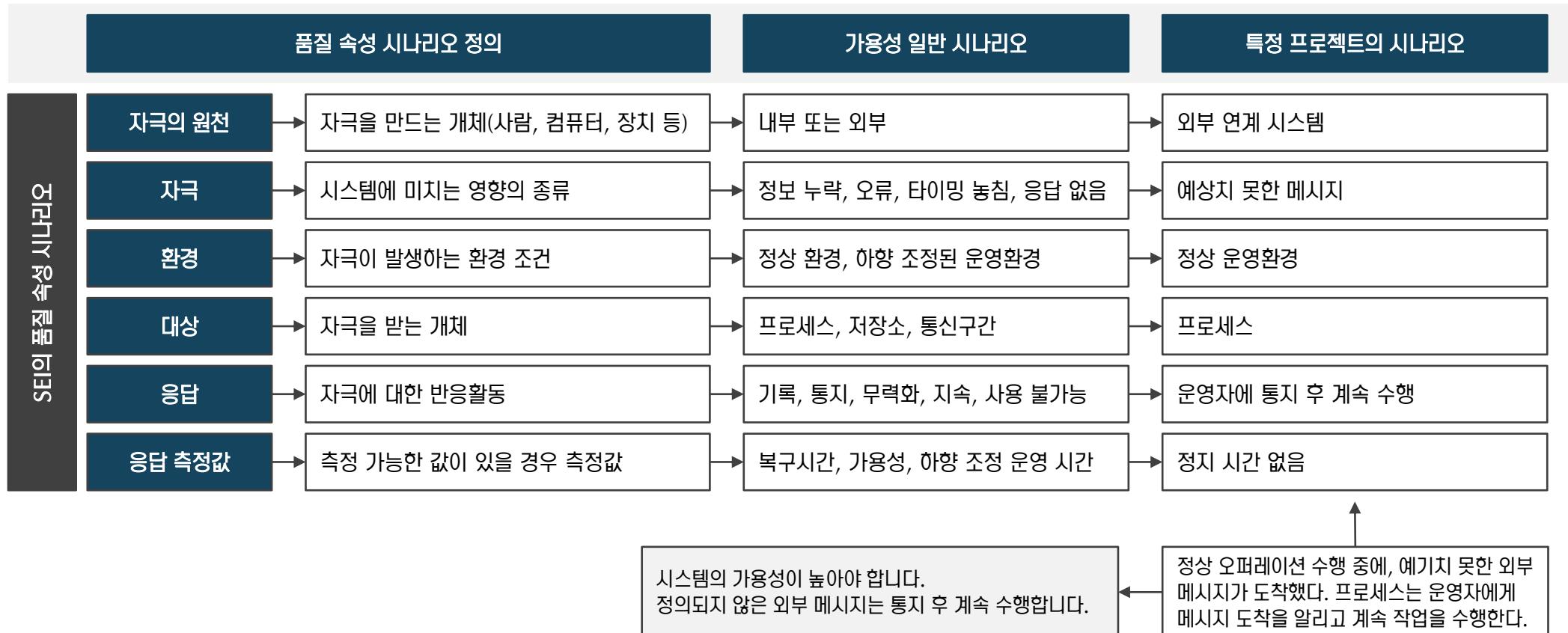
- ✓ 우리의 품질 표현방식은 이해관계자들과 소통하기에 충분한 정보를 담고 있는가?
- ✓ 품질이 중요하다면 보다 명확한 기준이 필요하지 않은가?
- ✓ 품질의 특성상 상세함의 표현에는 다양한 정도의 차이가 있을 수 있음



- ✓ 경고등은 0.1초 내에 동작하여야 한다.
- ✓ SMS 메시지는 반드시 전송되어야 한다.
- ✓ 고객등록은 24시간 365일 가능해야 한다.
- ✓ 비밀번호 확인은 신속히 이루어져야 한다.

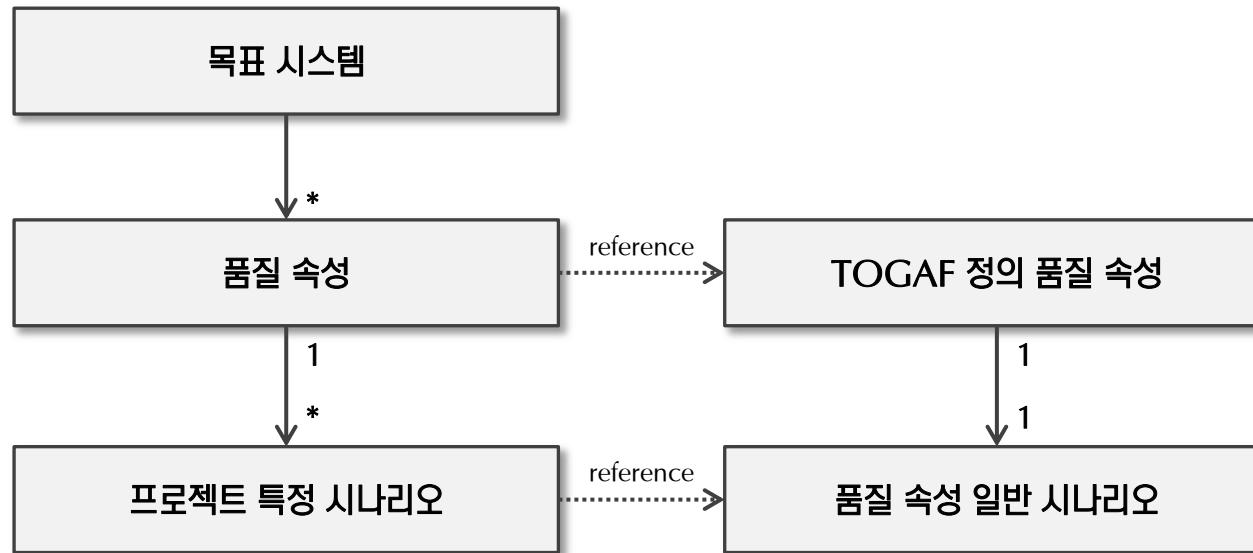
6. 품질속성 시나리오 (2/5) – 개요

- ✓ 품질을 어떻게 구체적으로 표현하는가에 대한 문제에 대한 대답이 필요함
- ✓ 품질을 여섯 가지 요소를 기준으로 표현하는 품질 속성 시나리오가 있음
- ✓ 품질 속성 시나리오는 품질 요구사항을 보다 명확하게 표현하는 틀을 제공함



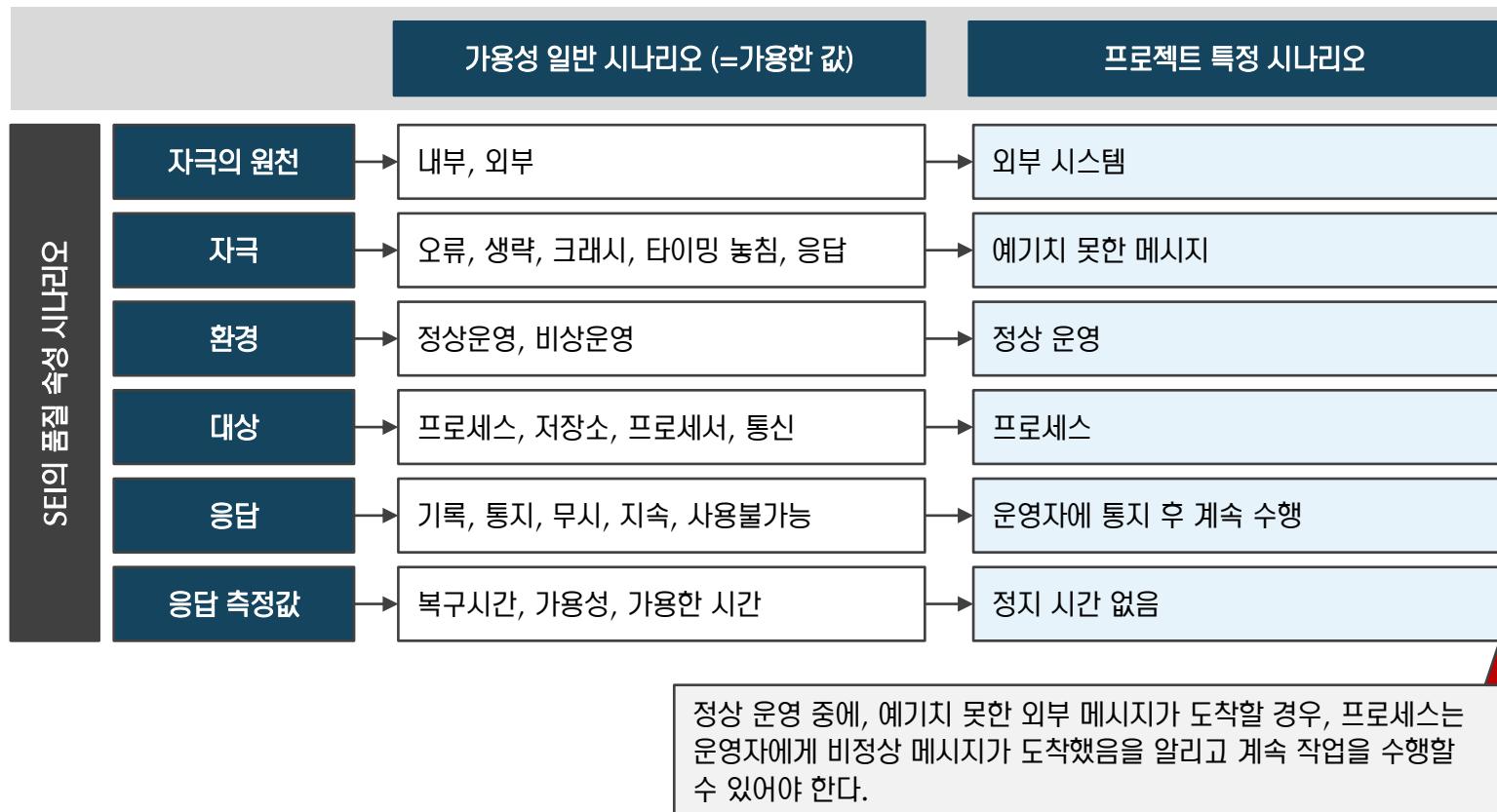
6. 품질속성 시나리오 (3/5) – 개념모델

- ✓ 목표 시스템은 조직에서 참조하거나 미리 정의해 놓은 품질 속성 여러 개를 가질 수 있음
- ✓ 품질 속성 일반 시나리오는 프로젝트 특정 시나리오를 구성할 때, 선택 가능한 값들을 모아 놓은 것임
- ✓ 품질속성에 대한 프로젝트 특정 시나리오의 개수에는 제한이 없지만, 반드시 필요한 시나리오로 필요한 최소한 구성함



6. 품질속성 시나리오 (4/5) – 예제 1

- ✓ 가용성은 시스템의 실패, 그리고 실패와 관련된 결과를 다룸
- ✓ 가용성(availability)은 필요할 때 운영 가능한 확률임, 따라서 가용성은 $(\text{실패평균시간}/(\text{실패평균시간}+\text{복구평균시간}))$ 임
- ✓ 관련 분야는 실패감지, 실패 빈도, 실패 결과, 실패 허용, 실패 방지, 실패 시 통지 등이 있음



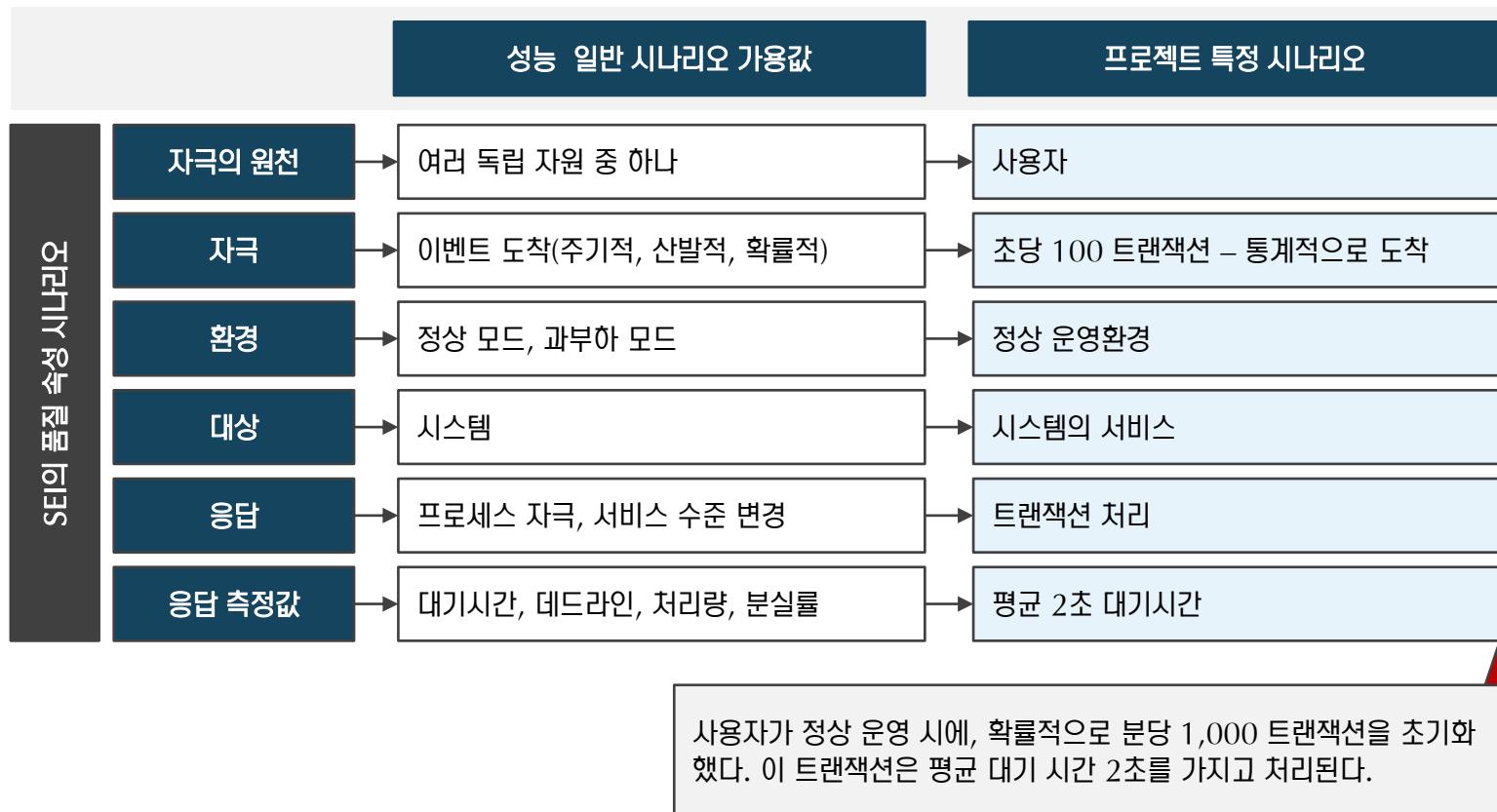
6. 품질속성 시나리오 (4/5) – 예제 2

- ✓ 변경 용이성 일반 시나리오는 프로젝트 특정 시나리오를 구성할 때 선택 가능한 값들을 가지고 있음
- ✓ 프로젝트 특정 시나리오는 문장으로 표현할 수도 있지만, 6가지 정보요소로 표현하면 보다 정확한 소통이 가능함
- ✓ 여섯 가지 항목이 항상 모두 필요한 것은 아니며, 때로는 당연하거나 중복이 될 수도 있는 정보로 채울 수 있음



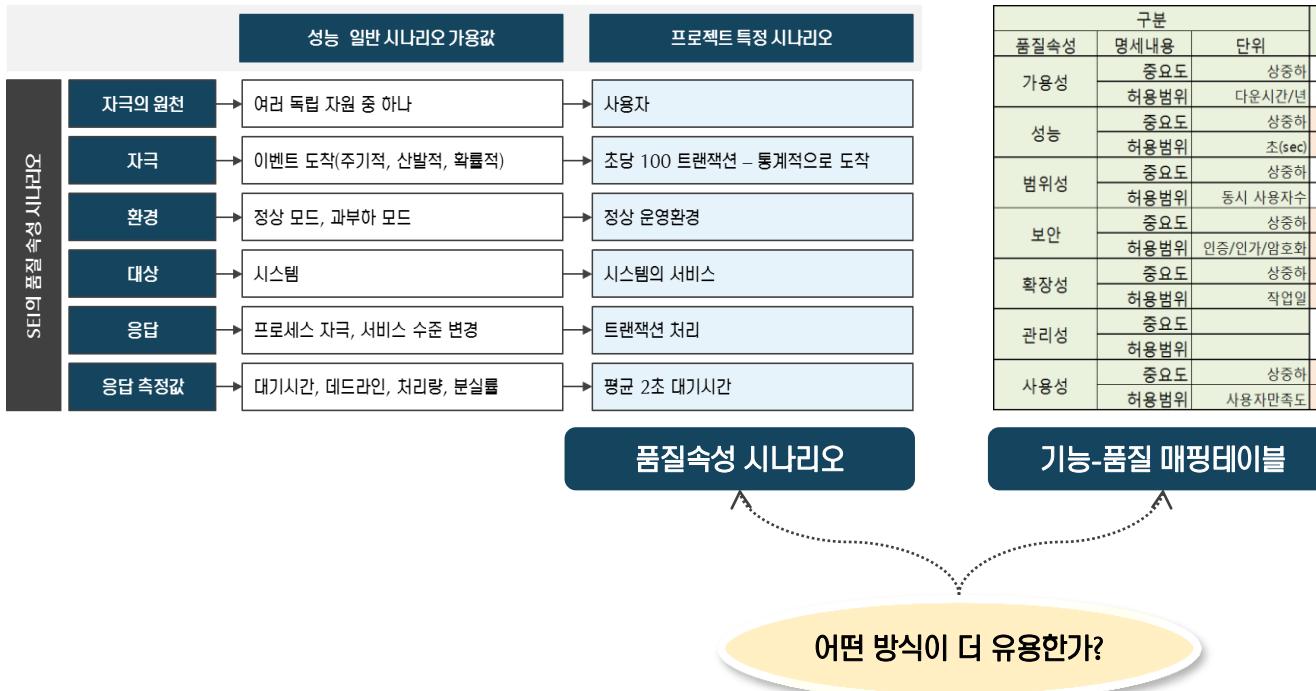
6. 품질속성 시나리오 (5/5) – 예제 3

- ✓ 성능 품질속성에 대한 일반 시나리오 마찬가지 선택 가능한 값들을 포함하고 있음
- ✓ “사용자가 정상 운영 시에, 확률적으로 분당 1,000 트랜잭션을 초기화했다. 이 트랜잭션은 평균 대기 시간 2초를 가지고 처리된다.”라는 문장을 여섯 가지 정보요소로 분리하여 표현을 했음



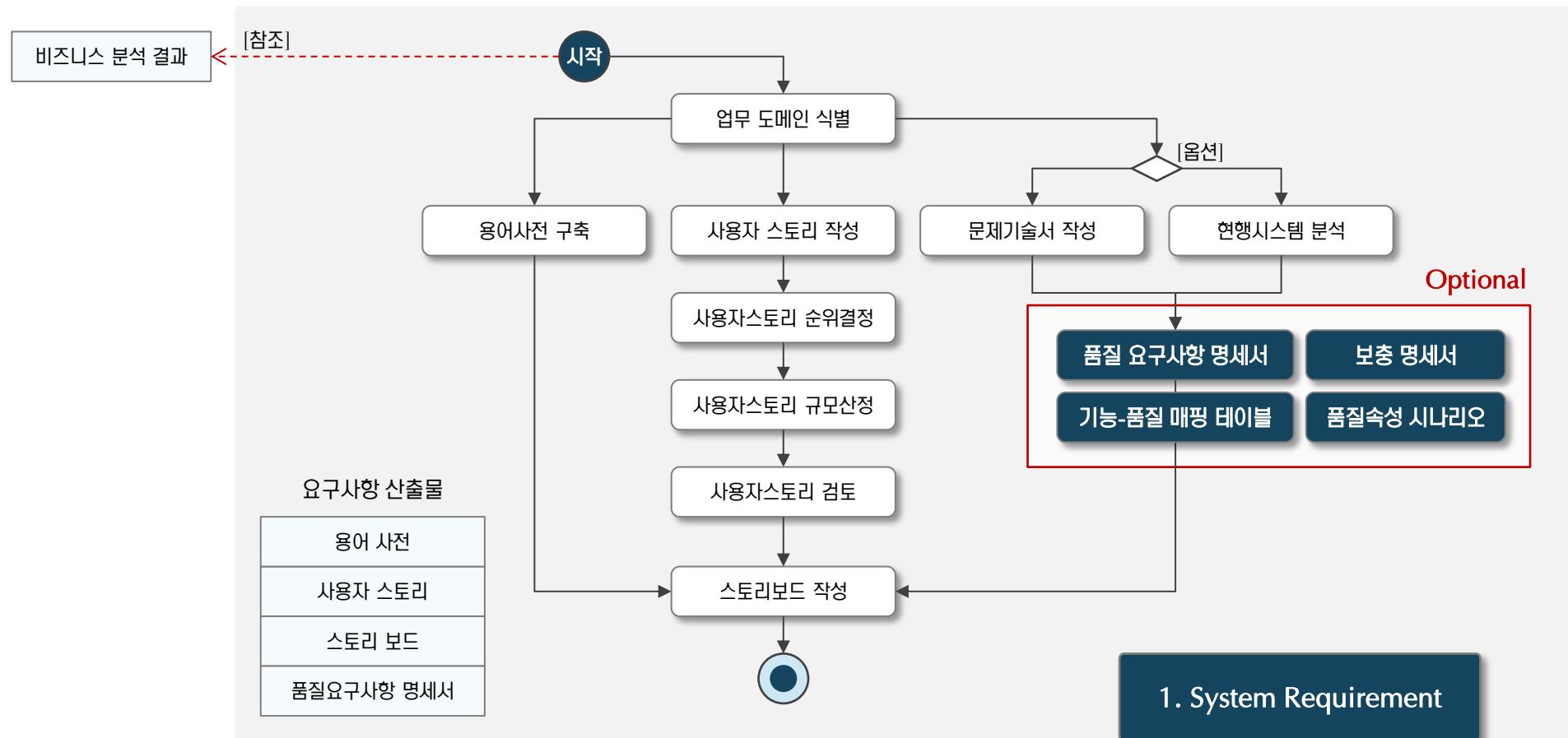
7. 품질속성 시나리오 효율성

- ✓ 품질속성 시나리오를 이용한 품질 목표를 표현하는 방법은 상대적으로 많은 작업을 요구함
- ✓ 품질 요구수준이 매우 높으며, 프로젝트 기간과 인력이 충분한 경우에 품질 속성 시나리오를 고려해볼 수 있음
- ✓ 대부분의 경우, 기능-품질 매핑 테이블을 이용하여 주요 기능별 품질속성 요구수준을 표현함



8. 품질속성과 프로세스

- ✓ 품질속성을 식별하는 활동은 시스템 요구사항 식별(trawling) 단계에서 수행함
- ✓ 표현하는 방법은 품질요구사항 명세서, 보충 명세서, 기능-품질 매핑 테이블 등의 형태로 다양함
- ✓ 어떤 형태로 표현이 되든 시스템의 비-기능(품질) 기대 수준을 정확히 표현하여야 하고 공유하여야 함



9. 요약

- ✓ 시스템 구축 시 요구사항에는 기능 요구사항과 비기능 (품질) 요구사항 두 가지가 있음
- ✓ 품질 요구사항은 아키텍처 설계의 직접 input이 되는 요구사항임, 즉, 아키텍처 설계를 통해서 품질목표를 달성해야 함
- ✓ 아키텍처 설계 시, 품질 요구사항 외에도 기능요구나 환경에 대한 이해가 선행되어야 함

- ✓ 시스템 요구사항을 기능과 품질로 명시적으로 나누어 정리할 준비가 되었습니까?
- ✓ 조직의 표준 품질속성은 정의되어 있습니까? 또는 참조 정의가 있습니까?
- ✓ 아키텍트 역할 담당자는 지정되었고, 요구사항 식별 활동에 참여하고 있습니까?
- ✓ 아키텍팅 프로세스는 준비되어 있습니까?



4. View와 Viewpoint

1. 3D Job
2. 대상과 관점
3. Viewpoint
4. View
5. Perspective
6. Aspect
7. 요약
8. 토의

1. 3D Job

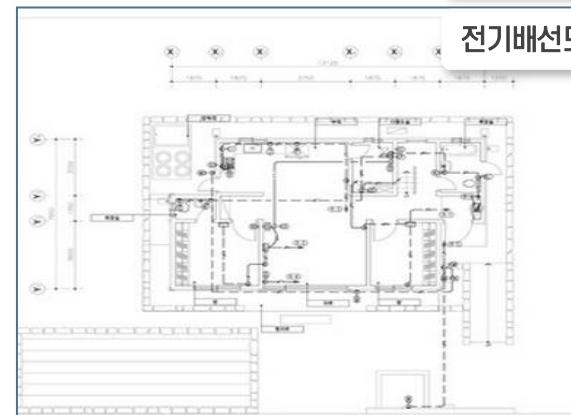
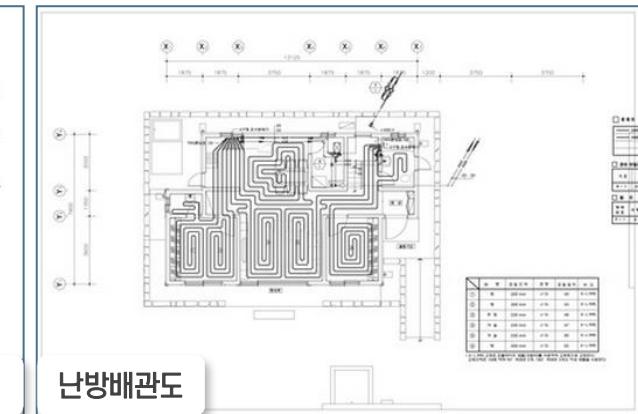
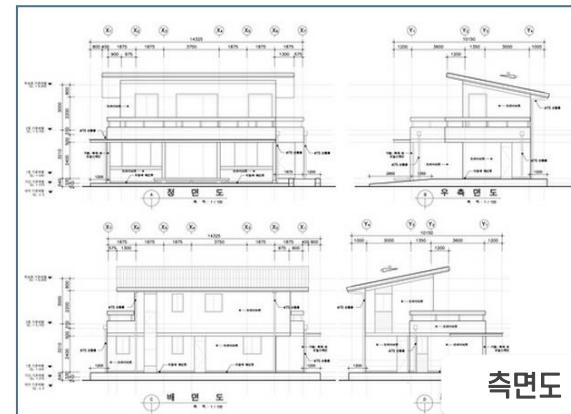
- ✓ 소프트웨어 엔지니어링 일은 3D Job입니다.
- ✓ 3D는 정말 어렵습니다.
- ✓ 3D는 많은 경험과 지식을 갖추어야 가능합니다.



2. 대상과 관점

- ✓ 집을 표현하는 방법은 누구를 위한 표현이냐에 따라 여러 가지가 있습니다.
- ✓ 그 집에서 사는 사람들이 원하는 표현 방식은 조감도를 보기로 희망합니다.
- ✓ 집을 짓는 사람들은 건축에 참여하는 역할에 따라 서로 다른 표현을 기대합니다.

농어촌 공사 표준주택설계도면(농림-09-41-가)
2009년, 철근 콘크리트 조



3. Viewpoint

- ✓ 엔지니어링에서 많이 사용하는 용어입니다.
- ✓ 소프트웨어 아키텍처 설계를 명확한 개념 이해가 필요합니다.
- ✓ 우선은 “관찰하는 위치”라는 의미로부터 시작합니다.

출처: 구글 사전

영어 > 한국어 사전에서 찾았습니다.

viewpoint [vju:pint] [US]

n.

[수식 어구를 동반해서] **견해**, 견지, 관점(standpoint) 《of》
a practical[religious] viewpoint 실질적[종교적] 견해[견지]
(어떤 것이) 보이는 지점, 관찰하는 위치

관련 구문

from the viewpoint of
...의 관점[견지]에서

동의어

noun: standpoint, point of view, angle, aspect, outlook, slant



4. View (1/5)

- ✓ 한 Viewpoint에서 여러 뷰를 가질 수 있습니다.
- ✓ 뷰는 눈에 보이는 것일 수도 있고, 눈에 보이지 않는 것 일 수도 있습니다.



출처: 구글 사진

영어 > 한국어 사전에서 찾았습니다.

view [vjuːɪ] [US]

n.

[sing.]

✓ 봄, 바라봄, 구경
개관, 개설, 통람 《of》

【법】 실지 검증

✓ 시력, 시각, 시선

✓ 시계(視界), 시야

objects in view 시야에 있는 것들, 보이는 것들

[a ~, the ~] 경치, 조망(眺望), 전망

a house with a view of the sea 바다가 바라보이는 집
관점, 견지

from a practical view 실제적인 면[견지]에서 보면
[보통 sing.;수식어와 함께] (특정한) 사고 방식, 보는 방식 《of》

a romantic view of life 삶에 대한 낭만적 사고 방식

(개인적) 의견, 견해, 생각 《on, about》 (Opinion 유의어)

I have[hold] other views about the boy's future. 아이의
장래에 관하여는 나대로의 다른 생각이 있다.

풍경화[사진];전망도(圖) 《of》

a view of the mountain 산이 있는 풍경화

a bird's-eye view 조관도, 전경;(사물의) 개관

a front[back] view 정면[후면]도

시찰, 관찰, 조사

목적, 계획, 의향, 의도;가망

소견, 인상, 감명, 느낌 《of》

4. View (2/5)

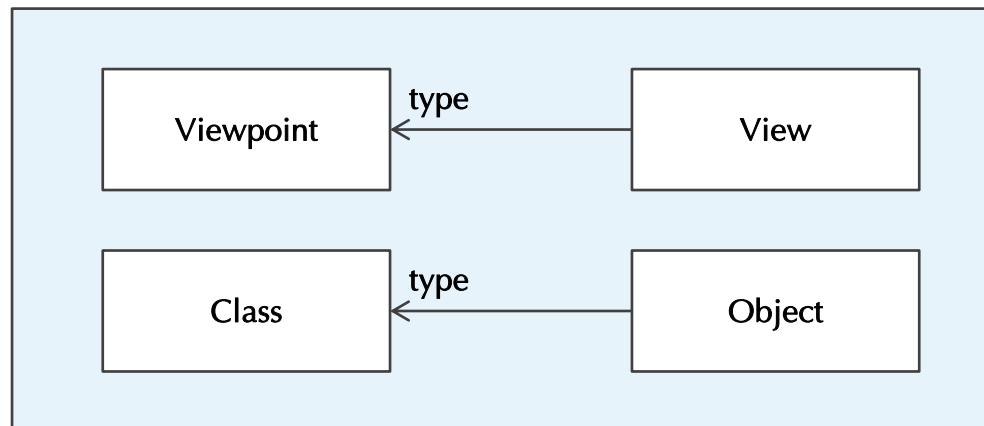
- ✓ 하나의 대상은 다양한 Viewpoint를 가질 수 있으며 거의 한계가 없을 정도입니다.
- ✓ 사진사가 폭포를 보았을 때, 물리학자가 폭포를 보았을 때 자신만의 Viewpoint에서 대상을 바라봅니다.
- ✓ 때로는 두 개 이상의 Viewpoint에서 교차하는 지점에 View를 형성하는 경우도 있습니다.



4. View (3/5) – Viewpoint 관계1

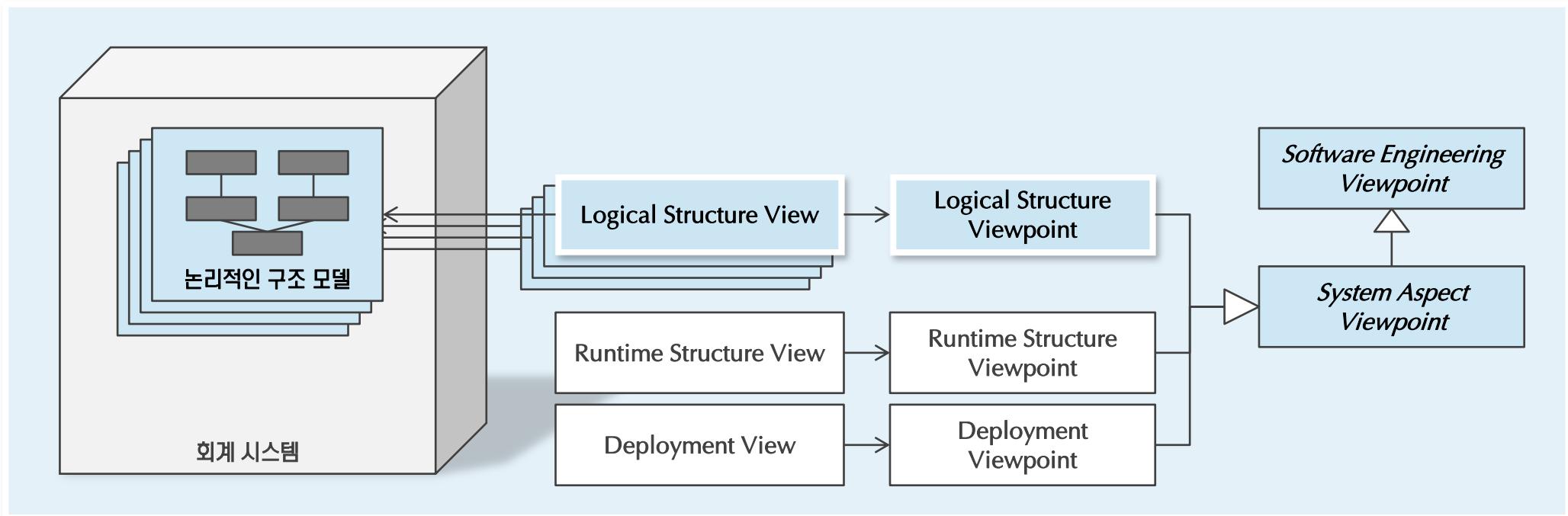
- ✓ 모든 View는 Viewpoint로부터 얻을 수 있습니다.
- ✓ 모든 View는 View Specification에 해당하는 Viewpoint를 가집니다.
- ✓ Viewpoint는 클래스에 View는 객체에 비유하기도 합니다.
- ✓ Viewpoint가 클래스라면 Abstraction 계층을 가질 수 있을까요?

Viewpoint : View :: Class : Object



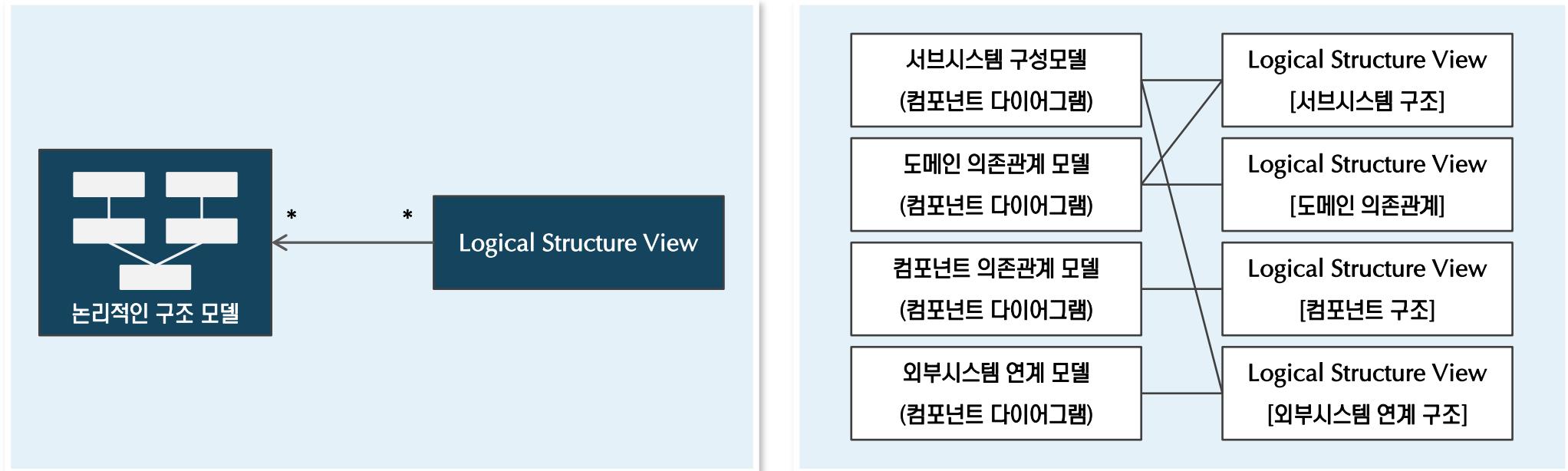
4. View (4/5) – Viewpoint 관계2

- ✓ 정보시스템을 다양한 뷰포인트에서 바라 보고 설명하거나 분석할 수 있습니다.
- ✓ 논리구조 Viewpoint는 시스템의 여러 측면 중에 논리구조만을 추출하여 바라봅니다.
- ✓ 논리구조 Viewpoint에서 바라보면 여러 개의 논리구조 View를 얻을 수 있습니다.
- ✓ 논리구조 View를 통해서 보이는 논리구조는 UML 모델로 표현할 수 있습니다. → 컴포넌트 다이어그램 등



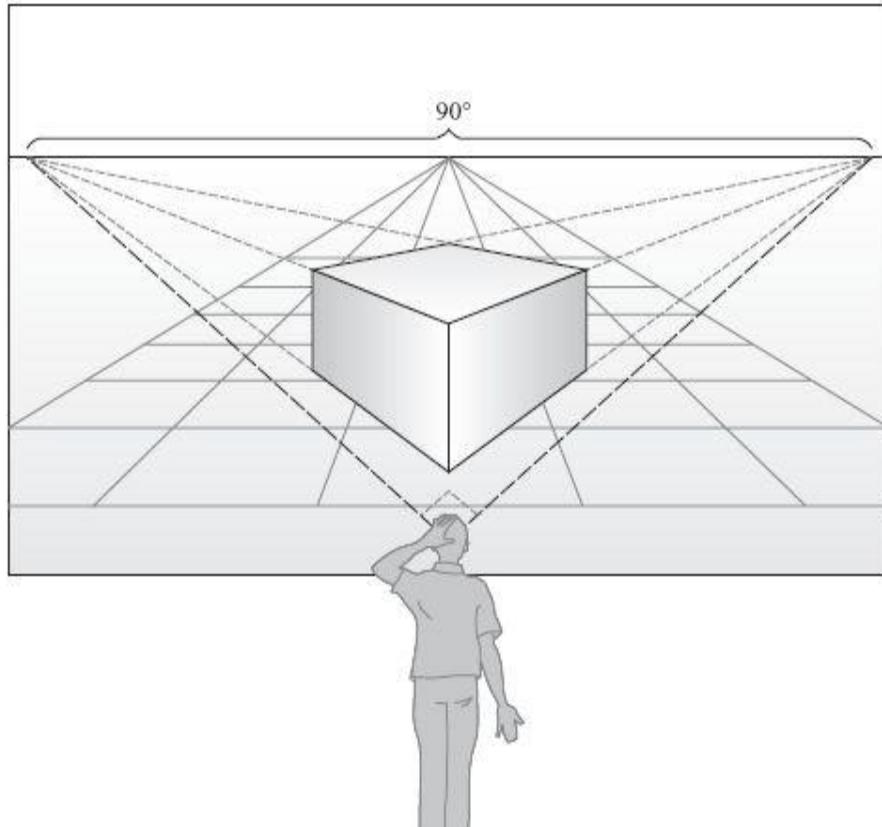
4. View (5/5) – Model 관계

- ✓ View는 개념이므로 시각적인 표현은 UML Model에 의존합니다. (소프트웨어 엔지니어링에서)
- ✓ 하나의 Viewpoint로부터 여러 View를 생성할 수 있고,
- ✓ 하나의 View는 여러 Model로 표현되고, 즉, 여러 Model로 구성되고(consist of),
- ✓ 하나의 Model은 여러 View에 참여할 수 있습니다. (IEEE 1471)



5. Perspective (1/4)

- ✓ 한 곳으로부터 지역이나 대상을 바라보았을 때, 시야 들어오는 경치를 의미합니다.
- ✓ 미술에서는 원근법, 또는 투시화법이라고 합니다.



출처: <http://www.answers.com/topic/perspective>

출처: 구글 사전

영어 > 한국어 사전에서 찾았습니다.

perspective [pər'spektiv] [US]

n.

↙ 원근법, 투시화법; ↘ 원근[투시]도

linear[angular] perspective 직선[사선] 투시도법

↙, ↘ 원근감, 균형;(사물을) 내다보는 힘, 균형 있게 보기; 시각, 견지
see things in the right perspective 사물을 바르게 보다

원경(遠景), 원근, 조망, 경치, 시야

전망, 전도, 가능성, 예상

견해, 관점, 사고방식

from a historical perspective 역사적인 관점에서

a., A

원근[투시] 화법의; 원근법에 의한

perspective representation 원근[투시] 화법

동의어

noun: prospect, vista, outlook, view

5. Perspective [2/4]

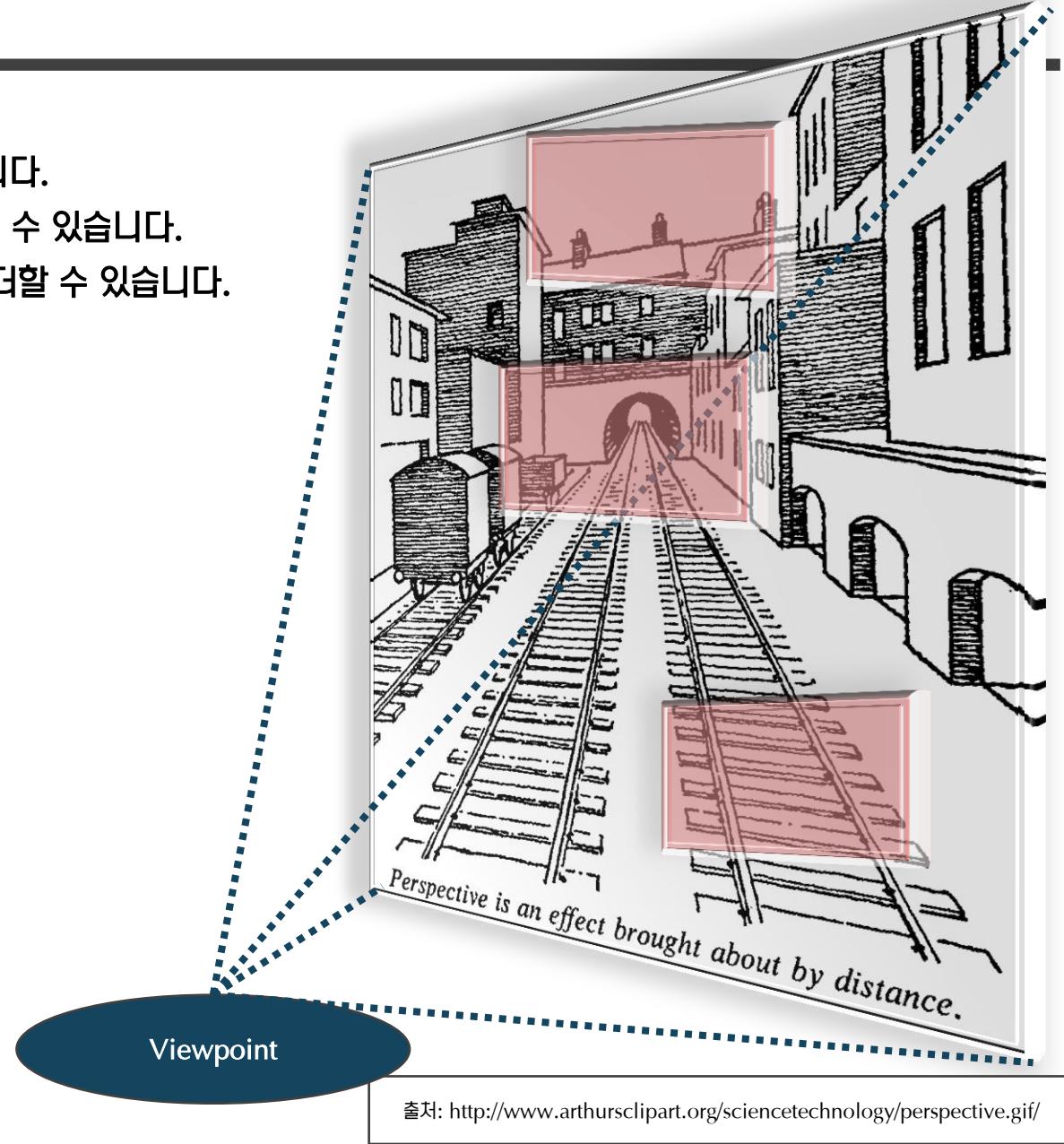
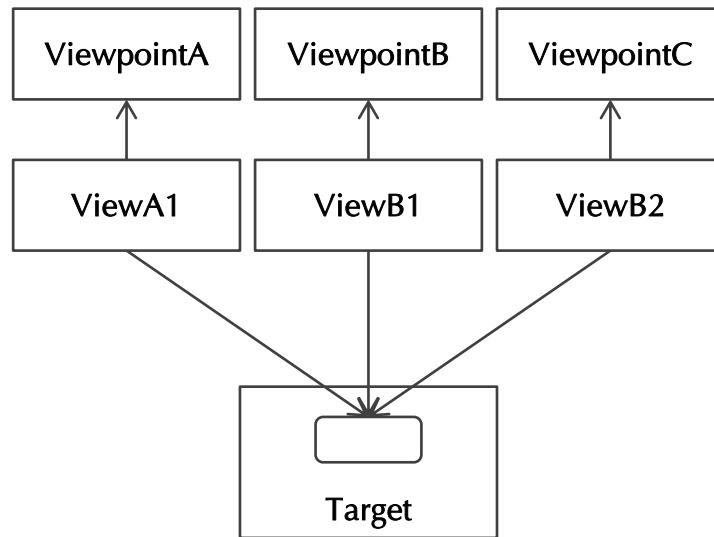
- ✓ Viewpoint를 기준으로 전방에 펼쳐 지는 장면, 경치 등을 의미합니다.
- ✓ 개념적으로는 전망, 예상 등으로 사용합니다.
- ✓ 견해, 관점, 사고방식 등의 의미로 사용할 때는 [개인의 주장이 약하거나 배제된] 것을 의미합니다.



출처: <http://www.flickr.com/photos/paulobrandao/>

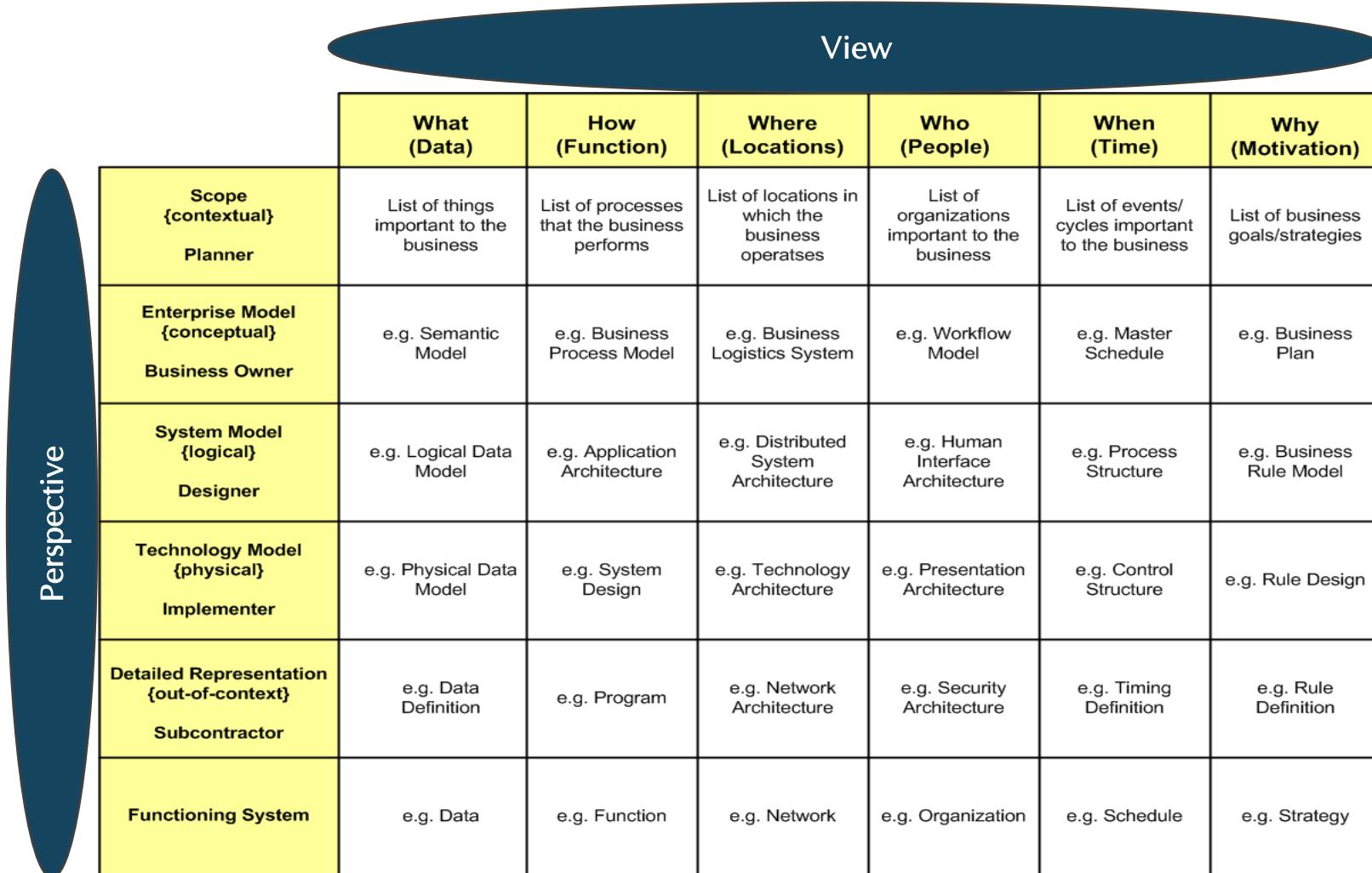
5. Perspective (3/4)

- ✓ Viewpoint로부터 Perspective를 얻을 수 있습니다.
- ✓ 하나의 Perspective 안에 여러 개의 View를 얻을 수 있습니다.
- ✓ 하나의 대상이 여러 Viewpoint로부터의 여러 View를 가질 수 있습니다.
- ✓ 다양한 뷰를 통해 대상에 대한 이해를 높이고 분석의 깊이를 더할 수 있습니다.



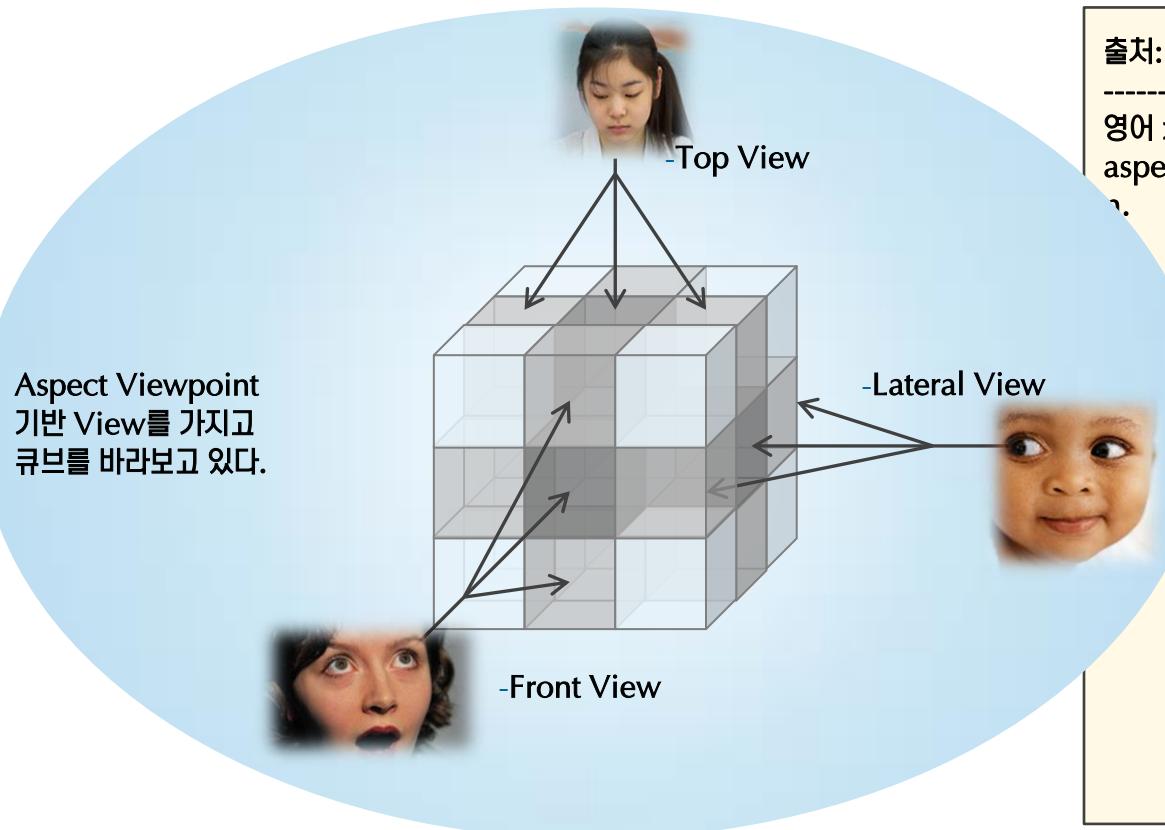
5. Perspective (4/4)

- ✓ Zachman 프레임워크에서 Perspective 영역을 볼까요?
- ✓ Perspective/ View



6. Aspect

- ✓ 어떤 사물의 [특징을 대표하는] 외관이라는 의미로부터 출합니다.
- ✓ 예제, diverse aspects of human life, diverse aspect of a information system.
- ✓ 큐의 특징적인 외관(aspect)은 어떤 것들이 있습니까?



출처: 구글 사전

영어 > 한국어 사전에서 찾았습니다.

aspect [ˈspekt] [US]

n.

양상, 외관, 모양, 경관

a mountain with a beautiful aspect 모습이 아름다운 산
(문제를 보는) 관점, 각도, 견지

consider a question in all its aspects [from every aspect] 문제를
모든 각도에서 고찰하다

(사물의) 면, 국면, 양상 (Opphase 유의어)

diverse aspects of human life 인생의 여러 양상

《문어》 (사람의) 용모, 생김새 (appearance), 표정; 태도

[방향을 나타내는 수식어를 수반하여] (가옥 등의) 방향, 방위

His house has a southern aspect. 그의 집은 남향이다.

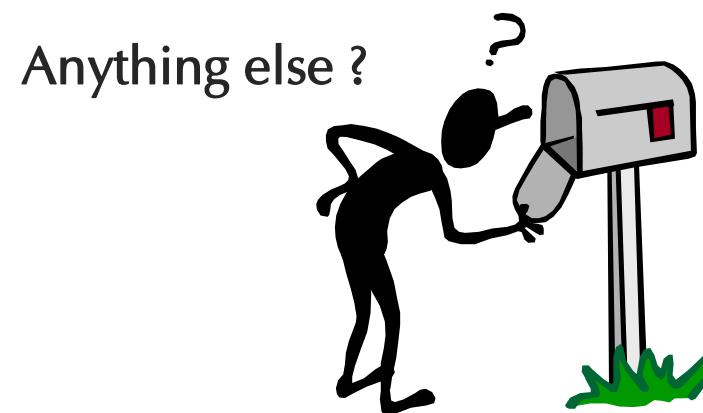
U 【문법】 (동사의) 상(相)

【천문】 각거리 《지구에서 본 하늘 위의 두 점간의 거리》; 【점성】 별의 상(相),
성위(星位)

7. 요약

✓ 3D 아키텍팅을 위한 핵심 개념 세 가지를 살펴보았습니다.

- View
- Viewpoint
- Perspective
- Aspect





5. IEEE 1471

1. 개요
2. IEEE 1471 Key Concepts
3. IEEE 1471 Close-up
4. IEEE 1471 Review
5. SA Summary
6. 토의

1. 개요 – 아키텍처 범람

- ✓ 너무나 많은 아키텍트의 존재로 인해 모두가 혼란스럽군요.
- ✓ 인사 담당자는 아키텍트 직무가 너무 많아 혼란스럽고,
- ✓ 프로젝트 관리자는 아키텍트가 너무 많이 필요해서 어려움이 있고,
- ✓ 엔지니어링 팀은 아키텍처 팀을 어떻게 구성해야 할 지 고민이 있습니다.

❖ 시스템 아키텍트

❖ 솔루션 아키텍트

❖ 보안 아키텍트

❖ 기술 아키텍트

❖ 네트워크 아키텍트

❖ 컴포넌트 아키텍트

❖ 애플리케이션 아키텍트

❖ 비즈니스 아키텍트

❖ 정보 시스템 아키텍트

❖ 데이터 아키텍트

❖ EAI 아키텍트

❖ 포탈 아키텍트

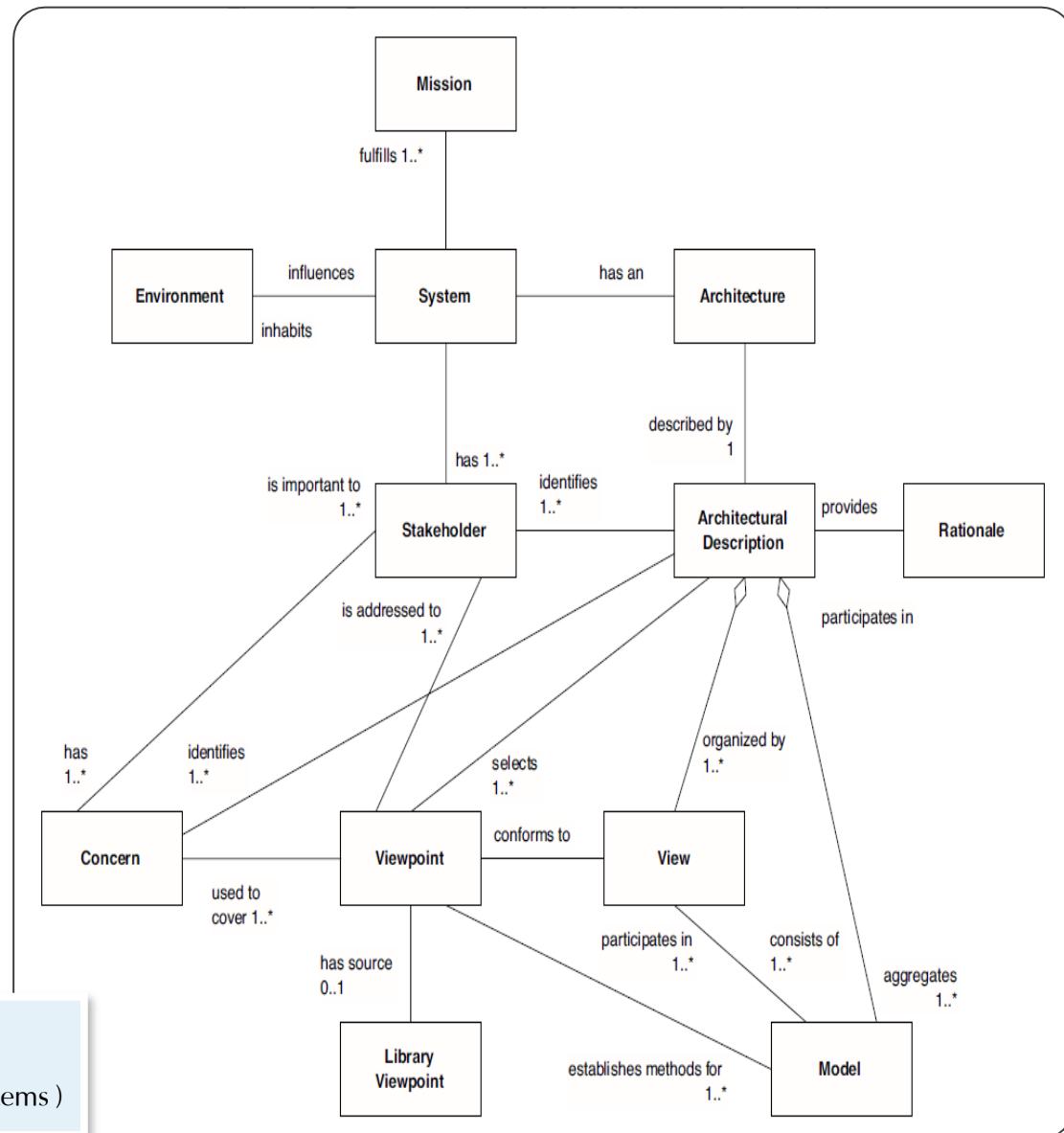
❖ 서비스 아키텍트

❖ 소프트웨어 아키텍트

[너무나 많은 아키텍트]

1. 개요 – Why IEEE 1471 ?

- ✓ IEEE 1471는 아키텍처 서술을 위한 권장 실행지침으로 표준에 준하는 위치를 가집니다.
- ✓ 아키텍트의 역할 논쟁을 표준을 근거로 조정하고 정리하는 것이 바람직합니다.
- ✓ IEEE 1471에서 아키텍처를 어떻게 정의하고 있는가로부터 출발 합니다.
- ✓ [EA와 TOGAF]에서 정의한 IT 영역 개념도 아키텍처 역할 정리에 필요합니다.

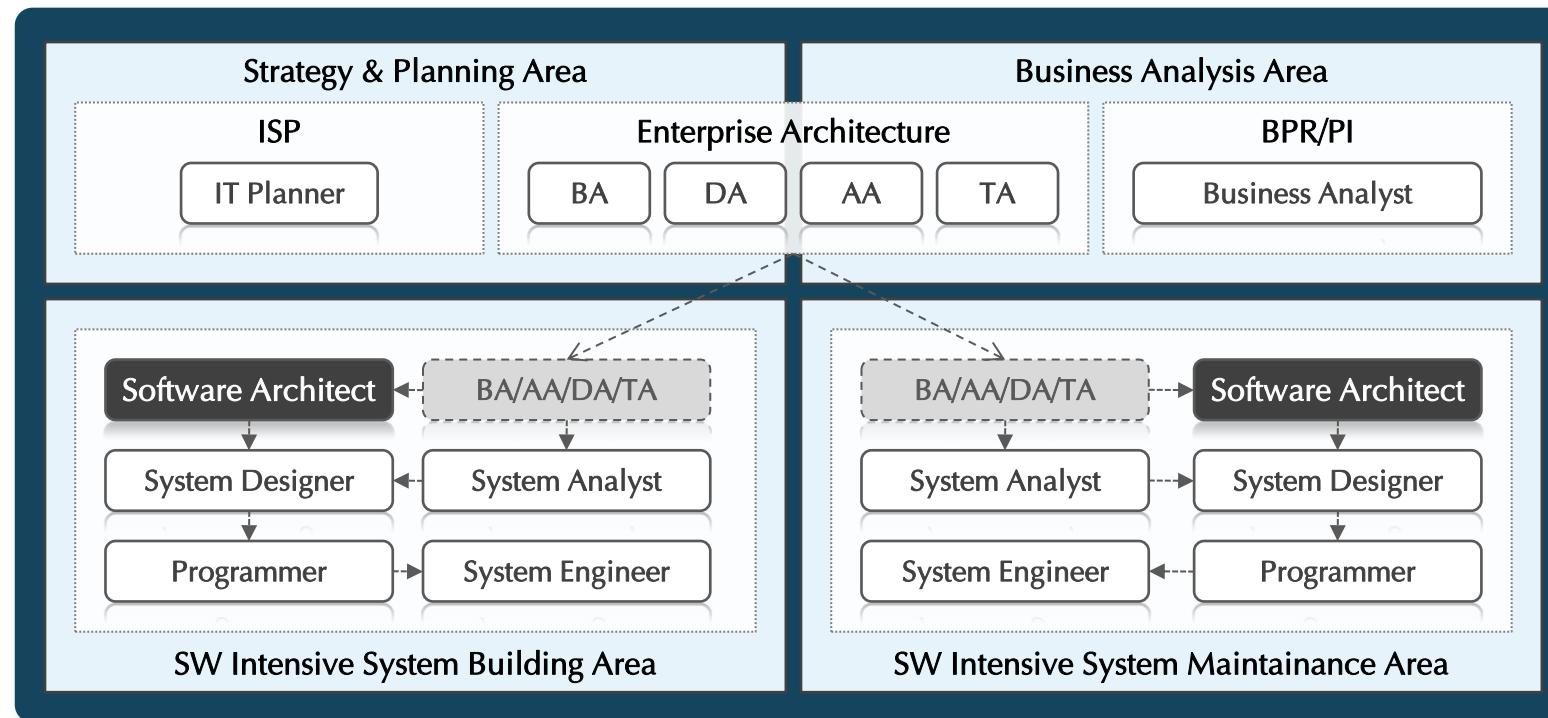


IEEE 1471 - 2000

(IEEE Recommended Practice for Architectural Description of Software-Intensive Systems)

1. 개요 - 혼란의 핵심

- ✓ 혼란의 핵심에는 EA에서 정의한 AA, DA, TA, BA가 있습니다.
- ✓ 일부 조직의 경우 Software Architect는 없고 AA, DA, TA, BA만 직무로 정의하고 있습니다.
- ✓ 특히 AA와 SA는 역할 중첩 논란이 항상 존재하고 있습니다.
- ✓ EA-AA는 거버넌스를 이유로 시스템 빌딩 영역에서도 지속적으로 활동을 해야 한다고 주장합니다.



1. 개요 - 소프트웨어 아키텍처 범위

- ✓ 소프트웨어 아키텍트는 [시스템 구축과 운영 영역]에서 정의하고 사용하며, 전 영역에서 참조하는 역할입니다
- ✓ 소프트웨어 아키텍처는 IEEE 1471에서 정의한 Software Intensive 시스템의 아키텍처를 의미합니다.
- ✓ 소프트웨어 아키텍트는 [Software Intensive System]을 위한 아키텍처를 설계하는 아키텍트입니다.
- ✓ 이러한 범위를 전제로 하고 IEEE 1471를 자세히 분해하여 이해하고자 합니다.



2. IEEE 1471 Key Concepts (1/4)

✓ Software Intensive System의 범위(coverage)는?

- 개인용 SW 애플리케이션
- 기업 정보 시스템
- 임베디드 시스템
- SW Product Lines, Product Families
- System-of- [Software intensive] System

What kind of systems does IEEE 1471 cover?

Its focus is software-intensive systems: any system in which software development and/or integration are dominant considerations (i.e., most complex systems these days). This includes computer-based systems ranging from individual software applications, information systems, embedded systems, software product lines and product families and systems-of-systems.

2. IEEE 1471 Key Concepts (2/4)

- ✓ Software Architecture란 무엇인가?
- ✓ CMU 사이트에 가면 아키텍처에 대한 정의를 60여가지 제시합니다.
- ✓ IEEE 1471에서 정의하는 아키텍처를 기준으로 합니다.
- ✓ 컴포넌트 == 아키텍처 요소(architectural element)

아키텍처는 시스템의 근간을 이루는 틀(fundamental organization)로써 시스템을 구성하는 컴포넌트, 컴포넌트 간의 관계, 컴포넌트와 환경 간의 관계, 그리고 설계와 개발 진행을 관리하는 원칙의 형태로 나타난다.

(The fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution)

IEEE 1471-2000

2. IEEE 1471 Key Concepts (3/4)

✓ Software Architecture란 무엇인가? (계속)

아키텍처는 시스템의 본질적(essential) 것이거나 통합적(unifying)인 것이다. 즉, 시스템의 구조(structure), 행위, 가치, 비용, 그리고 리스크를 결정하는 시스템 속성(properties) 집합이다. 제안 실행지침에서 정의는 “아키텍처는 시스템의 기반을 이루는 구조(fundamental organization)이며, 그 구성요소인 컴포넌트, 컴포넌트 상호 관계, 환경과의 관계, 그리고 설계와 개발진행을 이끌어 주는 원칙을 포함한다. 이 정의에는 서너 가지 핵심 아이디어가 있다.

첫째, 아키텍처는 시스템을 마음 속에 그린 것이다. 따라서, 아키텍처는 기록되지 않고 존재할 수도 있다. 표준에서 아키텍처와 아키텍처 서술을 구분했다. “지도가 영토는 아니다”라고 말하듯이 “아키텍처 서술은 아키텍처가 아니다.” 아키텍처 서술은 산출물로 작성된 것이거나 유형의 작업 결과물이다. 즉, 다른 사람들을 위해 마음 속에 그린 것을 표현하려는 시도이다. 표준의 초점은 아키텍처 서술 요구에 있다.

둘째, 아키텍처는 시스템의 토대가 되는 것(fundamental thing)들을 전체적으로 안고 있다. 토대가 되는 것들이란 시스템의 본질 또는 핵심에 해당하는 속성들이다.

셋째, 아키텍처는 고립 상태가 아니라 컨텍스트 안에서 이해되어야 한다. 시스템의 기반 속성(즉, 아키텍처)을 이해하는 것은 시스템이 환경과 어떤 관계에 있으며, 어떤 입장에 놓여 있는지를 이해하는 것이다. 어떤 것을 위한 기반(fundamental)인지를 알지 못하고서는 시스템에 대한 기반이 무엇인지 알기 어렵다. 그러므로 “기반(fundamental)은 시스템 이해관계자의 컨텍스트와 그 환경 안에서 해석되어야 한다.

끝으로, 아키텍처는 시스템을 구성하는 물리적인 컴포넌트들로 구성된 구조 그 이상이다. 물리적인 구조가 흔히 시스템의 기본 모양(aspect)이긴 하지만, 반드시 있어야 하는 것은 아니다.

<http://www.iso-architecture.org/ieee-1471/ieee-1471-faq.html>

2. IEEE 1471 Key Concepts (4/4)

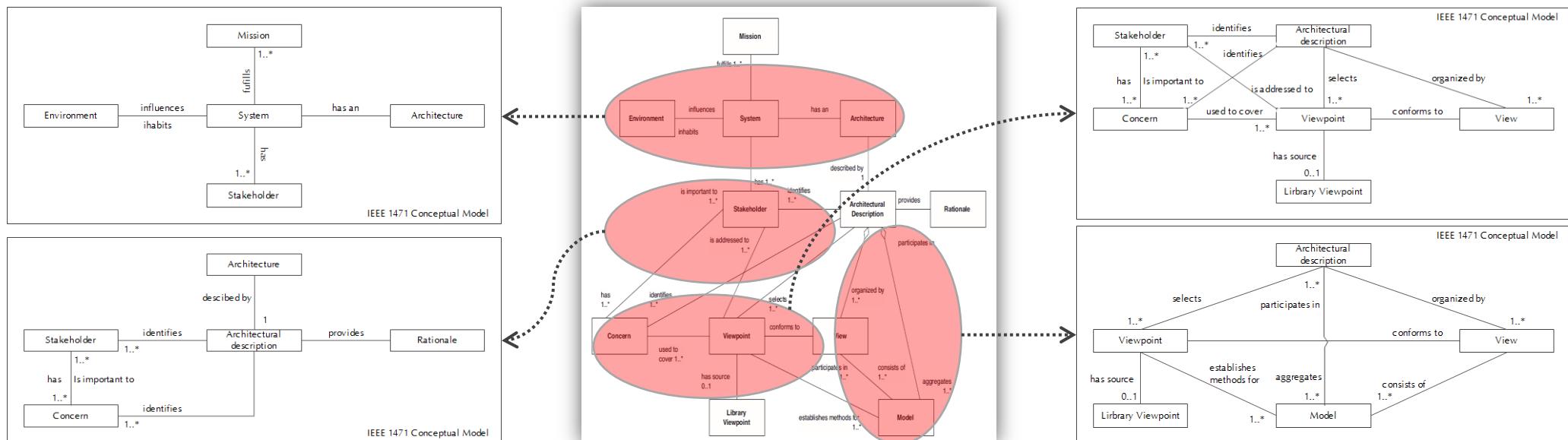
✓ 아키텍처 요소(Architectural Element)란 무엇인가?

- 시스템의 틀을 구성하는 요소를 의미합니다.
- 아키텍처 설계 후에 이러한 아키텍처 요소를 획득(개발, 구매, 오픈소스선택)하여 실행 가능한 시스템을 구성합니다.
- 모든 비즈니스 컴포넌트를 아키텍처 요소라고 할 수 있는가?

아키텍처 요소 타입 / 획득방법	자체개발	오픈소스	솔루션 벤더
비즈니스 컴포넌트	가입설계, 계약 컴포넌트	OMG Party컴포넌트	
프레임워크	지연처리 프레임워크, 예외처리	Struts, myBATIS	T사 프레임워크
서비스 서버	Mini-EAI서버, 상품 팩토리	Artifactory, SVN	T사의 암호화서버, IM/AM서버
플랫폼 소프트웨어	운영체제	Linux	HP-UX, Sun Solaris
	개발 언어	Physon, Groovy	Java, C, C++
	실행 플랫폼	JBoss, Spring, JVM	WebLogic, PaaS
	데이터 플랫폼	MariaDB, MongoDB, Redis	Oracle RDBMS
	프로세스 플랫폼	ESBWorkflow, uEngine	BPM
	비즈니스 플랫폼	Compiere	SAP, Oracle ERP
관리용 소프트웨어 (제외)	컴포넌트 관리, 배포관리		Tivoli, OpenView

3. IEEE 1471 Close-up

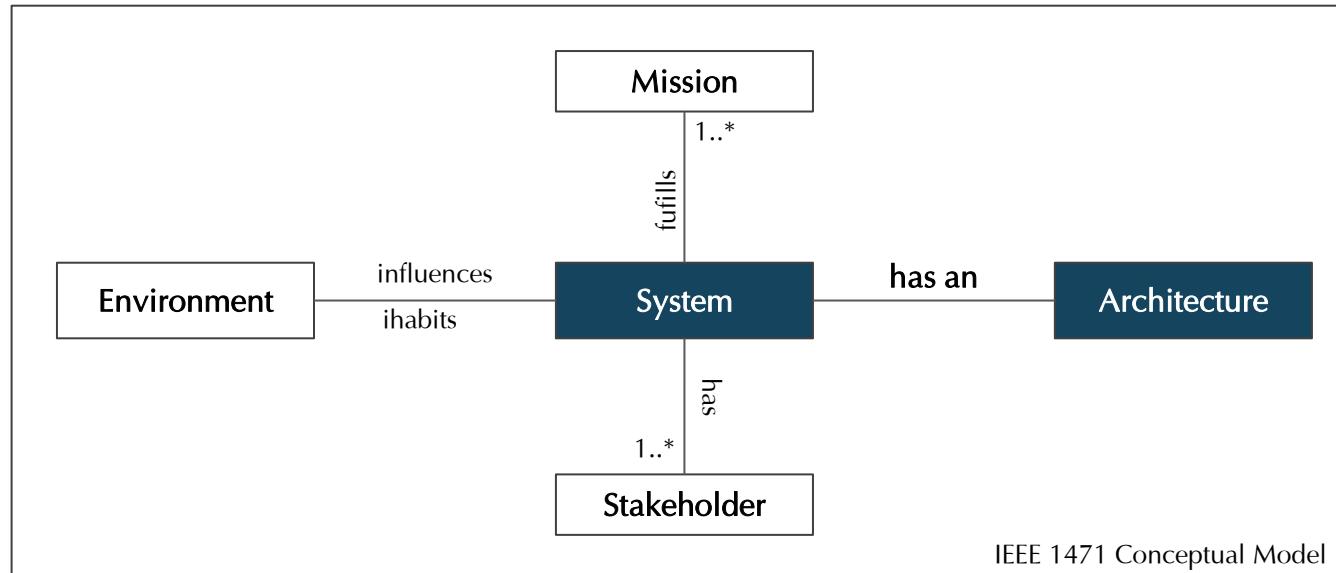
- ✓ IEEE 1471 개념 모델은 추상화 수준이 높은 모델로 한 번에 이해하기 어렵습니다.
- ✓ 따라서, 각 부분 별로 확대하여 부분별로 이해한 후 전체를 보는 것이 편리합니다.
- ✓ IEEE 1471 개념 모델을 네 개의 부분으로 나누어서 이해하고자 합니다.



[IEEE 1471 개념 모델 Close-up]

3. IEEE 1471 Close-up – System has an architecture.

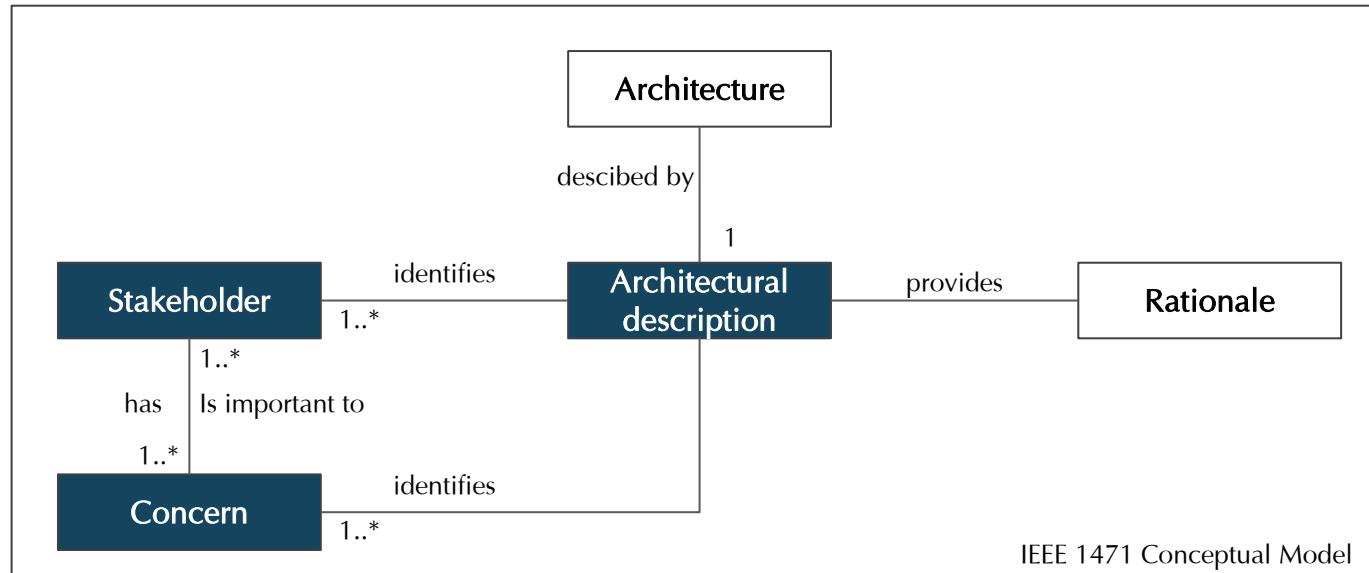
- ✓ 시스템은 [하나의] 아키텍처를 가진다.
- ✓ 시스템은 어떤 환경 속에서 미션을 수행한다. 이 환경은 언제나 제약조건을 가지고 있다.
- ✓ 시스템을 둘러싸고 있는 많은 이해관계자가 존재한다.
- ✓ 시스템은 아키텍처 [하나]를 가진다.



[IEEE 1471 Close-up : System has an Architecture]

3. IEEE 1471 Close-up – Architectural Description

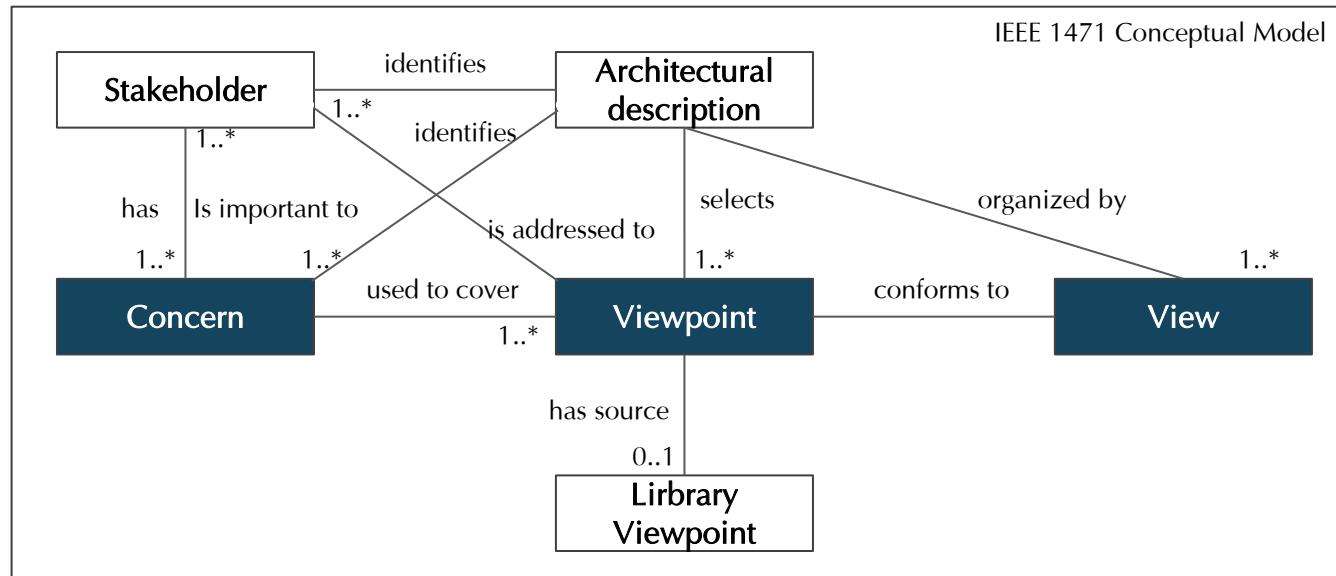
- ✓ 아키텍처는 아키텍트의 머릿속에 존재하는 구상(conception)으로 아키텍처 서술로 표현합니다.
- ✓ 아키텍처 서술은 논리적인 근거 또는 타당성(rationale)을 제공해야 합니다.
- ✓ 아키텍처 서술에서 이해관계자와 그들의 관심사(concern)를 식별하고 명세합니다.
- ✓ 이해관계자는 여러 관심사를 가지고 있으며, 관심사는 아키텍처 설계를 통해서 해결하여야 합니다.



[IEEE 1471 Close-up : AD identifies many Concerns]

3. IEEE 1471 Close-up – Viewpoint and concerns.

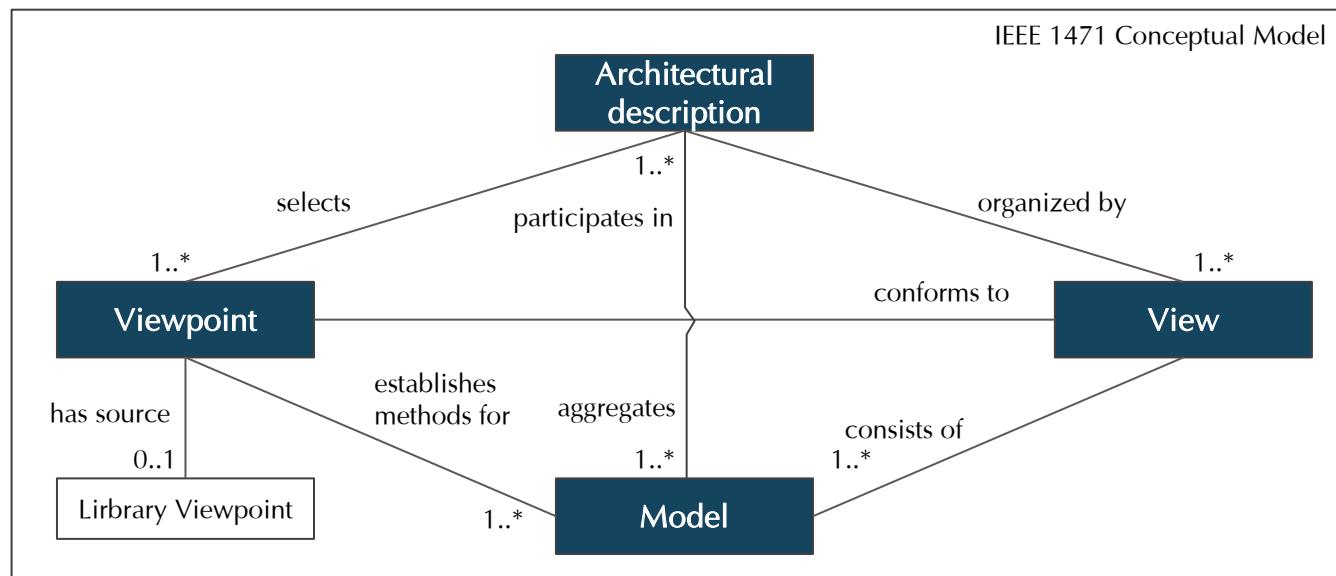
- ✓ AD는 여러 개의 Viewpoint를 가지고, Viewpoint는 관심사를 표현하는데 사용합니다.
- ✓ Viewpoint는 유사한 관심사의 묶음 단위로 설정됩니다.
- ✓ View는 Viewpoint 명세를 준수합니다.
- ✓ 조직에서 미리 정의한 표준 Library Viewpoint로부터 프로젝트를 위한 Viewpoint를 가져올 수 있습니다.



[IEEE 1471 Close-up : Viewpoint is used to cover the Concerns]

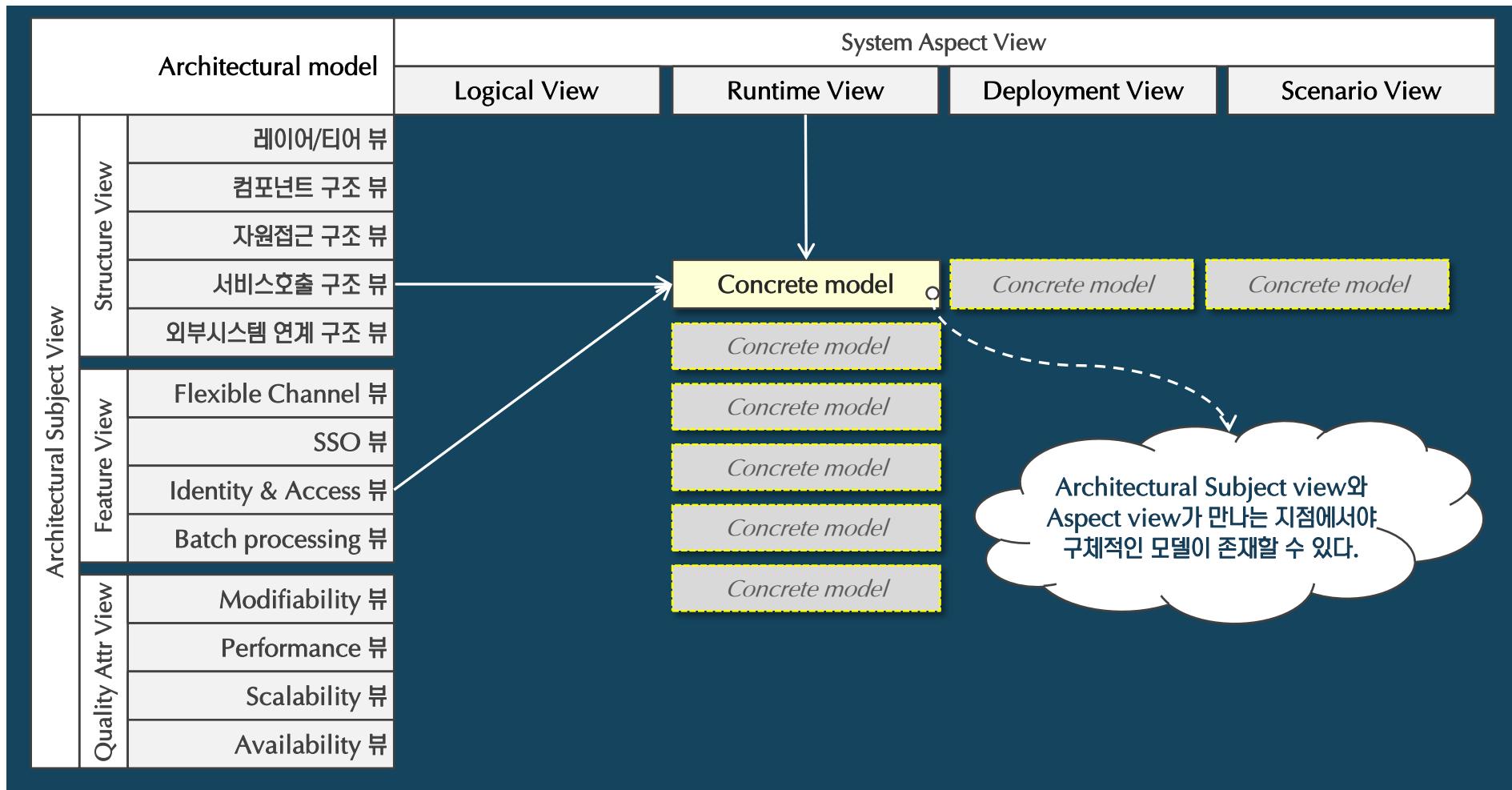
3. IEEE 1471 Close-up – View and Model (1/2)

- ✓ AD에는 여러 Model이 필요합니다.
- ✓ Model과 View는 n:n 관계를 가집니다.
- ✓ View와 Model 관계 이해는 IEEE 1471 개념 모델을 이해하는 데 매우 중요합니다.
- ✓ 여기서 Model은 어떤 Model을 의미할까요?



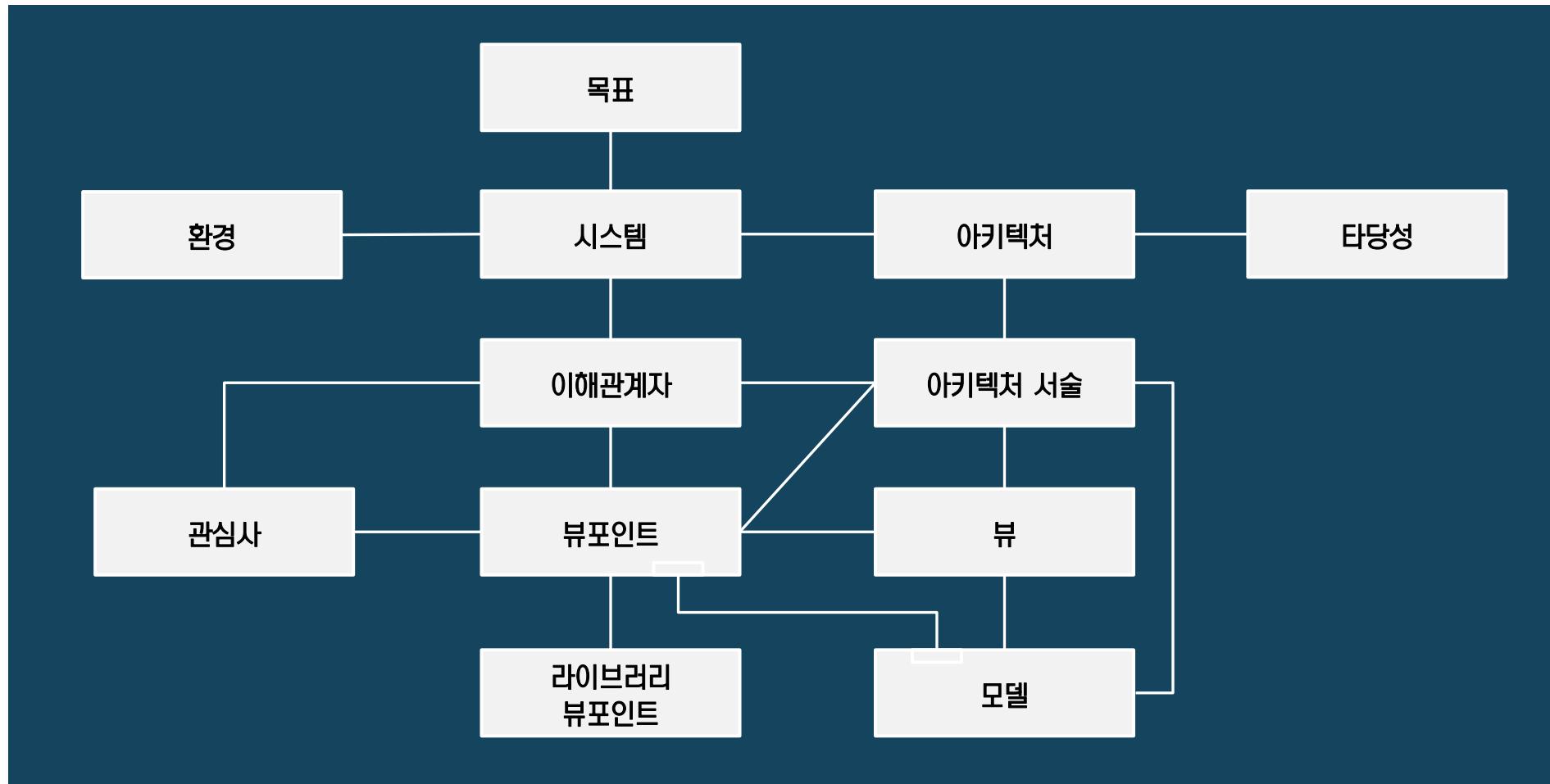
3. IEEE 1471 Close-up – View and Model (2/2)

- ✓ View와 View 가 만나는 지점에서 구체적인 Model이 존재합니다.
- ✓ 아래 도표의 의미에 대해 토론합니다.



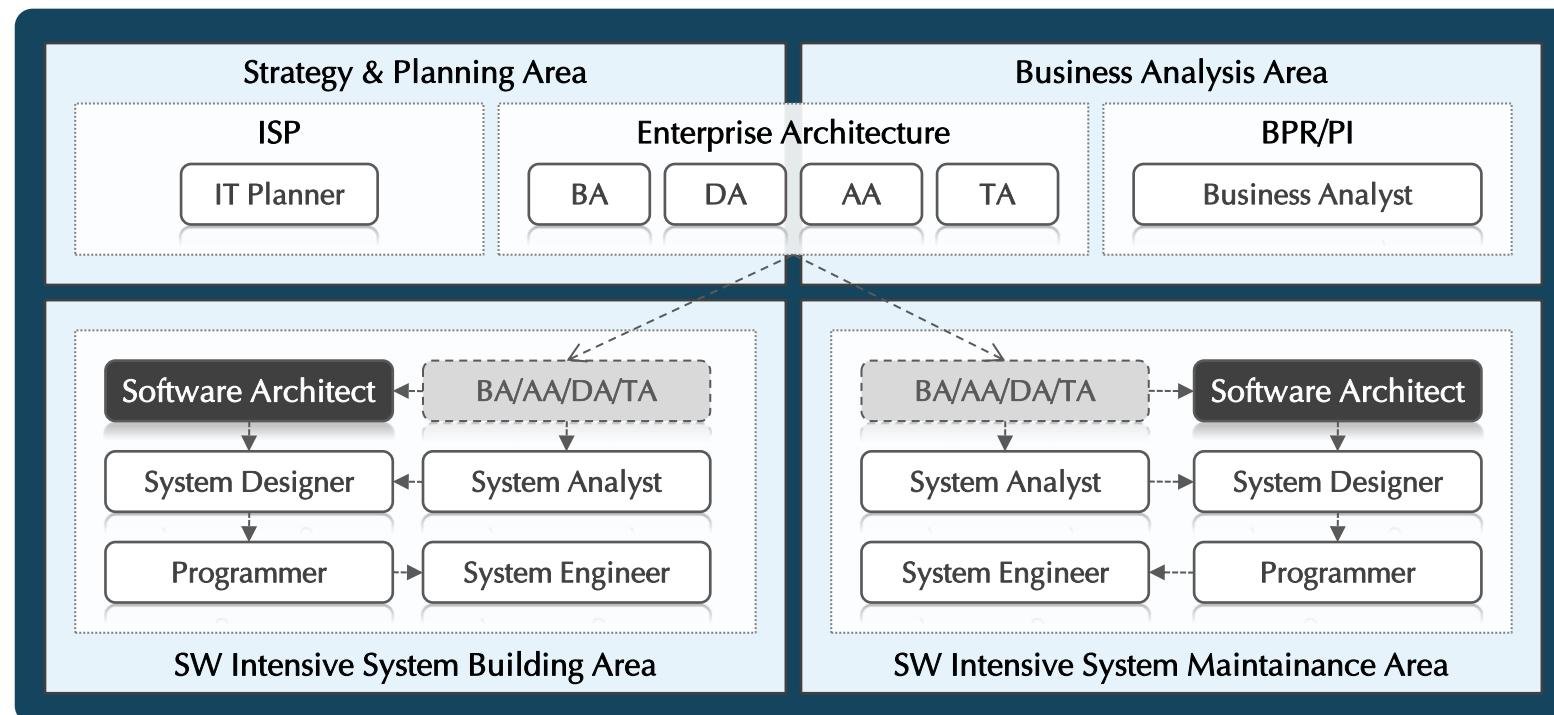
4. IEEE 1471 Review

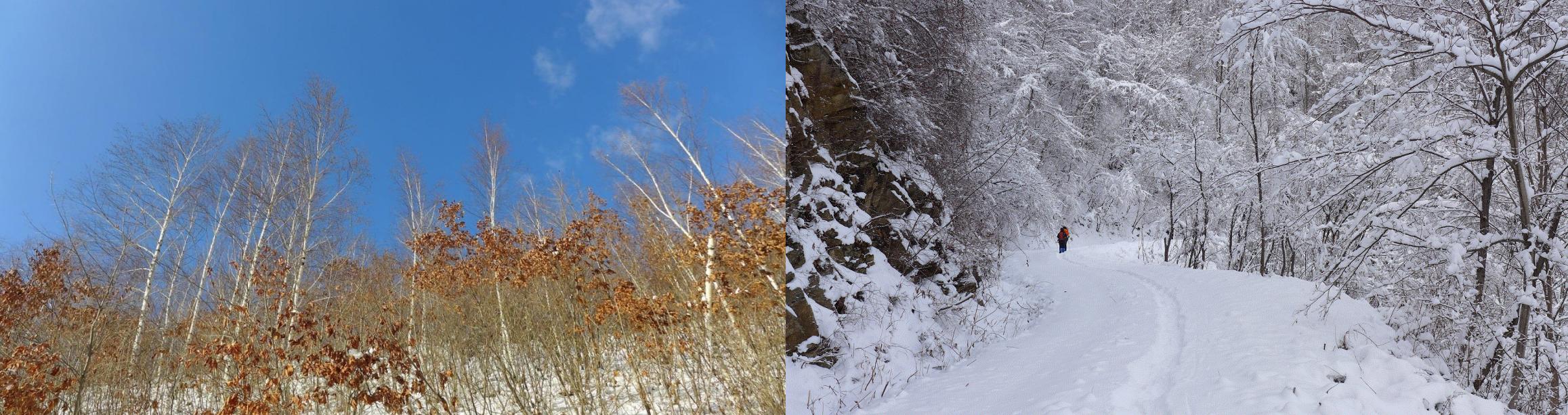
- ✓ 아키텍처 서술을 위한 개념 모델의 의미를 되새겨 봅니다.
- ✓ 이 개념 모델을 확장하여 아키텍처 설계를 위한 구체적인 모델로의 확장에 대해 토론합니다.



5. SA Summary

- ✓ AA, DA, TA, BA의 정확한 정의에 대한 정확하게 이해합니다.
- ✓ IEEE 1471 기반으로 Software Architecture를 정확하게 이해합니다.
- ✓ 이들 간의 차이에 대해 토론합니다.
- ✓ SA = [S]oftware intensive system's [A]rchitecture



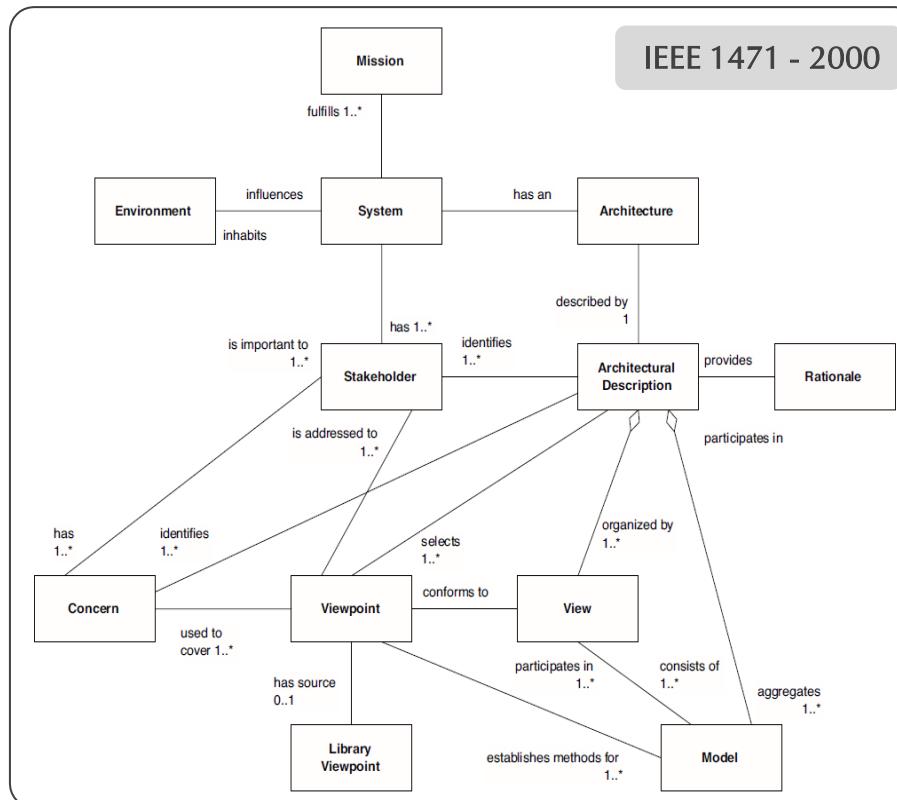


6. IEEE 1471 - Extension

1. 개요
2. IEEE 1471 – 확장
3. IEEE 1471 – 개념 모델 확장
4. New Architecting Process Example
5. SEI 개념 모델과 프로세스
6. IEEE 1471 – 확장 요약
7. 토의

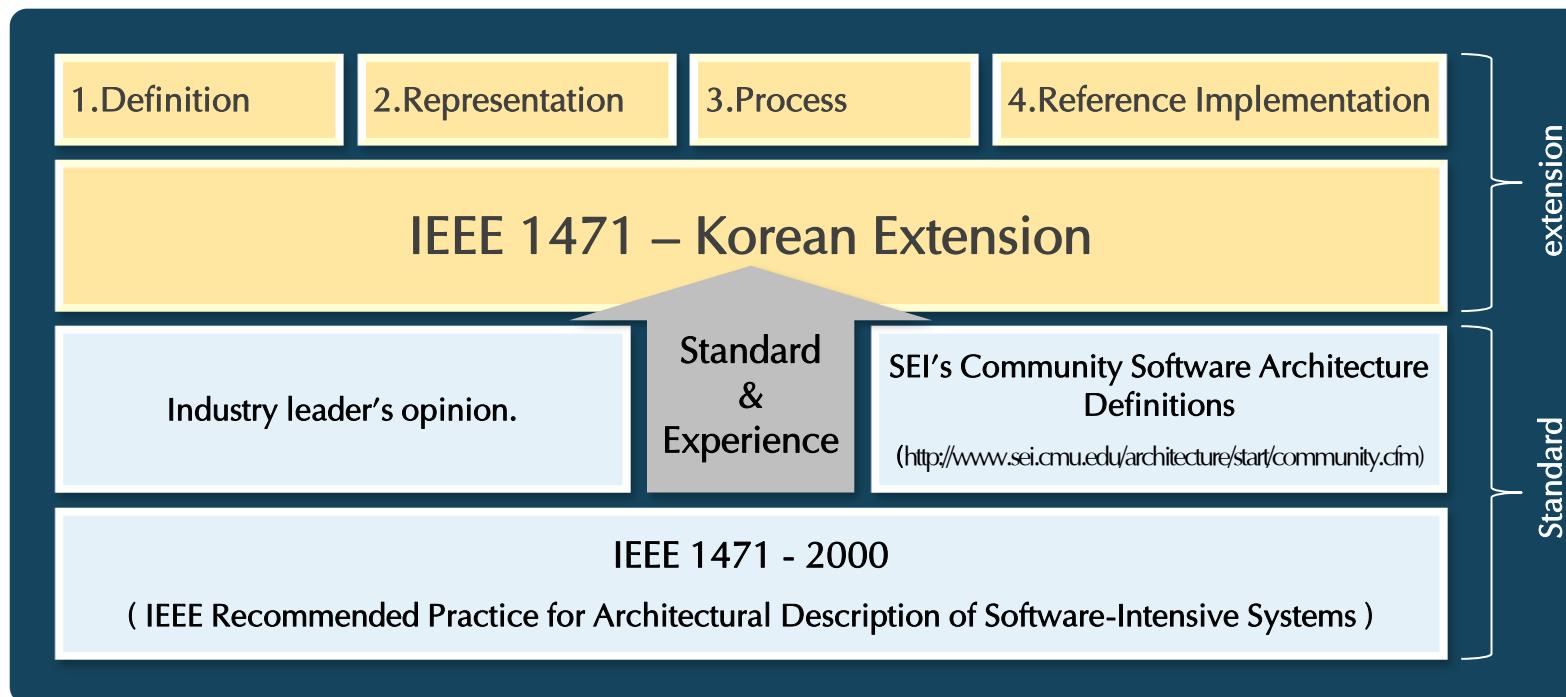
1. 개요

- ✓ IEEE 1471 개념 모델은 추상화 수준이 높은 모델입니다.
- ✓ 예 하나, 아키텍처 요구 사항은 단순히 [Concern]으로 표현하고 있습니다.
- ✓ 예 둘, 추상화 수준이 높은 Model과 추상화 수준이 높은 View 개념을 사용하였습니다.
- ✓ 즉, 대부분의 클래스는 추상 클래스 수준이어서 객체를 만들 수 없는 상태입니다.



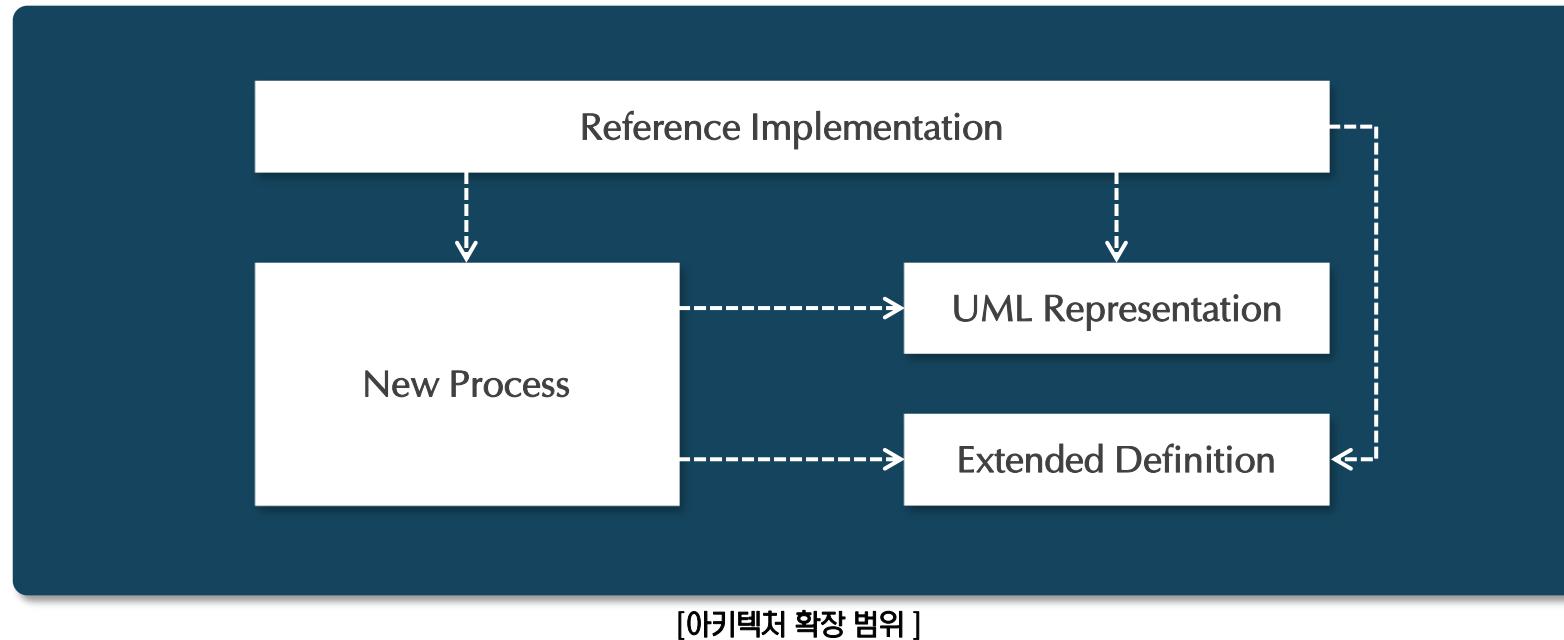
2. IEEE 1471 – Extension (1/2)

- ✓ IEEE 1471-2000을 바탕으로 아키텍처 관련 개념을 정의하고 서술을 통해 구체화 합니다.
- ✓ 아키텍처 분야 리더들의 축적된 경험과 지식을 반영하여 Extension을 명세합니다.
- ✓ 표준(standard), 원칙(principle), 논리(logical), 사실(fact)을 벗어나지 않아야 합니다.
- ✓ 현장에서 적용 가능한 실용적인(practical) 활동 결과물을 목표로 하여야 합니다.

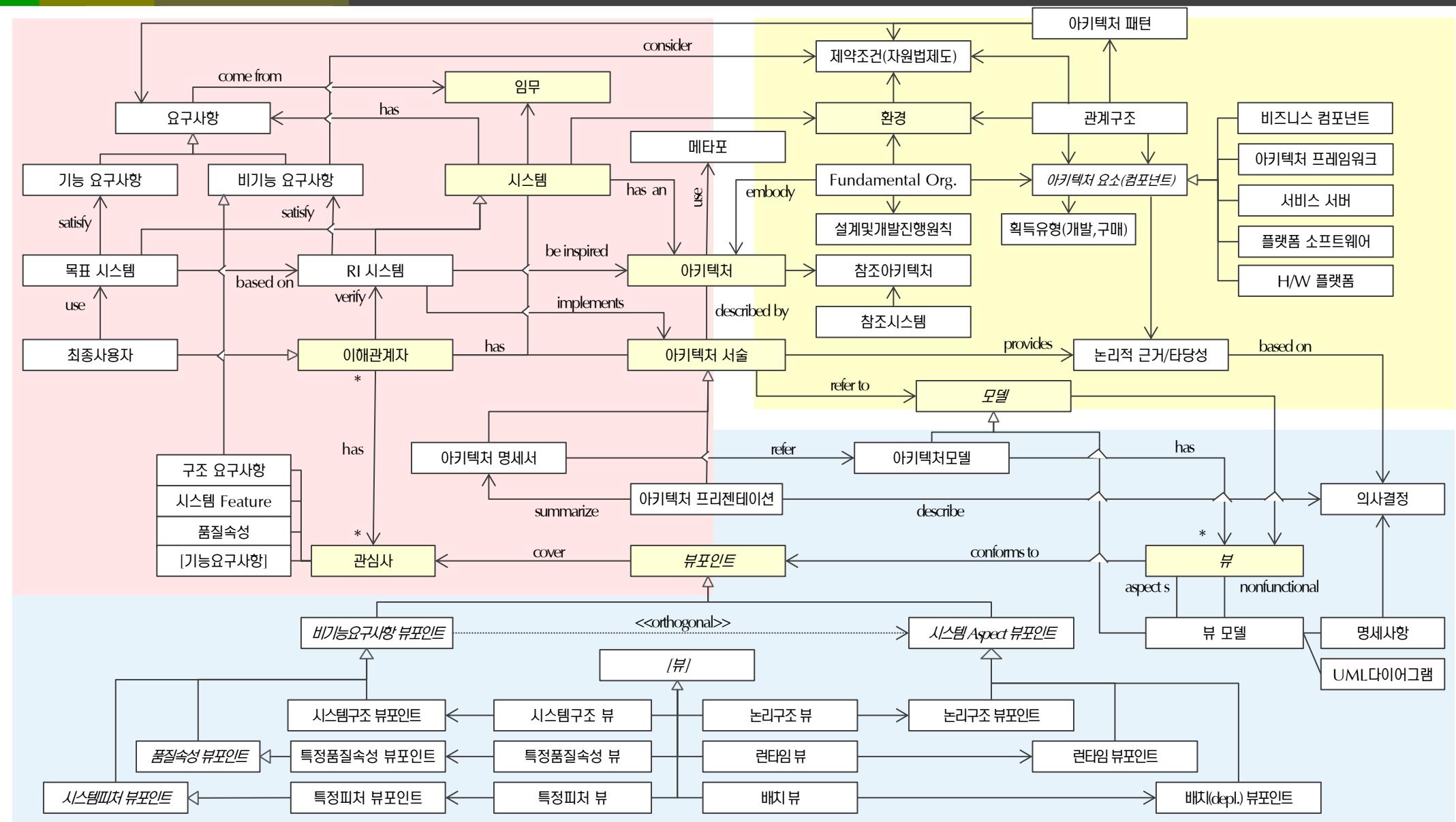


2. IEEE 1471 – Extension (2/2)

- ✓ 정의(definition) – 소프트웨어 아키텍처를 둘러싼 다양한 개념을 정의하고 서술합니다.
- ✓ 표현(representation) – 서술과 UML을 기반으로 아키텍처를 어떻게 표현할 것인가를 정의합니다.
- ✓ 절차(process) – 아키텍처 설계 절차를 정의하되 [정의]와 [표현]을 바탕으로 합니다.
- ✓ 예제(reference implementation) – 실무자들이 직접 참조할 수 있는 완전한 예제를 제시합니다.

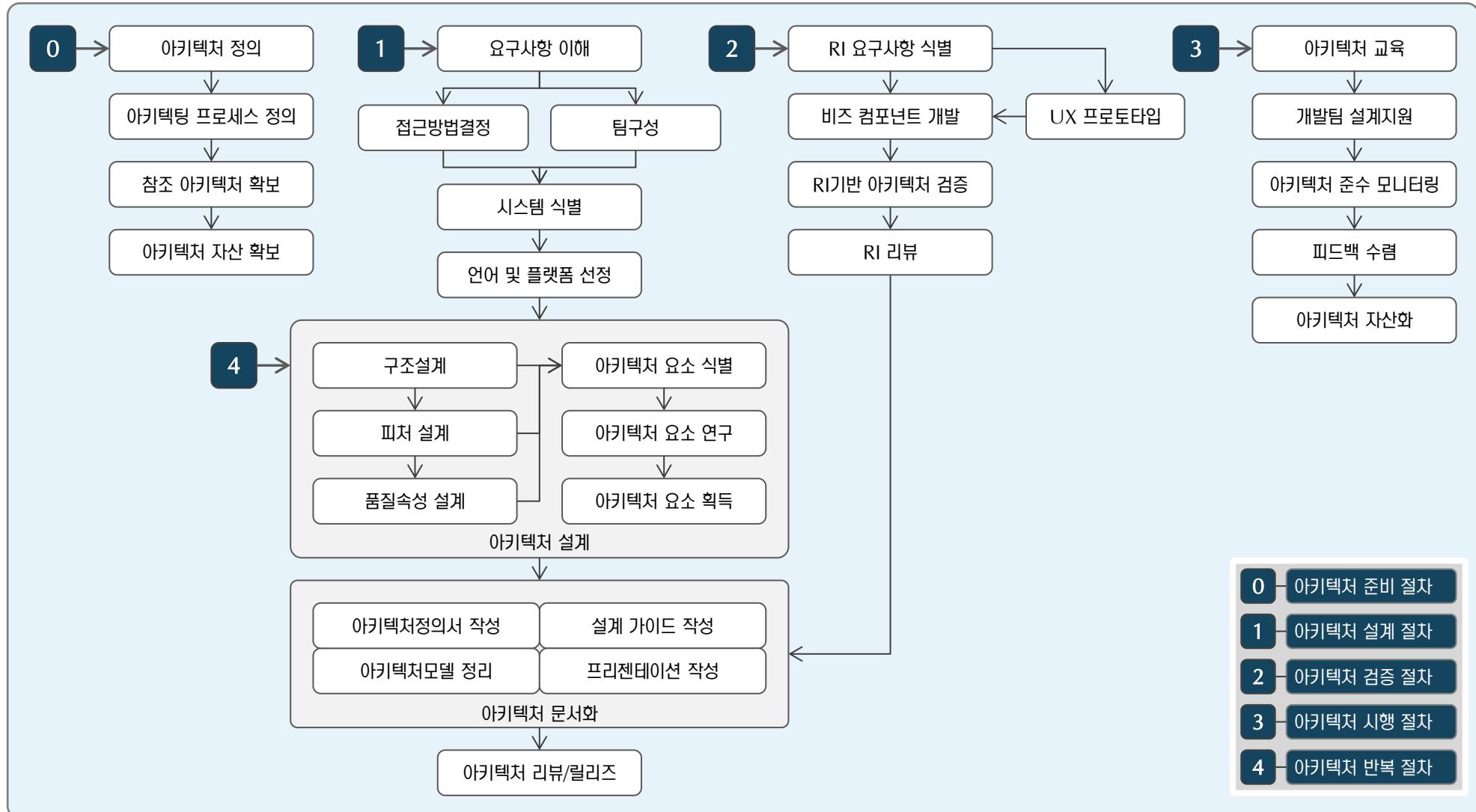


3. IEEE 1471 – 개념 모델 확장 [예]



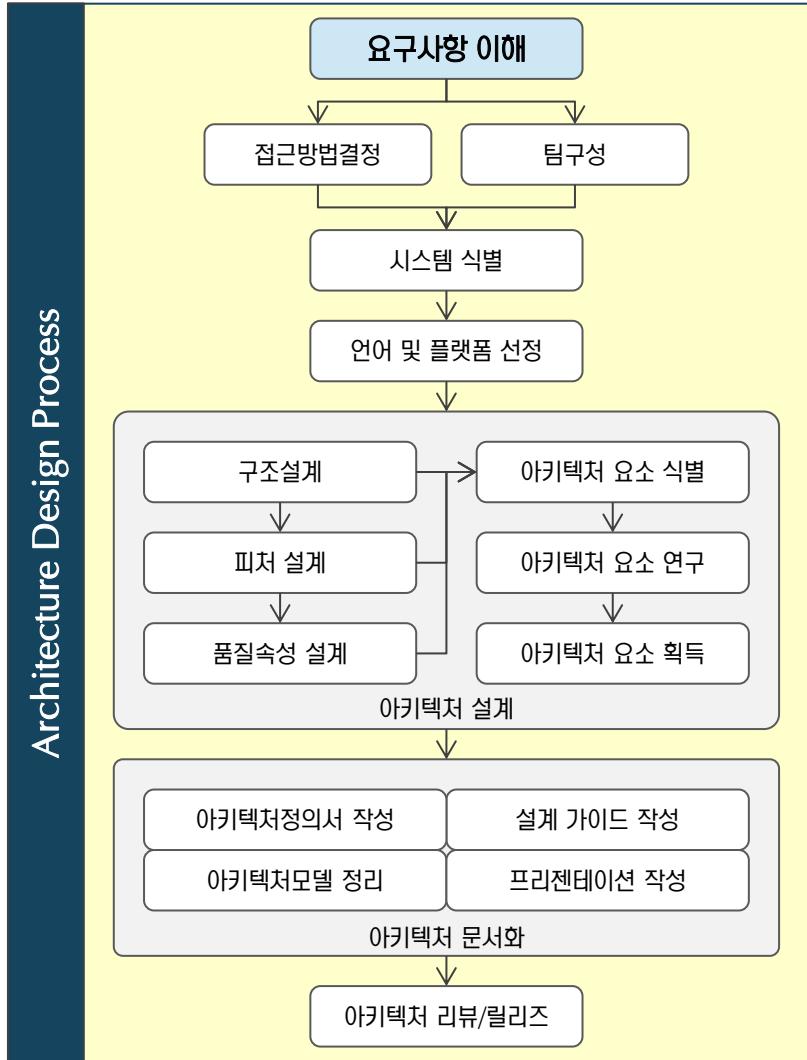
4. New Architecting Process Example(1/5)

- ✓ 아키텍팅 프로세스는 IEEE 1471 확장 모델을 바탕으로 합니다.

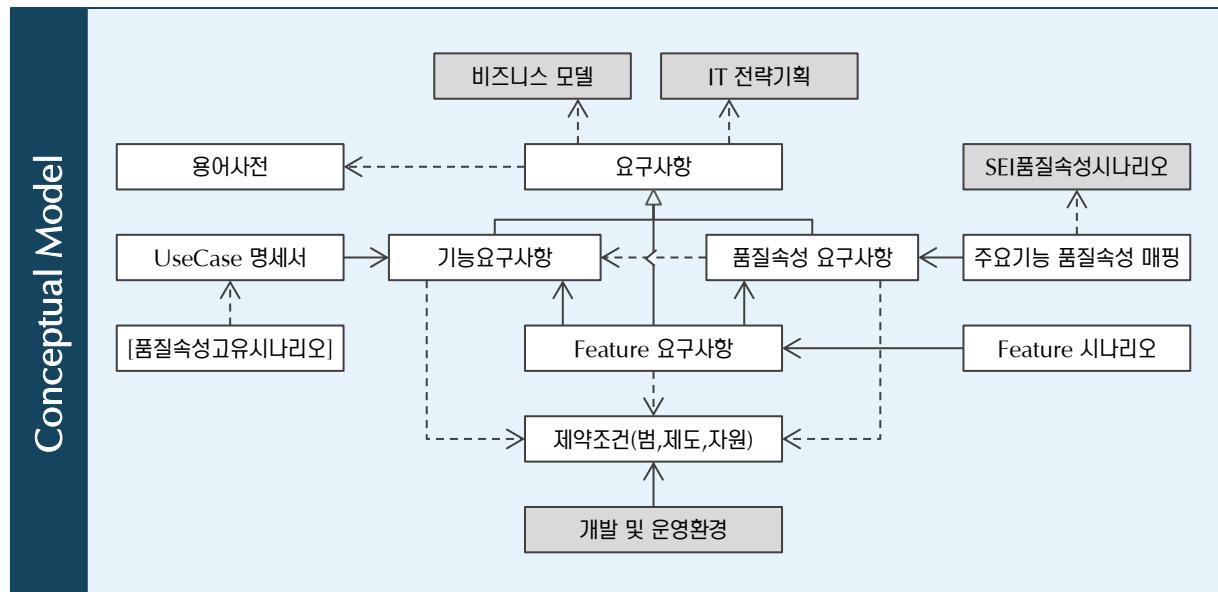


4. New Architecting Process Example(2/5)

✓ 아키텍처 요구사항을 이해하고 기록합니다.

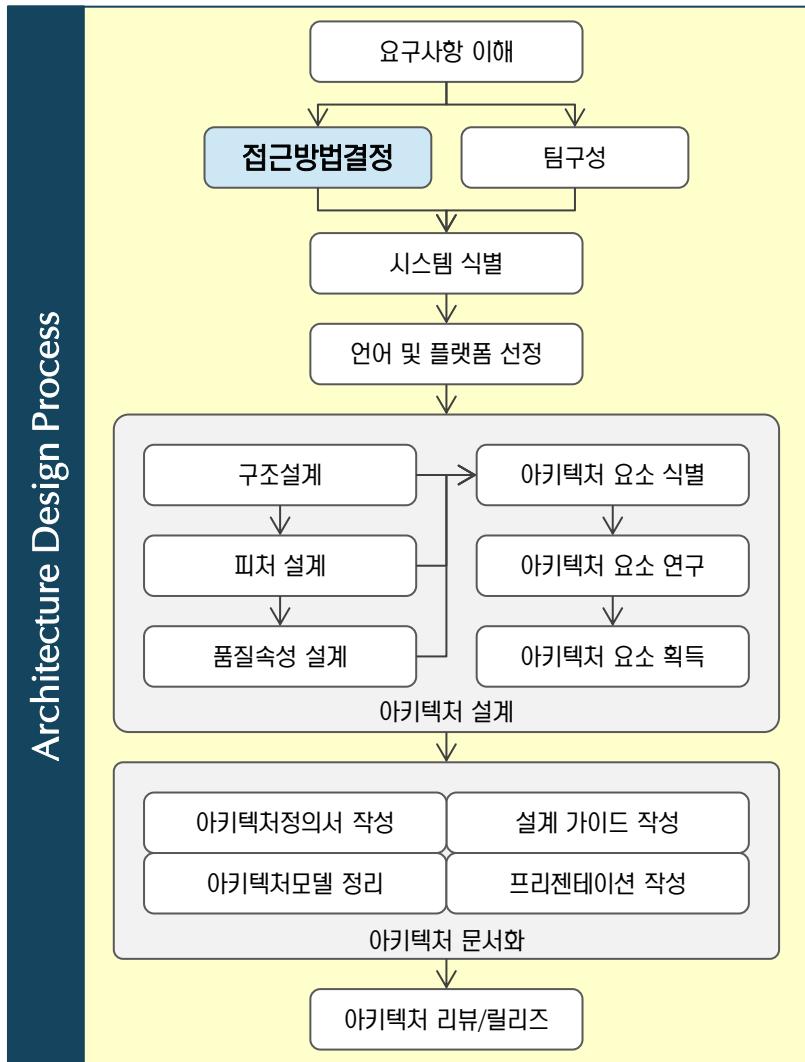


- ❖ 아키텍처 요구사항은 품질속성과 피처(Feature) 임
- ❖ 아키텍처 요구사항은 환경이 제공하는 제약조건을 고려함
- ❖ 품질속성 요구는 주요 기능-품질속성 맵으로 정리함
- ❖ SEI 스타일의 품질속성 시나리오를 사용할 수도 있음
- ❖ Feature 요구는 시나리오를 활용함

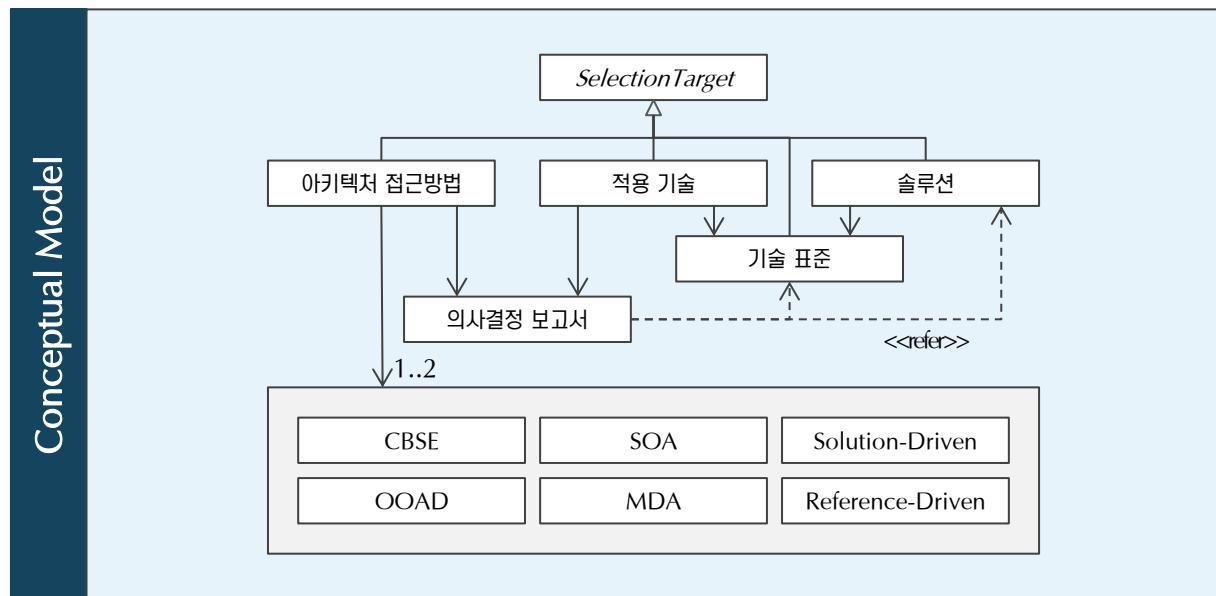


4. New Architecting Process Example(3/5)

- ✓ 시스템 구축을 위한 접근방법, 기술, 표준, 솔루션 등에 대한 방향을 결정합니다.

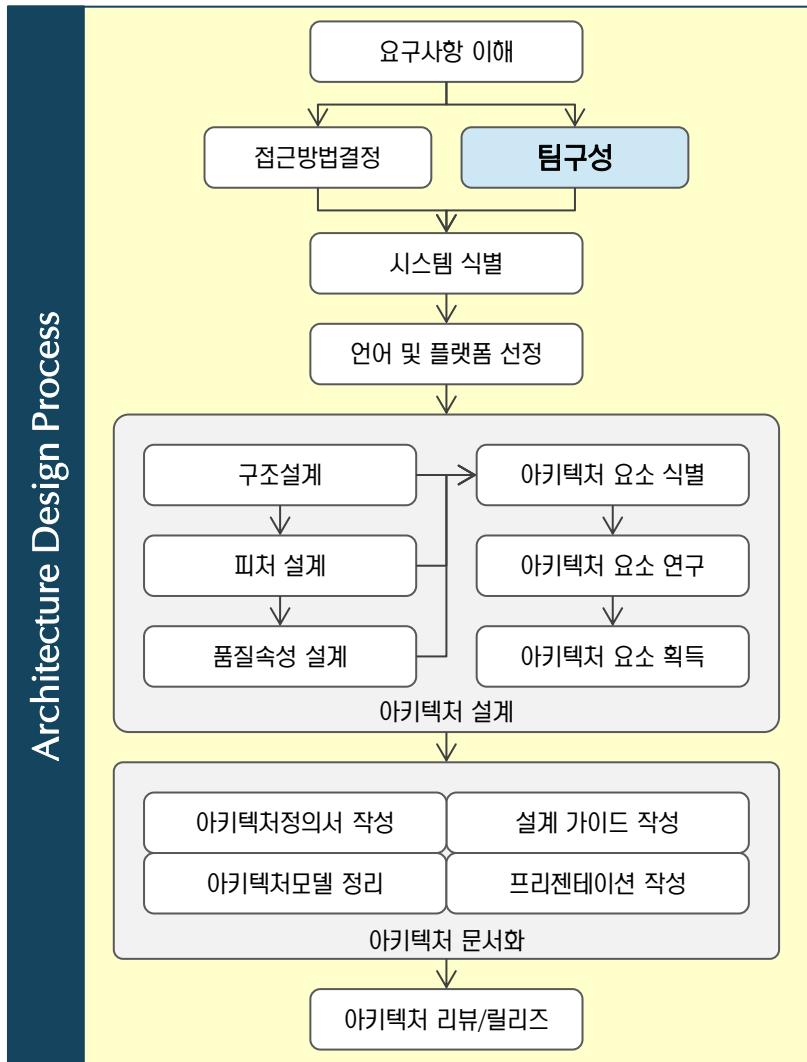


- ❖ 기준이 되는 접근방법을 의사 결정보고서를 통해 결정함
- ❖ 적용기술, 관련 표준, 핵심 솔루션 등에 대한 Study를 진행함
- ❖ [주의] 초기에 모든 솔루션을 검토할 필요는 없음

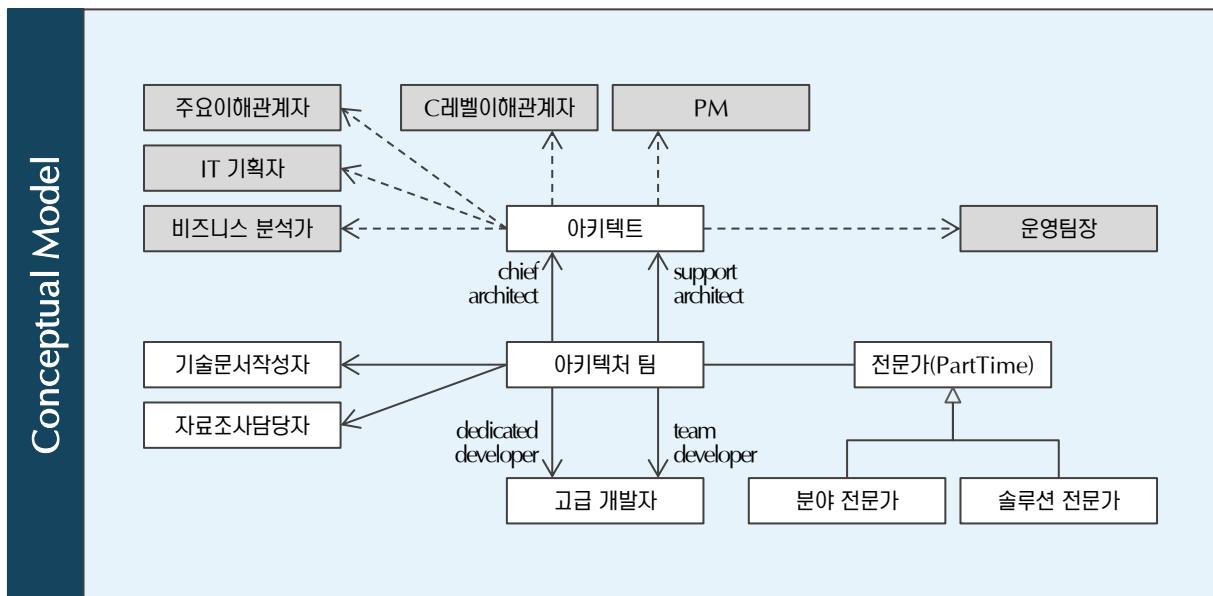


4. New Architecting Process Example(4/5)

✓ 아키텍처 팀을 구성합니다.

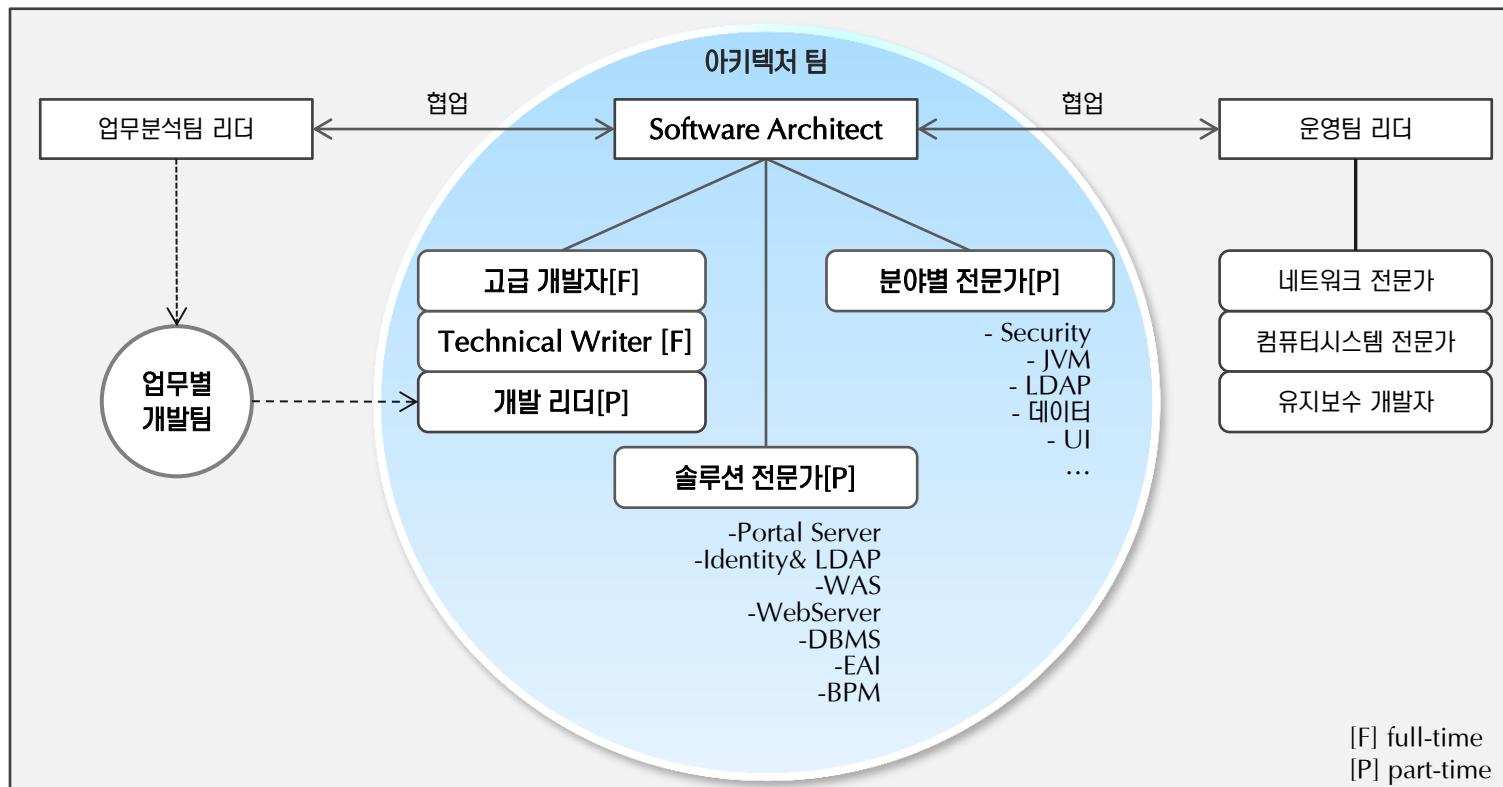


- ❖ 기준이 되는 접근방법을 의사결정보고서를 통해 결정함
- ❖ 적용기술, 관련 표준, 핵심 솔루션 등에 대한 Study를 진행함
- ❖ [주의] 초기에 모든 솔루션을 검토할 필요는 없음



4. New Architecting Process Example(5/5)

- ✓ 아키텍처 팀 구성은 AA, DA, TA, BA 역할로 채워서는 안됩니다.
- ✓ 아키텍처 팀의 핵심은 고급 개발자와 파트-타임으로 참여하는 솔루션 전문가, 또는 분야별 전문가입니다.
- ✓ 하드웨어, 네트워크 등은 아키텍처 팀의 범주를 넘어섭니다.
- ✓ 프로젝트와 사용 기술 특성에 따라 아키텍처 팀 규모를 결정합니다.



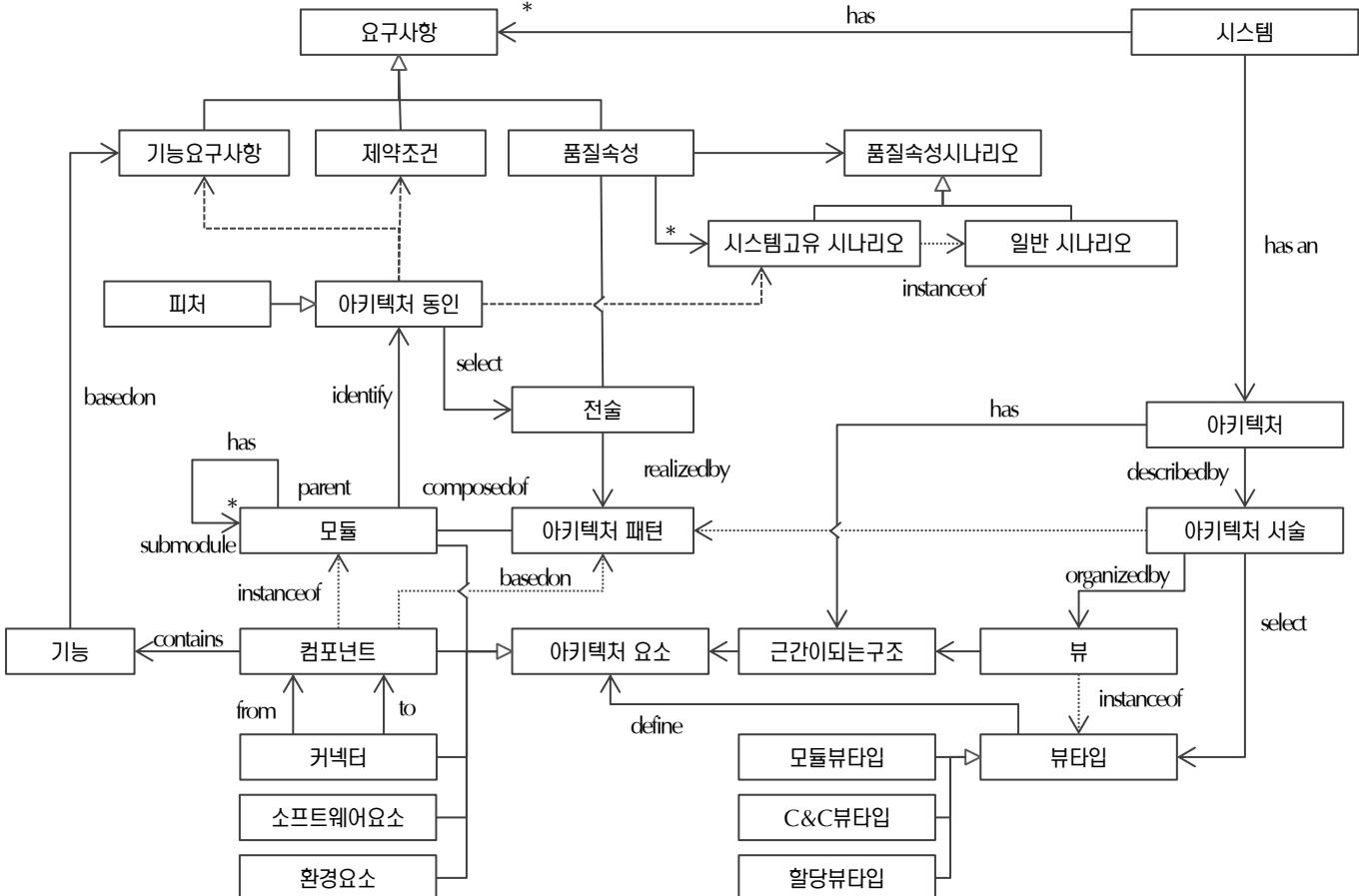
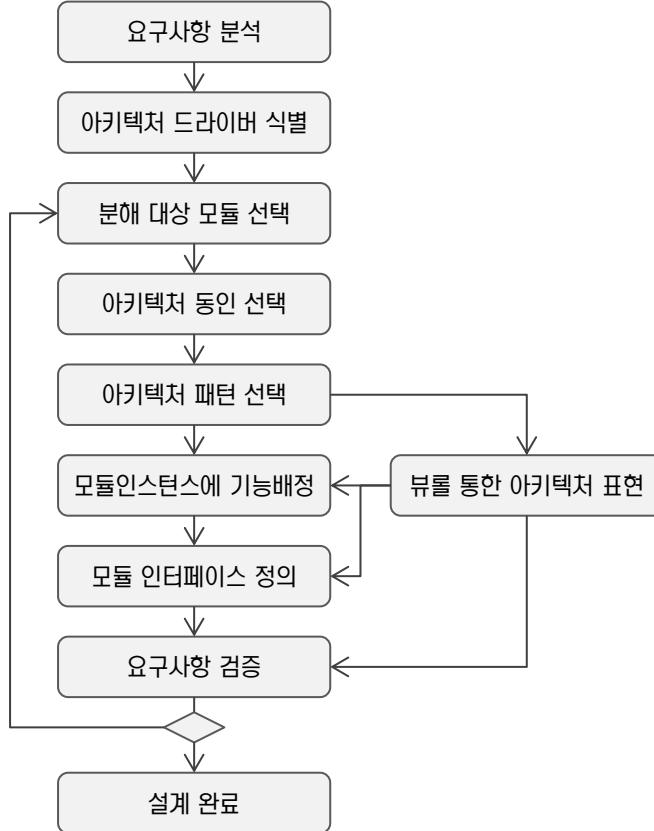
5. SEI 개념모델과 프로세스

✓ 아키텍처 기반 설계 방법으로 아키텍처 설계와 기능 설계를 구분하지 않았습니다.

✓ 아파트 한 동과 같은 단일 시스템에서는 적절하지만, 대규모 아파트 단지와 같은 복수 기능 컴포넌트를 가진 시스템에는 적절하지 않습니다.

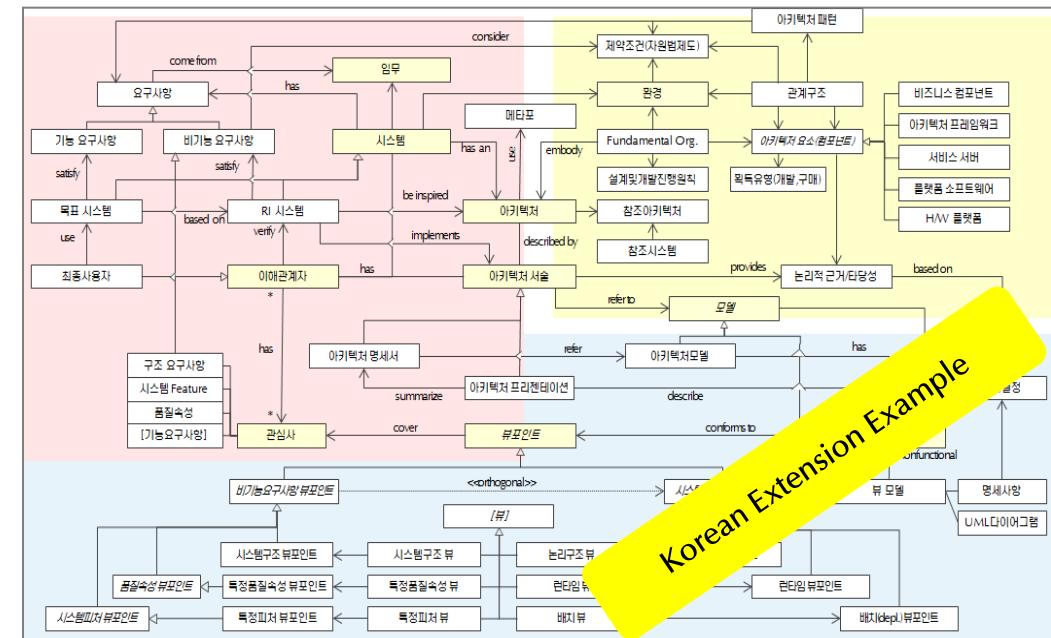
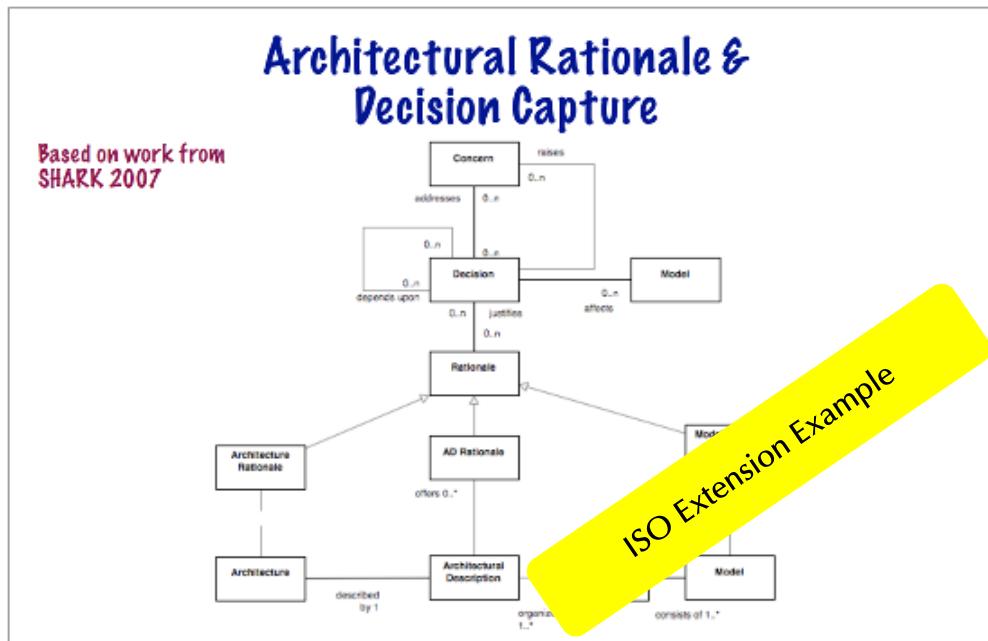
- 시스템 고유 시나리오
 - ✓ 문가폐를 위한 장치(device)와 제어기는 제품 라인에서 다양한 제품 유형별로 다른
 - ✓ 서로 다른 제품에 사용되는 프로세스는 서로 다른
 - ✓ 문을 내리는 중에 장애물(사람이나 물체)을 발견하면, 0.1초내에 정지해야 함
 - ✓ 차고문가폐기는 제품 고유의 진단 프로토콜을 사용하는 흡 정보 시스템이 관리 및 진단을 위해 접근할 수 있어야 함

[0] 아키텍처동인 실시간수행성능[0.1초], 제품 라인 지원을 위한 변경용이성, 온라인 진단



6. IEEE 1471 – 확장 요약

- ✓ IEEE 1471 표준 자체는 매우 추상화 수준이 높은 모델입니다.
- ✓ IEEE 1471는 일부 확장이 이루어지고 있긴 하지만 매우 제한된 범위를 가집니다.
- ✓ 따라서, IEEE 1471를 확장하여, 즉 도메인에 특화(specialization)하여 활용하여야 합니다.
- ✓ 한국아키텍트연합회에서 IEEE 1471 – Korean Extension 정의 작업을 진행하였음, 현재 중지된 상태입니다.



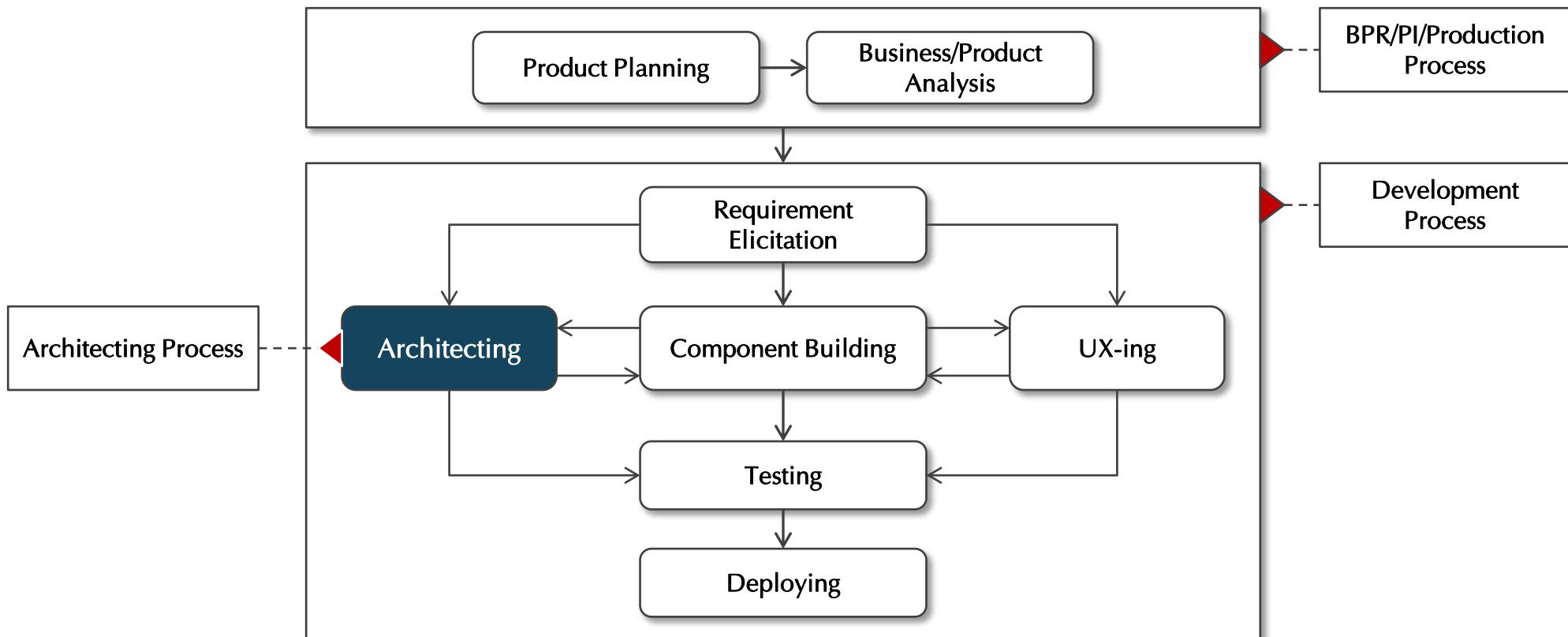


7. Architecting Process

1. Development process
2. Why Architecting Process ?
3. Architecting Process History
4. Reference Architecting Process - ADD
5. IEEE 1471 개념 모델 확장
6. New Architecting Process
7. 토의

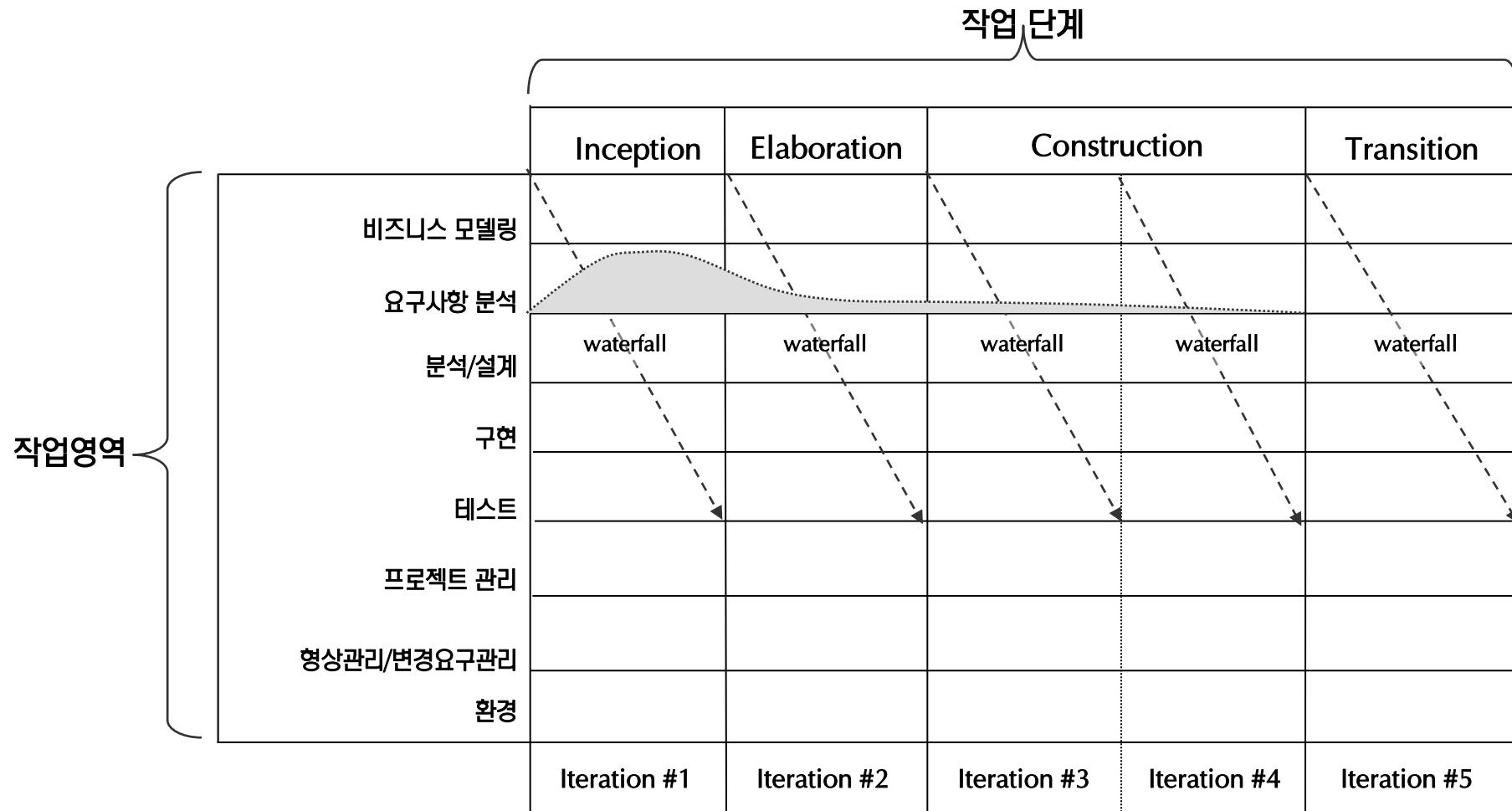
1. Development Process – Nextree 스타일

- ✓ SW를 기반으로 하는 제품이나 업무용 Application을 개발할 때, 다양한 프로세스가 필요합니다.
- ✓ 아키텍팅 활동의 목표 시스템의 틀과 원칙을 설계하는 중요한 활동입니다.
- ✓ 아키텍팅 프로세스는 개발 프로세스의 하위 프로세스이며, 아키텍처 설계, RI 등을 포함합니다.



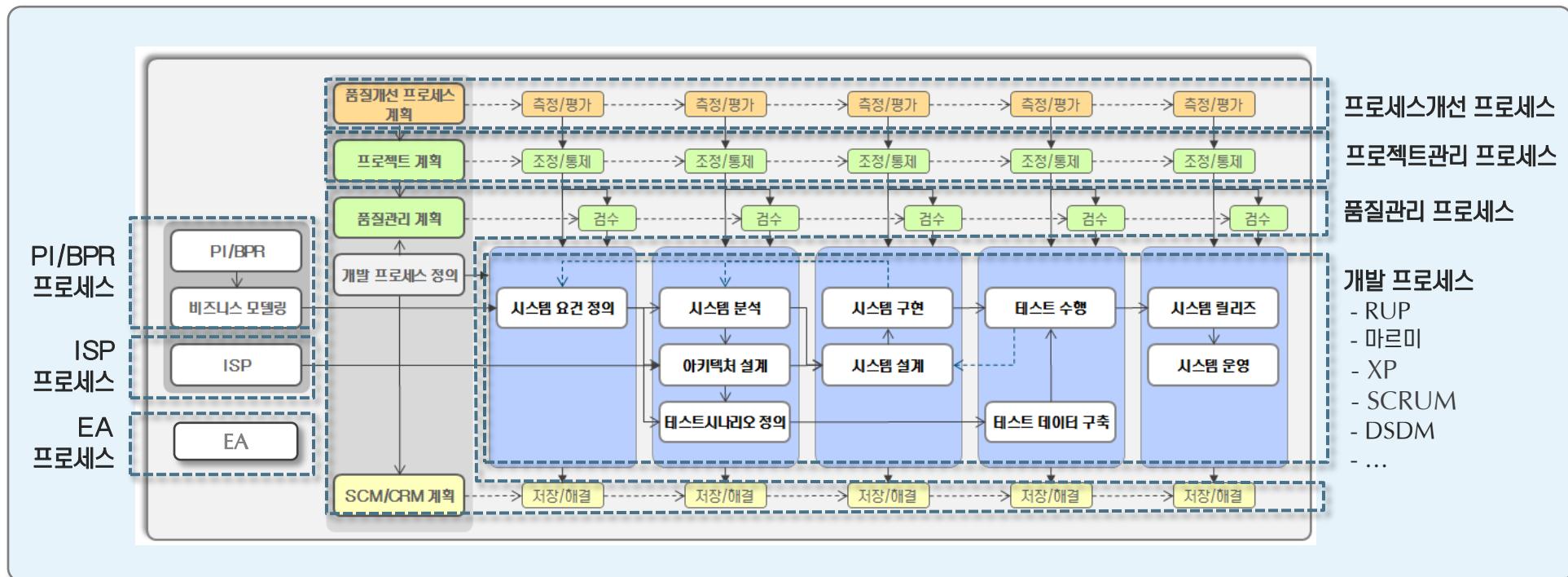
1. Development Process – RUP 스타일

- ✓ RUP는 Objectory를 모태로 하여 가장 완성된 형태의 개발 프로세스로 발전했습니다.
- ✓ Iterative & Incremental 사상을 기반으로 개발을 진행함으로써, Waterfall을 탈피하기 시작합니다.



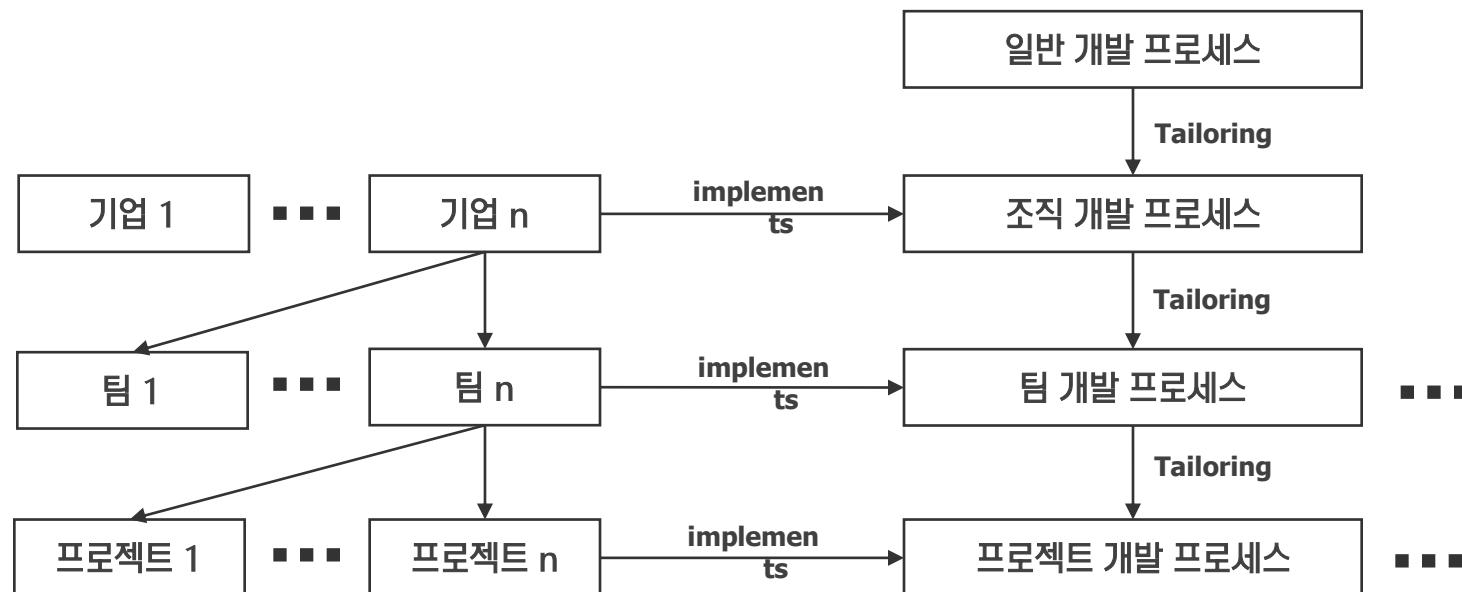
1. Development Process – 대규모 조직

- ✓ 대규모 조직은 다양한 종류의 프로세스를 가지고 있으며, 개발 프로세스는 그 중에 하나입니다.
- ✓ 프로젝트 유형에 따라 개발 프로세스를 선택하여 조정한 후에 사용하는 것이 일반적입니다.
- ✓ 프로세스는 역할 그룹(관리, 개발, 도메인, 솔루션, 시스템 엔지니어링 등) 별로 서로 다릅니다.



1. Development Process – 프로세스 조정

- ✓ 조직 개발 프로세스는 일반 개발 프로세스로부터 조정(tailoring)한 결과입니다.
- ✓ 팀 개발 프로세스는 조직의 표준 개발 프로세스를 팀의 개발작업 특성에 맞추어 놓은 것입니다.
- ✓ 프로젝트 개발 프로세스는 팀의 개발 프로세스를 해당 프로젝트의 특성에 맞추어 놓은 것입니다.



2. Why Architecting Process ?

- ✓ 프로세스는 태권도의 품새(Form)와 비슷합니다.
- ✓ 품새는 순서, 자세를 강조하지만, 실전에서는 스피드, 파워, 순간 판단력이 중요합니다.
- ✓ 실전은 품새가 조정(Tailoring)된 결과로 볼 수 있습니다.



2. Why Architecting Process ?

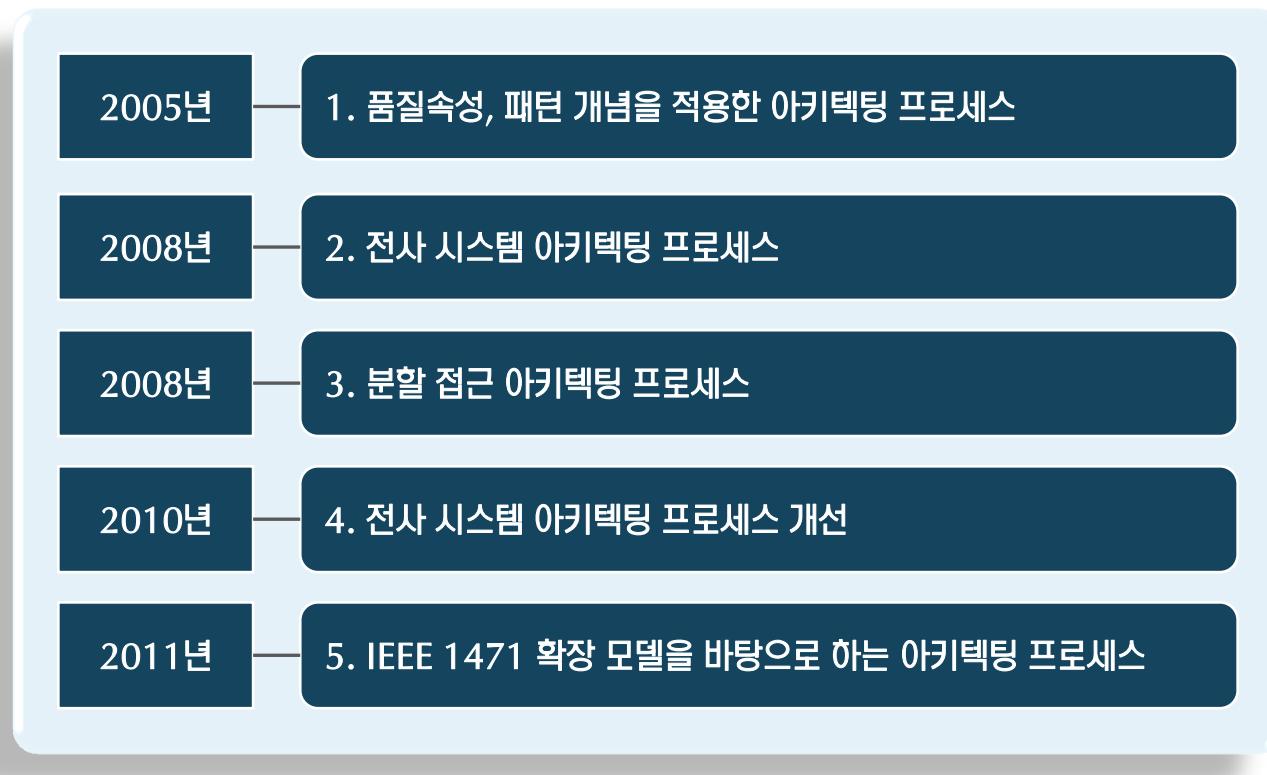
- ✓ 일단 품새가 만들어지면,
- ✓ 경험자의 노하우를 정형화된 형태로 품새에 담을 수 있으며,
- ✓ 수련자는 경험자의 기술을 품새를 통해서 전달받을 수 있습니다.
- ✓ 이런 과정을 통해서 경험과 지식 축적되고 전달되는 선순환 과정을 반복하면서 작업 성숙도가 높아집니다.



▲ 세계태권도연맹(WTF)이 품새 경기에 프리스타일 종목 도입을 추진하고 있다. (자료사진)

3. Architecting Process History (1/5)

- ✓ 우리는 아키텍팅 프로세스가 필요하다. 경험을 축적하고 전달하는 수단으로.
- ✓ 2005년 이후 다섯 차례에 걸쳐 아키텍팅 프로세스를 발전시켜 왔습니다.
- ✓ 개선할 때마다, 품질 속성, 전사 시스템, 분할 설계 등의 키워드가 있었습니다.
- ✓ 최근에는 IEEE 1471 – 확장모델을 기반으로 개발 프로세스를 개선하였습니다.

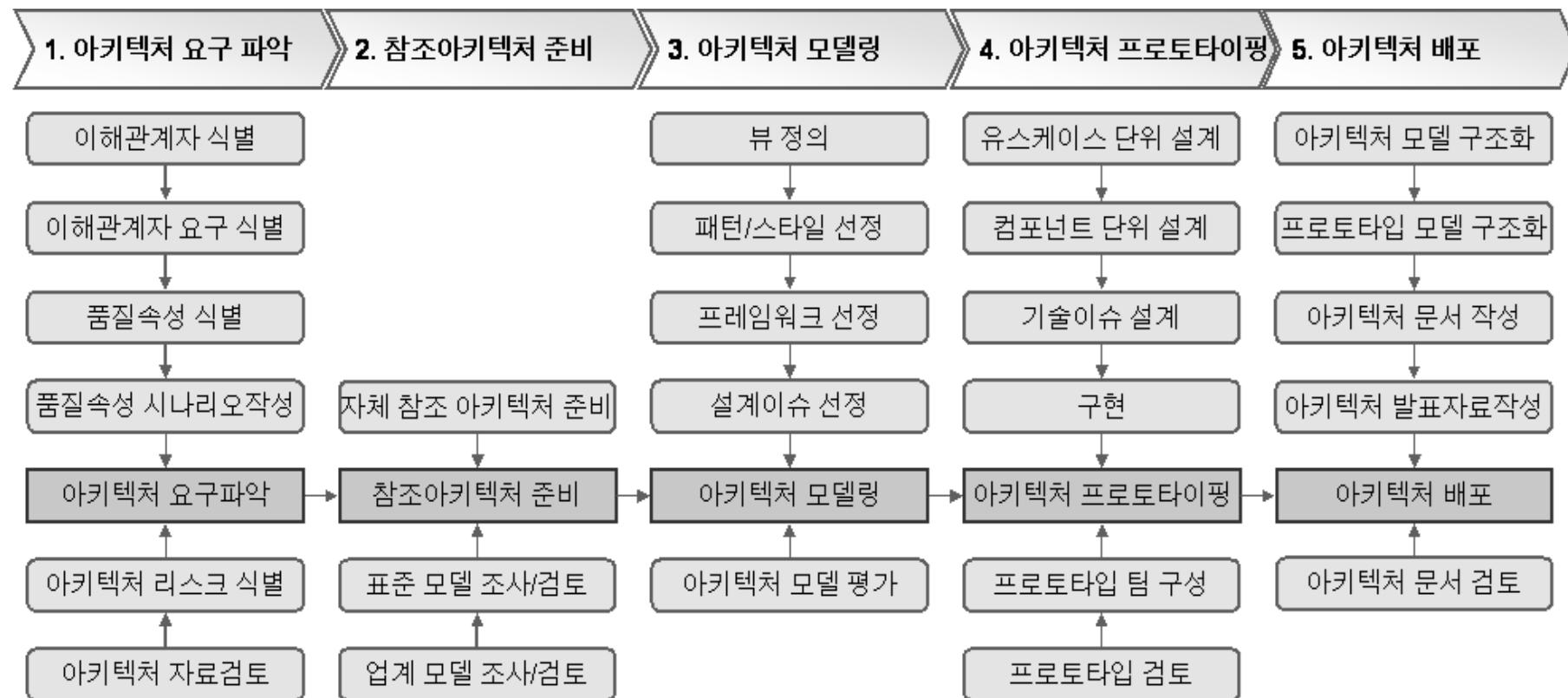


3. Architecting Process History (2/5)

- ✓ 2005년 금융회사의 차세대 프로젝트 과정에서 정의했습니다.
 - ✓ 품질속성, 패턴, 프로토타입을 특징으로 하는 프로세스입니다.
 - ✓ 전사 범위로 기획을 했으나 전사 시스템 아키텍팅 프로세스의

2005년

1. 품질속성, 패턴 개념을 적용한 아키텍팅 프로세스



3. Architecting Process History (3/5)

- ✓ 전사 시스템 아키텍팅의 순환적인 특성을 반영했습니다.
- ✓ 독립형 시스템 아키텍팅 프로세스는 [아키텍처 요소 획득]으로 한정하였습니다.
- ✓ 아키텍처 적용(enforcement) 개념을 프로세스에 도입하였습니다.

2008년

2. 전사 시스템 아키텍팅 프로세스

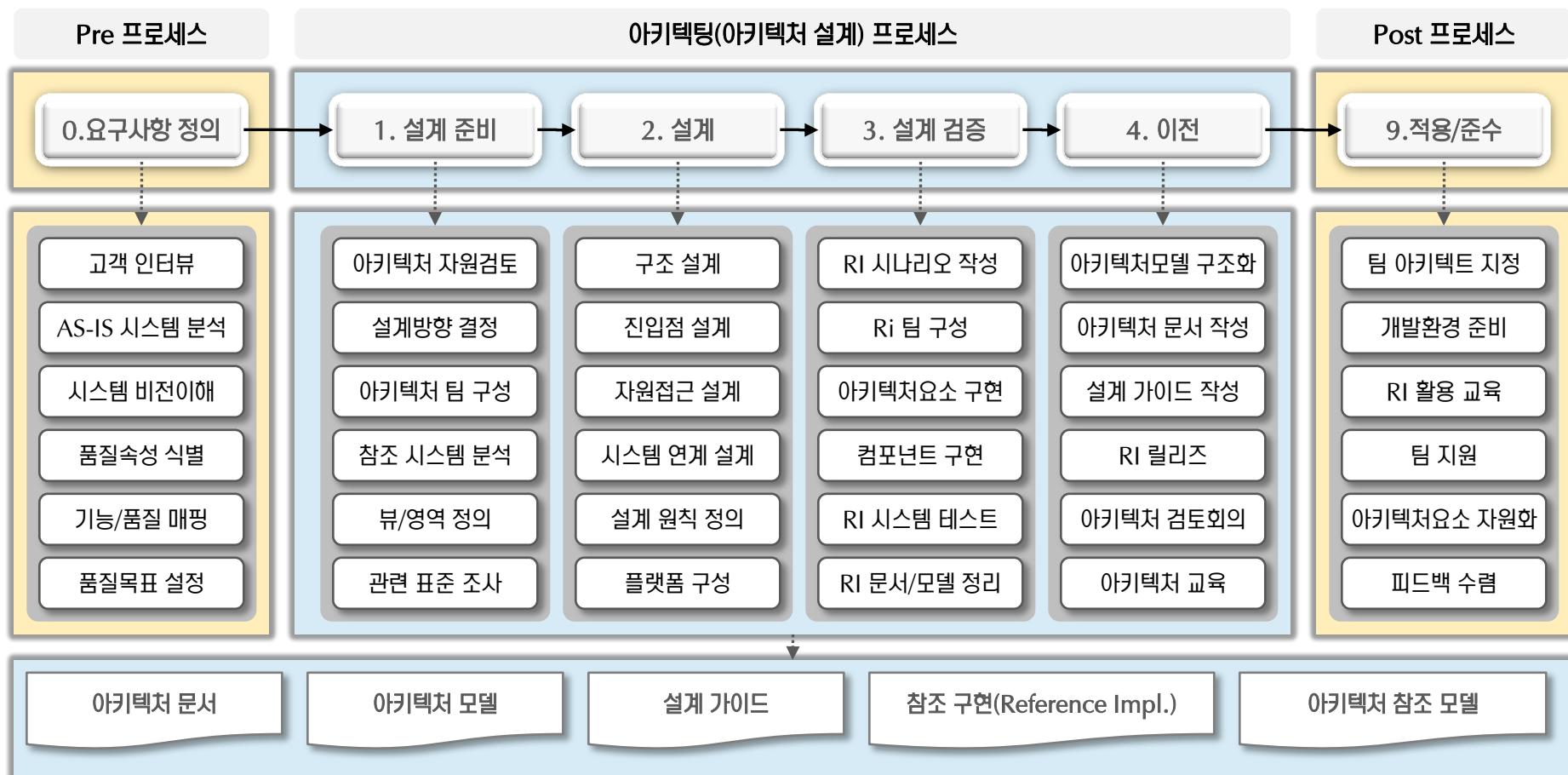


3. Architecting Process History (4/5)

- ✓ 독립형 애플리케이션 아키텍처 설계 시 분할-정복개념 적용하였습니다.
- ✓ 구조, 진입점, 자원접근, 시스템 연계, 플랫폼 적용, 설계 원칙 등으로 나누어 설계하였습니다.
- ✓ 뷰는 4 + 1 로 고정하였습니다.

2008년

- 3. 분할 접근 아키텍팅 프로세스



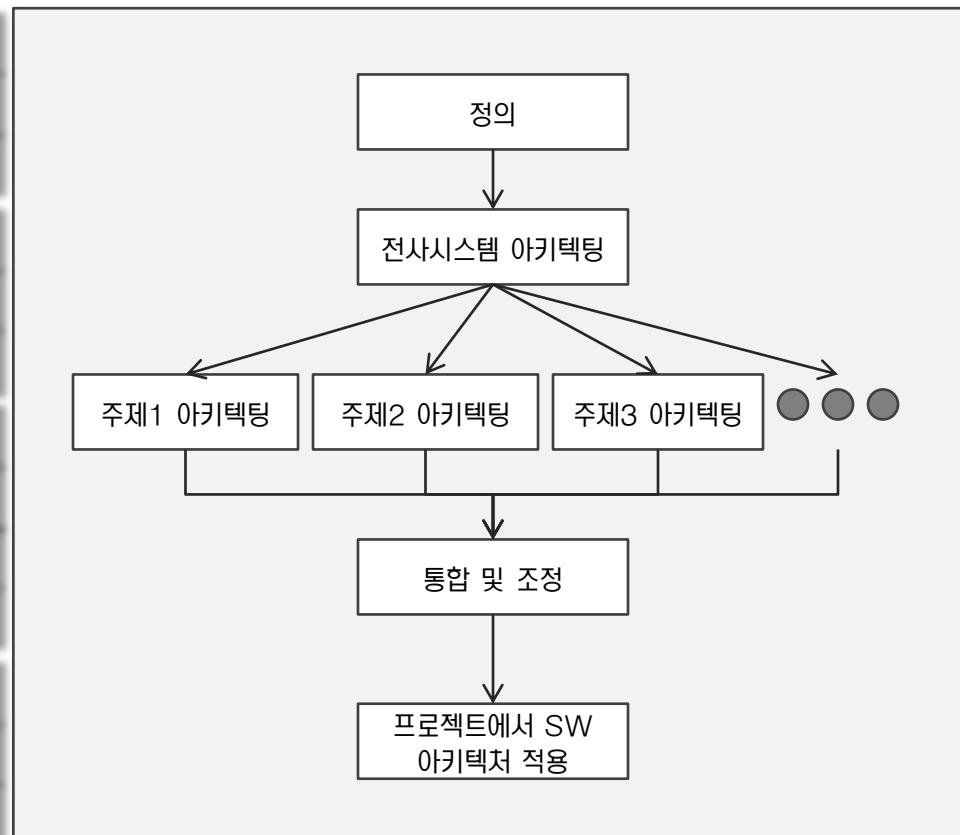
3. Architecting Process History (5/5)

- ✓ 기존의 전사 시스템 아키텍팅 프로세스를 개선하였습니다.
- ✓ 00사 차세대 프로젝트 준비를 위해 기존의 아키텍팅 프로세스를 개선하였습니다.
- ✓ 주제별 아키텍팅이라는 개념을 그대로 사용했습니다.

2010년

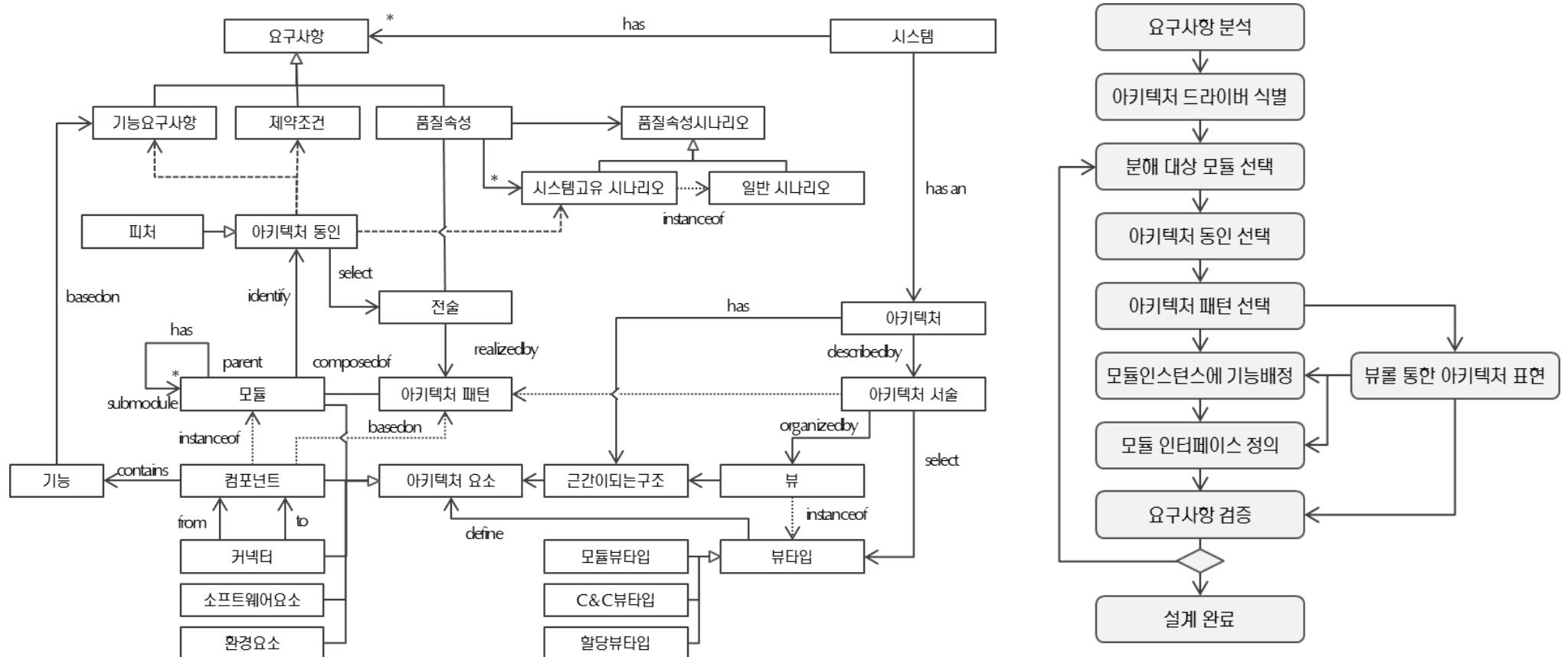
4. 전사 시스템 아키텍팅 프로세스 개선

1. 정의	1.1 다양한 아키텍처 개념 정의 및 조정 1.2 전사 시스템 아키텍팅 프로세스 정의 1.3 아키텍팅 접근방법 정의 – 뷰와 모델
2. 전사 시스템 아키텍팅	2.1 AS-IS 시스템 아키텍처 표현 및 분석 2.2 전사 시스템 아키텍처 설계 2.3 아키텍팅 주제 식별 및 우선순위 결정
3. 주제별 아키텍팅	3.1 주제 영역 연구 및 접근방법 결정 3.2 SW 아키텍처 설계 3.3 아키텍처 요소 획득 3.4 RI를 통한 검증
4. 통합 및 조정	4.1 주제별 아키텍팅 결과 통합 및 조정 4.2 전사 아키텍처 가이드 업데이트 4.3 아키텍처 적용(enforcement) 계획 수립

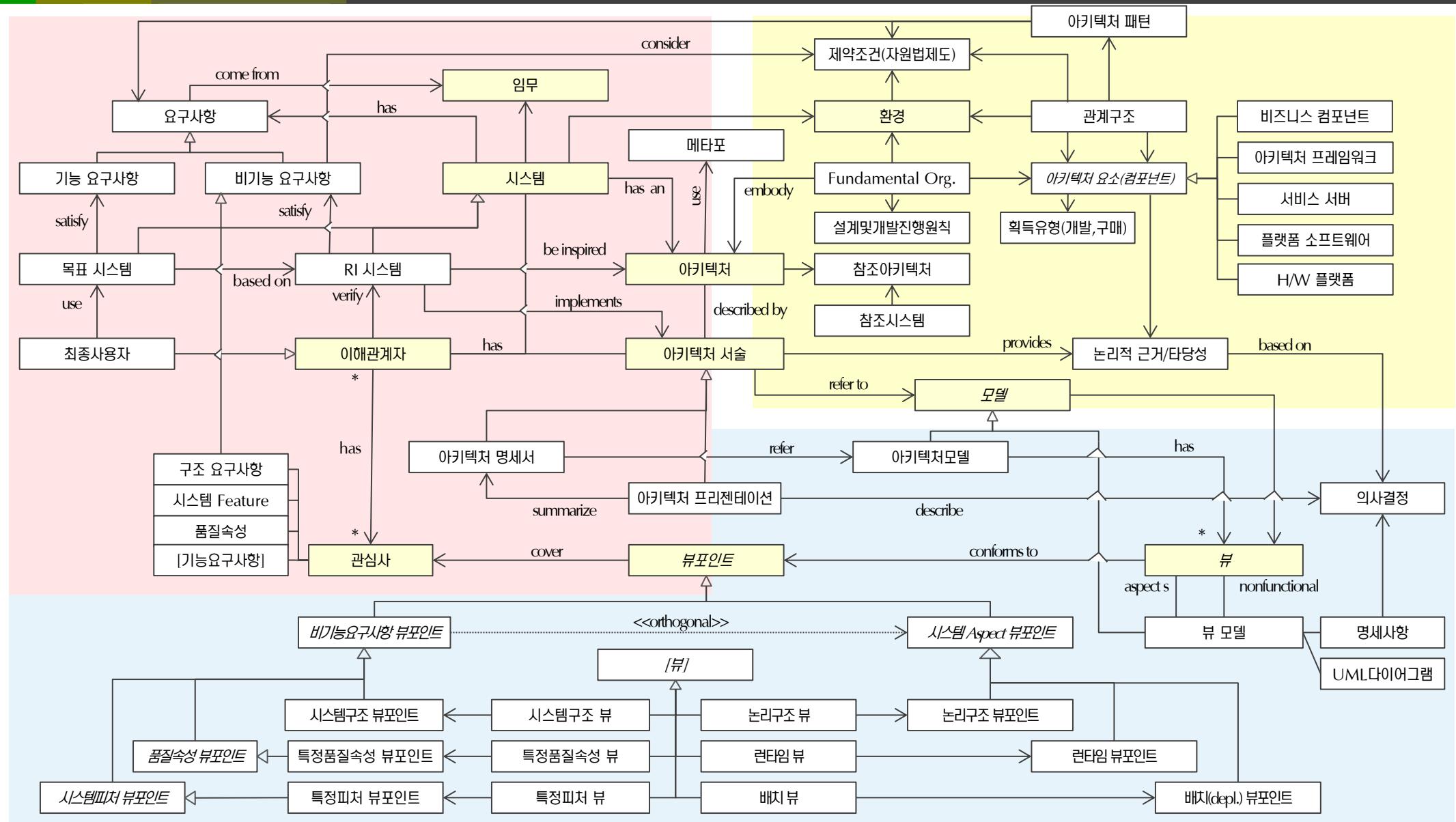


4. Reference Architecting Process – ADD

- ✓ ADD 프로세스는 현대의 다양한 아키텍팅 프로세스의 바탕이 되는 프로세스입니다.
- ✓ 소규모 시스템 또는 Embedded 시스템 개발에서는 아직도 활발하게 사용되는 프로세스입니다.
- ✓ 프로젝트에서 개발하려는 목표 시스템과 개발팀의 특성에 맞추어 조정(tailoring) 하여야 사용합니다.



5. IEEE 1471 – 개념 모델 확장

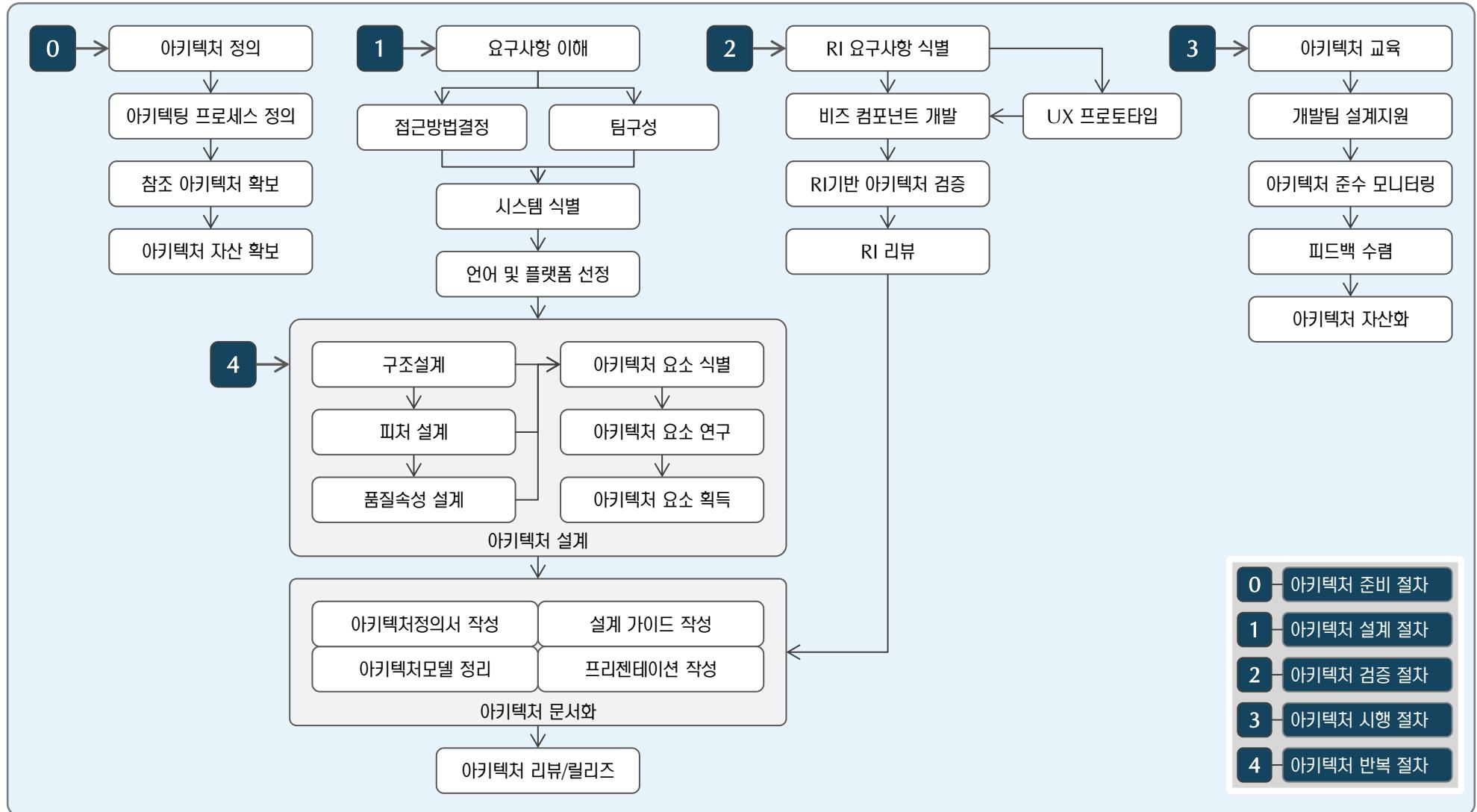


6. New Architecting Process

- ✓ 아키텍팅 프로세스는 IEEE 1471 확장 모델을 기반으로 합니다.

2011년

5. IEEE 1471 확장 모델 기반 아키텍팅 프로세스

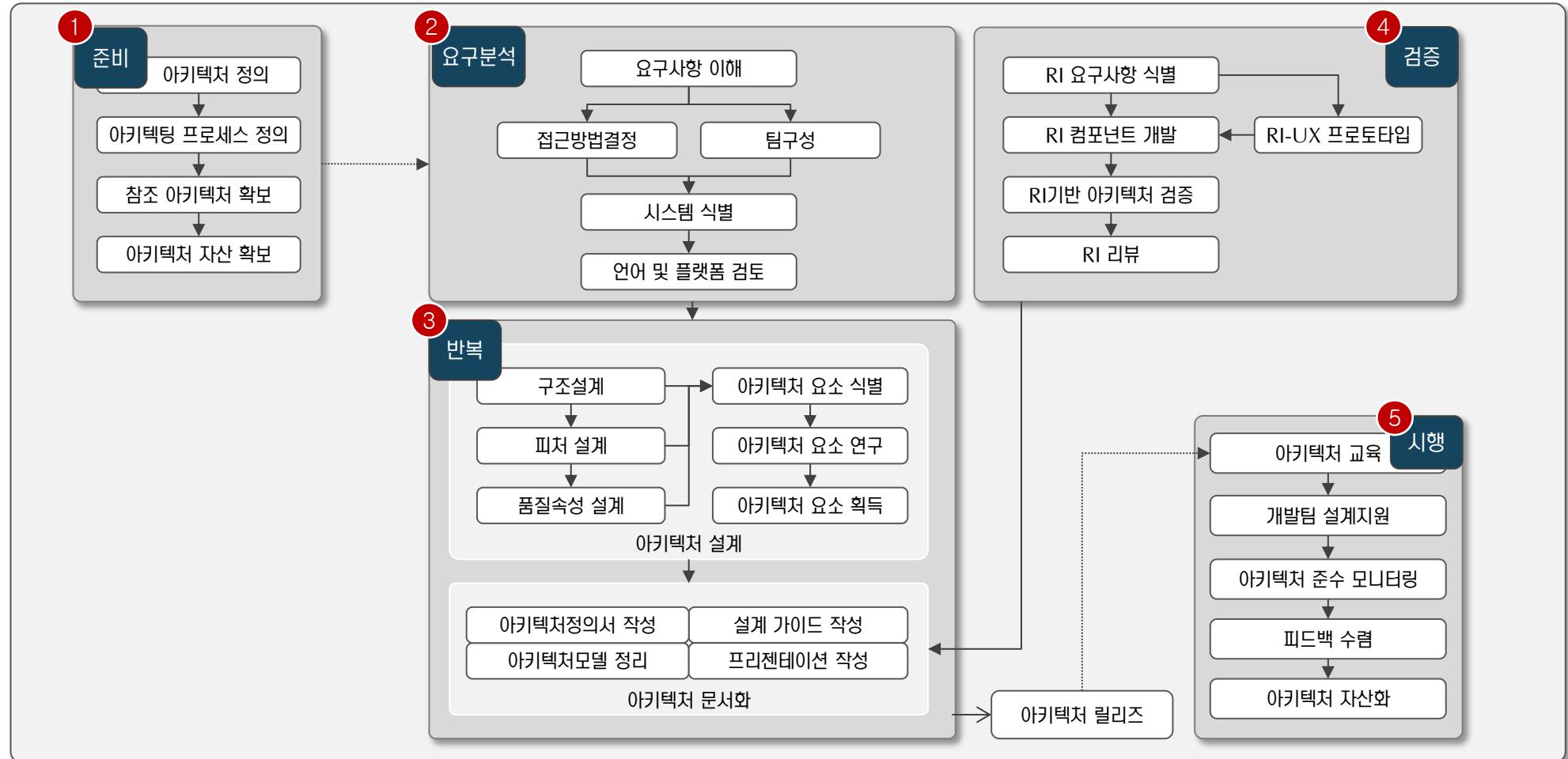


6. New Architecting Process[refined]

- ✓ 아키텍팅 프로세스는 IEEE 1471 확장 모델을 기반으로 합니다.

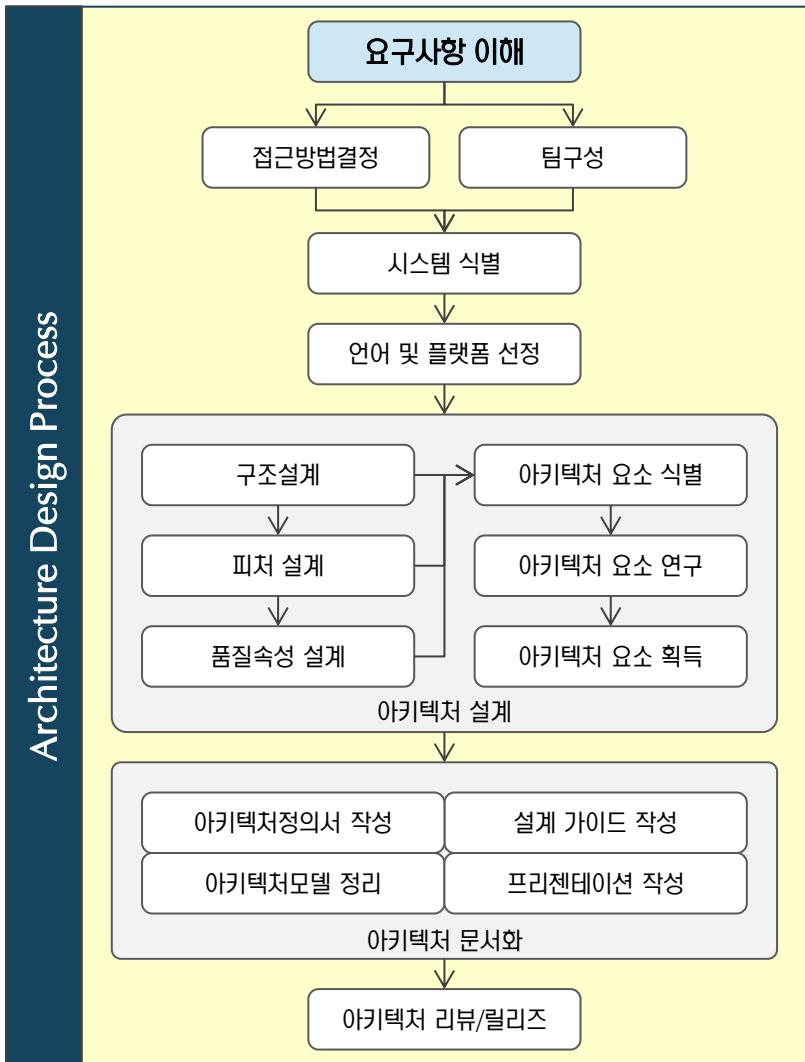
2011년

5. IEEE 1471 확장 모델 기반 아키텍팅 프로세스



6. New Architecting Process – 요구사항 이해

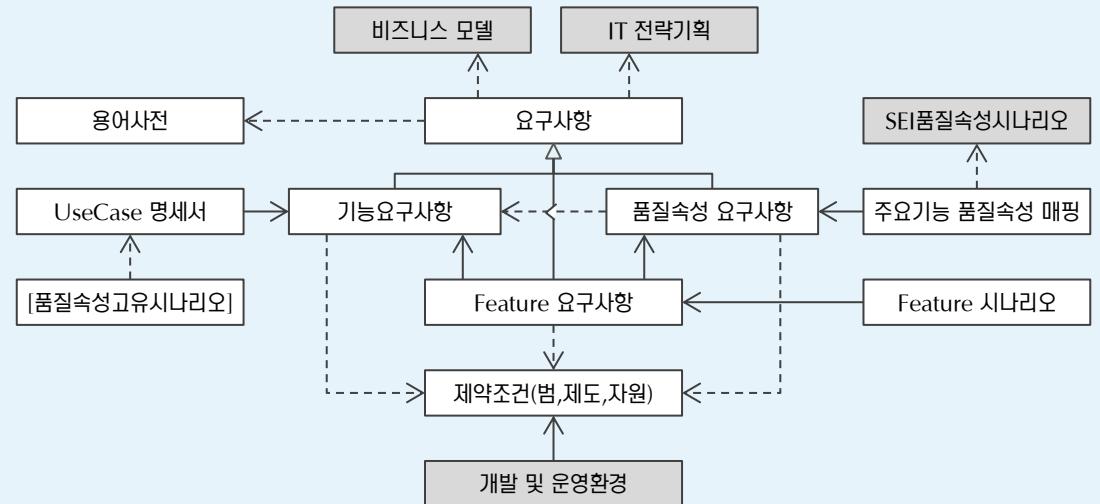
✓ 아키텍처 요구사항을 이해하고 기록합니다.



Activity Guide

- ❖ 아키텍처 요구사항은 품질속성과 피처(Feature) 임
- ❖ 아키텍처 요구사항은 환경이 제공하는 제약조건을 고려함
- ❖ 품질속성 요구는 주요 기능-품질속성 맵으로 정리함
- ❖ SEI 스타일의 품질속성 시나리오를 사용할 수도 있음
- ❖ Feature 요구는 시나리오를 활용함

Conceptual Model



6. New Architecting Process – 요구사항 이해

✓ 품질과 기능의 차이와 역할에 대한 이해는 아키텍처 설계의 출발점입니다.

✓ 기능과 품질은 직교(orthogonal) 함

- 상관:기능

- 교각:품질

✓ 품질은 주로 Cross-cutting 이슈입니다.

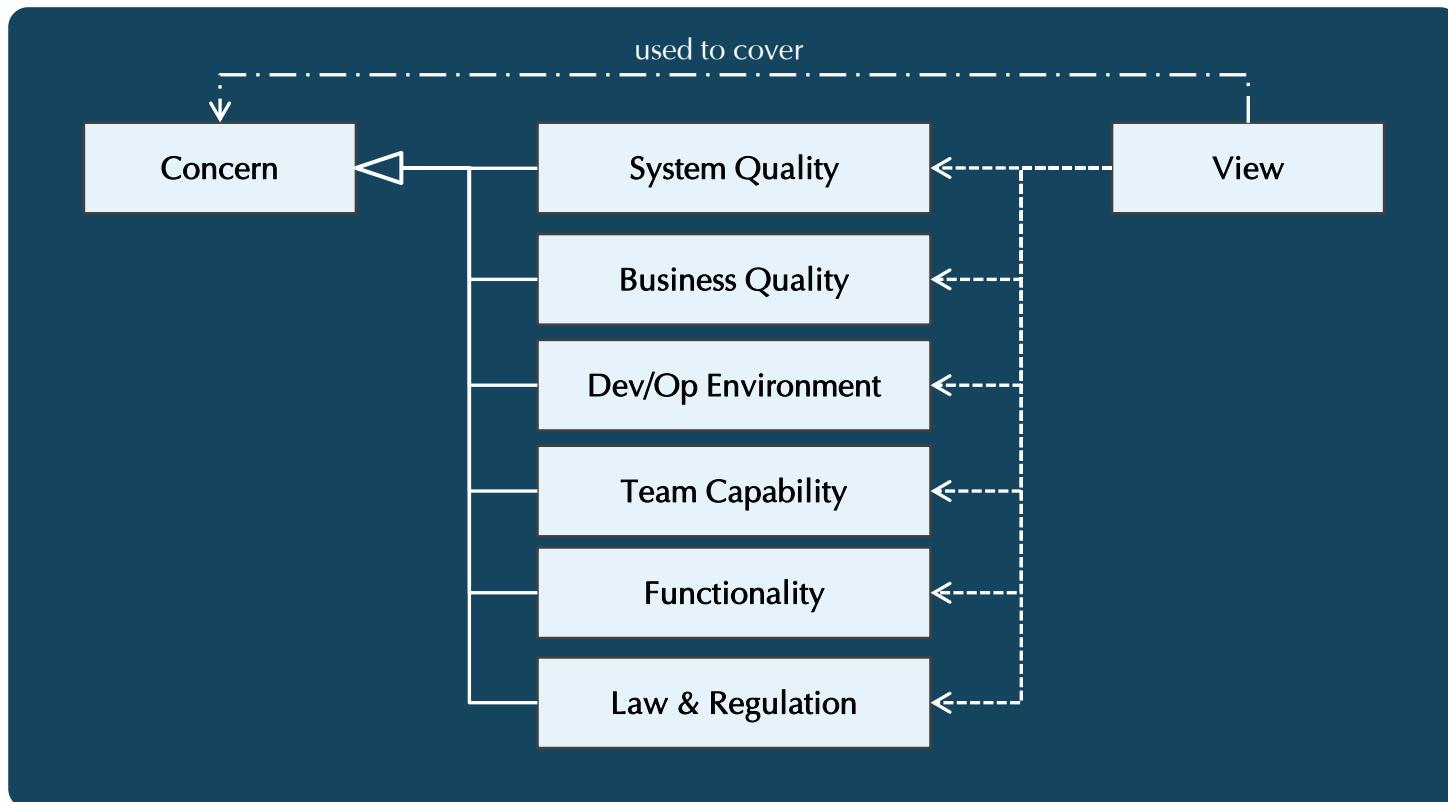
✓ 두 경계선이 흐릿한 경우도 있습니다.

✓ 피처(Feature)는 두 속성의 조합입니다.



6. New Architecting Process – 요구사항 이해

- ✓ 이해관계자의 관심사는 매우 다양하며, 시스템 영역에서 포함되는 것과 그렇지 않은 것들이 있습니다.
- ✓ 여러 관심사를 특정 기준으로 묶음짓고 이를 바라보는 관점을 정의하는데 그것을 뷰(view)라고 합니다.
- ✓ 관심사는 포괄적인 개념으로 시스템 품질이 주를 이루고 있지만, 기능성, 비즈니스 환경, 운영 환경, 법과 제도를 포함하고 있습니다.



6. New Architecting Process – 요구사항 이해

- ✓ 품질 속성은 조직 별, 컨텍스트 별로 서로 다른 정의를 가질 수 있습니다.
- ✓ 모든 품질 속성은 관심사가 될 수 있지만, 모든 관심사가 품질 속성은 아닙니다.
- ✓ 프로젝트 팀은 목표 시스템과 관련된 품질 속성을 정의함으로써 의사소통을 원활하게 할 수 있습니다.

시스템의 품질 속성(From SEI)

- 가용성(availability)
- 변경 용이성(modifiability)
- 수행성능(performance)
- 보안성(security)
- 테스트 용이성(testability)
- 사용성(usability)
- 상호운영성(interoperability)
- 이식성(Portability)
- 범위성(Scalability)

다양한 품질속성...

비즈니스의 품질속성 (From SEI)

- 시장 출하 시기(time to market)
- 비용과 장점(cost and benefit)
- 시스템의 예정 생명주기(projected lifetime of the system)
- 목표 시장(targeted market)
- 최초 공개 일정(rollout schedule)
- 기존 시스템과의 통합(integration with legacy systems)

시스템의 품질 속성(From TOGAF)

Availability

- manageability
- serviceability
- performance
- reliability
- recoverability
- locatability

Assurance:

- security
- integrity
- credibility

Usability

- International operation

Adaptability

- interoperability
- scalability
- portability
- extensibility

6. New Architecting Process – 요구사항 이해

- ✓ 하나의 기능은 여러 품질 속성과 관련이 있습니다.
- ✓ 서비스(기능) 제공 목표를 달성하기 위해, 요구되는 품질 목표를 달성하여야 합니다.
- ✓ 때로는 품질 목표가 기능 설계 또는 업무 구조 또는 절차와 관련이 있을 수 있습니다.
- ✓ 따라서, 아키텍팅 활동을 위해 특정 업무 기능에 대한 이해가 필요할 수도 있습니다.

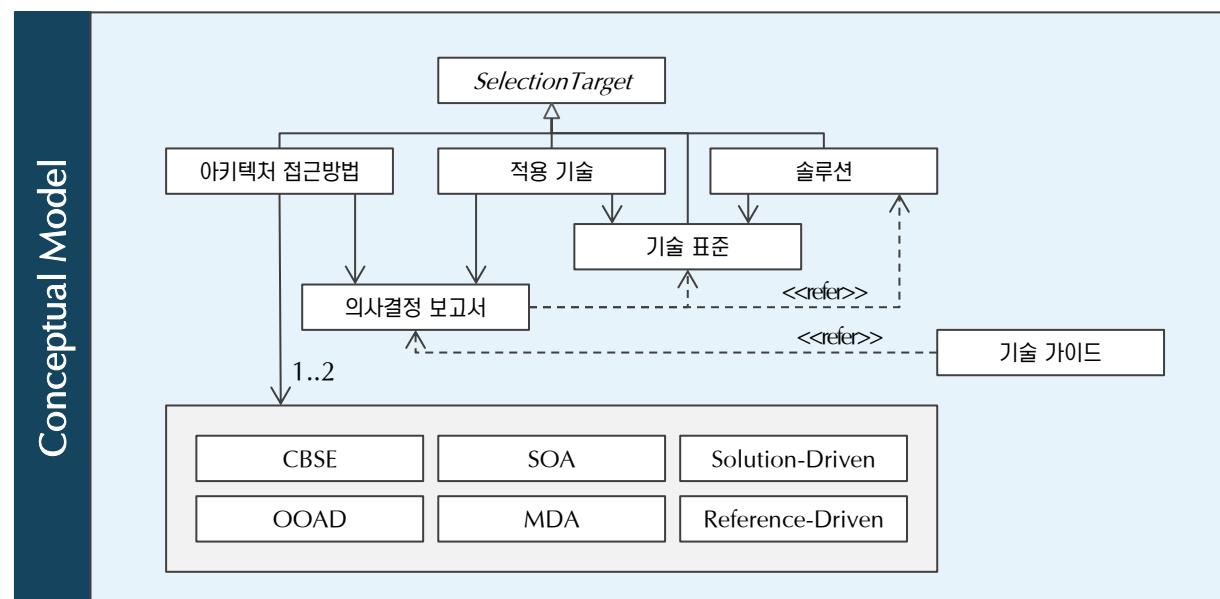
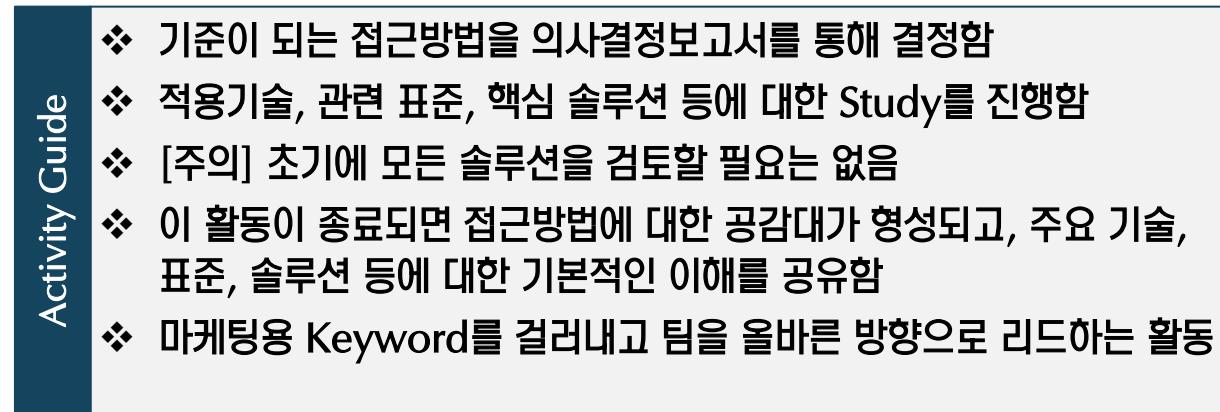
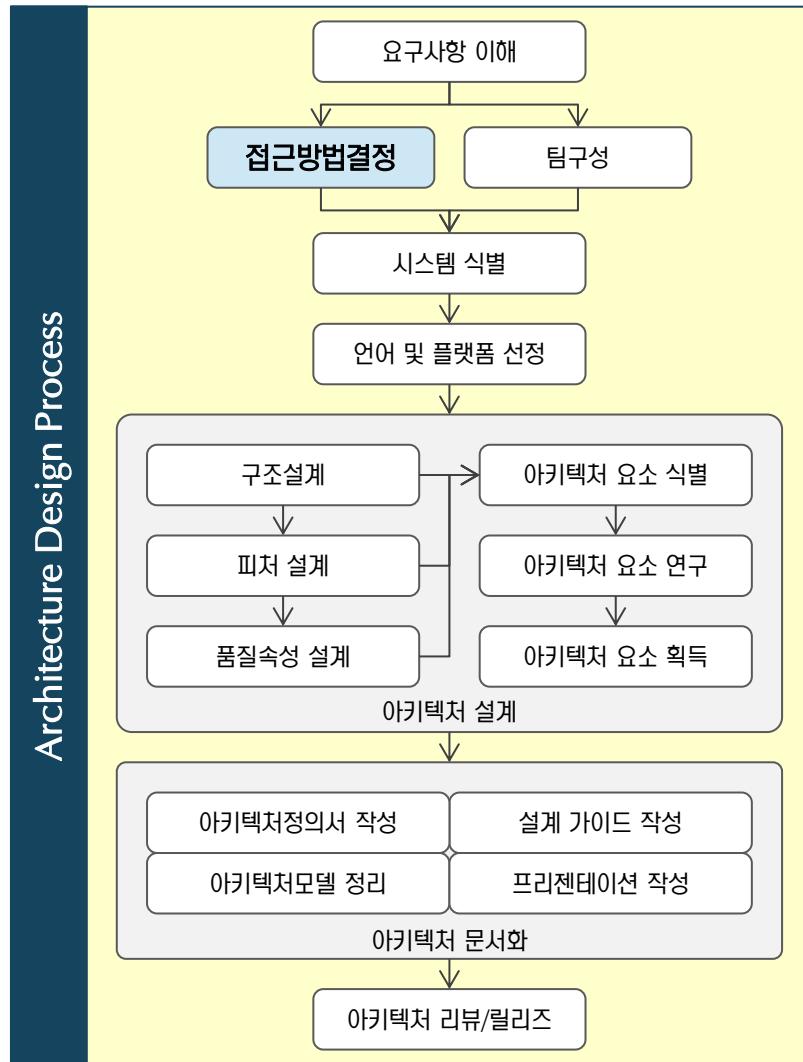
구분			경매등록	경매물품조회	입찰	낙찰금액지불	경매마감
품질속성	명세내용	단위					
가용성	중요도	상중하	중	상	상	중	상
	허용범위	다운시간/년	30H	10H	20H	30H	1H
성능	중요도	상중하	하	상	상	하	상
	허용범위	초(sec)	3	1	1	5	1
범위성	중요도	상중하	하	상	중	하	하
	허용범위	동시 사용자수	5천	10만명	1만명	5천명	5천명
보안	중요도	상중하	중	하	중	상	하
	허용범위	인증/인가/암호화	인증/인가	인증	인증/인가	인증/인가/암호화	해당사항없음
확장성	중요도	상중하	중	상	중	중	하
	허용범위	작업일	3	3	3	5	3
관리성	중요도				Cross-cutting 이슈		
	허용범위						
사용성	중요도	상중하	중	상	상	중	하
	허용범위	사용자만족도	80%	95%	90%	80%	해당사항없음

[주요 기능과 품질 속성 간의 연관관계]

아키텍팅
포인트

6. New Architecting Process – 접근방법 결정

- ✓ 시스템 구축을 위한 접근 방법, 기술, 표준, 솔루션 등에 대한 방향을 결정합니다.



6. New Architecting Process – 접근방법 결정

- ✓ 프로젝트 초기 팀이 필요로 하는 것은 정확한 방향성과 접근방법입니다.
- ✓ 접근방법이 결정된 후, 주요 기술에 대한 조사 및 정리 후 공유하는 과정이 필요합니다.
- ✓ 기술은 관련 표준과 관련 솔루션을 동반하는 경향이 있으므로 너무 깊이 들어가지 않도록 주의합니다.
- ✓ 조직의 아키텍트 그룹에서 [주요 기술 동향 및 분석 보고서]를 정기적으로 발간하면 도움이 됩니다.

[접근방법] CBD가 아닌 CBSE 접근방법

1. 컴포넌트 개요
2. 컴포넌트 정의
3. 컴포넌트 식별
4. 컴포넌트 관계 규칙
5. 컴포넌트 내부 구조
6. 컴포넌트 개발 프로세스

사례참조

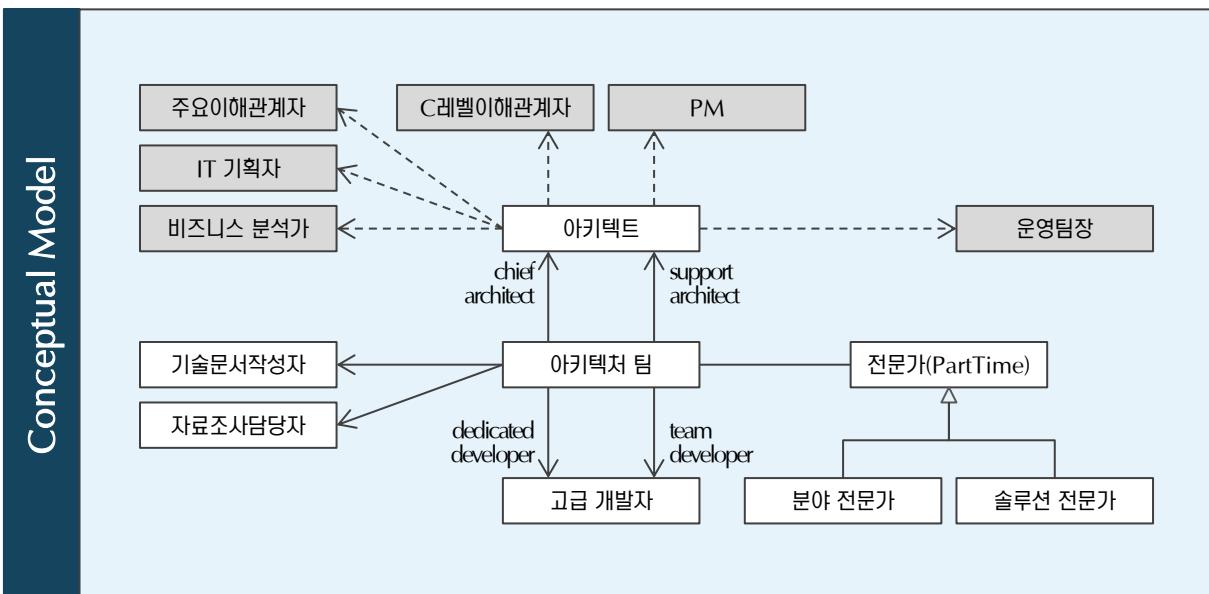
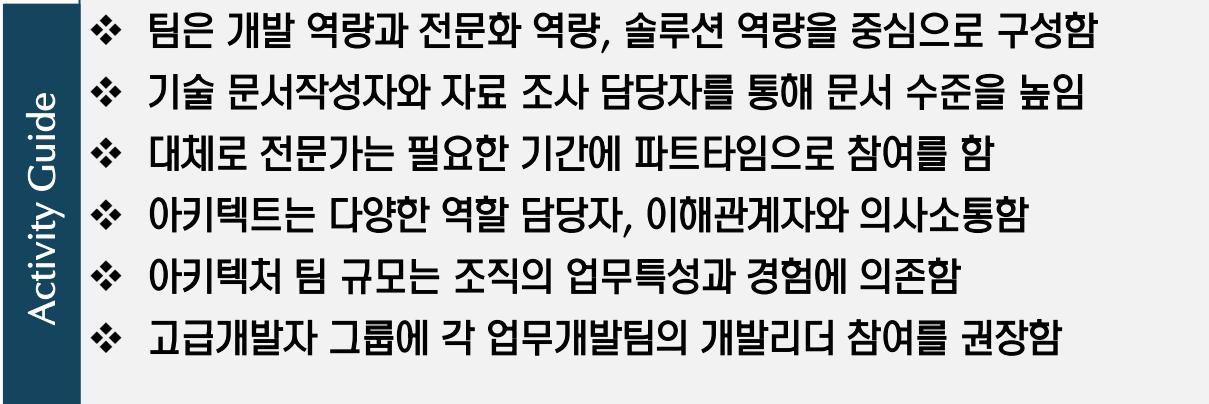
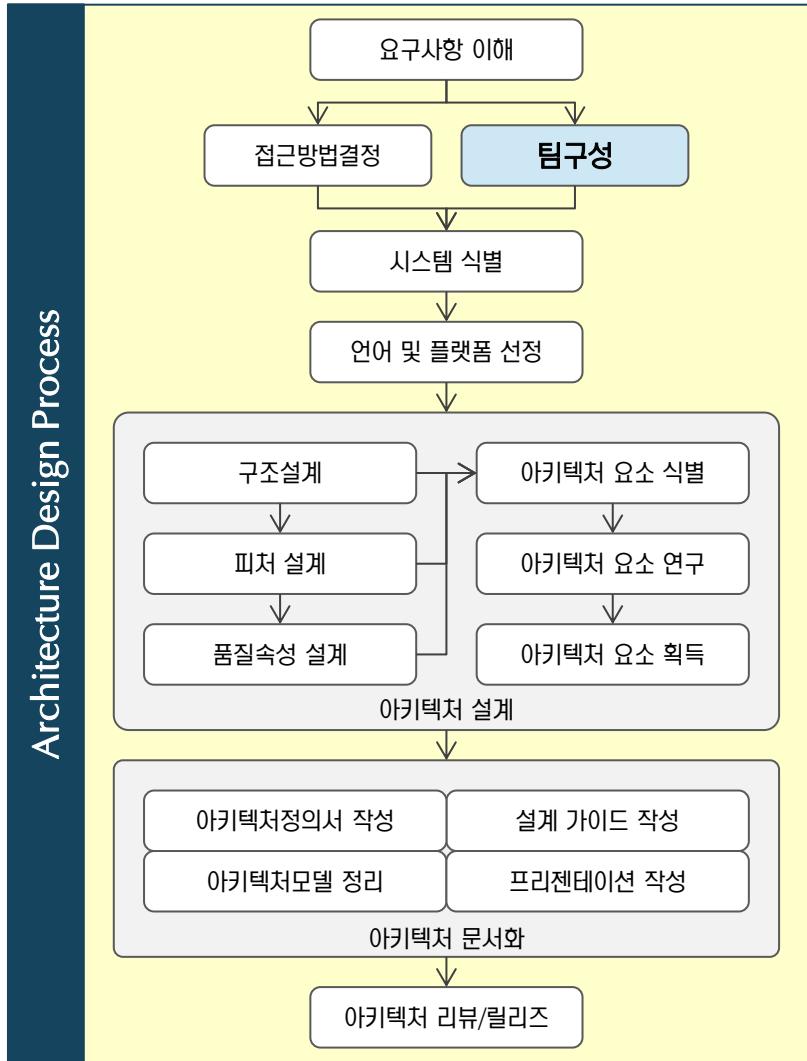
[기술] Identity & Access Management

1. Identity 관리 개념 정의
2. Identity 관리 프로젝트 접근방법
3. 정의 단계
4. Identity 관리
5. Access 관리
6. Identity 관리 이슈사항
7. 시나리오

사례참조

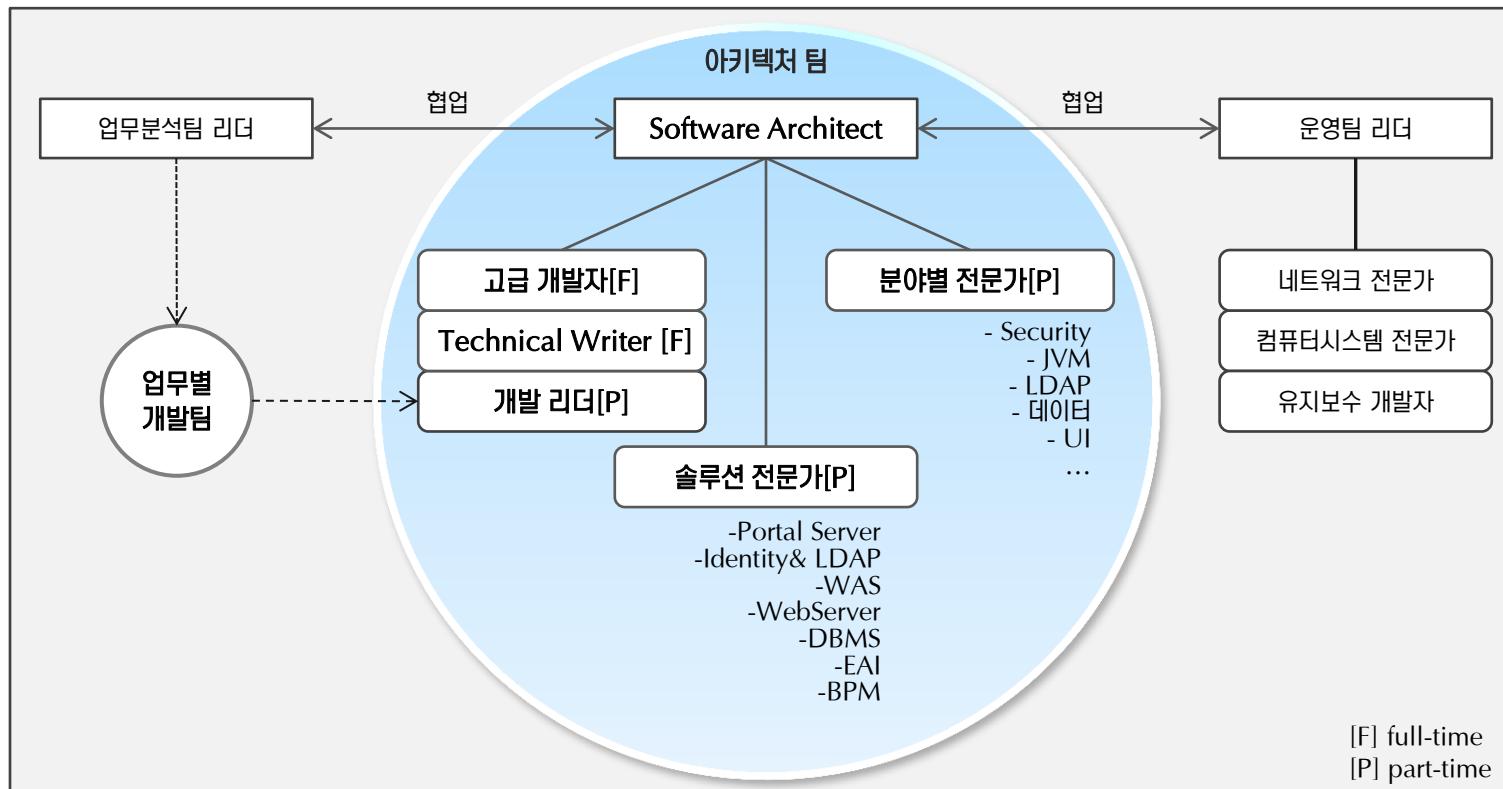
6. New Architecting Process – 팀구성

✓ 소프트웨어 아키텍처 팀을 구성합니다.



6. New Architecting Process – 팀구성

- ✓ 아키텍처 팀 구성은 AA, DA, TA, BA 역할로 채워서는 안됩니다.
- ✓ 아키텍처 팀의 핵심은 고급 개발인력과 파트 타임으로 참여하는 솔루션 전문가, 또는 분야별 전문가입니다.
- ✓ 하드웨어, 네트워크 등은 아키텍처 팀의 범주를 넘어섭니다.
- ✓ 프로젝트와 사용 기술 특성에 따라 아키텍처 팀 규모를 결정합니다.



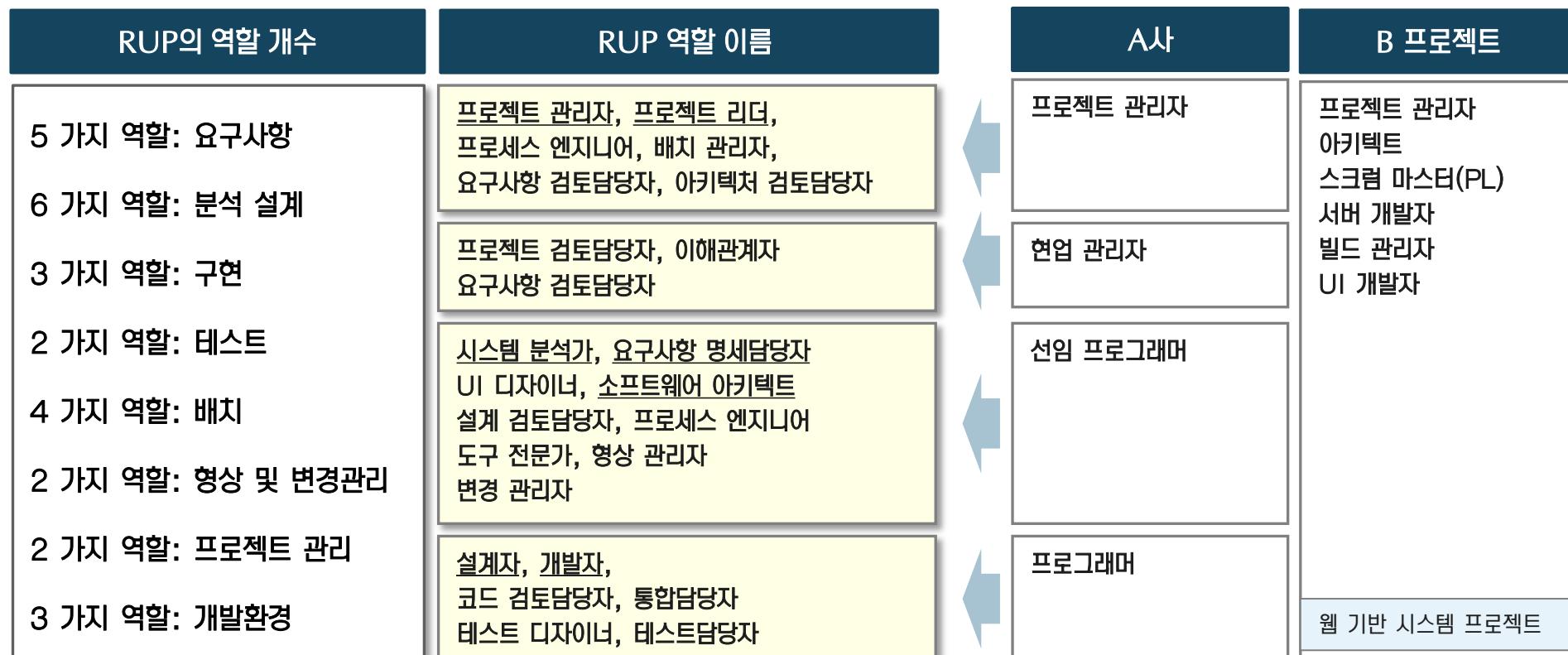
6. New Architecting Process – 팀 구성

- ✓ 어떤 유형의 개발이든 최소 1명의 아키텍트 역할을 지정해야 합니다.
- ✓ 애플리케이션 유형: 독립형 시스템, 엔터프라이즈 시스템(System of System)
- ✓ 개발 유형: 신규 개발, 확장 개발, 솔루션 맞춤(customization)
- ✓ 아키텍처 팀의 활동 범위에 따라 규모가 달라질 수 있습니다. ← 아키텍팅 프로세스 조정에 따라

시스템 유형 / 개발 유형		독립형 시스템		엔터프라이즈 시스템	
		소규모(10명내외)	중규모(50명내외)	중규모(100명내외)	대규모(200명이상)
신규 개발	신기술 (10%내외)	2명	5~7명	12명	20명이상
	기존 기술 (5%내외)	1명	3명	7명	12명이상
확장 개발	신기술 (5%이상)	1명	3명	6명	10명이상
	기존 기술 (5%이내)	1명	2명	4명	8명이상
솔루션 맞춤		1명	1명	2명	4명이상

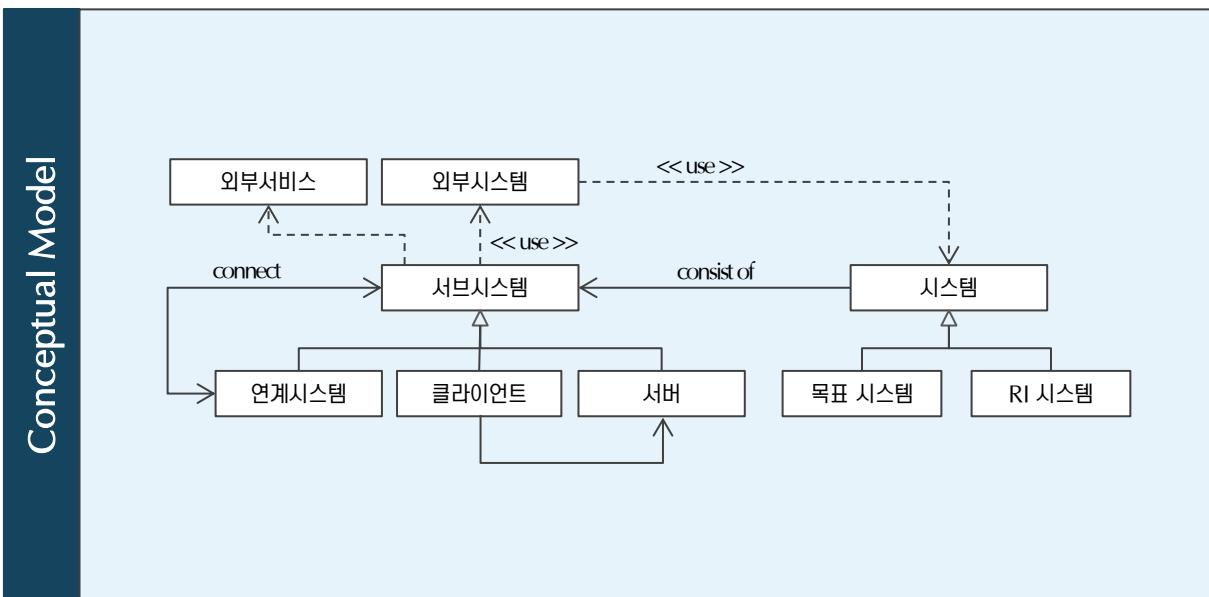
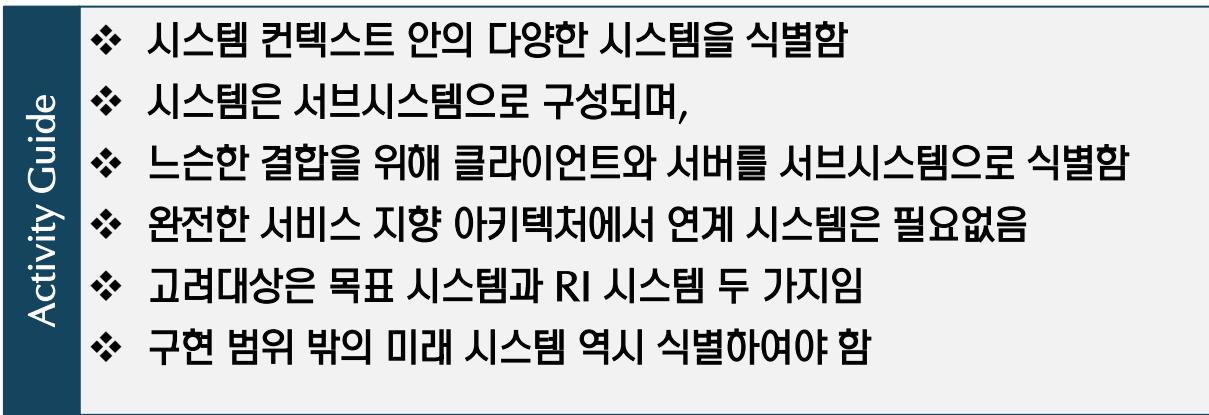
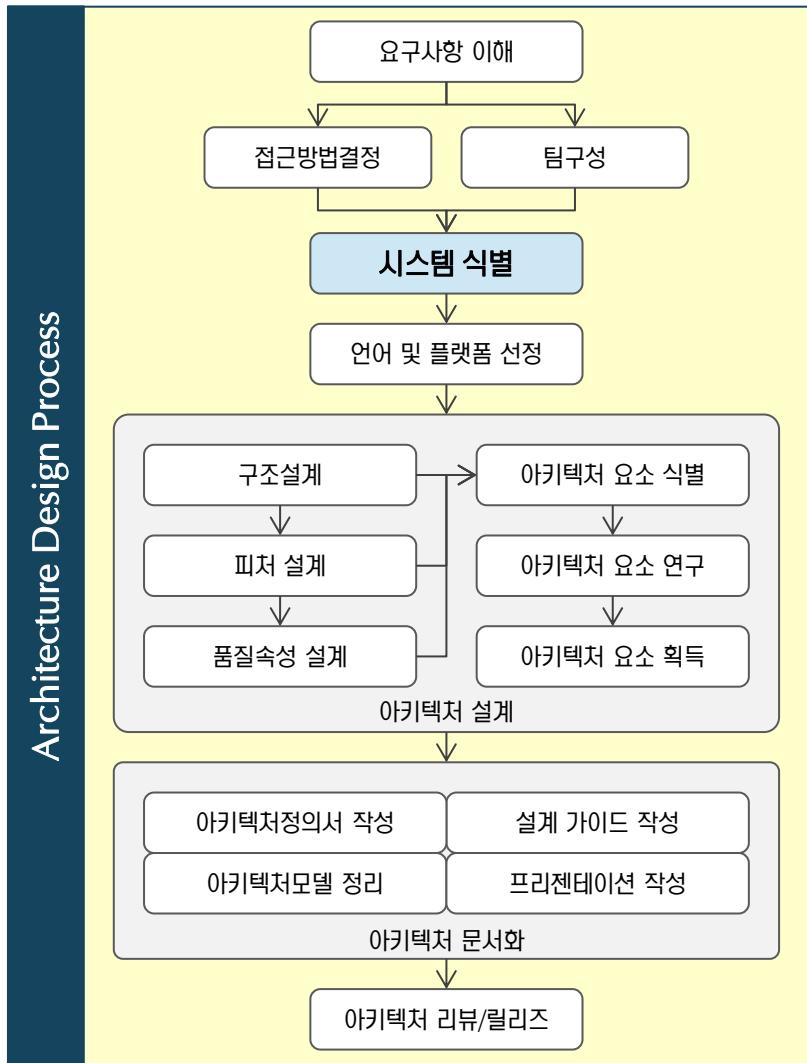
6. New Architecting Process – 개발 팀과 역할

- ✓ 소프트웨어 공학 기준, 표준 역할이 있으며, 개발 관련 프로세스는 역할 이름을 준수하여야 합니다.
- ✓ 조직의 특성과 프로젝트의 상황에 맞추어 역할을 정의합니다.
- ✓ 프로젝트에 사용한 기술에 따라 역할을 재 정의할 필요가 있습니다. , 예, UI integrator, 웹 퍼블리셔, 웹 기획자



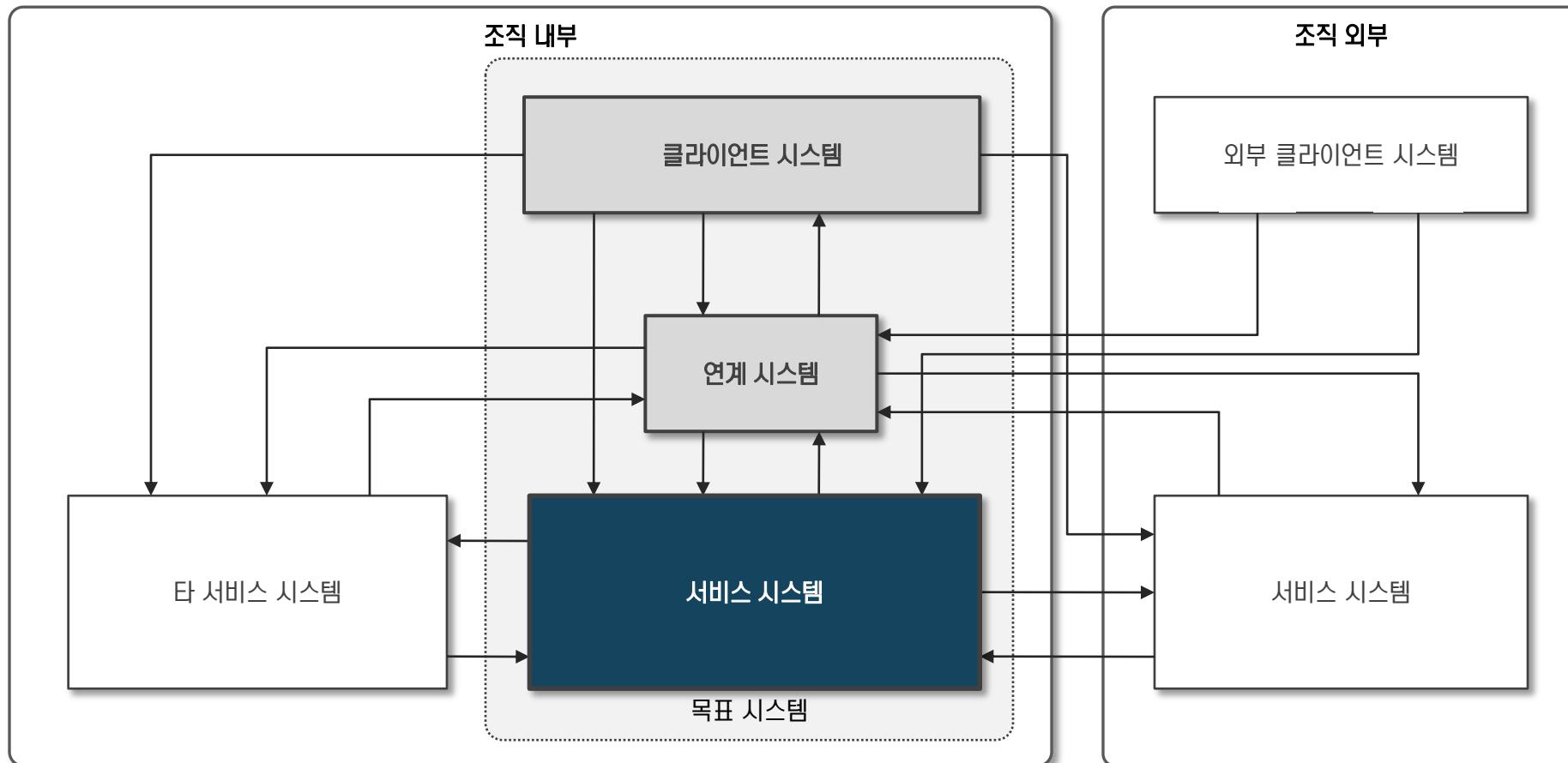
6. New Architecting Process – 시스템 식별

✓ 목표 시스템을 식별합니다.



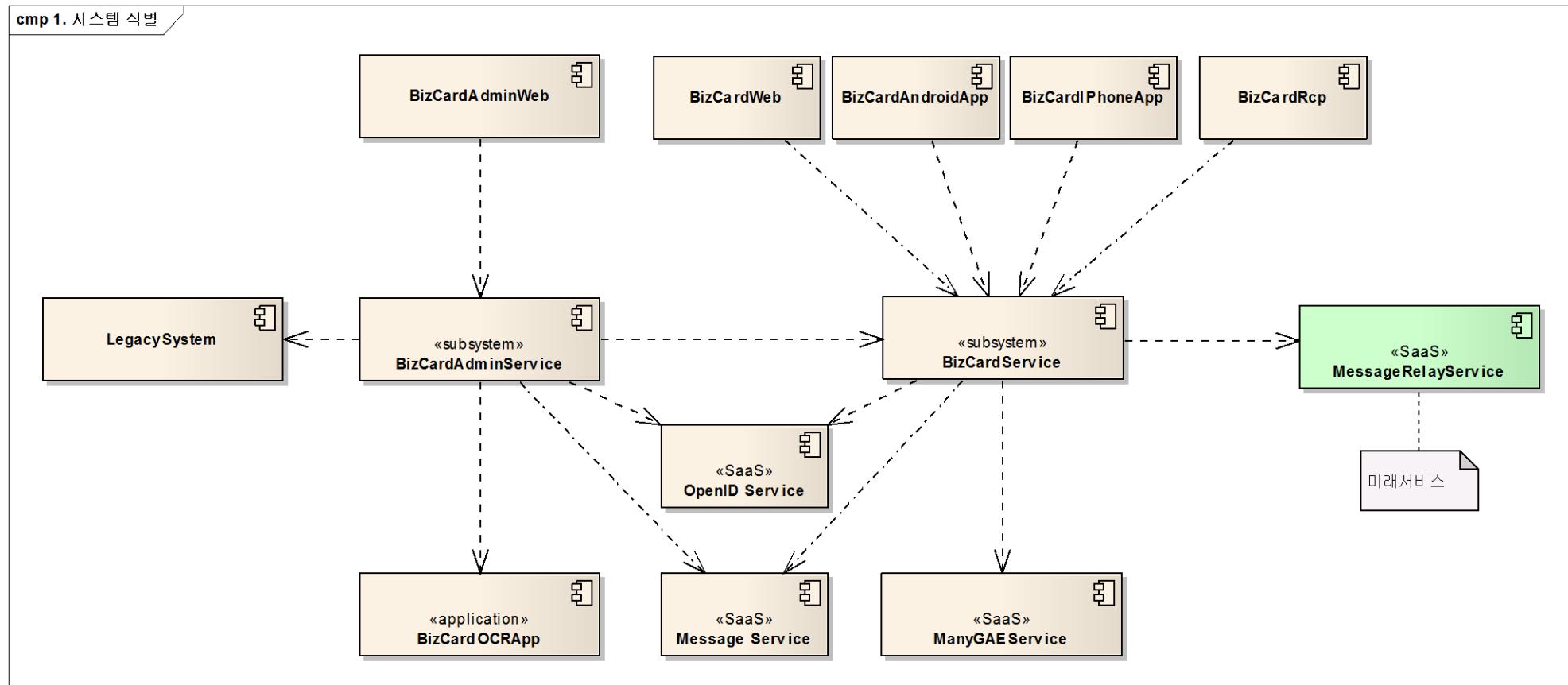
6. New Architecturing Process – 시스템 식별

- ✓ 우선 시스템 유형을 정의하고, 정의된 기준에 근거하여 시스템을 식별합니다.
- ✓ 조직 내부의 시스템과 외부의 시스템을 나누어 식별하는 것이 바람직합니다.



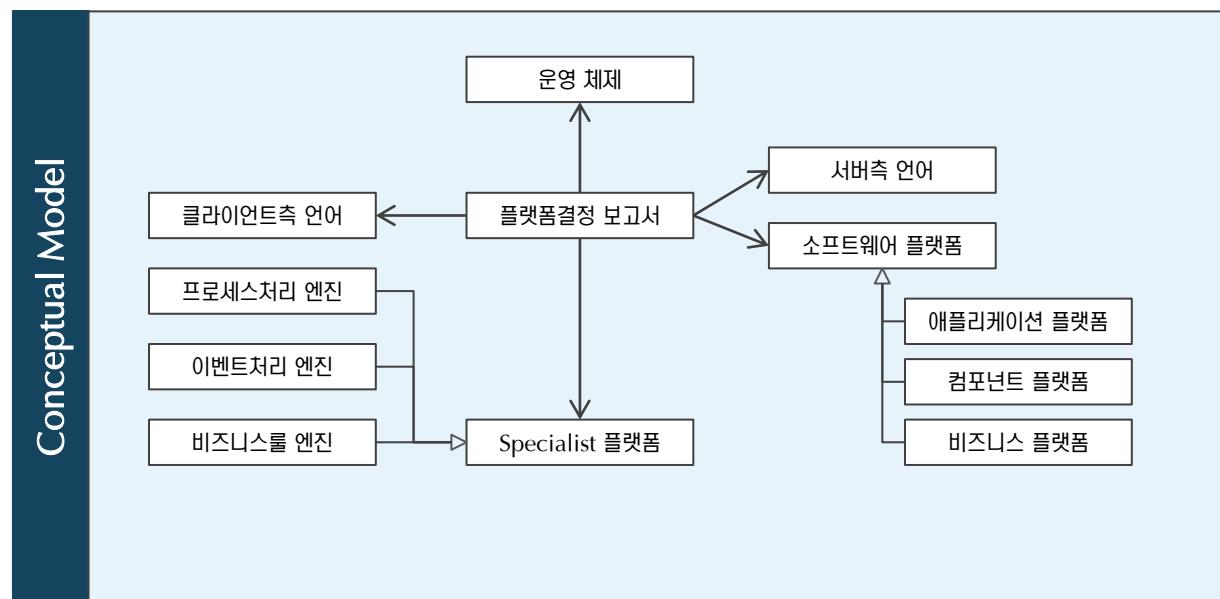
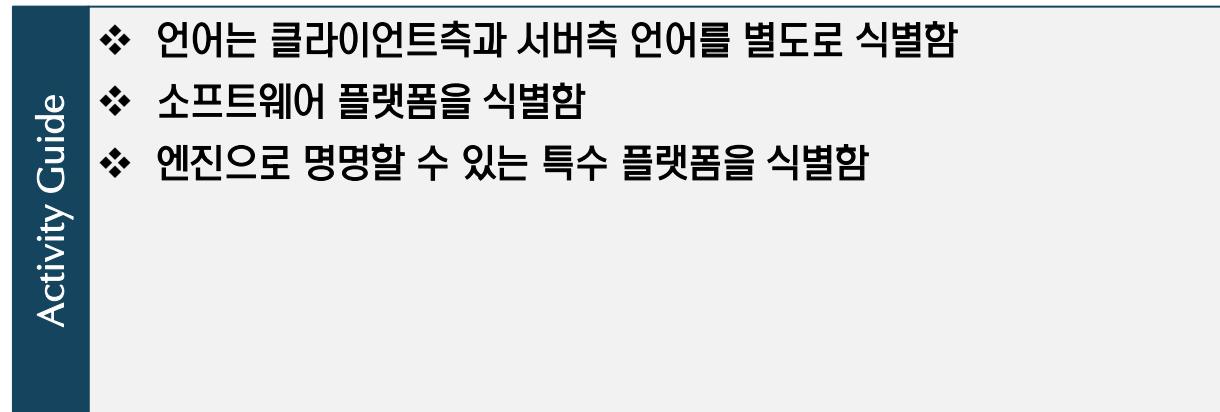
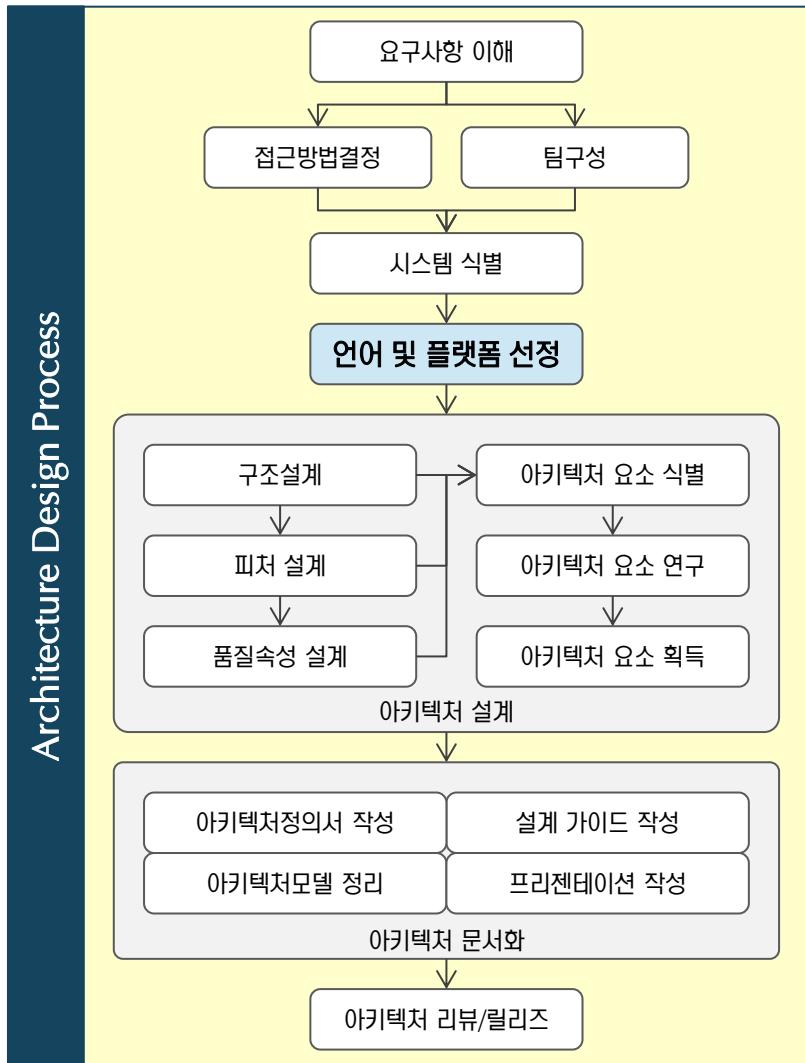
6. New Architecturing Process – 시스템 식별

- ✓ 예제, 비즈카드 시스템에서 시스템 식별
- ✓ SEI에서 모듈 분해(Decomposition)와 유사한 작업입니다.



6. New Architecting Process – 언어 및 플랫폼 선정

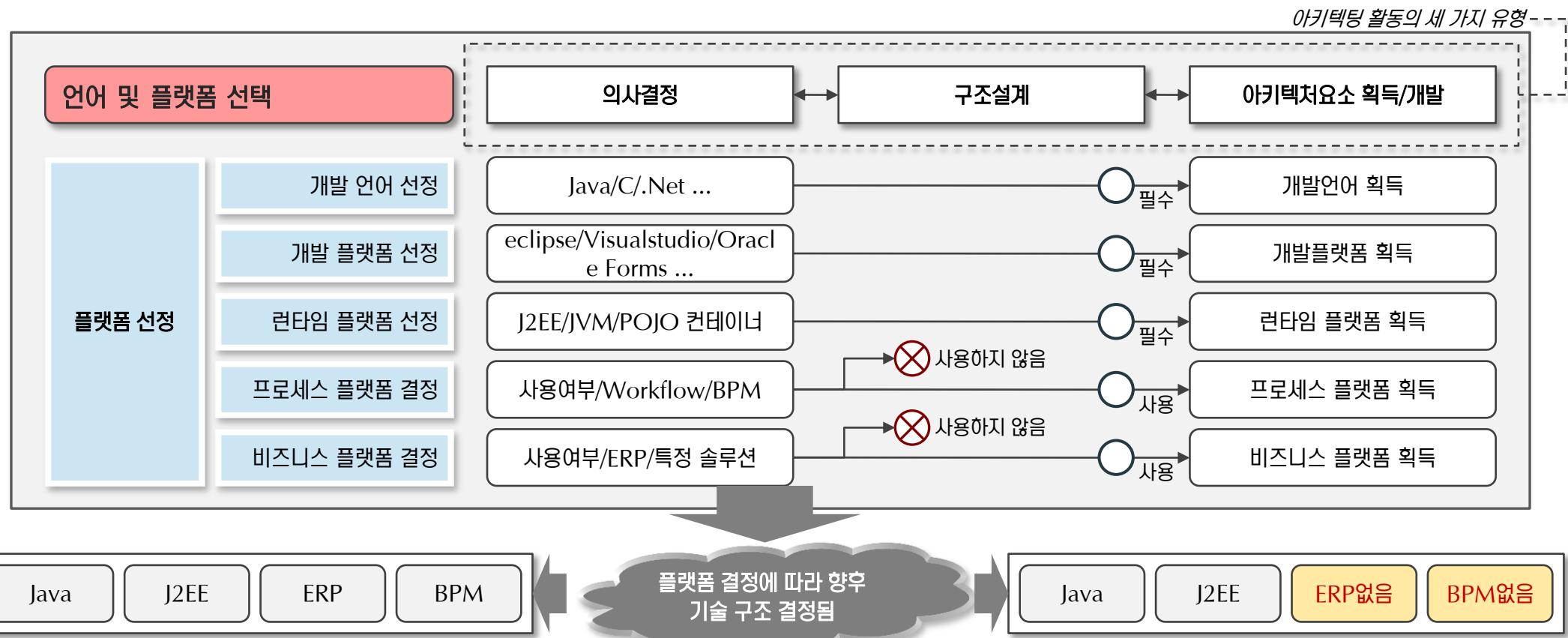
- ✓ 아키텍처 설계를 위한 프로그램 언어 및 플랫폼을 선정합니다.



6. New Architecting Process – 언어 및 플랫폼 선정

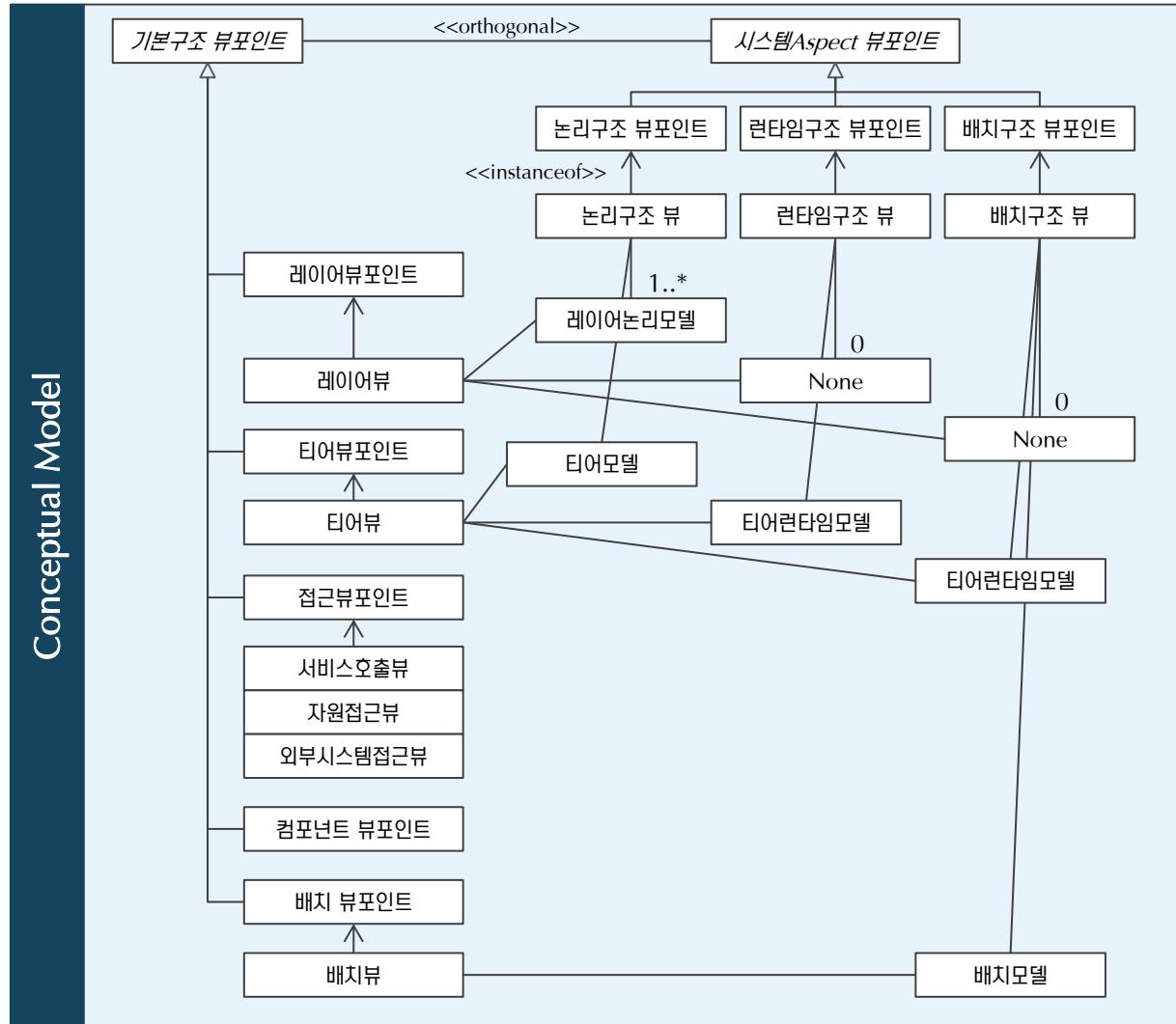
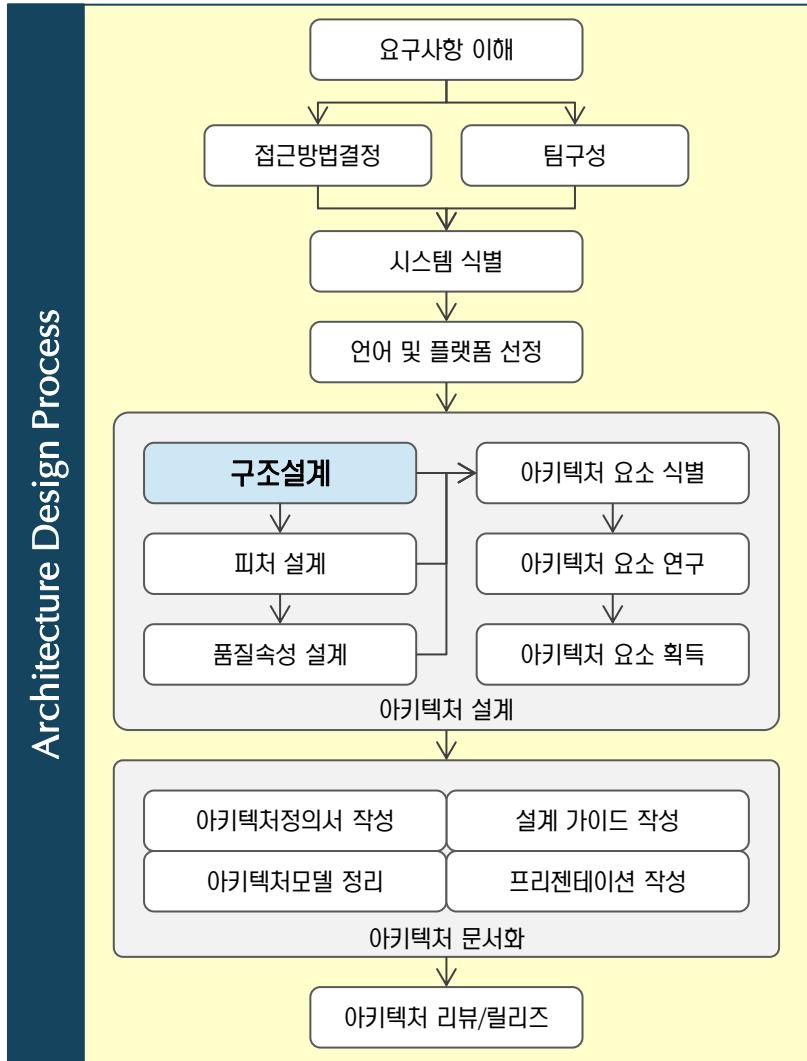
✓ 잘못된 플랫폼 선택은 많은 문제의 근본 원인을 제공하고 있습니다.

- 개발언어와 런타임 플랫폼은 필수 선택항목이며, 프로세스 플랫폼과 비즈니스 플랫폼은 옵션항목입니다.
- 선택된 플랫폼은 기술구조, 개발인력, 사용 소프트웨어에 제약조건을 가하므로 매우 신중해야 합니다.



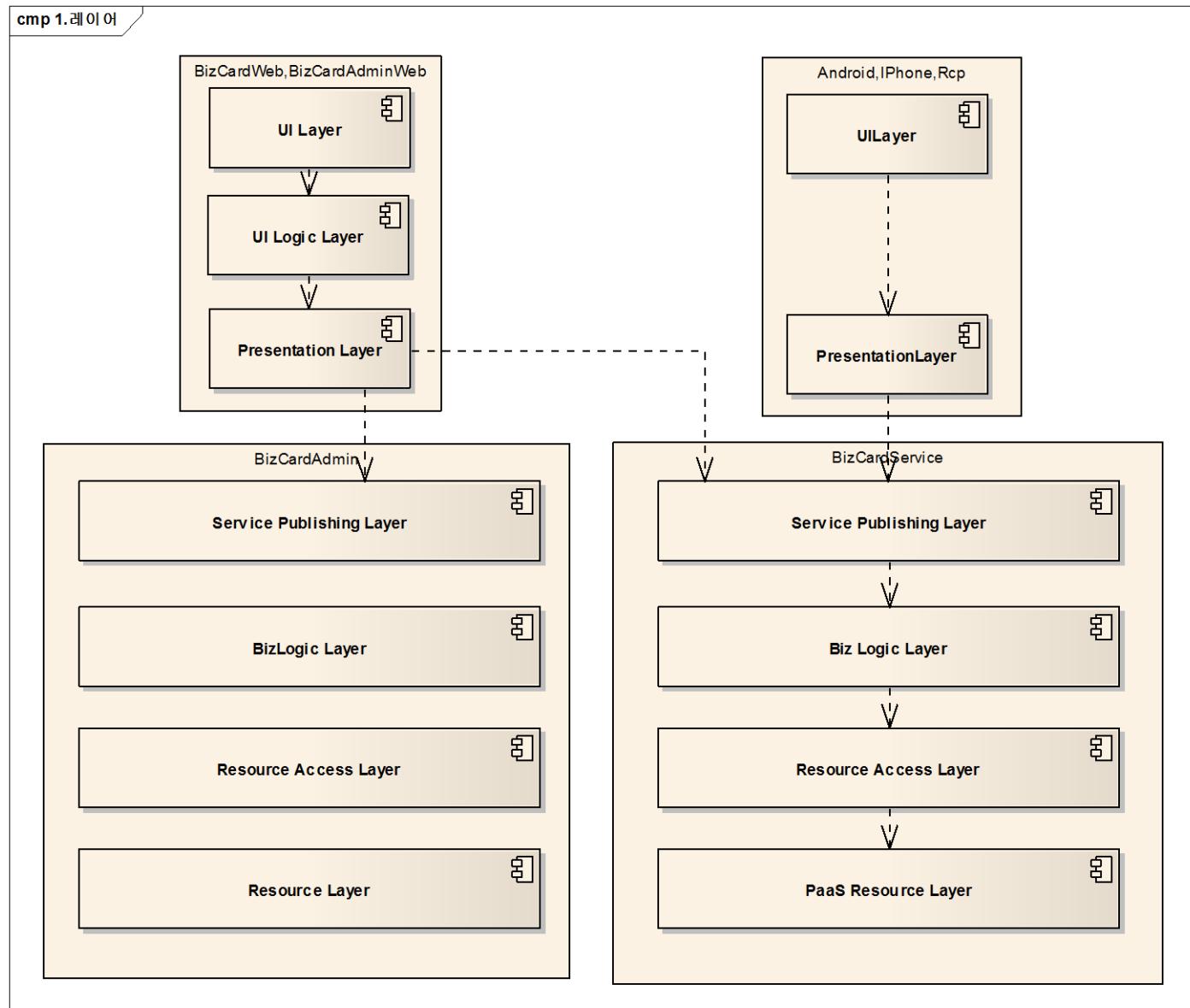
6. New Architecting Process – 구조설계

- ✓ 구조설계는 품질요구사항을 고려하지 않더라도 기본적으로 설계되어야 함



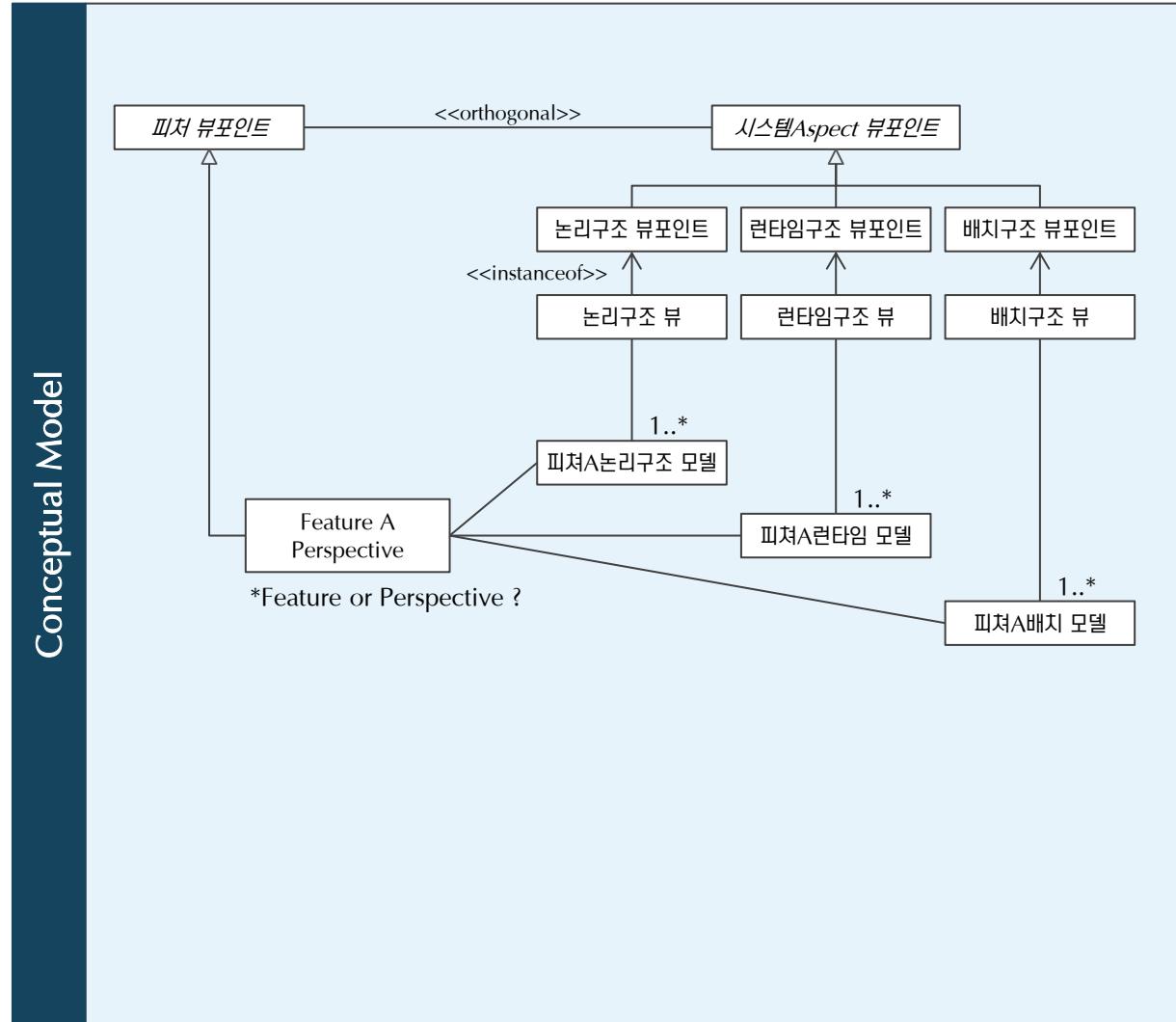
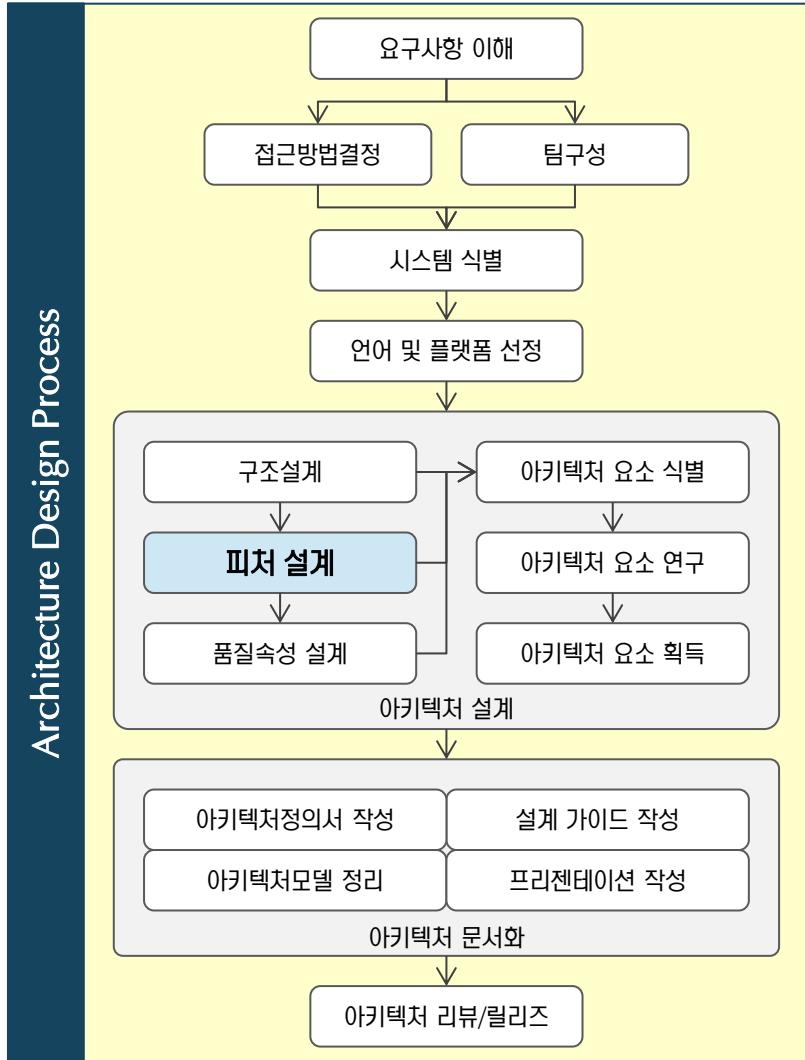
6. New Architecting Process – 구조 설계

- ✓ 구조설계의 첫 번째는 레이어 설계입니다.



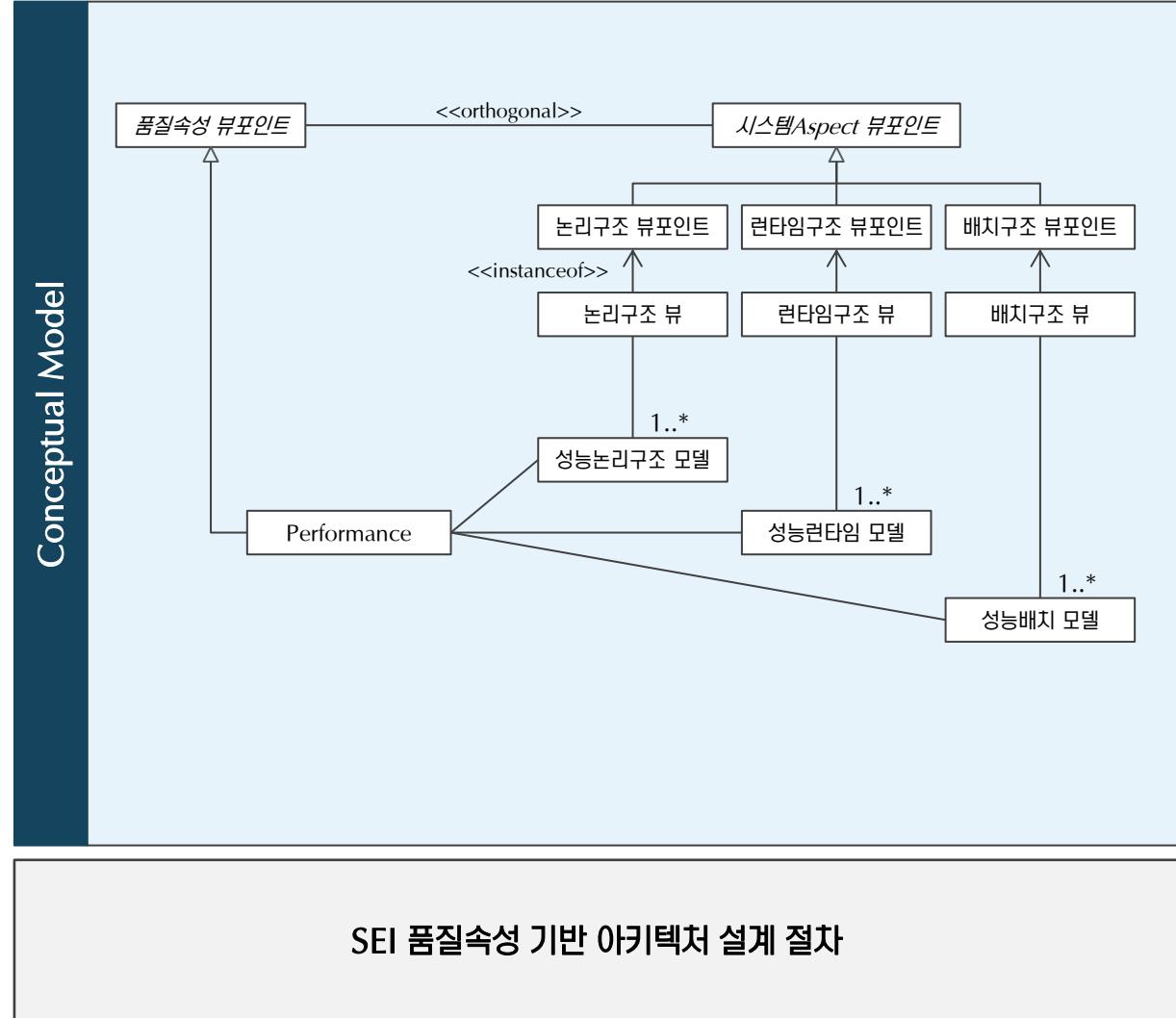
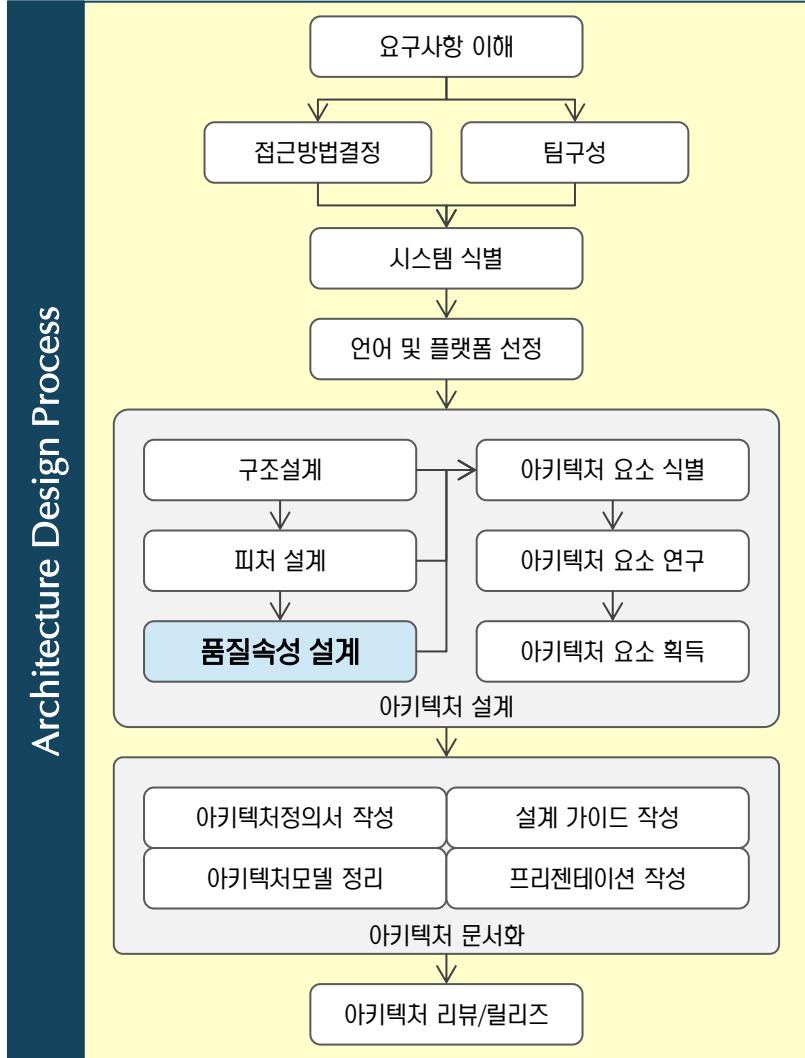
6. New Architecting Process – 피처 설계

- ✓ 피처는 기능과 품질이 다양한 비율로 결합된 형태의 요구사항입니다.



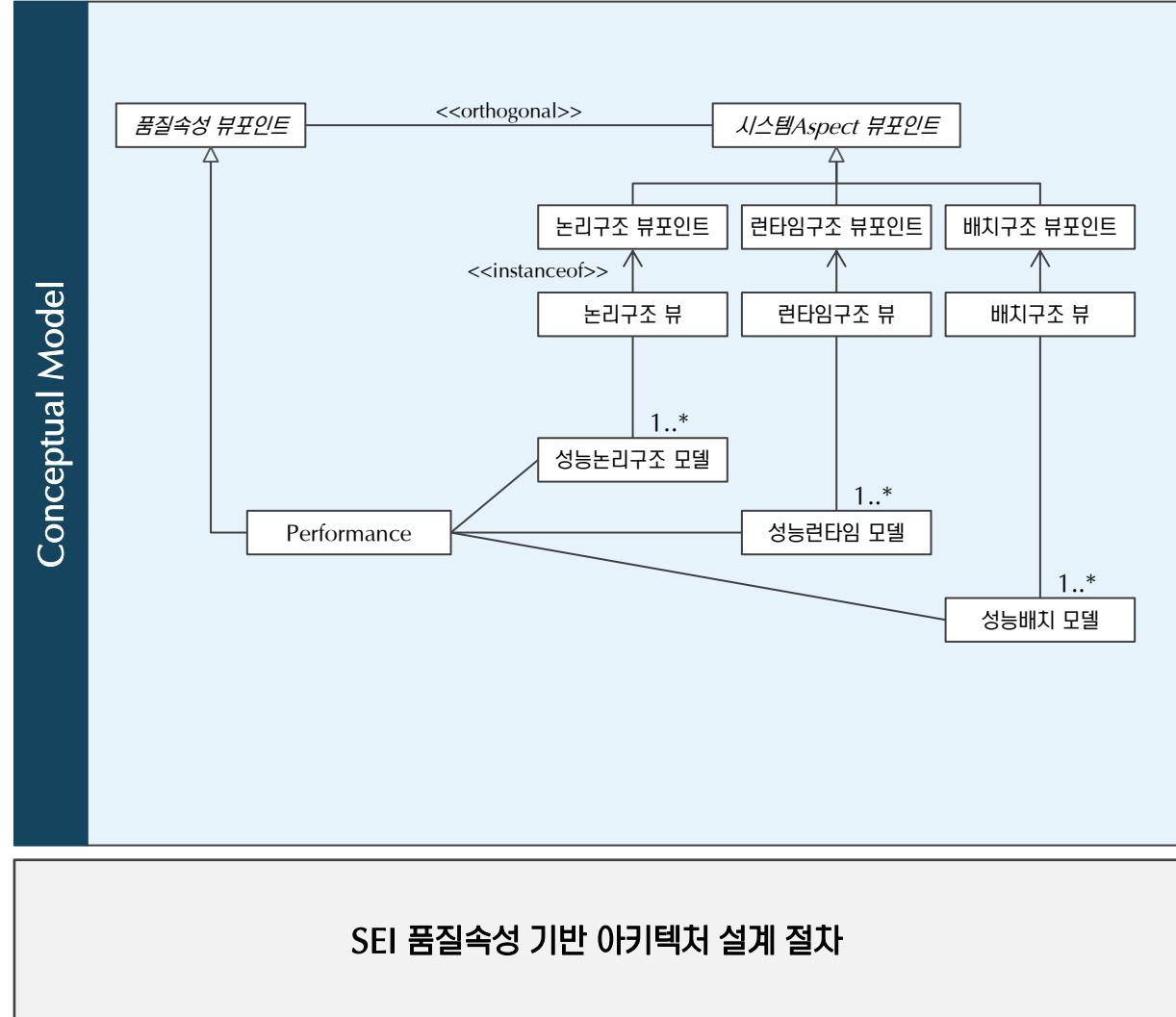
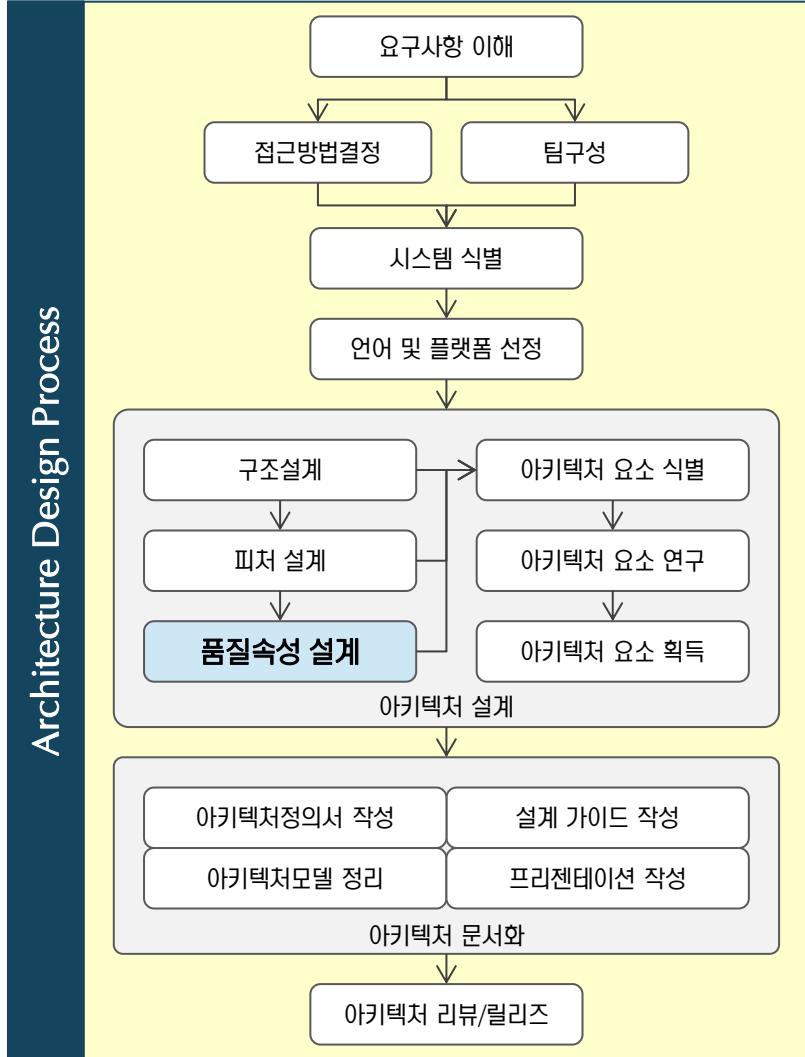
6. New Architecting Process – 품질속성 설계

- ✓ 품질속성은 조직이나 프로젝트 별로 서로 다른 세트를 선호하는 경향이 있음



6. New Architecting Process – 품질속성 설계

- ✓ 품질속성은 조직이나 프로젝트 별로 서로 다른 세트를 선호하는 경향이 있음



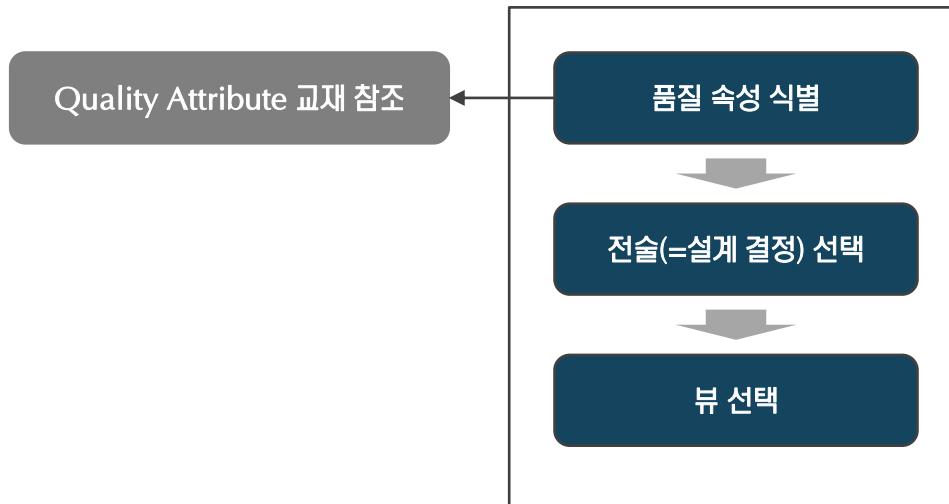


8. Architecting Process – SEI/ADD

-
- 1. SEI 프로세스
 - 2. 전술(설계 결정)
 - 3. 뷰
 - 4. 뷰선택
 - 5. ADD 프로세스
 - 6. ADD 프로세스 예제
 - 7. ADD 프로세스 요약
 - 8. 토의

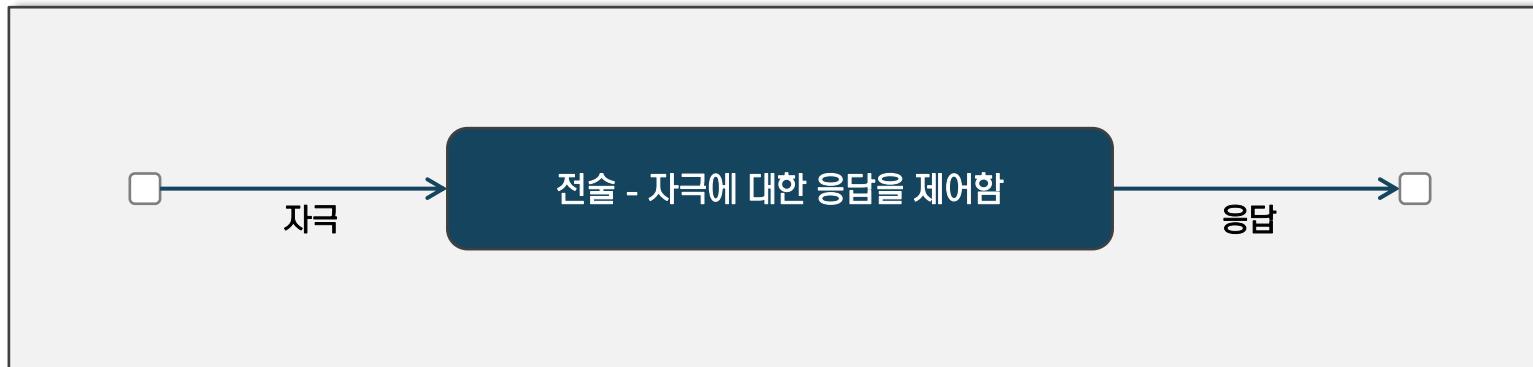
1. SEI 프로세스

- ✓ SEI에서 두 책(Software Architecture in Practice, Documenting Software Architecture)을 통해 제시합니다.
- ✓ 품질속성으로부터 뷰 선택에 이르는 일련의 아키텍팅 활동에 필요한 지식을 정리합니다.
- ✓ SEI는 소규모, 비-객체지향 개발 환경에서 사용할 수 있는 기본 아키텍팅 프로세스입니다.



2. 전술 [1/5]

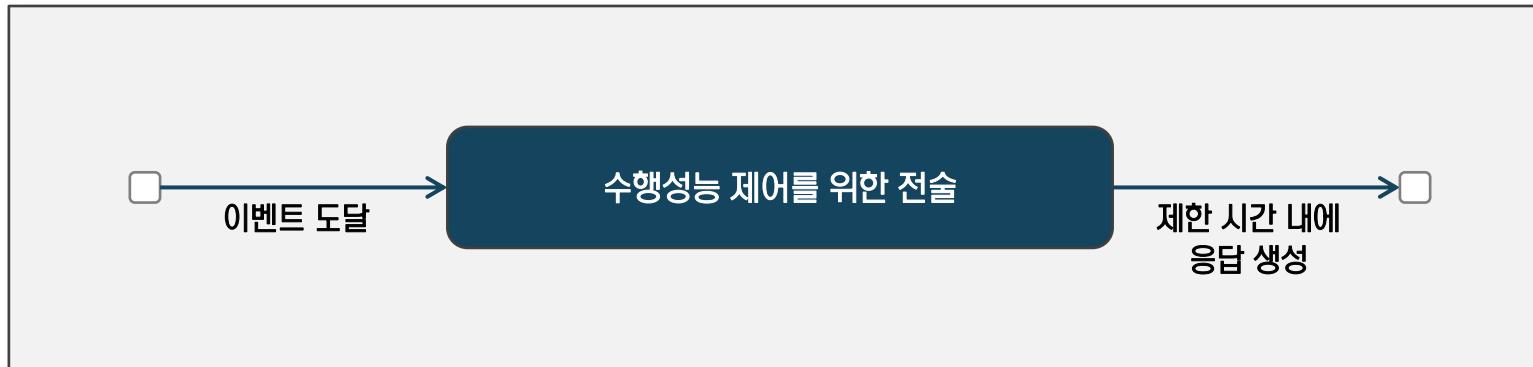
- ✓ 품질 목표 달성 여부는 아키텍처 전술에 의존함, 아키텍처 전술의 집합이 아키텍처 전략임
- ✓ 따라서, 각 전술은 결국 설계 옵션이며, 선택 간에 trade-off 이 존재함
- ✓ 예를 들면, 가용성을 높이기 위해 서버를 늘리면 동기화 문제가 발생함



2. 전술 (2/5) – 수행성능 [1/2]

✓ 수행성능(performance) 전술의 목표

- 주어진 시간 안에 시스템에 도달한 이벤트에 대한 응답을 생성함
- 이벤트 종류 – 메시지 도착, 만기 시간 도래, 상태 변화 감지
- 대기 시간(latency) – 이벤트 도달과 응답 생성 간의 시간

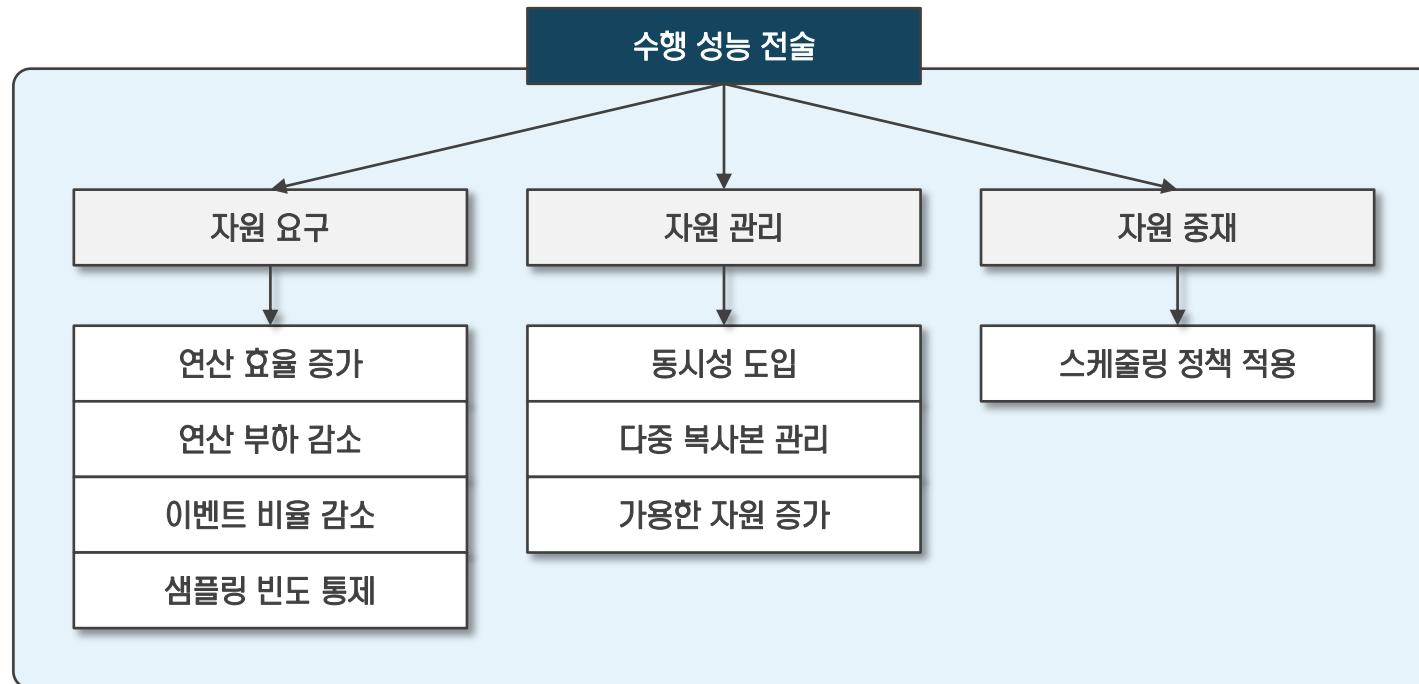


✓ 응답시간을 결정하는 요인

- 자원 소비
- 차단 시간 (blocked time) – 자원 경쟁, 자원 가용성 제한, 타 컴퓨팅 자원에 의존

2. 전술 [3/5] – 수행성능 [2/2]

- ✓ 자원 요구 – 대기시간을 줄이기 위해 연산 효율을 높이거나, 처리할 이벤트를 줄이거나, 자원을 제한함
- ✓ 자원 관리 – 동시성을 도입하거나 컴퓨팅 자원의 여러 복사본을 유지하거나, 가용자원을 늘림
- ✓ 자원 중재 – 스케줄링을 도입하여 자원 경합을 피함, 다양한 우선순위(고정, 동적, 정적) 스케줄링을 할 수 있음



2. 전술 (4/5) – 변경용이성 [1/2]

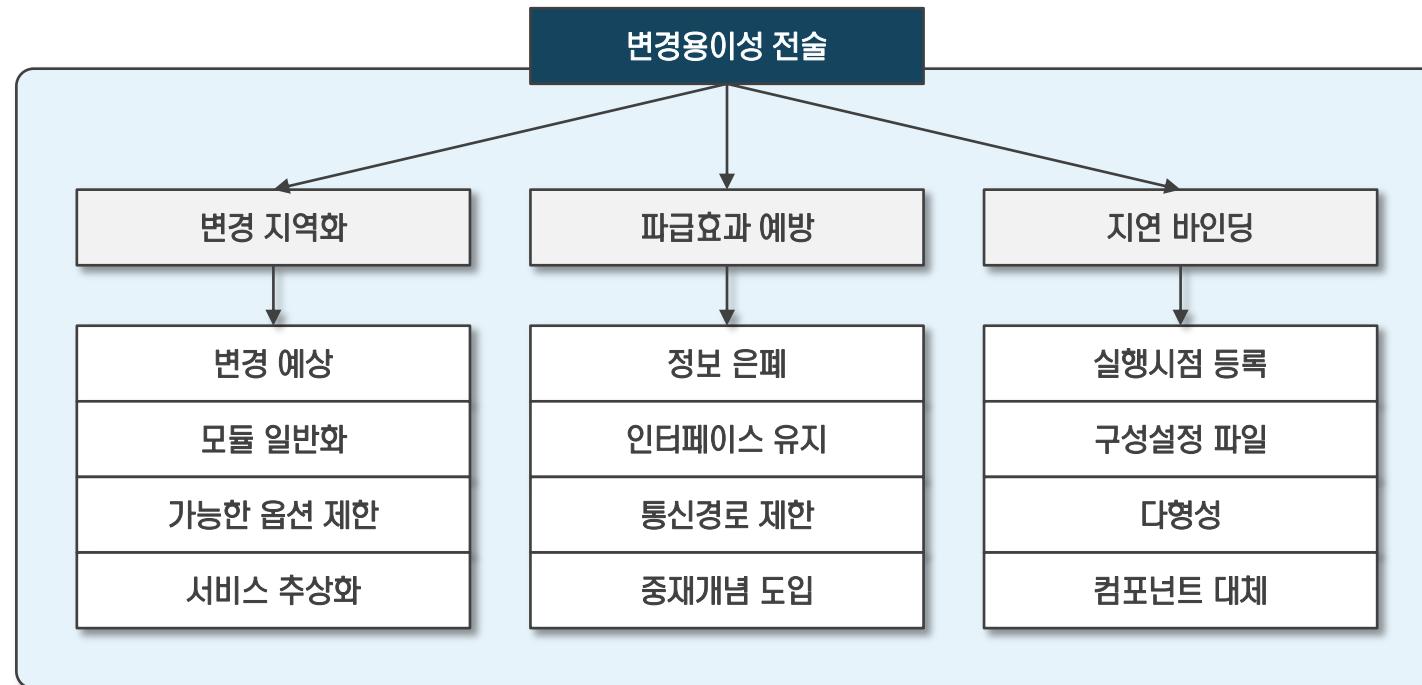
✓ 변경용이성(modifiability) 전술의 목표

- 설계, 구현, 테스트, 배치 변경으로 인한 시간과 비용을 제어



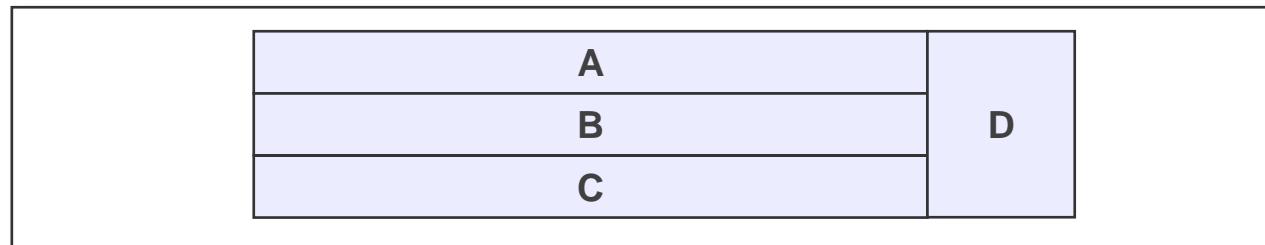
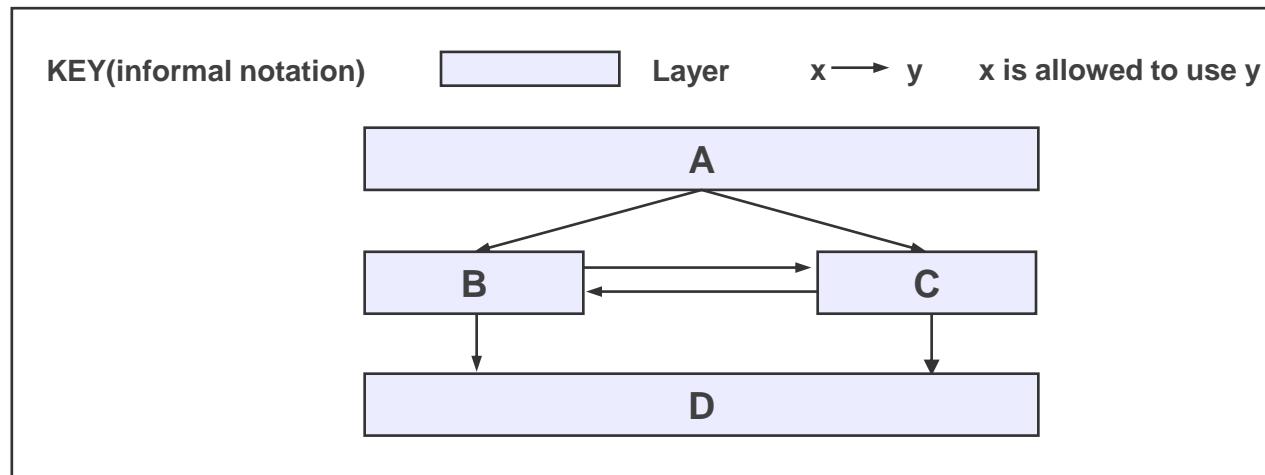
2. 전술 (5/5) – 변경용이성 [2/2]

- ✓ 변경의 지역화 - 변경에 직접 영향을 받는 모듈의 개수를 줄임
- ✓ 파급효과 예방 - 변경범위를 지역 모듈 안으로 제한
- ✓ 바인딩 연기 – 배치 시간과 비용을 통제



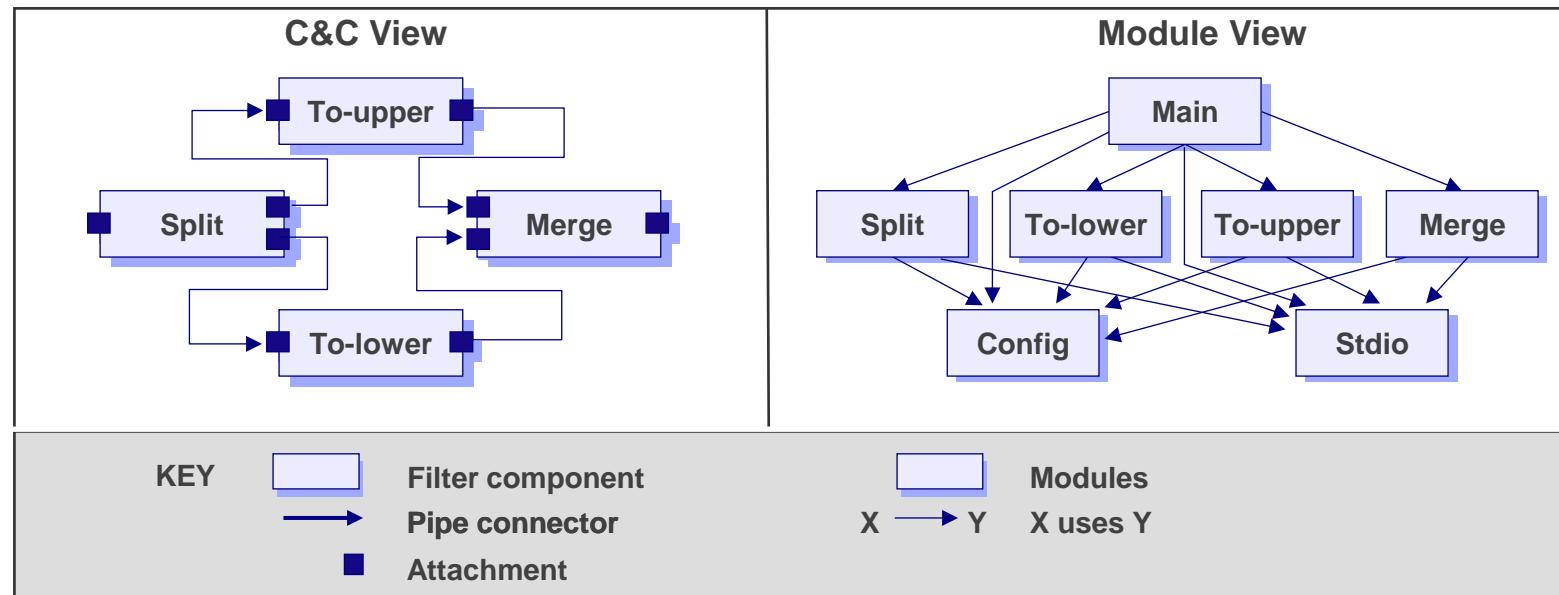
3. 뷰 (1/4) – 모듈 뷰타입

- ✓ 4가지 모듈 뷰타입 스타일 – 분할 스타일, 사용 스타일, 일반화 스타일, 레이어 스타일
- ✓ 모듈은 책임의 집합이며, 구현 단위이며, 기능의 묶음임
- ✓ 모듈은 세 가지 관계를 가지고 있음 – is-part-of 관계, depends-on 관계, is-a 관계



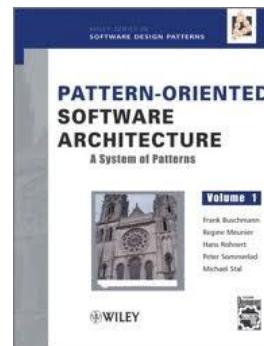
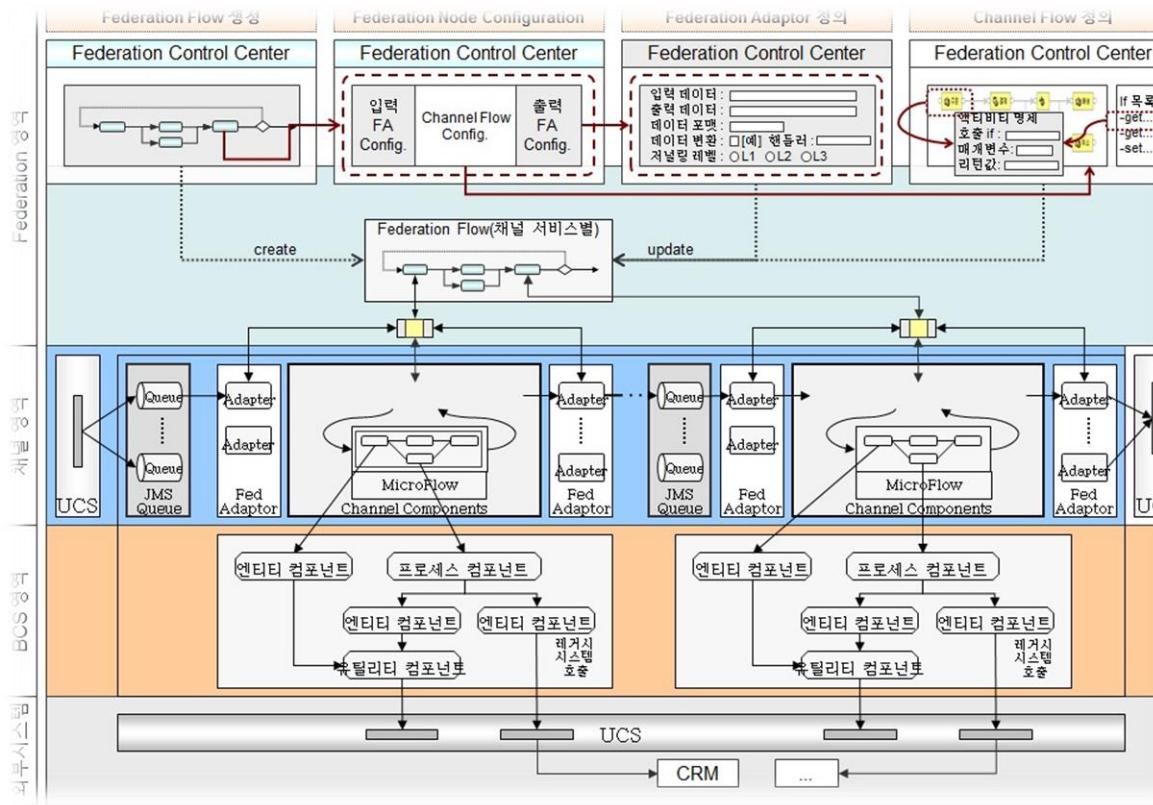
3. 뷰 (2/4) – C&C 뷰타입

- ✓ 다양한 스타일 – 파이프-필터 스타일, 공유-데이터 스타일, 발행-구독 스타일, 클라이언트-서버 스타일, 피어-피어 스타일
- ✓ 컴포넌트는 실행되는 컴퓨팅 요소나 저장소이며, 커넥터는 여러 컴포넌트 간의 상호작용 경로임
- ✓ 컴포넌트와 커넥터는 부착(attachment) 관계 만 존재함



3. 뷰 (3/4) – C&C 뷰타입

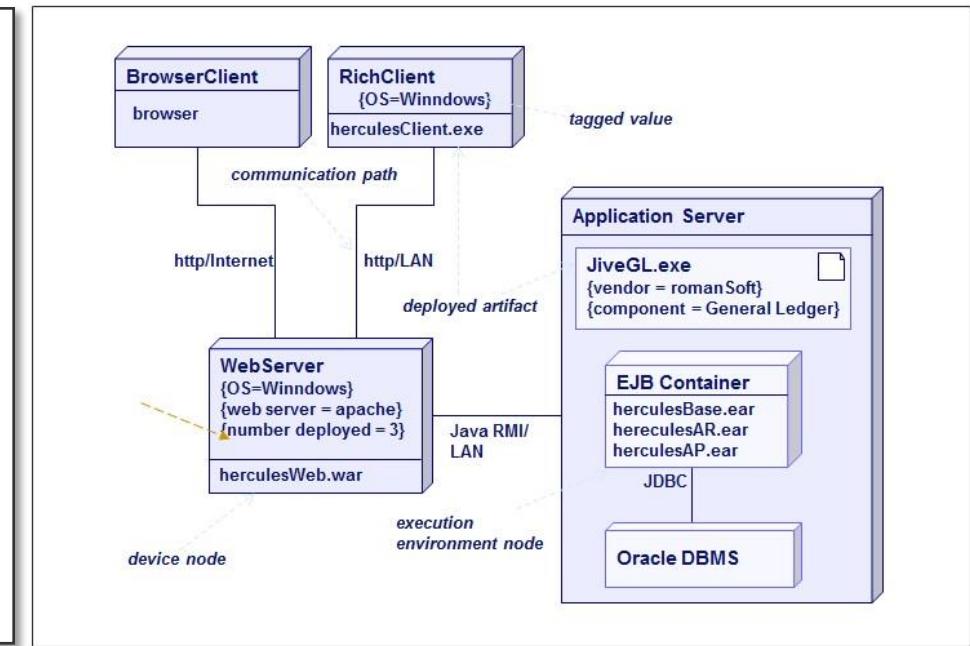
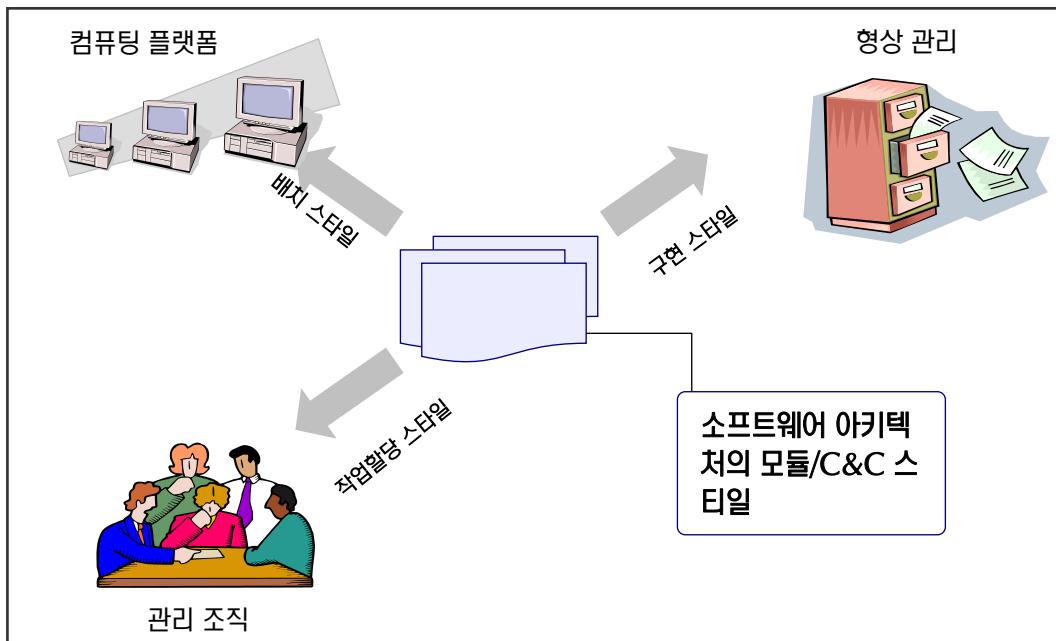
- ✓ 파이프-필터 스타일을 적용한 채널 시스템 설계 사례,
- ✓ C&C 뷰타입의 스타일은 아키텍처 패턴이라고도 불리며, 현대의 소프트웨어 설계에 많이 응용하고 있음
- ✓ POSA^(*) 패턴은 아키텍처 패턴의 주류를 이루고 있음



(*)POSA – Pattern Oriented Software Architecture

3. 뷰 (4/4) – 할당 뷰타입

- ✓ 할당 뷰타입 스타일은 배치 스타일, 구현 스타일, 작업할당 스타일 세 가지가 있음
- ✓ 할당 뷰타입은 소프트웨어 아키텍처와 외부 환경 간의 맵핑을 표현함, 아키텍처는 HW, 시스템, 팀의 구조에 영향을 줌
- ✓ 소프트웨어 요소와 환경 요소로 구성되어 있으며, 할당(allocation) 관계를 가짐



4. 뷰선택 (1/6) – 이해관계자와 문서화

- ✓ 목표 시스템을 둘러싸고 있는 다양한 이해관계자를 식별하고 그들의 관심사를 확인함
- ✓ 관심의 정도에 따라 뷰와 스타일을 분류한 후, 제거할 것은 제거한 후, 통폐합을 함

이해관계자	모듈 뷰				C&C 뷰	할당 뷰			기타						
	분할	사용	일반화	레이어	Various	배치	구현	작업할당	인터페이스명 세서	컨텍스트 다이어그램	뷰 간 매핑	다양성 가이드	분석 결과	타당성 제약조건	
프로젝트 관리자	s	s		s		d		d		o					s
개발 팀원	d	d	d	d	d	s	s		d	d	d	d			s
테스터와 통합담당	d	d	d	d	s	s	s		d	d	s	d			s
다른 시스템 설계담당									d	o					
유지보수 담당	d	d	d	d	d	s	s		d	d	d	d			d
프로덕트라인 빌더	d	d	s	o	s	s	s		s	d	s	d			s
고객						o	o		o						s
최종 사용자					s	o	o								s
분석가	d	d	s	d	s	d			d	d		s	d		s
인프라 지원 담당	s	s				s	d					s			
신규 이해관계자	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
현재와 미래 아키텍트	d	d	d	d	d	d	s	s	d	d	d	d	d	d	

Key: d=상세 정보(detailed information), s=일반 정보(some details), o=개요 정보(overview information), x=anything

4. 뷰선택 (2/6) – 절차

✓ 후보 뷰 리스트 생성

- 이해관계자/뷰 테이블 구축
- 아키텍처에 관심을 가진 이해관계자를 행에 나열
- 시스템에 적용될 뷰를 열에 나열
- 각 셀에 이해관계자가 각 뷰의 정보를 얼마나 많이 요구하는지 표시(none, overview only, moderate detail, high detail)

✓ 뷰 통합

- 관리 가능한 크기로 뷰 통합
- 개요만을 요구(overview only)하거나 적은 수의 이해관계자와 관련된 뷰 선택
- 통합 뷰를 위한 후보 선택

✓ 우선 순위 결정

- 먼저 수행할 뷰 선택, 프로젝트에 대한 상세 정도 기준
- 고려사항1: 다른 뷰를 시작하기 전에 하나의 뷰를 완결할 필요는 없음
- 고려사항2: 너비-우선(breadth-first) 접근방법 적용
- 고려사항3: 어떤 이해관계자의 관심이 다른 이해관계자의 관심을 우선함
- 고려사항4: 프로젝트 관리자나 회사 경영진이 더 일찍 그리고 더 자주 정보를 요구하는 경우
- 고려사항5: 아키텍처가 목적에 대한 적합성에 대해 평가되거나 확인되지 않은 경우 이 작업을 우선함

4. 뷰선택 (3/6) – 예제(A-7E) – 1. 후보 뷰 리스트 생성

✓ 이해관계자 목록

- 현재와 미래의 아키텍트(current and future architect)
- 프로젝트 관리자(project manager)
- 개발 팀 멤버(a member of the development team)
- 테스터와 통합자(testers and integrators)
- 유지보수자(maintainers)
- 프로젝트 투자 기관(funding agency)

✓ 품질 속성

- 수정 용이성(modifiability)
- 실시간 성능(real-time performance)
- 점증적 부분집합 배치 능력(ability to be fielded in incremental subset)

✓ 뷰 목록

- 모듈 뷰타입 - 분할 뷰, 사용 뷰, 레이어 뷰
- 컴포넌트-커넥터 뷰타입 - 통신-프로세스 뷰, 공유 데이터 뷰
- 할당 뷰 - 배치 뷰, 구현 뷰, 작업 할당 뷰

4. 뷰선택 (4/6) – 선택 예제(A-7E) – 1. 후보 뷰 리스트 생성

- ✓ 이해관계자와 아키텍처 뷰 테이블

이해관계자	모듈 뷰			C&C 뷰		할당 뷰		
	분할	사용	레이어	통신처리	데이터공유	배치	구현	작업할당
현재와 미래의 아키텍트	d	d	d	d	d	d	s	s
프로젝트 관리자	s	s	s	o	d	o	d	d
개발 팀원	d	d	d	d	d	s	s	d
테스터와 통합 담당		d		d	s	s	d	
유지보수 담당	d	d	d	d	d	s	s	s
수행성능 분석 담당	d	d	d	d	s	d		
변경 용이성 분석 담당	d	d	d	s	s	d	o	o
설정 용이성 분석 담당	d	d	d	s	s	d		o
편당 예이전시	o	o	o	d	d			

Key: d=상세 정보(detailed information), s=일반 정보(some details), o=개요 정보(overview information), x=anything

4. 뷰선택 (5/6) – 선택 예제(A-7E) – 2. 뷰통합

✓ 뷰 선택

- 배치 뷰 제외 - 단일 프로세서 하드웨어 환경
- 작업 할당 뷰, 구현 뷰를 모듈 분할 뷰와 통합 - 모듈을 작업 할당과 개발 파일 구조의 기본 단위로 수용
- 컨설턴트들은 공유 데이터 뷰가 무용하다고 결론
- 성능 분석가들은 통신-프로세스 뷰를 사용하여 성능 모델 개발 가능
- 개발자, 테스터, 통합 담당자, 유지보수 담당자는 모듈 분할 뷰와 레이어 뷰 사용

✓ 최종 뷰 목록

- 분할 뷰
- 사용 뷰
- 레이어 뷰
- 통신-프로세스 뷰

4. 뷰선택 (6/6) – 선택 예제(A-7E) – 3. 우선순위 결정

✓ 모듈 분할 뷰와 레이어 뷰

- 대부분의 이해관계자는 모듈 분할 뷰와 관련
- 모듈 분할 뷰는 작업 할당과 관련
- 작업 할당을 위해 어떤 모듈 사용 가능한지 알아야 함 - 레이어 뷰 우선 할당

✓ 통신-프로세스 뷰

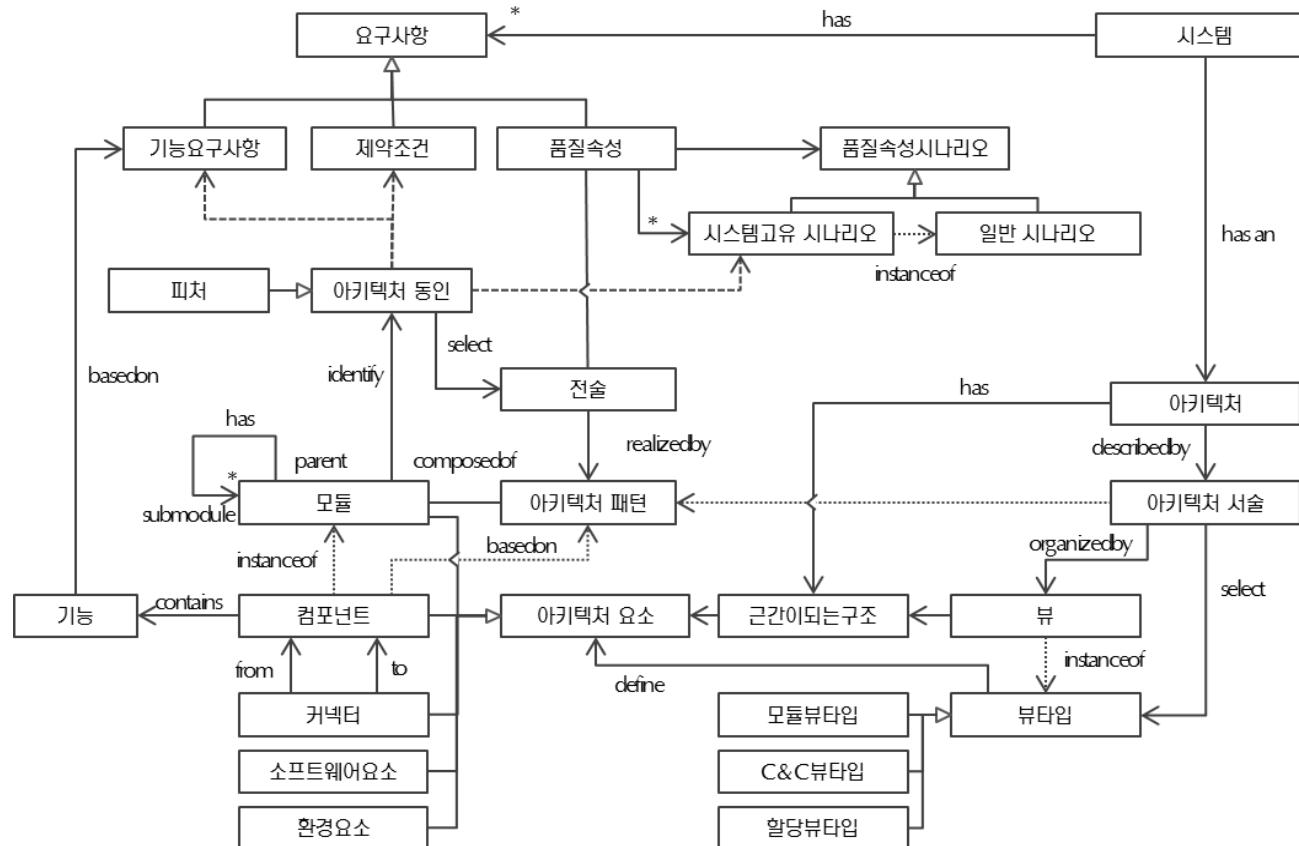
- 통신과 동기화가 결정되고 문서화될 정도로 모듈 분할이 완료된 후 작업

✓ 사용 뷰

- 가장 낮은 우선순위
- 레이어 뷰의 사용 가능 관계는 사용 뷰의 제약사항

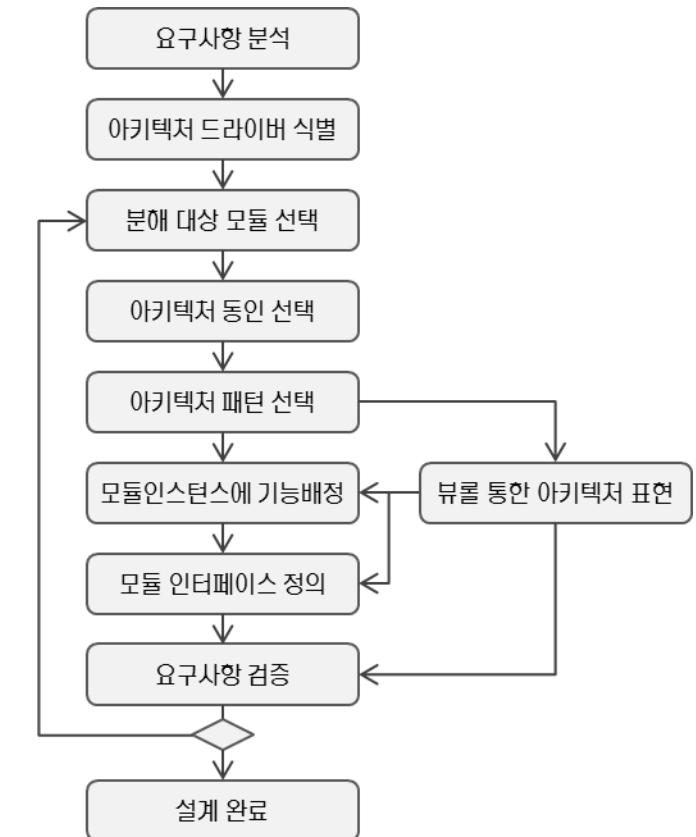
5. ADD 프로세스 (1/2)

- ✓ ADD – [Quality] Attribute Driven Design
- ✓ 아키텍팅 프로세스의 목표를 품질과 기능요구 만족으로 정함
- ✓ ADD input은 품질 속성 시나리오, 품질 목표 달성과 아키텍처 간의 관계에 대한 지식임



5. ADD 프로세스 (2/2)

- ✓ 품질 속성 상의 분해(decomposition) 절차를 기반으로 함
- ✓ 재귀적인 분해 프로세스
 - 각 단계에서, 품질 시나리오를 만족하기 위해 전술과 아키텍처 패턴을 선택
 - 패턴이 제공하는 모듈 타입을 인스턴스화하기 위해 기능을 배정
- ✓ 요구사항 분석 후 아키텍처 동인이 명확히 식별되었을 때 시작
- ✓ ADD 출력 :
 - 아키텍처에 대한 최초 몇 가지 모듈 분해 뷰들
 - 시스템은 기능과 기능들 간의 상호작용을 위한 컨테이너로 간주됨
 - 아키텍처에 대해 최초로 선을 그음 – 거친 입자 수준
 - 아키텍처 설계와 구현을 위한 상세 설계 간의 차별화



6. ADD 프로세스 – 예제 (1/9)

- ✓ 예제 – 홈 정보시스템의 차고 문 개폐를 위한 아키텍처
- ✓ ADD 입력
 - 기능 요구사항과 제약조건
 - 품질 요구사항을 시스템-고유의 품질 시나리오의 집합으로 표현
 - 일반 시나리오로부터 애플리케이션을 위해 시스템-고유 시나리오로 상세화
 - 차고 문을 위한 품질 시나리오
 - 시나리오1 - 문 개폐를 위한 장치와 제어기는 제품 라인에서 다양한 제품 유형별로 다름
 - 시나리오2 - 서로 다른 제품에 사용되는 프로세서는 서로 다름
 - 시나리오3 - 문을 내리는 중에 장애물(사람이나 물체)을 발견하면, 0.1초 내에 정지해야 함
 - 시나리오4 - 차고 문 개폐기는 제품-고유의 진단 프로토콜을 사용하는 홈 정보 시스템이 관리 및 진단을 위해 접근할 수 있어야 함
- ✓ ADD 시작
 - 아키텍처 동인(driver) 식별 완료
 - 요구 변경의 결과 또는 요구에 대한 보다 깊은 이해 후 아키텍처 동인은 변경 가능함
 - 동인이 되는 요구가 명확히 식별되면 ADD를 시작



6. ADD 프로세스 – 예제 [2/9]

✓ 예제 – 홈 정보시스템의 차고 문 개폐를 위한 아키텍처

✓ ADD 단계

- 분해할 모듈 선택
- 다음 단계에 따라 모듈 재정의

단계1: 구체적인 품질 시나리오와 기능 요구로부터 아키텍처 동인 선택

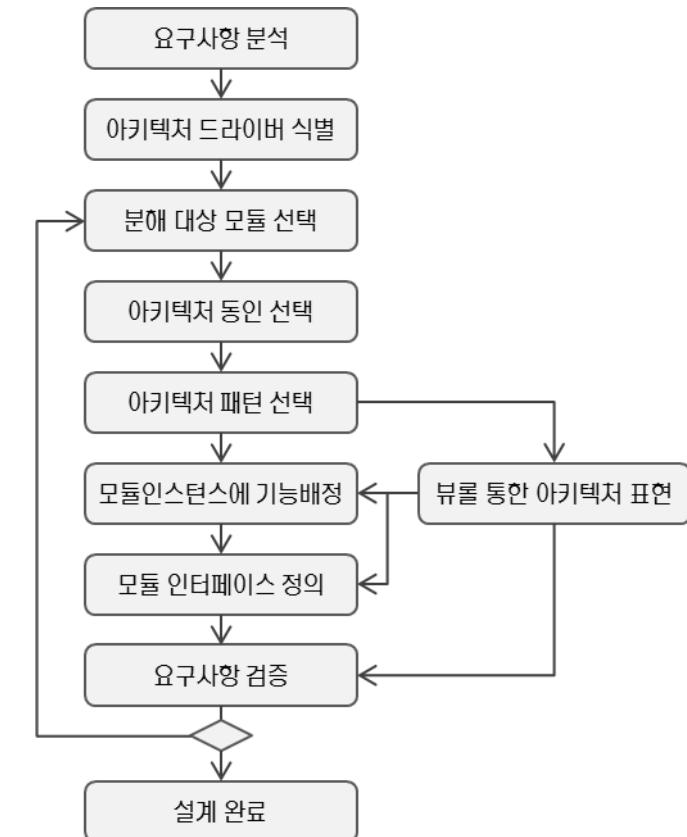
단계2: 아키텍처 동인을 만족하는 아키텍처 패턴 선택

단계3: 모듈을 인스턴스화 하고 기능을 할당하며 여러 뷰를 사용하여 표현

단계4: 하위 모듈의 인터페이스 정의. 분해는 모듈 상호작용에 대한 모듈과 제약조건을 제공. 이 정보를 각 모듈을 위한 인터페이스 문서에 문서화

단계5: 유스케이스와 품질 시나리오를 검증하고 정의하고 이 제약조건을 하위 모듈에도 적용

- 추가 분해를 필요로 하는 각 모듈에 대해 위의 절차를 반복



6. ADD 프로세스 – 예제 (3/9)

✓ 분해할 모듈 선택

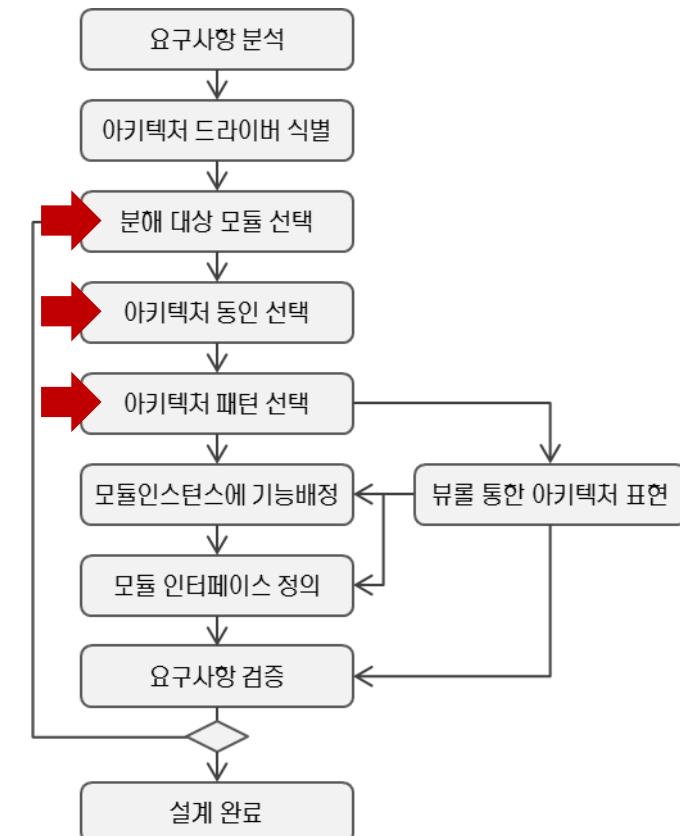
- 시스템 -> 하위 시스템 -> 하위 모듈
- 예제의 모듈 : 차고 개폐기(시스템)
- 제약 조건 : 개폐기는 홈 정보 시스템과 상호작용해야 함

✓ 아키텍처 동인 선택

- 아키텍처 동인 = 아키텍처를 이끄는 기능과 품질 요구사항의 조합
- 예제의 요구 : 실시간 수행성능[0.1초], 제품 라인 지원을 위한 변경 용이성, 온라인 진단
- 아키텍처 프로토타입 요구

✓ 아키텍처 패턴 선택

- 각 품질속성 별 전술이 존재, 전술 구현을 위해 사용할 패턴 존재
- 전술은 하나 이상의 품질에 영향을 주며, 패턴은 다른 품질 속성에 영향을 줌
- 아키텍처 설계에서, 여러 품질 간의 균형을 위해 여러 전술의 조합을 사용 (계속)



6. ADD 프로세스 – 예제 (4/9)

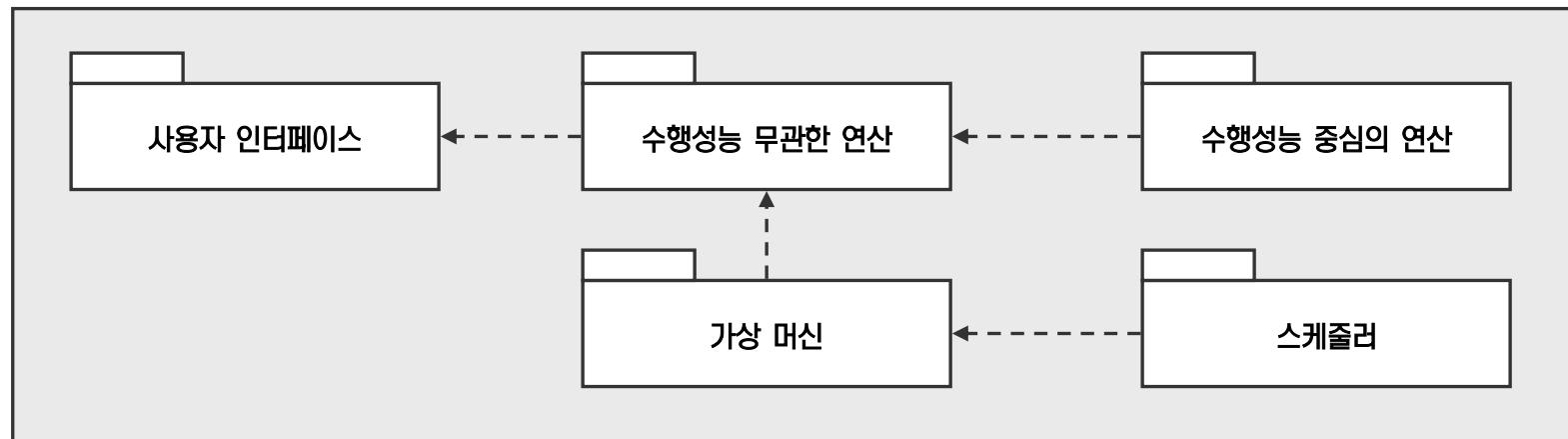
✓ 아키텍처 패턴 선택 (계속)

- 이 단계의 목표 : 모듈 타입을 구성하는 전체적인 아키텍처 패턴을 수립
- 전술 선택의 두 가지 가이드 : 동인, 적용된 패턴이 다른 품질에 주는 부작용(side effect), Side effect 의 예 : 변경 용이성을 위해 HTML 해석기를 사용, 이는 수행성능에 영향을 줌
- 예제(변경 용이성, 수행성능)를 위한 전술 세 가지

전술1: 정보 은폐, 모듈의 분리 - 사용자 인터페이스, 통신, 진단, 센서를 다루는 모듈, 각 모듈을 가상 머신 이라 함

전술2: 연산 효율성 증대

전술3: 스케줄링 정책 사용

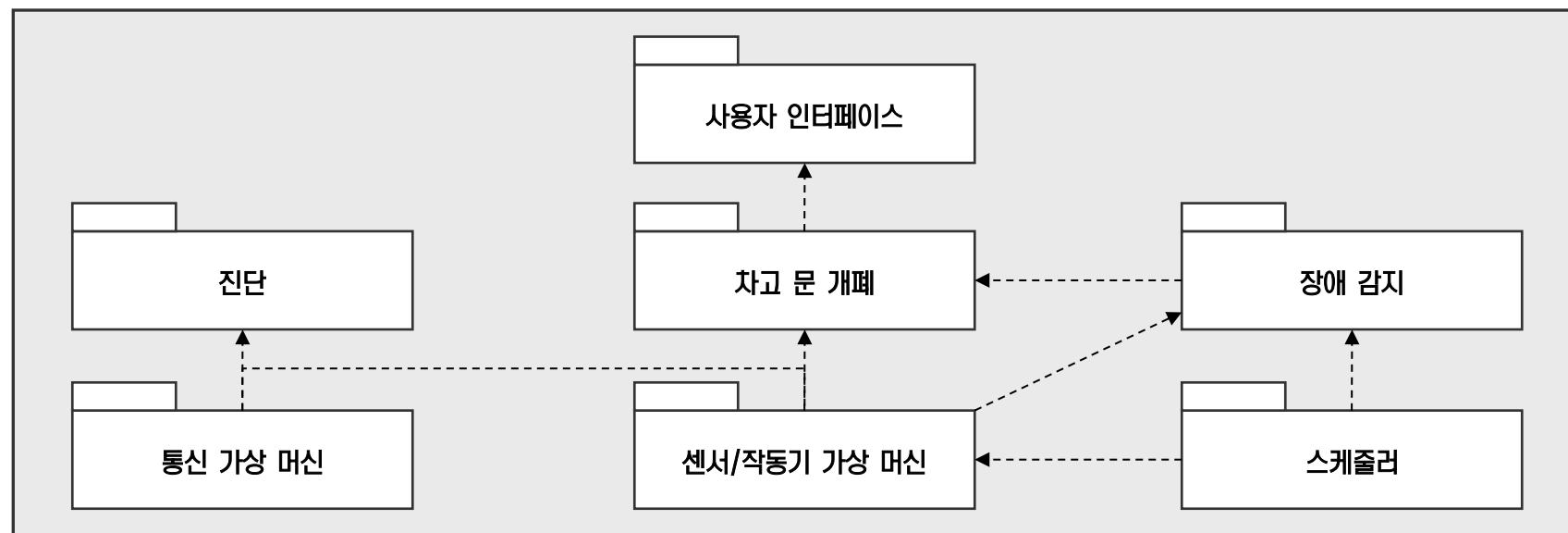


[차고 문 동인 달성을 위한 전술을 활용하는 아키텍처 패턴]

6. ADD 프로세스 – 예제 (5/9)

✓ 모듈 인스턴스화

- 수행 성능과 무관한 연산은 통신과 센서를 관리하는 가상 머신 위에서 실행
- 가상 머신 위에서 실행되는 소프트웨어 = 애플리케이션
- 실제 시스템에서 하나 이상의 모듈을 가지며, 기능의 각 “그룹”에 대해 하나의 모듈이 존재
- 예제를 위한 할당
- 장애율 감지 관리와 차고 문 정지 부분을 수행성능 중심 모듈에 할당
- 차고 문 일반 개폐는 수행성능 무관 모듈에 할당
- 진단 부분은 수행성능 무관 모듈에 할당



[차고 문 개폐기의 레벨 1 분해도]

6. ADD 프로세스 – 예제 [6/9]

✓ 기능 할당

- 상위 모듈에 관계된 유스케이스 적용은 기능 분산에 대한 이해를 높임
- 필요한 기능을 수행하기 위해 하위 모듈을 추가하거나 제거함
- 상위 모듈에 적용되는 유스케이스는 하위 모듈 안에서 일련의 책임으로 표현되어야 함
- 책임 할당을 통해 생산자/소비자 관계 형성
- 이 단계에서 정보 교환 방법, 메시지 타입 등은 설계 단계로 넘김
- 초점 : 정보 자체와 역할(생산자/소비자)
- 생산자와 소비자 간에 다양한 상호작용 패턴이 나타남(publish-subscribe)
- 품질 층족여부 확인을 위해 동적 실행시간 배치 정보 등이 추가로 필요함 (뷰를 통해 표현)

6. ADD 프로세스 – 예제 (7/9)

✓ 뷰를 통한 아키텍처 표현

- 세 가지 뷰로 충분 – 모듈 분해, 동시성, 배치
- 방법론 자체는 선택된 특정 뷰에 종속되지 않음, 따라서 필요한 뷰는 추가 가능
- 모듈 분해 뷰 – 각 모듈은 책임을 담는 컨테이너이며, 각 모듈 간의 데이터 흐름을 파악
- 동시성 뷰 – 병렬 활동과 동기화와 같은 시스템의 특성인 다음을 모델링 함, 자원 경쟁 문제, 데드락, 데이터 일치성
- 동시성 문제에 대한 고찰
 - . 두 사용자가 동시에 유사한 일을 함 - 자원 경쟁이나 데이터 일치성 문제를 다룸 [한 사람은 원격으로 문을 닫고, 다른 사람은 스위치를 이용하여 문을 염]
 - . 한 사용자가 동시에 여러 활동을 수행함 – 데이터 교환이나 활동 제어 문제를 다룸 [사용자가 문을 열면서 동시에 진단을 수행함]
 - . 시스템을 시작함 – 시스템 안에서 실행 활동에 대한 개요와 초기화 방법을 보여 줌 [차고 문 개폐 시스템은 홈 정보 시스템의 가용성에 의존하는가? 차고 문 개폐 시스템은 항상 작동하며 신호를 기다리는가?]
 - . 시스템을 종료함 – 일관된 시스템 상태의 달성이거나 저장과 같은 마무리(cleaning up) 문제를 다름
- 배치 뷰 – 다중 프로세서와 특수한 하드웨어가 사용됨, 배치 뷰 사용을 통해 원하는 품질 달성을 지원하는 배치를 결정하고 설계하는데 도움을 줌

6. ADD 프로세스 – 예제 (8/9)

✓ 하위 모듈의 인터페이스 정의

- 모듈의 인터페이스는 필요한 서비스와 특성을 보여 줌
- 구조(모듈 분해 뷰), 동적인 특성(동시성 뷰), 실행시간(배치 뷰)의 관점에서 분해를 분석하고 문서화 함으로써 하위 모듈 간의 상호작용 가정을 보여 줌
- 모듈 뷰의 문서화 내용
 - .정보의 생산자/소비자,
 - .서비스를 제공하고 사용하기 위한 모듈을 필요로 하는 상호작용 패턴
- 동시성 뷰의 문서화 내용
 - .서비스를 제공하고 사용하는 모듈에 대한 인터페이스를 리드하는 쓰레드 간의 상호작용
 - .활동 컴포넌트에 대한 정보
 - .동기화하고 직렬화 하는 컴포넌트에 대한 정보
- 배치 뷰의 문서화 내용
 - .하드웨어 요구사항 – 특수 목적의 하드웨어
 - .처리 성능 요구사항 – 프로세서의 처리 속도가 10 MIPS는 되어야 함
 - .통신 요구사항 – 1초에 1회 이상 갱신되어야 함

6. ADD 프로세스 – 예제 [9/9]

- ✓ 하위 모듈을 위한 제약 조건인 유스케이스와 품질 시나리오를 검증하고 재정의 함

- 기능 요구

- .차고 문 개폐기는 요청에 따라 지역적으로 또는 원격으로 문을 열고 닫음
- .장애가 감지되었을 경우 0.1초 이내에 멈추어야 함
- .홈 정보 시스템과 상호작용하고 원격 진단을 지원하여야 함

- 책임은 다음의 모듈로 분해

- .사용자 인터페이스, 문 개폐 모듈, 장애물 감지
- .통신 가상 머신, 센서/작동기(actuator) 가상 머신
- .스케줄러, 진단

- 제약 조건

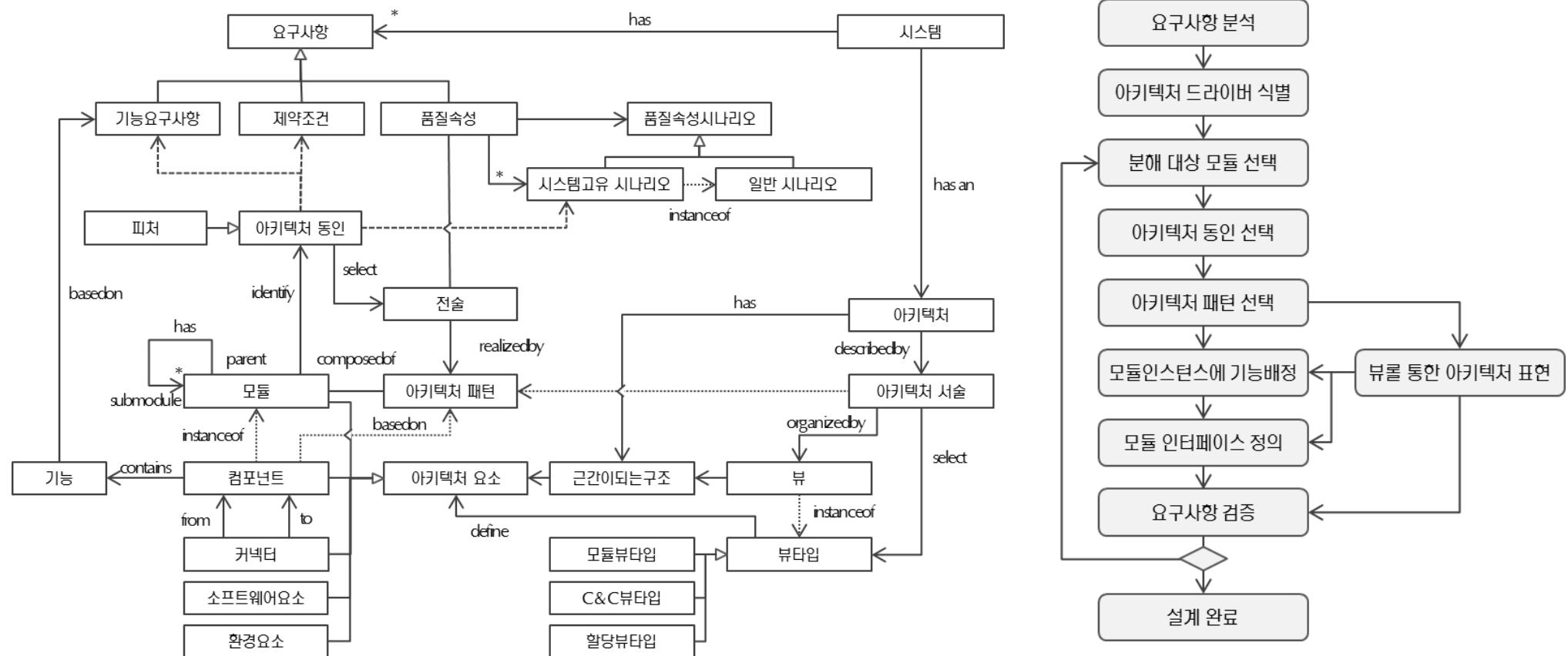
- .분해는 제약조건을 만족하여야 함
- .제약조건은 단일 하위 모듈에서 만족
- .제약 조건은 여러 하위 모듈에서 만족, 예, 홈정보 시스템과 통신이 유지되어야 함 – 통신가상머신은 통신 단절을 감지함

- 품질 시나리오

- .문을 열고 닫기 위한 장치와 제어기는 서로 다른 제품 라인에서 서로 달라야 함
- .서로 다른 제품에서 사용되는 프로세서는 달라야 함
- .차고 문을 내리는 동안 장애물을 감지하면, 문은 0.1초 이내에 멈추어야 함
- .차고 문 개폐기는 홈 정보 시스템 안에서 제품-고유의 진단 프로토콜로 접근되어야 함

7. ADD 프로세스 요약

- ✓ ADD 프로세스는 현대의 다양한 아키텍팅 프로세스의 바탕이 되는 프로세스임
 - ✓ 소규모 시스템 또는 Embedded 시스템 개발에서는 아직도 활발하게 사용되는 프로세스 임
 - ✓ 프로젝트에서 개발하려는 목표 시스템과 개발팀의 특성에 맞추어 조정(tailoring) 하여야 사용함



- ✓ 질의 응답
- ✓ 토론

감사합니다....

- ❖ 넥스트리컨설팅(주)
- ❖ CEO 송태국 / 대표 컨설턴트
- ❖ tsong@nextree.co.kr