P0. *Software Architecture Introduction*

2014

Sungwon Kang

Engraving ***The Confusion of Tongues*** by Gustave Doré (1865)

"The Tower of Babel failed not because of technological limitations or lack of resources.

It failed because of communication problems and disorganization that followed because of the lack of communication."

☛ Architecture enables communication

☛ Crucial to the success of software development

From this picture we can see ...

- For the success of a large enterprise, communication is the single most important case of failure.
- Software architecture has important lessons from this picture as it has the same goal, i.e. constructing a large system.
- Keeping this in mind, . . .
- And because our topic, software architecture, is an important way to construct and evolve software
- Let's investigate the question "What is (computer) software?"

  => A useful way of looking at it is to view it as a means of realizing "services"

서비스

"Man is by nature a social animal;

an individual who is unsocial naturally and not accidentally is

either beneath our notice or more than human.

Society is something that precedes the individual.

Anyone who either cannot lead the common life or is so self-sufficient as not to need to, and therefore does not partake of society,

is either a beast or a god. " — *Aristotle* in Politics

- **"인간은 사회적 동물이다"**
- 서비스**: "A do X for B"**

=> **"인간활동은 서비스이다" <–> "Self-service"**

서비스의 예

A	X	B
나	문을 연다	다른 사람
선생님	가르친다	학생
시계	alarm을 울린다	주인에게
물레(바퀴)	올린다	방아
은행	대출을 해준다	고객

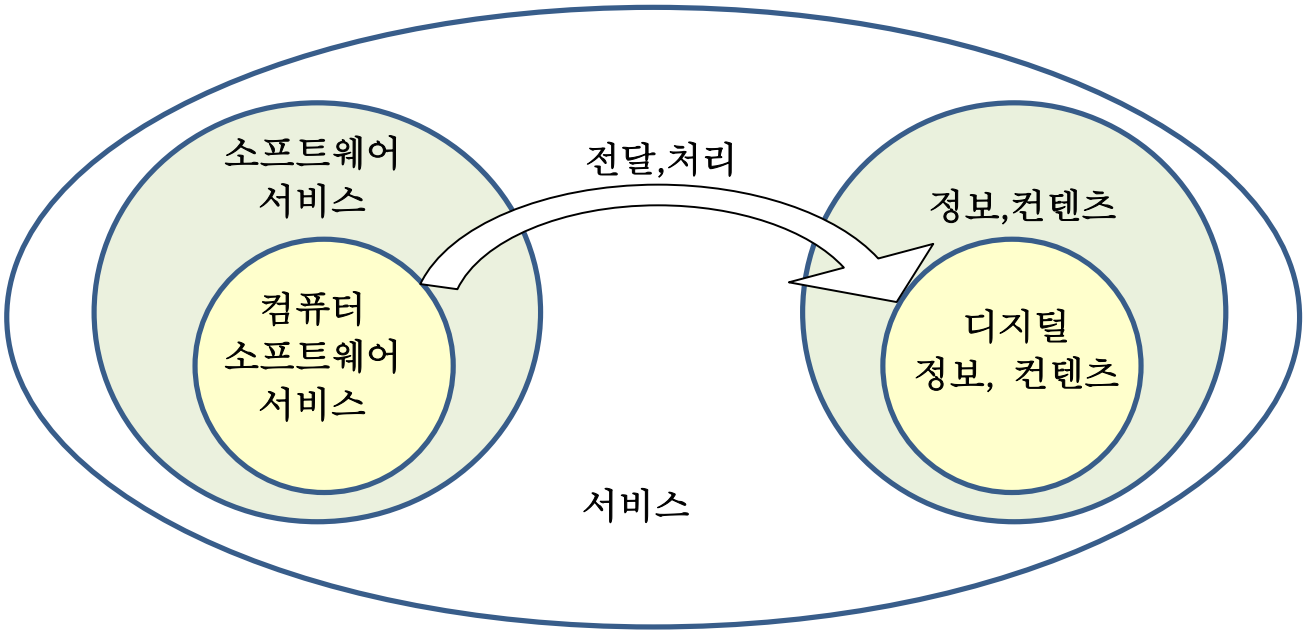
컴퓨터소프트웨어와 서비스

- 컴퓨터 소프트웨어에 우리는 여러 가지를 담는다 **(=** 구현한다**).**
  그러나 정보, 데이터, 콘텐츠는 담는 방법이 비교적 간단하지만,
  서비스를 담는 것은 간단하지 않다.

- 하드웨어에 담을 수 있는 것은**?**
  - 힘, 운동, 서비스 등
  - 그러나 아주 간단한 서비스만 담을 수 있다
    (Hardware + Logic (= Software))
    예**)** 시계, 자명종, 자전거 등.

☛ 하드웨어를 "서비스"관점으로 생각하는 것은 부적절

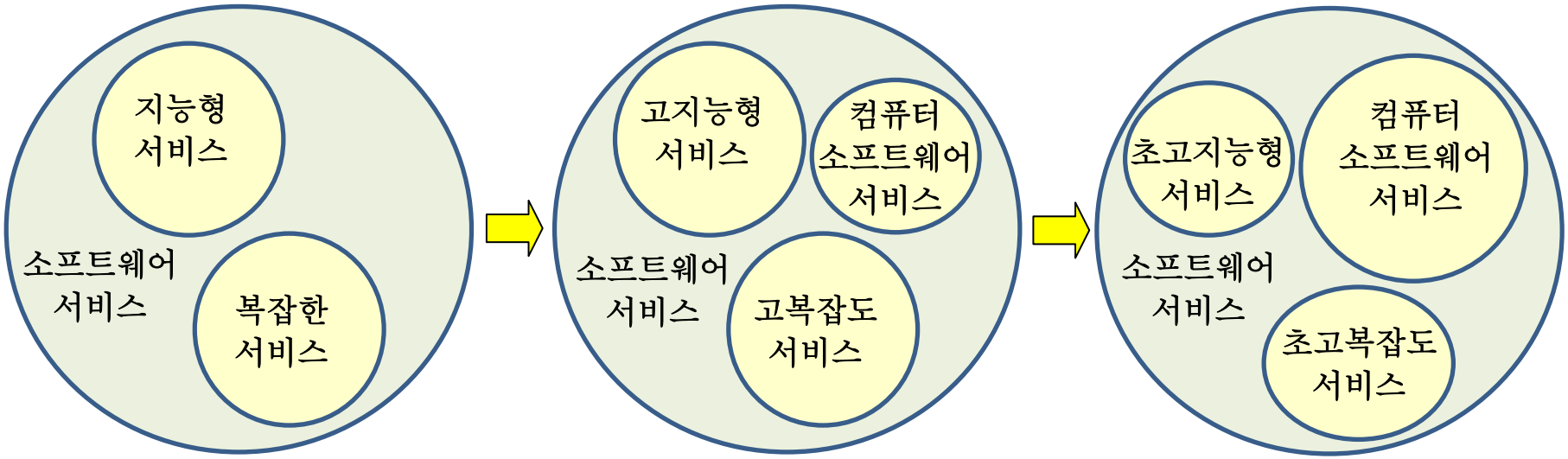
서비스, 소프트웨어 서비스, 컴퓨터 소프트웨어 서비스



- **소프트웨어**: 프로그램, 절차
- **소프트웨어 서비스**: 서비스를 위한 소프트웨어
- 소프트웨어서비스가 아닌 서비스?
  - "I live for Y."
  - Non-programmable 서비스
- **컴퓨터 소프트웨어 서비스**: 컴퓨터 소프트웨어에 의해 수행되는 서비스

소프트웨어 ≠소프트웨어 서비스

(생활과 산업 속의) 컴퓨터 소프트웨어 역할의 발전



(a) 과거 (b) 현재 (c) 미래

소프트웨어 서비스 중 간단한 것이 먼저 컴퓨터소프트웨어에 의한 서비스로 바뀌었다.

컴퓨터 소프트웨어 서비스의 시대

- Pervasive Computing Era (Mark Weiser) !
  - ☛ Need much of computer software service
- "Internet of Things" in Future Internet
  - ☛ Every "thing" is a "service(s)" provider

- Those who can handle computer software will dominate the world !
  - "***Our civilization runs on software***." - Stroustrup

- How can we make computer software well?
  - Many skills needed
  - The most important one is “software (architecture) design”.
    => See the definitions of software architecture.

소프트웨어 아키텍처 개요

1. 소프트웨어 아키텍처의 개념과 중요성
2. 소프트웨어 아키텍처의 정의
3. 소프트웨어 아키텍처의 설계 방법
4. 아키텍처 설계의 근본 원리들

1. 소프트웨어 아키텍처의 개념과 중요성

- 개념: Design vs. Architecture
- 중요성: Role of architecture in Software Development

What is software architecture?

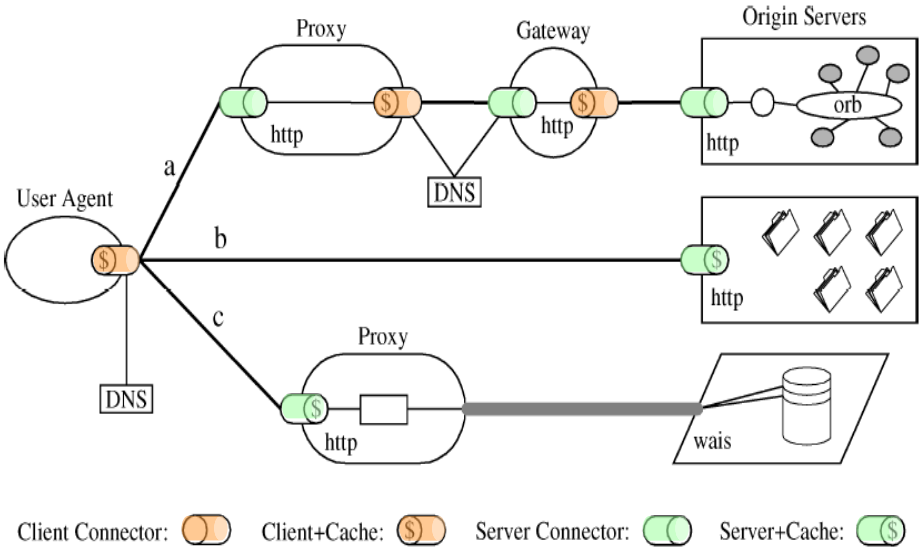
- **Before constructing a building, we design its architecture first.**

- **Building Architecture**
  - Supports the whole building to stand
    <= construction decisions
    => It is difficult and (sometimes) dangerous to change architecture
  - Is an abstract entity that exists before actually constructing building with material such as metal, mud or bricks

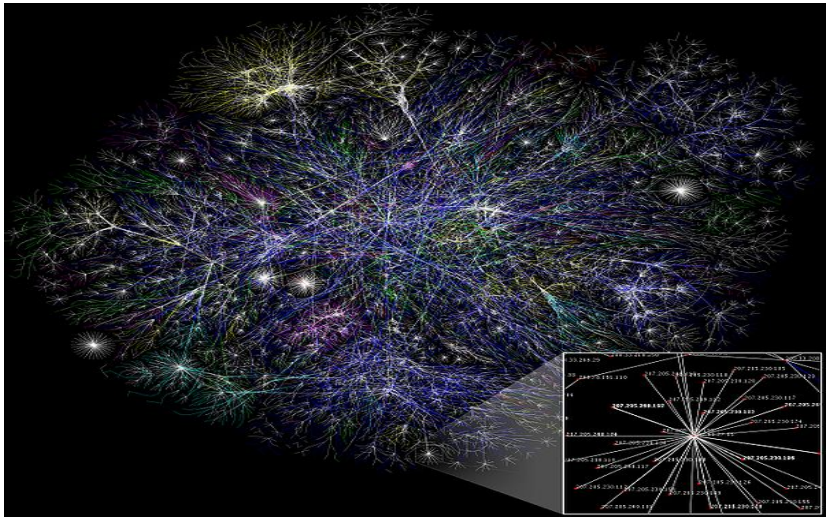
우리가 원하는 모습(=> 기능)과 특징(=>속성)을 사전에 검증가능

What is software architecture?

- Shows the overall relation between components of a software system and includes important development decisions
- Determines performance and quality of the system
- Helps us understand the whole system “before” it comes into existence



WWW Architecture



Internet Architecture

Slide Source: Mary Kathryn Thompson

What do we Design? (Man does this.)

- Things
  - Bridges
  - Toys
  - Sculptures
  - Meals
- Systems
  - Governments
  - Universities
  - Factories
  - Hospitals
- Ideas
  - Laws
  - Philosophies
  - Classes
- Documents
  - Books
  - Newspapers
  - Websites

make

build

create

write

Far beyond engineering

KAIST

Slide Source: Mary Kathryn Thompson

What is Design?

Design is a human activity which combines resources (knowledge, skills, experiences, creativity, tools, materials, etc.) to meet a need, accomplish a goal, or create an artifact.

Focus is usually on new, different or better.

ksw- What is missing here when "we" think of design as software designers ?

KAIST

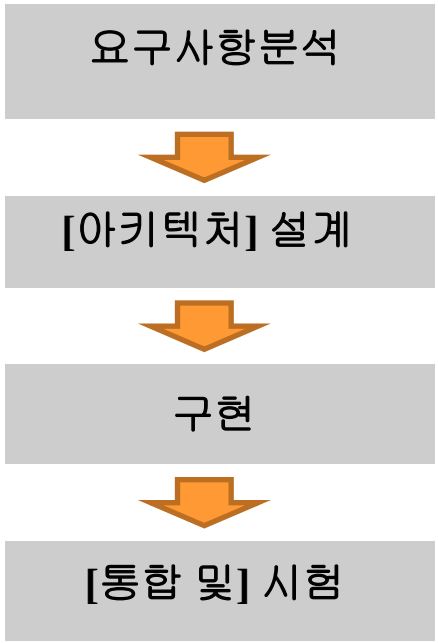
Design vs. Architecture

- Although the previous discussion touches important aspects of design, such as
  - Design as a creative activity
  - To some extent, problem solving
- It lacks something from the software development point of view.

- ksw: "*In other words, 'architecture design' is a design activity but not all design activities are architectural design.*"

Design vs. Architecture

Accepting that
"design" is an essential step in software development.



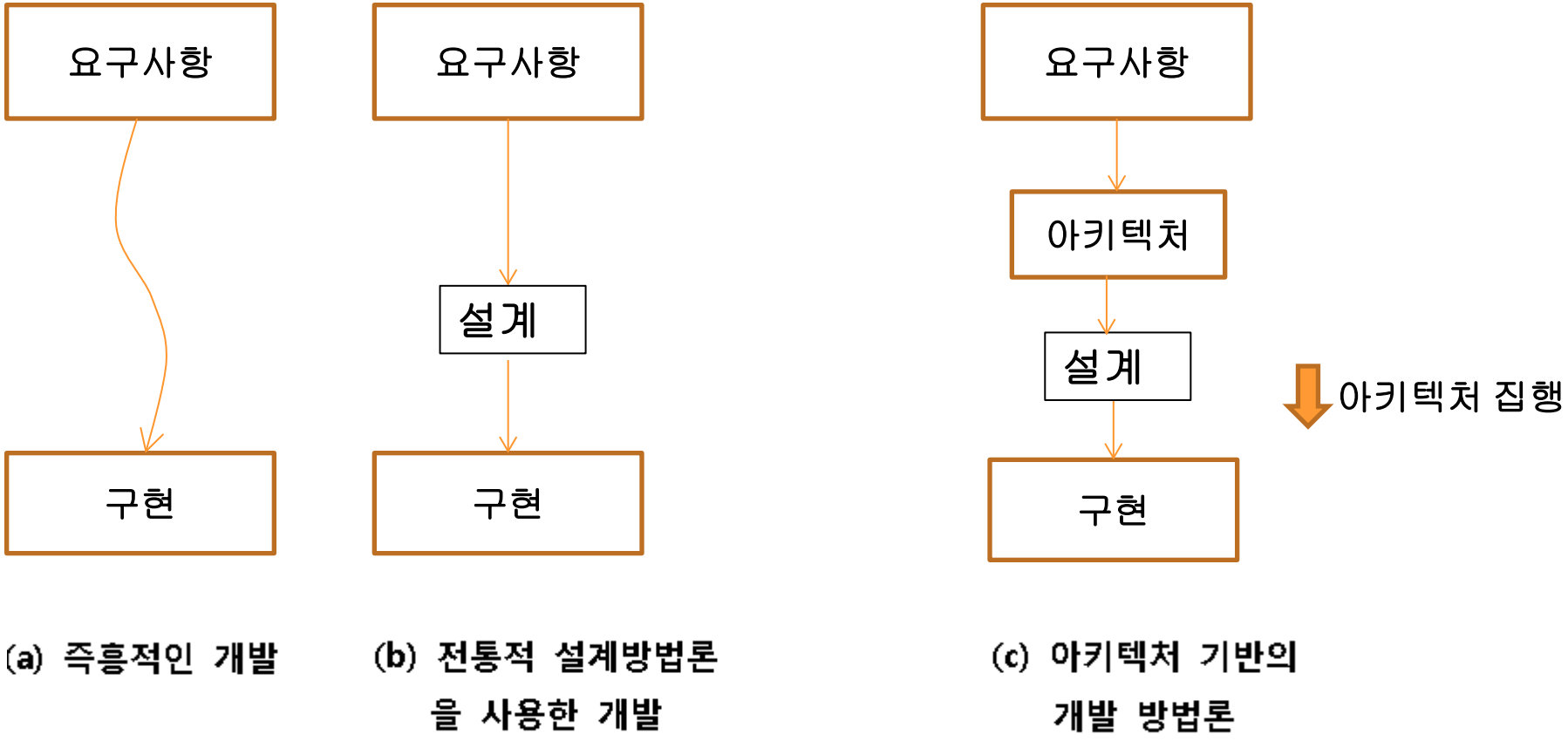
개발순기

- It helps problem solving of a complex problem.
  (=> complexity)
- It helps tackling large problems.
  (= scale)

(=> Abstraction,
Divide & Conquer)

아키텍처 설계의 중요한 원리들

아키텍처 기반 개발의 장점



(a) 즉흥적인 개발

(b) 전통적 설계방법론을 사용한 개발

(c) 아키텍처 기반의 개발 방법론

- 기존의 설계와 아키텍처 설계의 차이

  - Bottom-up vs.Top-down

  - 특히 객체지향 설계에서는 객체가 우선적인 도출 단위이고 컴포넌트나 서브시스템은 사후적인 결정으로 얻어짐.

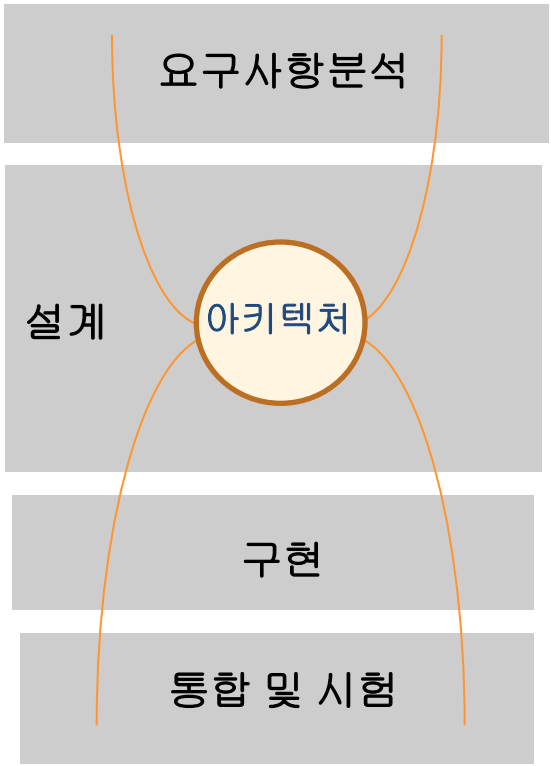
  - 기존의 설계 방법은 기능을 실현하는데, 초점을 맞추었으나 아키텍처 설계는 Quality-centric

소프트웨어 개발의 중심축으로서의 아키텍처

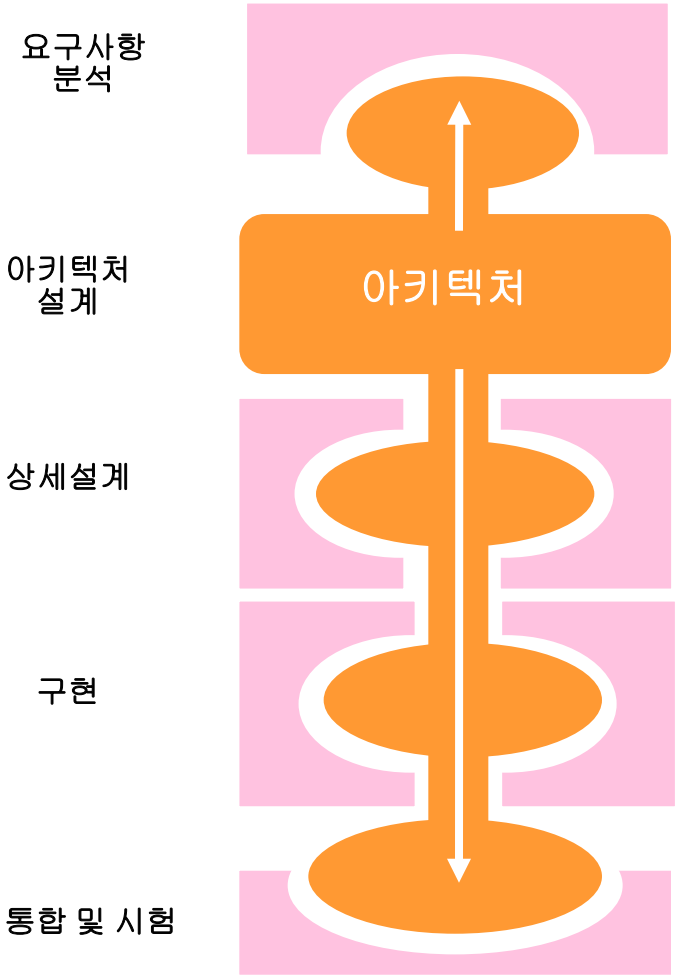
Software architecture plays a pivotal role in software development.

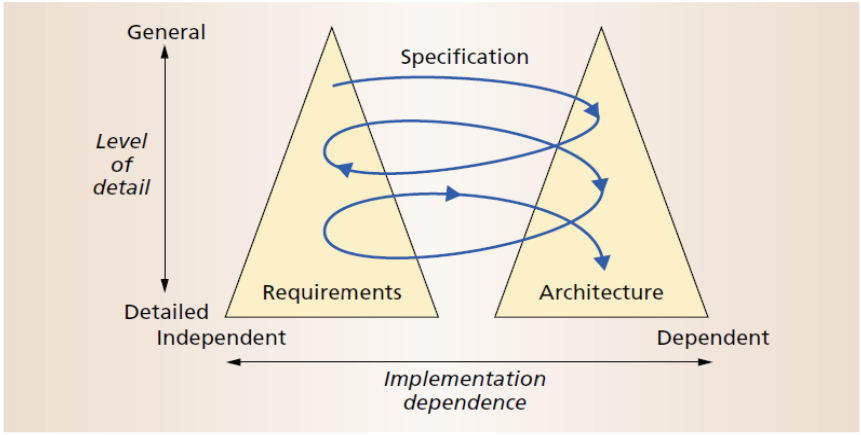
This is not what we “must” do but if we want to do well, we had better do this way.

소프트웨어 개발의 중심축으로서의 아키텍처



개발순기에 있어서 아키텍처의 역할





- 요구사항과의 **interplay**
- 이어지는 단계에 대한 중요한 제약을 제시

소프트웨어 아키텍처의 역할

- Ian P. Sharp at 1969 NATO Conference on SSE Techniques :

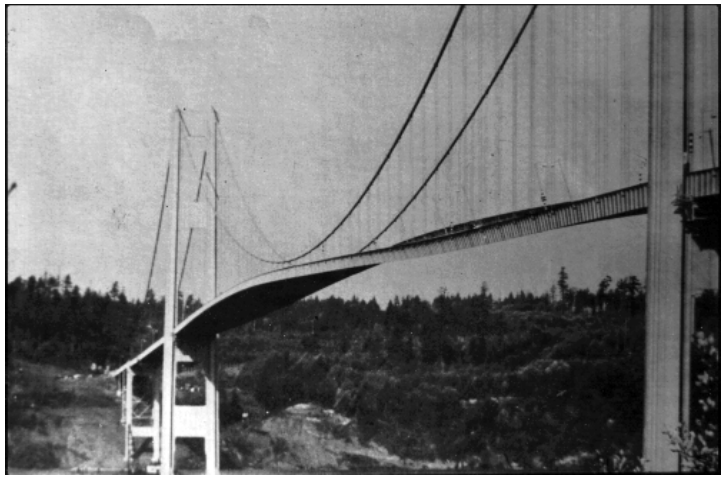
"우리가 필요한 것은 소프트웨어공학 이외에도 더 있다. ...
그것은 바로 소프트웨어 아키텍처이다.
아키텍처는 공학과 다르다. ...
예를 들어 ... OS/360은 부분부분 매우 코딩이
잘 되었다. 세부적으로 들어가 보면 OS의 부분부분은 우리가 좋은
프로그래밍 기법이라고 보는 기법들과 모든 개념들을 쓰고 있다.
그런데 OS/360가 형체 없는 프로그램 덩어리(amorphous lump of program)인 것은 아키텍트가 없었기 때문이다.
설계는 많은 그룹의 엔지니어들에게 위임되었고, 그들은 각자의
아키텍처를 반영하여야 하였다.
그리고 이 덩어리 들은 최종적으로 통합하였을 때에는
전체가 부드럽고 아름다운 소프트웨어가 되지 못하였다." [Kruchten 06]

Example of a Good Architecture

Lasted for more than 1200 years under heavy pressure from the above and the sides.

Sukgoolam (석굴암): Built in AD 751.

Example of a Bad Architecture



Tacoma Narrows Bridge
Opened: July 1, 1940
Collapsed: Nov. 7, 1940

The old Tacoma Narrows Bridge twisted and vibrated violently under 64 km/h winds on the day of the collapse.

http://en.wikipedia.org/wiki/Wikipedia:Featured_picture_candidates/Tacoma_Narrows_Bridge_Collapse

2. 소프트웨어 아키텍처의 정의

- 아키텍처 와 소프트웨어 아키텍처

  - 아키텍처라는 용어는 컴퓨터 아키텍처의 의미로 1980년대까지 쓰였고, Fred Brooks, Butler Lampson, David Parnas, John Mills등에 의하여 시스템의 "소프트웨어 조직"의 의미로 연구되었다.

소프트웨어 아키텍처의 다양한 정의

- "소프트웨어 아키텍처는 소프트웨어 시스템들의 큰 규모의 구조와 실행에 관한 연구이다" [Shaw 89].
- 소프트웨어 아키텍처란 "시스템의 근본적인 조직형태로써, 그것은 구성컴포넌트들과, 그들 서로와 환경에 대한 관계 그리고 그 설계와 진화를 관장하는 원칙들에 담겨있다" [IEEE 1471 00].
- 소프트웨어 아키텍처란 "프로그램 혹은 컴퓨팅 시스템의 소프트웨어 아키텍처란 소프트웨어 구성요소, 이들 구성요소의 가시적인 속성, 그리고 구성요소 사이에 관계로 구성된 시스템의 전체적인 구조 또는 구조들을 말한다“ [Bass 03].
- 소프트웨어 "아키텍처란 다음과 같은 것에 관한 **중요한 결정들의 집합**이다. 1)시스템의 구조를 나타내는 구조적 구성요소와 그들을 결합시키는 인터페이스, 2)그들의 협동을 통해 나타나는 구조적 구성요소와 행위적 구성요소들의 결합을 점진적으로 서브시스템에 맵핑을 하는 것, 3)구조를 이끌어 나가는 아키텍처 스타일의 선택" [Jacobson 99][Kruchten 01]

우리의 "소프트웨어 아키텍처" 정의

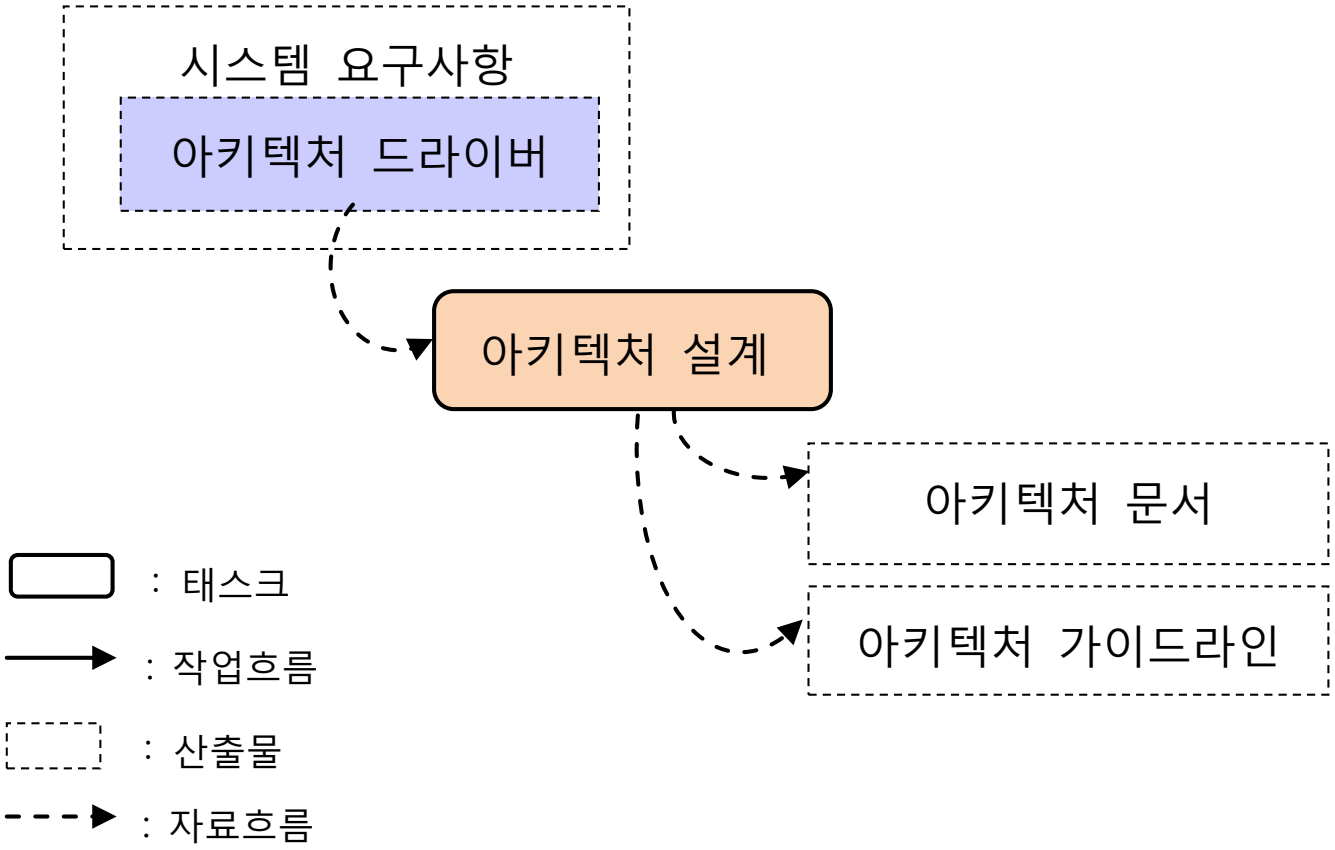
소프트웨어 아키텍처는
소프트웨어 시스템이 요구되는 기능과 품질을 궁극적으로 갖도록,
소프트웨어 시스템을 용이하게 구축하고
또한 지속적인 사용과 개선을 위하여 필요한 진화성을 갖도록 하는
소프트웨어 시스템의 구조 와 이어지는 개발에 관한 중요한 결정이다.

☚ 묵시적인 품질요구사항을 명시적으로 표현

- 구축용이성
- 진화성

3. 소프트웨어 아키텍처의 설계

아키텍처 설계의 입력물과 출력물

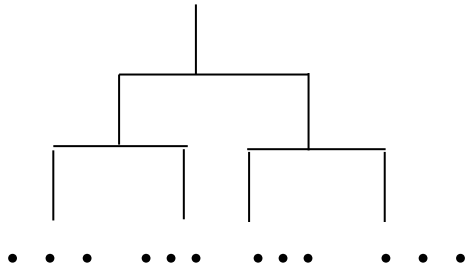


The result (Include "As is")

The rest (Include "To be")

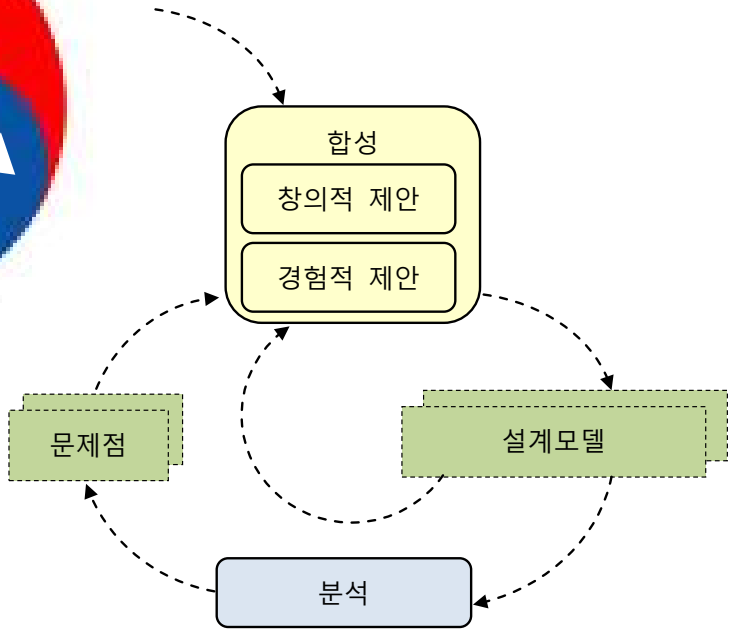
3. 소프트웨어 아키텍처의 설계

1. 분할 정복 으로서의 설계활동

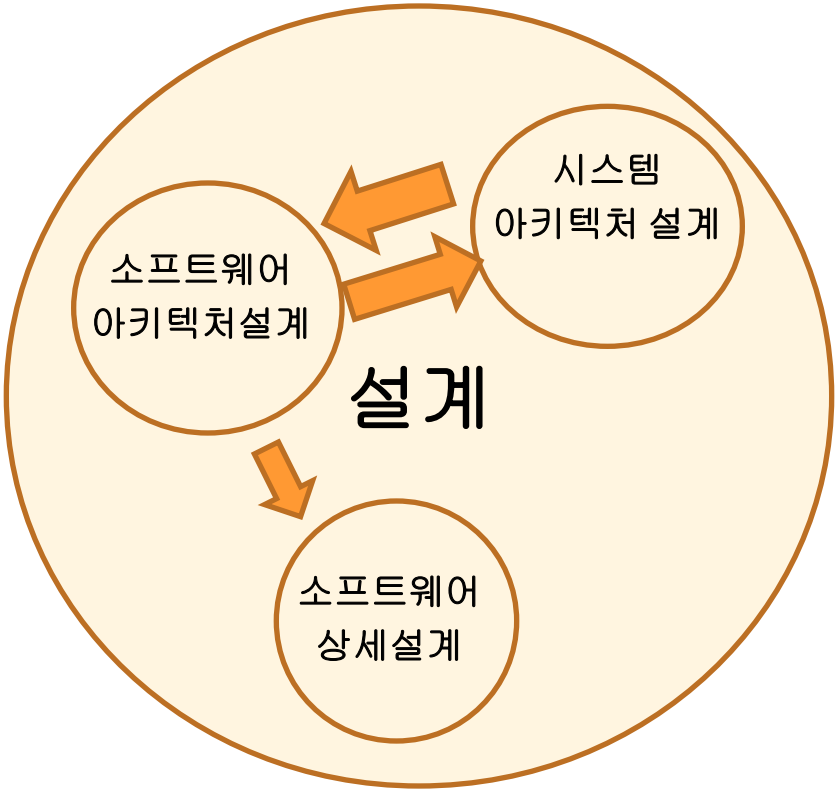


2. 합성과 분석의 반복으로 이루어진 과정으로서의 설계활동

3. 경험과 창의성의 결합으로서의 설계활동



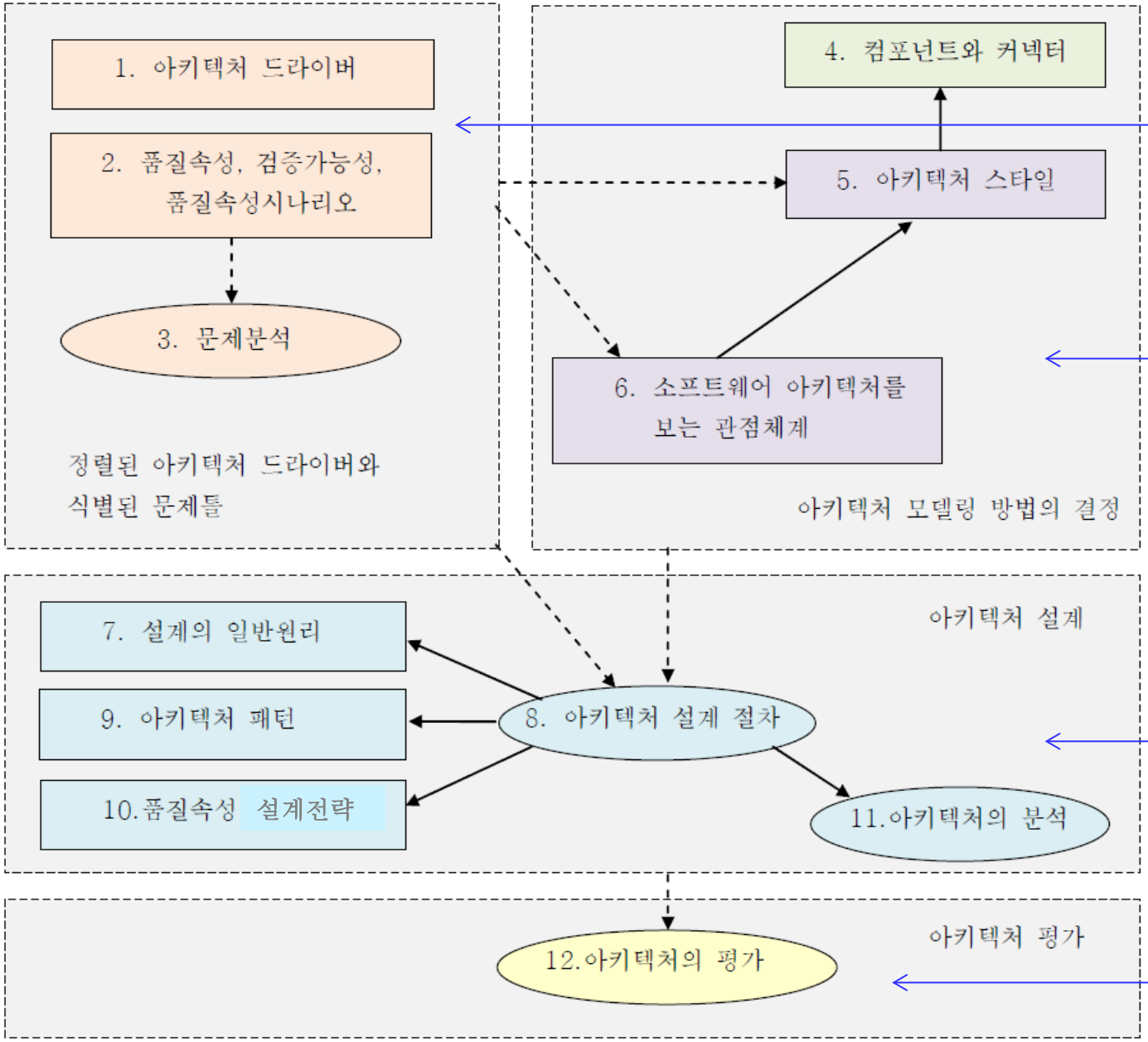
다양한 종류의 설계들 사이의 관계



4. 소프트웨어 아키텍처 설계의 근본 원리들

1. 아키텍처 드라이버
2. 품질속성, 검증가능성, 품질속성시나리오
3. 아키텍처 문제분석
4. 컴포넌트와 커넥터
5. 아키텍처 스타일
6. 소프트웨어 아키텍처를 보는 관점체계
7. 설계의 일반원리
8. 아키텍처 설계 절차
9. 아키텍처 패턴
10. 품질속성 설계전략
11. 아키텍처의 분석
12. 아키텍처의 평가

Experience cannot be replaced but knowing key principles will speed up the learning process and give a firm understanding. - ksw


1. 아키텍처 드라이버
2. 품질속성, 검증가능성, 품질속성시나리오
3. 문제분석
정렬된 아키텍처 드라이버와 식별된 문제들
4. 컴포넌트와 커넥터
5. 아키텍처 스타일
6. 소프트웨어 아키텍처를 보는 관점체계
아키텍처 모델링 방법의 결정
Requirements
How to model architecture
7. 설계의 일반원리
9. 아키텍처 패턴
10. 품질속성 설계전략
8. 아키텍처 설계 절차
11. 아키텍처의 분석
아키텍처 설계
Modeling process
12. 아키텍처의 평가
아키텍처 평가
Evaluation of the modeling results
: 활동 원리
: 산출물 원리
A → B: : A는 B에 의존한다 (혹은 A는 B를 이용한다.)
A - - → B: : A는 B의 입력이다.

A “Meta-” Principle – Documentation !

- "Meta" means "one level up".
  It is "meta" because without it other principles would be useless.
- Architecture and its documentation provides a **communication medium** containing the most important development decisions.
- This will enable us to solve the Tower of Babel Problem **!**

Textbooks

[Kang 12] 강성원, 소프트웨어 아키텍처로의 초대 - 소프트웨어 아키텍처 설계의 근본 원리들, 홍릉과학출판사, **2012.10.**

[Hofmeister 00] Hofmeister, C., Nord, R., and Soni, D., *Applied Software Architecture*, Addison-Wesley, 2000.

[Bass 13] Bass, L., Clements, P., Kazman, R., *Software Architecture in Practice*, 3rd ed., Addison-Wesley, 2013.

[Taylor 09] Taylor, R. N., Medvidovic, N., Dashofy, E., *Software Architecture: Foundations, Theory, and Practice*, Wiley, January 2009.

[Clements 11] Clements, P., Bachmann, F., Bass, L., Garlan, D., Ivers, J., Little, R., Merson, P., Nord, R., and Stafford, J., *Documenting Software Architectures: Views and Beyond*, 2nd Ed., Addison-Wesley, 2011.

Questions?