A01. Documenting Software Architecture

2014

Sungwon Kang

ToC

1. IEEE1471 *Recommended Practice for Architectural Description*
2. Documenting and Formalizing Architectures (David Garlan)
3. 아키텍처 설계 문서 템플릿

References

- **[Kang 12]** 강성원**,** 소프트웨어 아키텍처로의 초대 **-** 소프트웨어 아키텍처 설계의 근본 원리들**,** 홍릉과학출판사**, 2012.10. (**제 **III**부**)**
- [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.
  - https://wiki.sei.cmu.edu/sad/index.php/Main_Page
- [IEEE 1471] IEEE-Std-1471-2000, *Recommended Practice for Architectural Description of Software-Intensive Systems.*
- [ISO42010 07] ISO/IEC 42010, *Systems and Software Engineering - Recommended Practice for Architectural Description of Software-Intensive Systems*, 2007.)
- [ISO42010 11] ISO/IEC/IEEE 42010, *Systems and Software Engineering - Architecture Description*, 2011.

1. IEEE 1471

Recommended Practice for Architectural Description of Software-Intensive Systems

What is IEEE 1471?

- **"IEEE-Std-1471-2000, *Recommended Practice for Architectural Description of Software-Intensive Systems"***
- **An *architectural description (AD)* is a collection of products to document an architecture.**
- **Specify certain (minimal) required content of an AD reflecting current practices and consensus.**
- **Does NOT specify**
  - The format or media for an architectural description
  - particular architecture description languages
  - required views or models
  - required formal consistency or completeness criteria
- **Does NOT assume or prescribe a specific life cycle .**

Organization of IEEE 1471

1. Overview
- 1.1 Scope
- 1.2 Purpose
- 1.3 Intended users
- 1.4 Conformance to this standard

2.References

3.Definitions and acronyms

4.Conceptual framework
- 4.1 Architectural Description in context
- 4.2 Stakeholders and their roles
- 4.3 Architectural activities in the life cycle
- 4.4 Uses of ADs

5. Architectural Description practices
- 5.1 Architectural documentation
- 5.2 Identification of stakeholders and concerns
- 5.3 Selection of architectural viewpoints
- 5.4 Architectural views
- 5.5 Consistency among architectural views
- 5.6 Architectural rationale
- 5.7 Example use

A Bibliography

B Notes on terminology

C Examples of viewpoints

D Relationship to other standards

1.1 The Scope (of IEEE 1471)

a) Expression of the system and its evolution
b) Communication among the system stakeholders
c) Evaluation and comparison of architectures in a consistent manner
d) Planning, managing, and executing the activities of system development
e) Expression of the persistent characteristics and supporting principles of a system to guide acceptable change
f) Verification of a system implementation's compliance with an architectural description
g) Recording contributions to the body of knowledge of software-intensive systems architecture

1.2 Purpose (of IEEE 1471)

- To facilitate the expression and communication of architectures and
- Thereby lay a foundation for quality and cost gains through standardization of elements and practices for architectural description.

1.3 Intended users (of IEEE 1471)

- Those that use, own, and acquire the system (users, operators, and acquirers, or *clients*)
- Those that develop, describe, and document architectures (architects)
- Those that develop, deliver, and maintain the system (architects, designers, programmers, maintainers, testers, domain engineers, quality assurance staff, configuration management staff, suppliers, and project managers or *developers*)
- Those who oversee and evaluate systems and their development (CIOs, auditors, and independent assessors)

1.4 Conformance to this standard

- An architectural description conforms to this recommended practice **if** that description meets the requirements listed in Section 5.

2. References

- IEEE Std 610.12
  IEEE Standard Glossary of Software Engineering Terminology, 1990.

- IEEE/EIA Std 12207.0
  IEEE/EIA Standard—Industry Implementation of ISO/IEC 12207: 1995, Information Technology—Software life cycle processes, 1996.

3. Definitions and acronyms

(Skipped)

4. Conceptual framework

- *Conceptual framework* for architectural description establishes terms and concepts pertaining to the content and use of architectural descriptions.

4.1 Architectural Description in context
4.2 Stakeholders and their roles
4.3 Architectural activities in the life cycle
4.4 Uses of ADs

4.1 Architectural Description in context (1/3)

- The environment is also called *context*.
- *Concerns* are those interests which pertain to the system's development, its operation or any other aspects that are important to one or more stakeholders. Include system considerations such as performance, reliability, security, distribution, and evolvability.
- A system exists to fulfill one or more missions in its environment.
- Every system has an architecture.
- The architecture of a system, which is conceptual, vs. particular descriptions of that architecture, which are concrete products or artifacts.
- Architectural descriptions (ADs) are the subject of this recommended practice.

4.1 Architectural Description in context (2/3)

- An architectural description is organized into one or more (*architectural*) *views*
- Each view addresses one or more of the concerns of the system stakeholders.
- The term *view* is an expression of a system's architecture with respect to a particular viewpoint.
- Other information, not contained in any constituent view, may appear in an AD, as a result of an organization's documentation practices.

  E.g.
  - System overview
  - System context
  - Stakeholders and their key concerns
  - Architectural rationale

4.1 Architectural Description in context (3/3)

- A ***viewpoint***
  - establishes the conventions by which a view is created, depicted and analyzed.
  - determines the languages (including notations, model, or product types) to be used to describe the view, and any associated modeling methods or analysis techniques.
- An **architectural description** selects one or more viewpoints for use, typically based on consideration of the stakeholders.
- A (new) viewpoint definition may originate with an AD, or, in the case of a (commonly used) ***library viewpoint***, it is defined elsewhere.
- A **view** may consist of one or more *architectural models.*

Problem: No design decisions
No connection with Rational

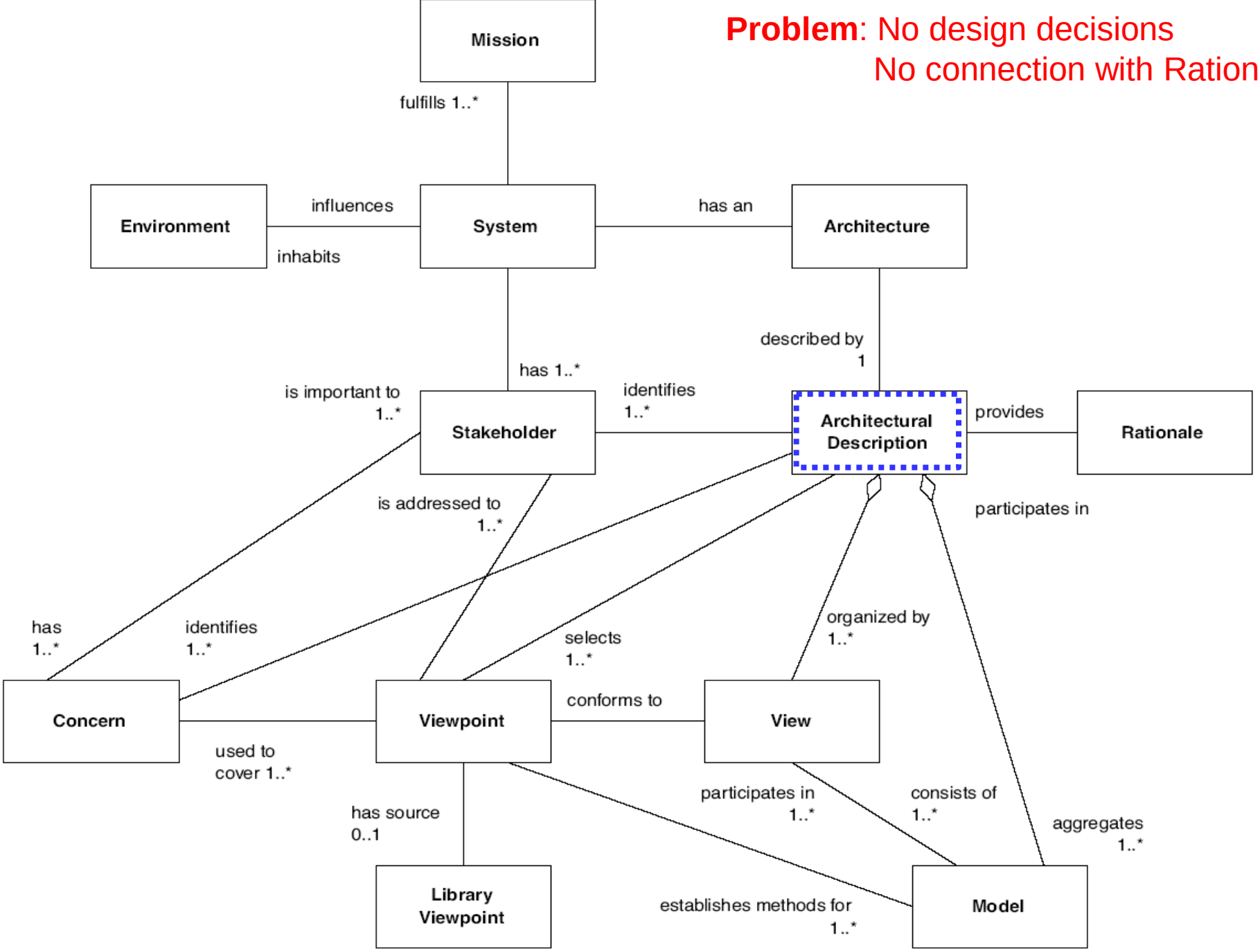


Figure 1—Conceptual model of architectural description

4.2 Stakeholders and their roles

- **Stakeholders**: clients, users, the architect, developers, and evaluators.

- Two key **roles** among stakeholders:
  - the acquirer (or client)
  - the architect.

- The architect records the architecture for the system's stakeholders in the form of an architectural description.

- In this form, the architecture serves to guide the system's developers.

4.3 Architectural activities in the life cycle

- Architecting contributes to the development, operation, and maintenance of a system from its initial concept until its retirement

- Detailed systems engineering activities, such as detailed requirements definition and interface specification and the architecting of major subsystems typically follow development of the system architecture.

- Recommended uses of AD: 4.3.1 – 4.3.4
  - 4.3.1 Scenario: architecture of single systems
  - 4.3.2 Scenario: iterative architecture for evolutionary systems
  - 4.3.3 Scenario: architecture of existing systems
  - 4.3.4 Scenario: architectural evaluation

4.3.1 Scenario: architecture of single systems

- For new system construction, architecting is carried out in response to the user-acquirer's vision for the system, the system goals, and stakeholders' needs and constraints.

- The primary stakeholders are the user-acquirer and the system developers.

- **Usages:**
  - To predict the fitness of a system that conforms to the AD.
  - AD will typically evolve throughout the life cycle, and can thus act as a means for assessing changes to the system.

4.3.2 Scenario: iterative architecture for evolutionary systems (1/2)

- The architecture, the delivered systems, and the stakeholders co-evolve.
  - The set of stakeholders in the systems and architecture will change as the architecture evolves.

- An initial architecture is prepared in response to the current and expected user needs, using the current and estimated technological capabilities.

- The AD is used to guide each system as it moves through development, and appropriate AD versions are used to evaluate each system on delivery.

4.3.2 Scenario: iterative architecture for evolutionary systems (2/2)

- Architectures developed under this scenario will naturally emphasize flexibility and adaptability more strongly than single system architectures.

- ADs for evolving systems will typically be developed in an iterative pattern of version changes.
  - Can be established by the acquirer to organize the acquisition and evolution of a sequence of systems.

- One way to evolve is a product line architecture.

4.3.3 Scenario: architecture of existing systems

- System development has taken place without an AD.
  - The built system will have an architecture but lack an AD.
  - In this case, an AD can be created through a reverse-engineering effort and using this recommended practice.
  - This AD is then used to develop a new system, to guide maintenance or evolution activities, or to establish an evolutionary architecture as in 4.3.1 and 4.3.2.
- In some cases, a new AD is needed because the original description does not provide views that are necessary later in a system's life cycle.

4.3.4 Scenario: architectural evaluation

- Determine the quality of an AD and predict the quality of systems whose architectures conform to the AD.
- To evaluate conformance of the system's architecture to an AD, a process for reverse engineering is needed (see 4.3.3).

Do you think that reverse engineering is needed?

4.4 (Specific) Uses of ADs (1/2)

a) Analysis of alternative architectures
b) Business planning for transition from a legacy architecture to a new architecture
c) Communications among organizations involved in the development, production, fielding, operation, and maintenance of a system
d) Communications between acquirers and developers as a part of contract negotiations
e) Criteria for certifying conformance of implementations to the architecture
f) Development and maintenance documentation, including material for reuse repositories and training materials

4.4 (Specific) Uses of ADs (2/2)

g) Input to subsequent system design and development activities

h) Input to system generation and analysis tools

i) Operational and infrastructure support; configuration management and repair; redesign and maintenance of systems, subsystems, and components

j) Planning and budget support

k) Preparation of acquisition documents (e.g., requests for proposal and statements of work)

l) Review, analysis, and evaluation of the system across the life cycle

m) Specification for a group of systems sharing a common set of features, (e.g., product lines)

5. Architectural Description Practices

5.1 Architectural documentation

5.2 Identification of stakeholders and concerns

5.3 Selection of architectural viewpoints

5.4 Architectural views

5.5 Consistency among architectural views

5.6 Architectural rationale

5.7 Example use (Informative)

Conformance

- **Conforming ADs include the following elements:**
  - a) AD identification, version, and overview information (5.1)
  - b) Identification of the system stakeholders and their concerns (5.2)
  - c) Viewpoints that have been selected and the rationale for the selections (5.3)
  - d) One or more architectural views (5.4)
  - e) All known inconsistencies (5.5)
  - f) A rationale for selection of the architecture (5.6)

☛ This conformance lacks **self checking mechanisms** that ensure a traceable and evolvable AD such as **design process** and **intermediate artifacts.**

5.1 Architectural documentation

- **An AD shall contain:**
  a) Date of issue and status
  b) Issuing organization
  c) Change history
  d) Summary
  e) Scope
  f) Context
  g) Glossary
  h) References

=> Selected to satisfy the requirements of IEEE/EIA Std 12207.0–1996 *"Software life cycle processes"*.

5.2 Identification of stakeholders and concerns

- An AD shall **identify the stakeholders** considered by the architect.

- **Stakeholders should include** :
  a) Users
  b) Acquirers
  c) Developers
  d) Maintainers

  of the system

5.2 Identification of stakeholders and concerns

- An AD shall **identify the concerns** considered by the architect.

- **Concerns should include** :
  - The purpose or missions of the system
  - The appropriateness of the system for use in fulfilling its missions
  - The feasibility of constructing the system
  - The risks of development and operation to users, acquirers, and developers
  - Maintainability, deployability, and evolvability

- The specific intent of the terms *mission*, *appropriateness*, and *feasibility* will vary from system to system.

5.3 Selection of architectural viewpoints (1/2)

- **An AD shall identify the viewpoints selected.**

- **Each viewpoint shall be specified by**
  a) A viewpoint name
  b) The stakeholders to be addressed by the viewpoint
  c) The concerns to be addressed by the viewpoint
  d) The language, modeling techniques, or analytical methods to be used in constructing a view based upon the viewpoint,
  e) The source, in the case of a library viewpoint, which is defined elsewhere (the source could include author, date, or reference to other documents, as determined by the using organization).

5.3 Selection of architectural viewpoints (2/2)

- **Additional information :**
  - Formal or informal consistency and completeness tests to be applied to the models making up a view
  - Evaluation/analysis techniques to be applied to the models
  - Heuristics, patterns, or other guidelines to assist in synthesis of a view
  - May be incorporated by reference (such as to a suitable recommended practice or previously defined practice).

- An AD shall include a rationale for each viewpoint.
  - Addresses how the stakeholders and concerns are covered by the selected viewpoints.

5.4 Architectural views

- An AD shall contain one or more architectural views.
- Each view shall correspond to exactly one viewpoint.
  - Each view in an AD shall conform to the specification of its corresponding viewpoint.
- Each view shall include:

  a) An identifier and other introductory information

  b) Representation of the system constructed with the languages, methods, and modeling or analytical techniques of the viewpoint

  c) Configuration information
- An architectural view may consist of one or more architectural models.

5.5 Consistency among architectural views

- AD
  - should contain an analysis of consistency across all of its architectural views.
  - shall record any known inconsistencies among its architectural views.

5.6 Architectural rationale

- AD
  - shall include the rationale for the architectural concepts selected.
  - should provide
    - evidence of the consideration of alternative architectural concepts and
    - the rationale for the choices made.
- When the AD describes a system that pre-exists the development of the AD,
  - the rationale for the legacy system architecture shall be presented, if known.

2. Documenting and Formalizing Architectures

David Garlan
Carnegie Mellon University

Acknowledgement:

Some of these slides have been taken from the tutorial on Documenting Software Architecture developed by the SEI for ICSE 2003

Seven Principles of Sound Documentation

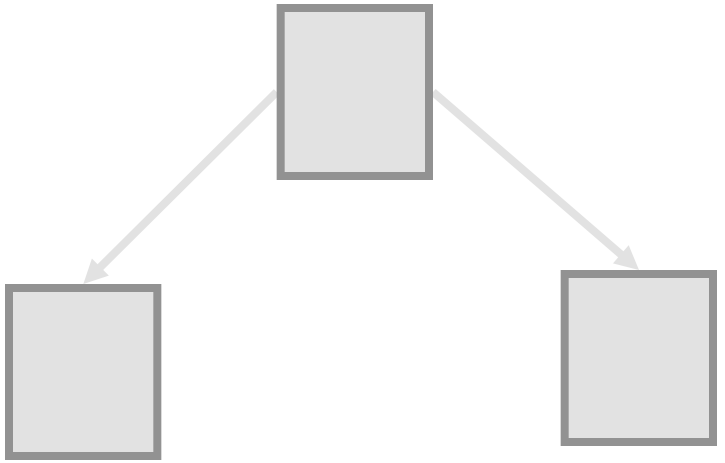
• **Certain principles apply to *all* documentation, not just documentation for software architectures.**

1. Write from the point of view of the reader
2. Avoid unnecessary repetition
3. **Avoid ambiguity**
4. Use a standard organization
5. Record rationale
6. Keep documentation current but not too current
7. Review documentation for fitness of purpose

3. Avoid ambiguity (1/2)

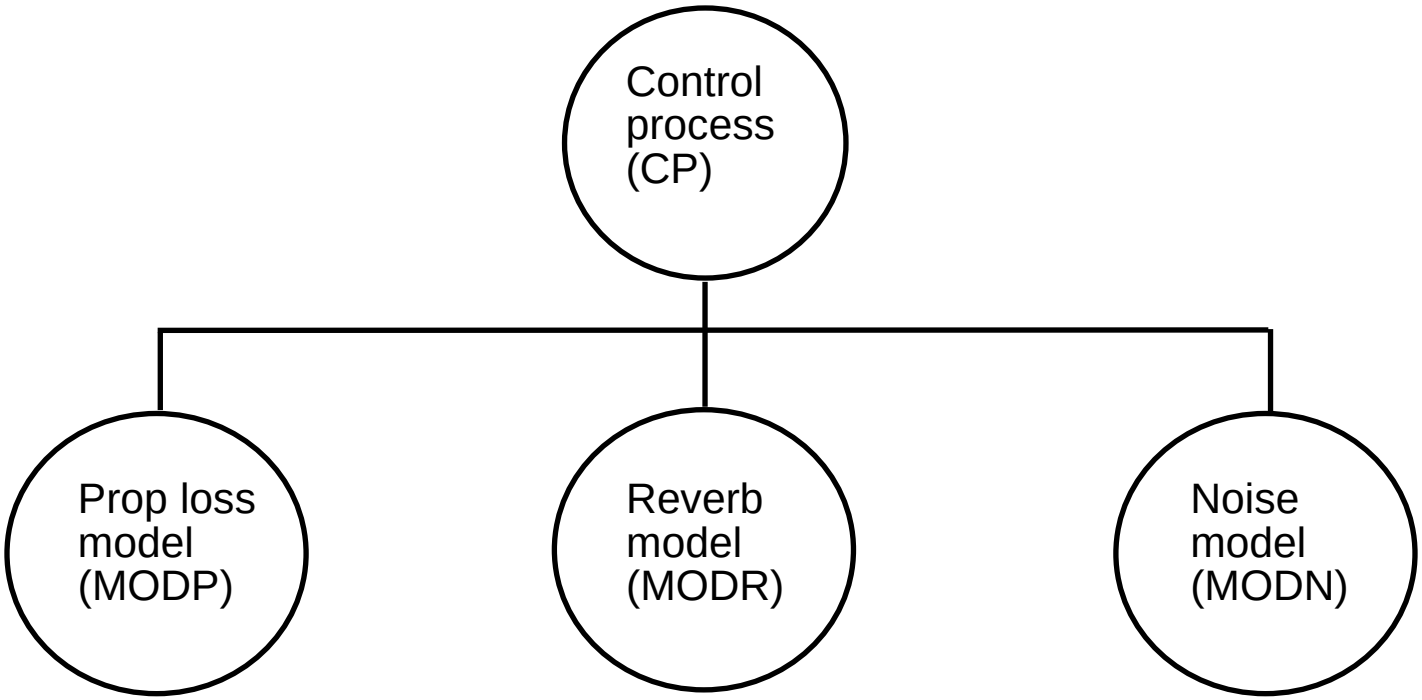
- **Documentation is for communicating information and ideas.**
  - If the reader misunderstands, the documentation has failed.
- **Precisely-defined notations/languages help avoid ambiguity.**
- **If your documentation uses a graphical language**
  - Always include a key/legend
  - Either point to the language's formal definition or give the meaning of each symbol.

3. Avoid ambiguity (2/2)



- Box-and-line diagrams are a common form of architectural notation.
- But what do they mean?
- These do not show an architecture, but only the beginning of one.
- If you use one, always define precisely what the boxes are and what the lines are.

A Software Architecture for a System (?)



What's Wrong with the Diagram?

- Many things are left unspecified:
  - What kind of components?
  - What kind of connectors?
  - What do the boxes and arrows mean?
  - What is the significance of the layout?
  - Why is control process on a higher level?

- Box and arrow drawings alone are not architectures; rather, they are a starting point.

Essential Elements of Good Architectural Documentation

- **Statement of requirements**
  - business context, rationale for product, domain
- **Description of context**
  - Systems with which it must interact, external interfaces
- **Use of architectural diagrams**
  - with suitable discrimination of boxes and lines
  - explanatory prose
  - include relevant views (see reading by Clements et al.)
- **Consideration of implementation-level constraints**
  - but only insofar as they impact architectural design
    - e.g., need to use given infrastructure, processor requirements
  - will often involve other structural views
- **Rationale for the architectural design**
  - how it addresses requirements & implementation constraints
  - alternatives

Other Aspects

- **Style/product-line issues**
  - what are the expected dimensions of variability?
  - what aspects of the architecture must/should not change?
- **Management issues**
  - implications on organizational structure of development team
  - use of architectural reviews
- **Other Design issues**
  - process structure
  - code reuse implications, use of standards
  - risk analysis
  - OA&M: operations, administration, and maintenance

Indicators of Good Quality

- **Lines and boxes have different shapes/colors and a key to explain what these mean**
- **Tables are used to summarize dimensions/choices**
- **Diagrams do not try to do too much**
  - each view of architecture fits on a page
  - separation into views, where necessary
  - use of hierarchy
- **Clear distinction between viewtypes**
  - separation of concerns
  - but with mappings, where appropriate

Indicators of Bad Quality

- Lines all look the same
- Arrows don’t mean anything
- Arrows mean many things
- Confuse implementation with architectural abstraction
- No key or legend
- Too many driving requirements

3. 아키텍처 설계 문서 템플릿

아키텍처 설계 문서 목차

아키텍처 설계 문서 제목

A. 시스템 개요
B. 시스템 요구사항
  B.1 기능요구사항
  B.2 품질요구사항
  B.3 제약사항
  [B.4 요구사항 분석모델]
C. 아키텍처 문제분석
  C.1 선정된 아키텍처 드라이버
  C.2 아키텍처 문제 분석
  C.3 아키텍처 문제 분석표
D. 설계절차
  [D.1 아키텍처 스타일]
  D.2 아키텍처 관점체계
  D.3 아키텍처 설계절차의 정의
E. 아키텍처 설계결과
F. 설계결과의 분석
  [F.1 아키텍처 추가 문제 분석표]
  F.2 아키텍처 설계 결정표
  F.3 아키텍처 분석 관점 및 분석 결과
G. 아키텍처 평가
  G.1 평가방법
  G.2 아키텍처 평가 결과
  G.3 대안 아키텍처 및 평가
H. 토의 사항

- "[ ]"은 필요 시 기술
- **H** 는 위의 항목들에서 다루어지지 않은 중요한 사항들에 대하여 논의

아키텍처 설계 원리와 아키텍처 문서 목차의 매핑

아키텍처 설계 원리들	아키텍처 문서의 적용 항목
ADP 1. 아키텍처 드라이버	B.2 B.3
ADP 2. 품질속성, 검증가능성, 품질속성시나리오	B.2
ADP 3. 아키텍처 문제 분석	C
ADP 4. 컴포넌트와 커넥터의 이분법	E의 전체에 걸쳐 나타남.
ADP 5. 아키텍처 스타일	D.1
ADP 6. 소프트웨어 아키텍처 관점체계	D.2
ADP 7. 설계의 일반원리	D.3, E 전체에 걸쳐 나타날 수 있음.
ADP 8. 아키텍처 설계 절차	D.3
ADP 9. 아키텍처 패턴	E 전체에 걸쳐 나타날 수 있음. F.1 에 적용이 나타나야 함.
ADP 10. 품질속성 설계전술	E 전체에 걸쳐 나타날 수 있음. F.1에 적용이 나타나야 함.
ADP 11. 아키텍처의 분석	F
ADP 12. 아키텍처의 평가	G

I hear and I forget.

I see and I remember.

I do and I understand.

Questions?