Software Design: An Introduction

– David Budgen

소프트웨어 모델링
200511346
이창현

contents

[1]The Role of Software Design

질 문

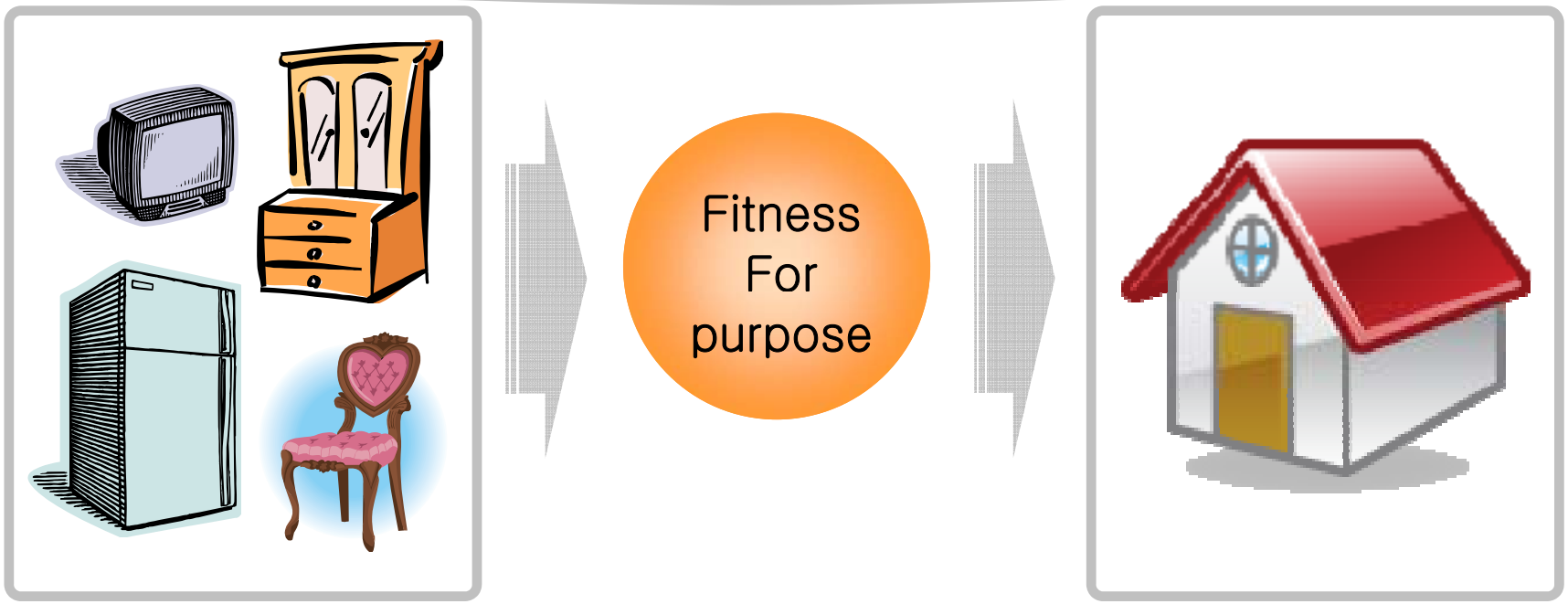
"디자인을 하는 목적이 무엇 인가?"

답 변

"주어진 문제에 맞는
수행 가능한 해결책을 생산해내기 위해서"

[1]The Role of Software Design

여러 Design을 비교 하고 현재 목적에
"가장 적절한 Design"을 하는 것이다.



[1]The Role of Software Design

1.1 The software design process

디자인의 모양과 과정은 이해하기 어렵고 까다로운 측면이 있다.

why?

Software의 특징

1. Complexity (복잡성)
2. Problem of conformity (문제의 유사성)
3. Ease of changeability (변하기 쉬운)
4. Invisibility (눈에 보이지 않는)

[1]The Role of Software Design

1.1 The software design process

Procedural manner

1. Representation part
2. Process part
3. Set of heuristics

Software Design을 제한하는 요소

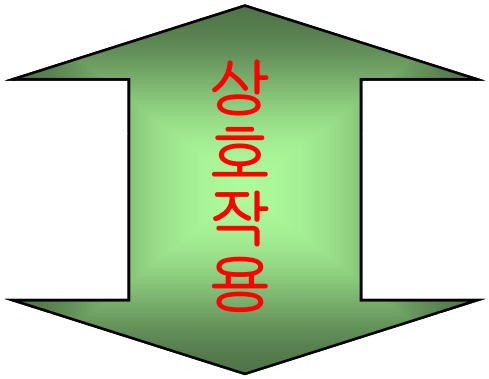
1. Programming Language to be Used
2. Execution Environment or Operating System
3. Performance Expectations
4. User Interface Needs

[1]The Role of Software Design

1.2 Design in the software development cycle

Analisys activity

사용자의 요구에 대처할 해결책의 "형식"을 제공



Software designer

형식에 따른 "해결책"을 제공

Design은 software 개발

형식에 영향을 미치는 요소로 작용

Designer need "think ahead"

Designer는 해결책을 제시하고
계획하는 것에 있어서 신중하고
앞선 생각과 자세가 필요하다.

[1]The Role of Software Design

1.3 Design qualities

Fitness for purpose가 design quality를

결정 하는 척도가 되지는 못한다.

Design quality 요소

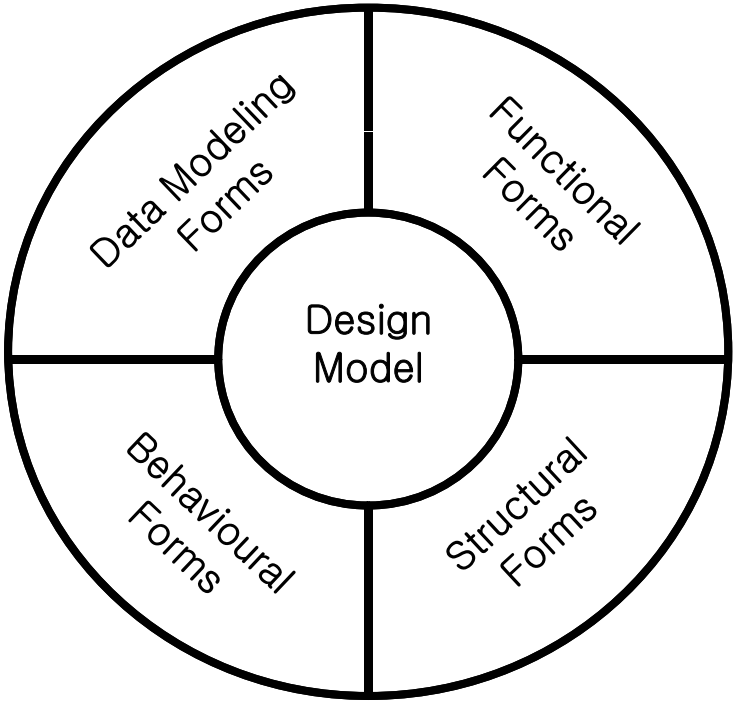
1. Reliability(신뢰성)
2. Efficiency(능률성)
3. Maintainability(유지성)
4. Usability(유용성)

[2]Describing Designs

2.1 Recording the design model : design viewpoints

Design Viewpoints(관점)

1. Behavioural
   - 외부 사건과 시스템 활동 사이의 연결
2. Functional
   - system이 무엇을 하는가?
3. Structural
   - 구조상 요소간의 상호 의존
4. Data modelling
   - data objects 사이에 존재하는 관계



Design viewpoints protected from the design model

[2]Describing Designs

2.2 Design representation forms

1. Textual descriptions

Headings, lists, indentation 등을 사용해 만든 form

- Limitations

1. List와 table위에 적절하게 위치 하지 못할 경우 명확하게 표현하기 어렵다.
2. Natural language는 애매한 경향이 있다.

=> 복잡하고 길어질 경우 사용을 피하는 것이 좋다.

2. Diagrammatical descriptions

도형들을 그려 도식화 하여 만든 form

- Property

1. A small number of symbols (circles, lines(arcs), boxes)
2. A hierarchical structure(계층적 구조)
3. Simple symbols => 의사소통을 쉽게 해준다

[2]Describing Designs

2.2 Design representation forms

3. Mathematical descriptions

- Mathematical하게 표현한 form
- 추상적인 생각을 간결하게 묘사하기에 매우 적합하다.

디자이너의 추상적 생각을 항상 묘사할 수 있는 것은 아니다.

[2]Describing Designs

2.3 Some example of design representation

The Statechart

객체의 상태와 상태를 변화시키는 이벤트를 표현한 것.

(상태에서 상태로의 통제 흐름)

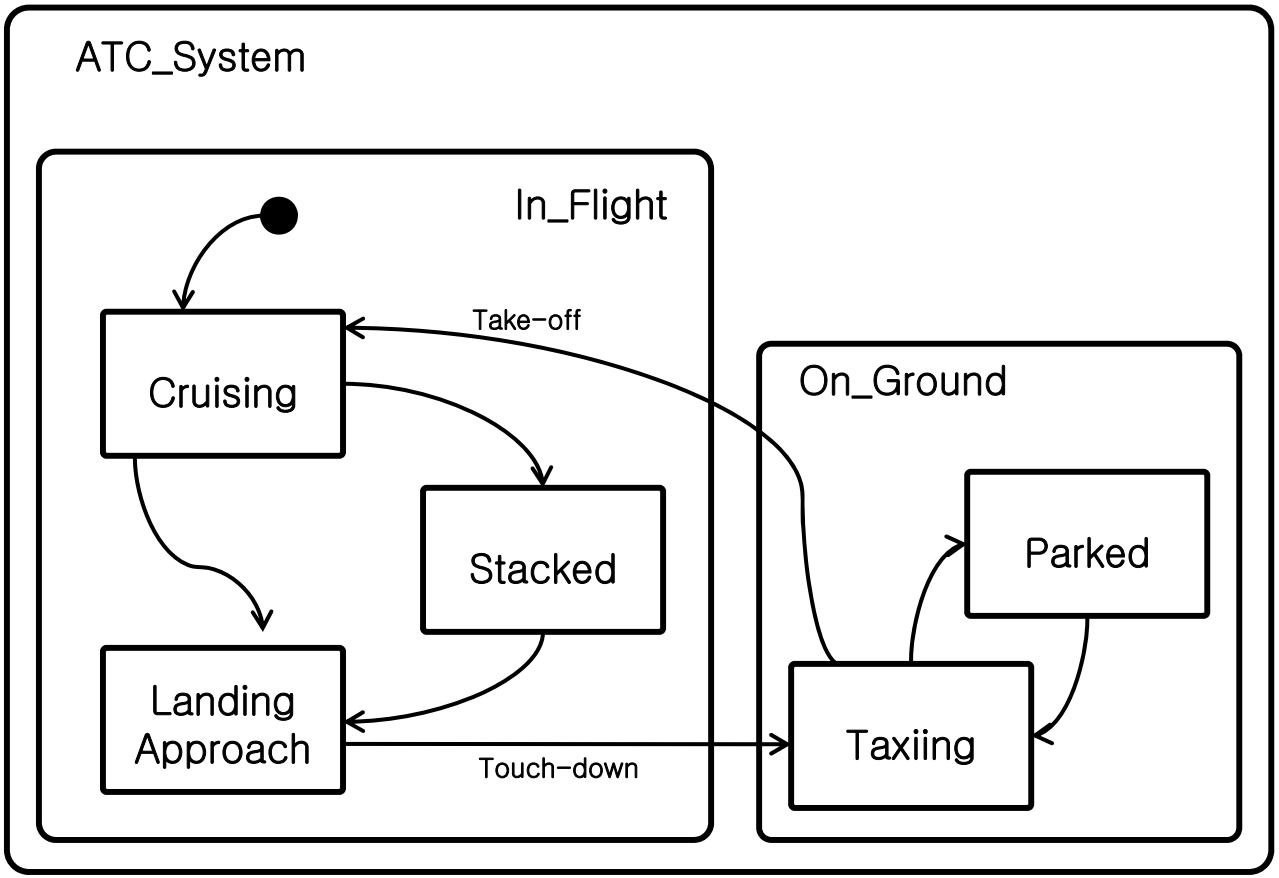
- 특 징

  1. 객체에 대해서 생성에서 소멸까지의 과정을 분석하기가 용이하다.
  2. 모서리가 둥근 사각형으로 표현 =>

[2] Describing Designs

2.3 Some example of design representation

An example Statechart



상태에서 상태로의 흐름!!

[2]Describing Designs

2.3 Some example of design representation

The Jackson Structure Diagram

1. 순차, 선택, 반복에 관한 데이터 구조를 설명

선택 => 

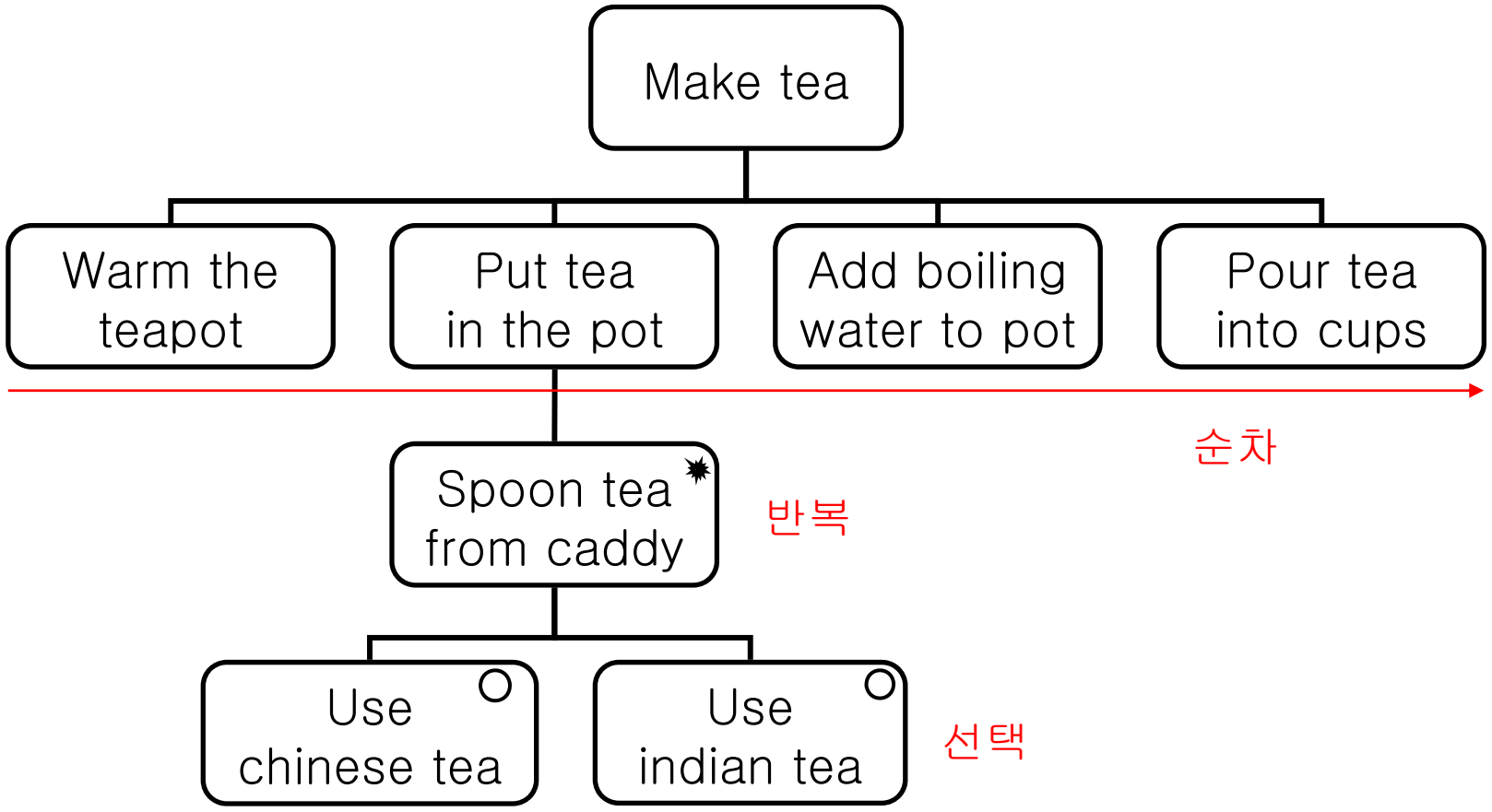
반복 => 

3. Data structure modelling, 시간 별로 지시된 행동을 묘사하는데 유용

[2] Describing Designs

2.3 Some example of design representation

An example of a Jackson Structure Diagram



[2]Describing Designs

2.3 Some example of design representation

The Structure Chart

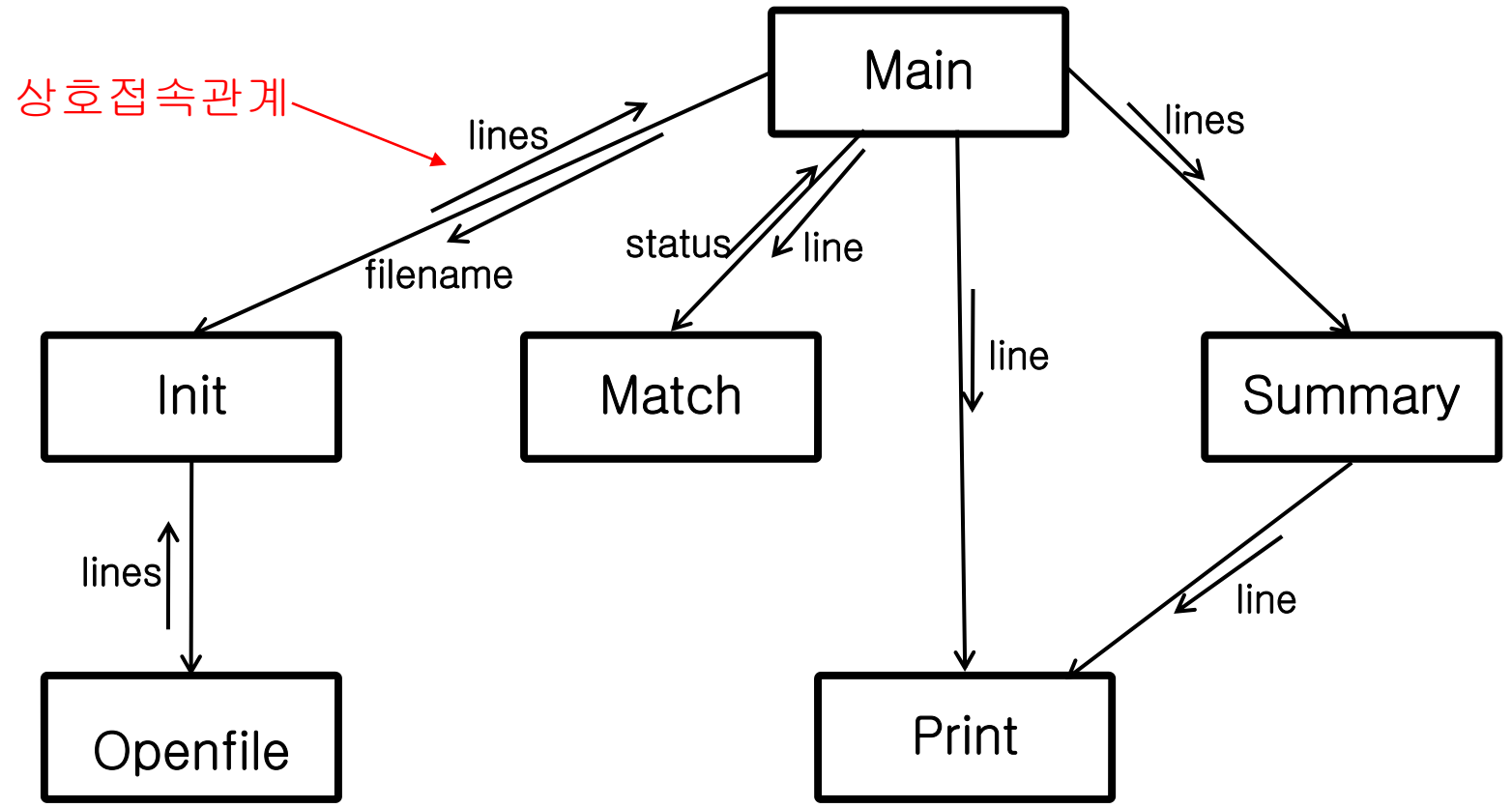
- 프로그램의 호출 구조(상호 접속 관계)를 설명

- Jackson Structure Diagram과 유사해 보이나 다르다!!
  - Elelments represent physical entities(subprogram)
  - Hierarchy(계층) shown is one of invocation(transfer of control)

[2] Describing Designs

2.3 Some example of design representation

A simple Structure Chart



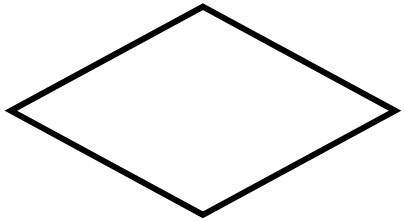
[2]Describing Designs

2.3 Some example of design representation

The Entity-Relationship Diagram

- Entity와 Entity사이의 관계를 표현

=> 주로 데이터 베이스를 분석 하는데 유용



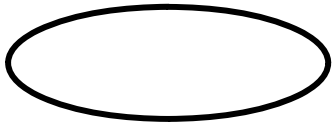
Relationship:

One-to-one

One-to-many

Many-to-many

Entity

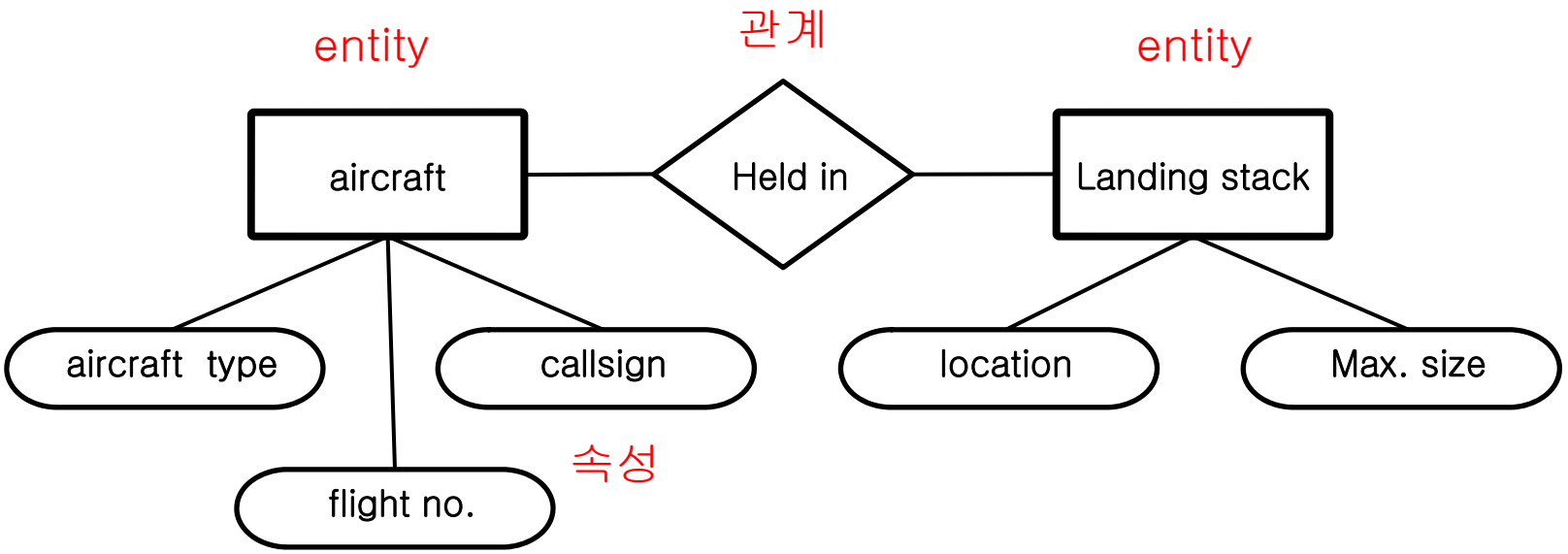


Attribute(속성)

[2] Describing Designs

2.3 Some example of design representation

A simple Entity-Relationship Diagram



[3]Software Design Practices and Design Methods

3.1 Rationale for software design methods

software design methods 역할

1. 일반적인 목표와 스타일을 정립하다.
2. 일관성을 부여
3. 명백한 문제점에 대해 형태를 만드는 것을 돕는다.

- 디자인 정보를 제공하는 가장 실용적인 수단

Methods의 유용함을 제한하는 요소

1. Methods에는 문제가 어떻게 풀어져야 하는지에 대해서 상세하게 규정되어져 있지 않다.
2. Methods가 desiner가 찾아낸 해결책과 대립될 수 있다.

[3]Software Design Practices and Design Methods

3.2 Design strategies

Top-down

separating a large problem into smaller ones

단점 : 중요한 구조상의 결정이 초기에 이루어 져야 한다.

Compositionnal

Involve “entities” that can be assembled to create a solution model

Organisational

Where the needs of the development organization and its management structures impose constraints upon the design process.

Template

Where some general paradigm describes a reasonably large domain of problems.

[4]Features of Some Software Design Methods

4.1 Jackson Structured Programming(JSP)

JSP

- 자료구조를 프로그램 구조에 맞추는 것에 중점을 둠
- is prvided by the Jackson Structure Diagram.

  => 구조의 설계를 잘못하면 적절하지 못한 프로그램이 된다.
- is used for modelling data structures.
- is used for functional modelling.

process part of JSP

Step1	Draw Structure Diagrams for inputs and output	Elaboration
Step2	Merge these to create the program Structure Diagram	Transformation
Step3	List the operations and allocate to program elements	Elaboration
Step4	Convert program to text	Elaboration
Step5	Add conditions	Elaboration

[4]Features of Some Software Design Methods

4.2 Structured System Analysis(SSA) and Structured Design(SD)

SSA / SD

- software design의 top-down 기법
- consist of “analysis” and “design”
- 기본 방식은 자료흐름도(DFD : data-Flow diagram)를 구조도(structure chart)로 체계 있게 변화시키는 것

● 단점 : has a large and relatively disjoint transformation step.

Process part of SSA and SD

Analysis	Step1	Develop a top-level description	Elaboration
	Step2	Develop a model of the problem (SSA)	Elaboration
design	Step3	Subdivide into DFDs describing transactions	Elaboration
	Step4	Transform into Structure Charts	Transformation
	Step5	Refine and recombine into system description	Elaboration

[4]Features of Some Software Design Methods

4.3 Jackson System Development(JSD)

JSD

- encourages the designer to create a design model around the notion of modelling the behaviour of active “entities”.
- these entities gradually will be elements of the solution.

JSD process part

1. Entity Analysis	Identify and model problem entities	Elaboration
2. Initinal Model Phase	Complete the problem model network	Elaboration
3. Interactive Function Step	Add new solution entities	Elaboration
4. Information Function Step	Add new solution entities	Elaboration
5. System Timing Step	Resolve synchronisation issues	Elaboration
6. Implementation	Physical design mappings	Elaboration

[4]Features of Some Software Design Methods

4.4 Object-Oriented Design

OOD

- 객체 지향적 설계
- concerned with developing an object-oriented system model to implement requirements.

장 점

1. Easier maintenance
2. Objects are potentially reusable components
3. For some systems, there may be an obvious mapping from real world entities to system objects.