Object-oriented system development

Survey of structured methods

6조 박진조

기본 개념

- 소프트웨어 공학
  - 요구하는 목표를 달성하기 위한 소프트웨어 개발 시 소프트웨어 분석, 설계, 구현, 및 테스트 등에 방법론, 도구, 및 기술들을 체계적으로 적용 하는 것
- 소프트웨어 개발 방법론
  - 소프트웨어를 생산하기 위해 반복적으로 수행될 실행방법을 정리한 것
  - 무 방법론 -> 방법론

기본 개념

- 구조적 개발 방법론
- 정보공학 방법론
- 객체 지향 개발 방법론

논문의 목적

- 객체 지향 개념에 대한 이해
- 객체 지향 방법론과 전통적 방법론의 비교를 통한 올바른 개념 이해
- 객체 지향 방법 소개
- 전통적 방법 소개
- 전통적인 개발방식에서 객체지향 방식으로의 전환의 필요
- 객체지향 개발방식에 대한 관심 증대

객체지향

- 객체 지향 개발은 소프트웨어 디자인의 신뢰성, 유지보수성, 개발과정을 좀더 효율적이게 하는 재 사용성을 향상시킨다.
- 위의 내용을 정당화 할 수 있는 개념

추상화	캡슐화	상속
-객체란 실세계의 일부를 추상화 한것이다. -객체지향 개론은 실세계 안의 구조들을 모델링하는것이다. -구조와, 행위의 통합된 유닛을 모델화한다.	-object가 자신의 내부를 사용자로 하여금 숨김 -유지/보수성의 향상	-부모클래스의 속성을 자식이 상속 -좀더 일반적이고 하이레벨의 object는 재사용이 가능해진다 -여러부모에게 상속받음으로써 생기는 다형성

Evaluation of modelling components

- Method간 비교를 위해 OO개념의 기본원리를 구조화된 방법의 전통적 모델과 비교의 필요
- Object의 정의와 전통적 개념(entities/function)과의 차이

	Traditional	OO approach
공통점	Entity - instances / type / class	동일
	Event - 무언가가 발생하는 것	Message
차이점	Rule - control을 명시한 rule과 entities를 분리하여 생각	Data + Activity

Object의 다양한 특성

- Object의 분류 : activity의 많고 적음에따라
  - Data-oriented object
  - Task-oriented object : 수학적 계산 등에 사용
- Booch에 의한 분류
  - Object : actors, agents, server 로 분류
    - Actors : action을 수행하는 주체 (=tasks, procedures)
    - Server : actors의 행동을 받는 대상 (=data base)
    - Agents : 두 특성의 결합
  - 반영의 예시
    - Real-time systems have more actors
    - Data retrieval systems have more servers

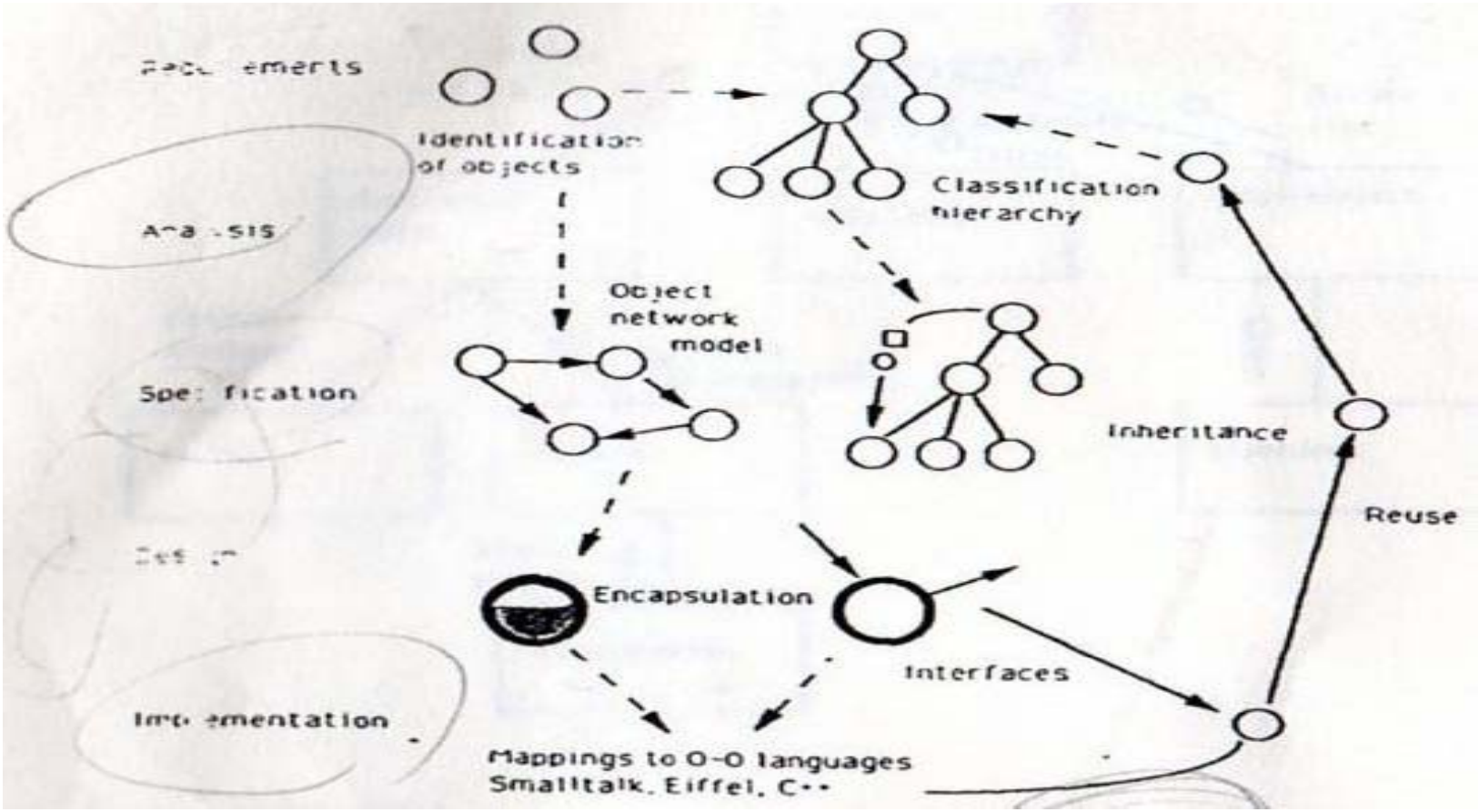
Evaluation procedure

- OO 개발의 meta-model
  - OO method가 OO개념에 잘 부합하는지를 평가하기 위한 틀
  - 평가에 사용되는 4가지 측면
    - Conceptual modelling
      - Method는 OO기준에 부합되며 모델링 하는 것을 포함하여야 한다.
    - Procedual guidance
      - Method는 분석가들에게 분석, 명세, 디자인을 행하는 방법을 말해주는 명확한 단계를 가져야 한다.

Evaluation procedure

- Transformation
  - Method는 specification이 변하는 것에대한 알고리즘을 design에 주어야한다.
- Design products
  - Design specification과 design의 결과는 명확히 code 로 묘사될 수 있어야 한다.

Object-oriented meta-model



Analysis of OO method

Table 1. Feature analysis of object-oriented methods

Method	Abstraction	Classification	Inheritance	Encapsulation	Coverage (R-A-S-D-I)
HOOD	Y	Y	Partial	Y	------
OOSD	Y	Y	Y	Y	------
OOSA	Y	Partial	-	-	-----
OOA	Y	Y	Y	-	------
ObjectOry	Y	Y	Y	Partial	-------

Key: Y = Yes.

R-A-S-D-I in coverage refers to Requirements Analysis. Analysis, Specification, Design, and Implementation. The measure of coverage is judged from the methods procedures and notations.

Object-oriented Method

- HOOD
  - 포괄적인 디자인 method, 분석단계의 부재
  - Object내의 복잡한 데이터구조의 명세를 필요로 하지 않는 real-time orientation이라 상속개념을 완벽히 지원하지 않는다.
- OOSD
  - 포괄적인 디자인 method, 부석단계의 부재
  - Interface 묘사, 캡슐화 등의 세밀한 표기가 제공된다
  - 표기가 많아져 읽기 어렵다

Object-oriented Method

- OOSA
  - Shaler and mellor's method
  - 초기의 추상화, object modelling에 도움
  - Entity – Relationship modilling과 유사
  - Activity가 data model에 합쳐지는 개념이라기보다 dataflow diagram(function)과 state transition(entity)이 합쳐진것이다.

OOSA

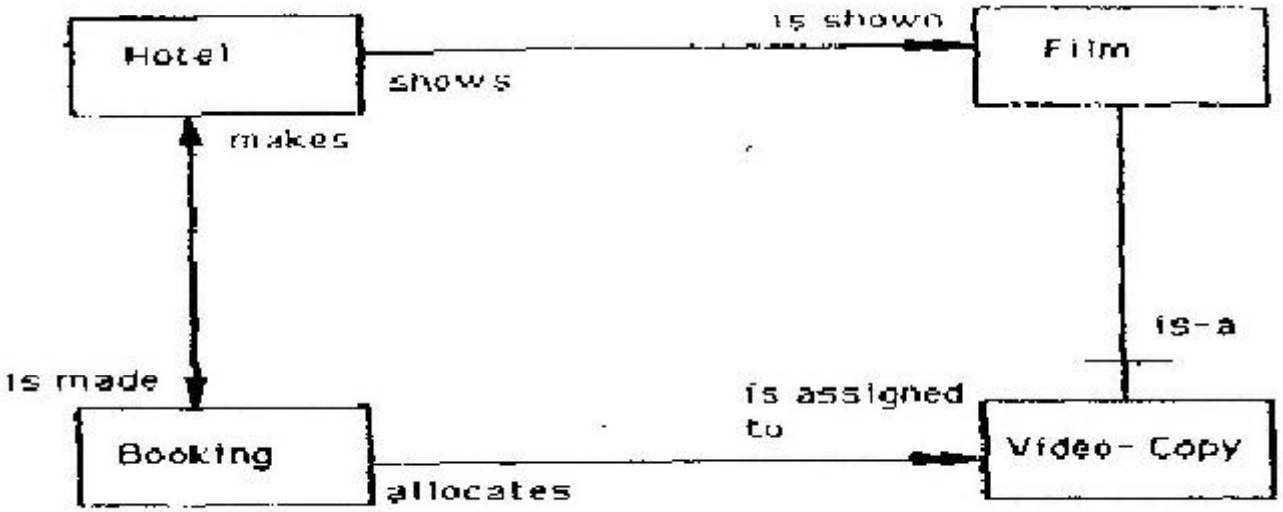


Figure 3. Object model of VI application produced by OOSA

Object-oriented Method

- OOA
  - Coad and Yourdon method
  - 분석 메소드
  - 추상화에 structure layer의 도움을 받는다
    - Subject, structure, attribute, service
- 즉, 완벽한 OO method는 존재하지 않는다

Traditional Method

Table 2. Summary of method specification models and approaches

Method	Functional process	Data relationship	Event sequence	Coverage (R-A-S-D-I)	Application
IE	Y	Y	Y	----------	IS
ISAC	Y	Y	N	------	IS
SASD	Y	N	Y	---------	IS
SSADM	Y	Y	Y	----------	IS
SADT	Y	Y	N	------	IS. RT
JSD	N	Y	Y	----------	IS. RT
NIAM	Y	Y	N	---------	IS (data intensive)
Mascot	Y	N	N	---------	RT

Key: Y = Yes, N = No.
Coverage of the life-cycle: Requirements (R), Analysis (A) Specification (S), Design (D), Implementation (I).
Application: IS = information systems, RT = real-time.

Traditional Method

- SASD
  - 기능적 분해이용
  - 데이터의 흐름을 지향하는 설계
  - 프로세스 목적에 관계없이 현 세계의 모델에 관계하는 모듈에 관심을 갖는 OO관점과는 다르게 이 method는 목적 관계된 분석을 하게된다.

Traditional Method

- JSD
  - 동시에 발생하는 프로세스의 네트워크 에 기초
  - 시스템 제어가 entities와 관련된 actions의 시간 순서의 관점에서 모델링된다
  - 데이터 분석에 중점을 두며, 그로 인해 data와 operations을 결합한 object model과 비슷하게 되었다.
  - 많은 면에서 OO method와 비슷하다
  - 분류와, 상속은 지원하지 않음

Traditional Method

- IE
  - 정보기술의 발전에 따른 새로운 방법론 필요
  - 정보시스템을 위해 데이터 모델링과 관련된 구조화된 method
  - OO개념과 많이 비슷하다
  - Object의 정적인 측면의 명세중심
  - Dynamic system components는 무시
  - 기능적 분할 방식

방법론의 정리

구조적 방법론	정보공학 방법론	객체지향 방법론
데이터 흐름에 따른 프로세스 위주의 분석과 설계 방식	정보기술발전에의해필요성도래/데이터 중심의 분석과 설계 방식	소프트웨어의 분석과 설계에 객체지향의 원칙을 적용
하향식,모듈화, 분할과 정복, 폭포수	하향식,모듈화, 분할과 정복, 폭포수	프로세스와 데이터의 통합하여 처리
분할 해결 방식과 단순한 모형(Model)의 사용 개발자와 사용자 모두 학습이 용이	대규모 정보시스템의 개발에 가장 적합한 방법론	완성시스템에 대한 요구사항의 신속한 확인결과 도출 /재사용성의 강조/ 쉽고 표준화된 표기법
단순한 모형의 사용으로 인해 개발단계 사이에 모형의 완전성 및 일관성을 유지시키기 어려움 재사용 불가능으로 인한 생산성 감소	경직되고 복잡한 구조	세분화된 객체 모델 (사용의 어려움) 재사용, 생산성 향상이 기대에 못 미침 객체 정의의 어려움
SASD	IE, SSADM	HOOD, OOSD, OOA등

결론

- 지금까지 구조적 시스템 개발방법을 알아보았고, 그것이 OO개발 방식과 얼마나 유사하며, 또 얼마나 차이가 있는지도 알아보았다
- 즉, 위로 미루어보아 structured method로부터 OO method로의 이동은 실현 가능하다
- OO개념에 좀더 많은 관심을 두어야 한다
- OO개론의 지지자들은 상업적 범위의 application내에서 평가함으로 OO의 타당성을 증명해야한다.


소프트웨어 엔지니어링의 비극은
소프트웨어 프로젝트를 어떻게 계획하고
수행하는지를 몰라서가 아니고
잘 알면서도, 이를 실행하지 않기 때문에
생긴다

Richard E. Fairley

The End