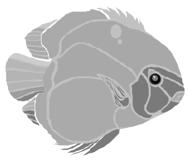
Object-Oriented Analysis

Object-Oriented Analysis Techniques
Coad's OOA Technique
Short History
Terminological Comparison
Postscript and Remarks

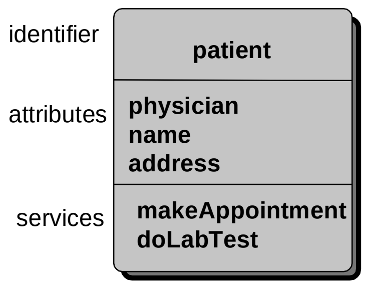
Object-Oriented Analysis

- Object-Oriented Analysis covers a host of techniques introduced in the mid-eighties for requirements modelling.
- Such techniques focus on the **things** that exist within the application domain, model them with **objects**
- These techniques use **classification**, **generalization**, **aggregation** to structure object assemblies
- **Actions** (services/activities) are associated with objects
- **State changes** are effected by actions performed on objects



Coad’s Object-Oriented Analysis

- Proposed by Peter Coad [Coad91].
- An **object** is defined as a real world entity related to the problem domain, with crisply defined boundaries.
- Objects are encapsulated along with their attributes and behaviour.
- In Coad’s model, there are five kinds of concepts for modelling an application: **objects**, **attributes**, **structures**, **services** and **subjects**.



Five Layers to OOA

- **Class/Object Layer**
- **Structure Layer**
- **Service Layer**
- **Attribute Layer**
- **Subject Layer**

■ Phase I: Identifying Problems & Opportunities

- **Confirm that a problem exists**
- **Carry out a study to determine if a system can be developed to solve the problem (2 days - 4 weeks)**
- **Produce the Feasibility Study**

Phase II: Info Requirements Analysis

- **Study existing procedures and information systems**
- **Define goals to be achieved by the new system**
- **Propose alternate business processes**
- **Define the boundaries of the information system**
- **Define non-functional requirements**

■ Phase III: Systems Analysis

- Chart input, processes, & outputs using Data Flow Diagrams
- Develop data dictionary listing all attributes of the system
- Analyse structured decisions using decision tables or decision trees
- Analyse semi-structured decisions. May need to develop decision support systems.
- Prepare System Proposal, Cost/Benefit Analysis of alternatives
- Recommend course of action

■ Phase IV: The Design Phase

- Specify an architecture and a detailed design for the proposed information system
- Ideal system specified first, meeting all functional requirements, then modified to meet non-functional requirements and other constraints
- Resources allocated for hardware equipment, personnel tasks and programming tasks
- Technical specifications are prepared

■ Phase V: Develop and Document

- The system is implemented on the basis of the design specification.
- Analyst works with programmers to develop any original software needed
- Analyst uses structured techniques (pseudo code, flowcharts,…) to communicate with the programmers
- Programming of the system is carried out
- Program testing is carried out
- Procedures, system manuals, software specifications and documentation are completed

■ Phase VI: System Testing

- Testing of the system as a whole is performed
- Users conduct acceptance testing

■ Phase VII: Implementation & Evaluation

- Equipment is acquired and installed
- Staff is trained
- Conversion from old system
- Evaluation of system

■ The Ongoing Maintenance Phase

- **Over 60% of a system’s resource can be spent in this phase**
- **Two flavours of maintenance:**
  - correction of software errors
  - enhancements to meet changing needs
- **Important that all previous phases keep easy maintenance as major goal.**

Systems Lifecycle cq. OOAD

- I. Identifying Problems & Opportunities
- II. Info Requirements Analysis
- III. Systems Analysis
- IV. Systems Design
- V. Development & Documentation
- VI. Systems Testing
- VII. Implementation & Evaluation
- On-going Maintenance Phase

OOAD only affects phase III, IV, and V.

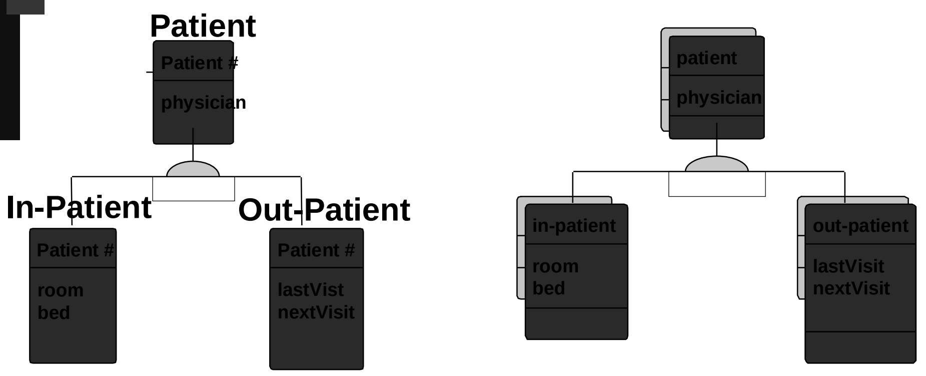
The other phases remain the same.

Five Layers to OOA

- Class/Object Layer
- Structure Layer
- Service Layer
- Attribute Layer
- Subject Layer

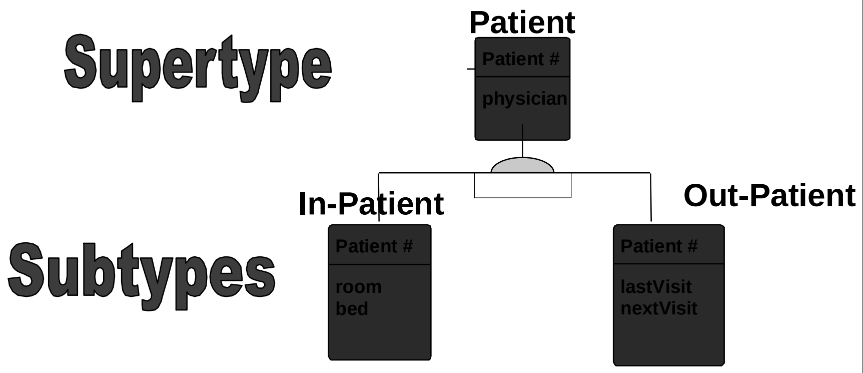
Structure: Basic Class Notation

ER Notation *OOA Notation*



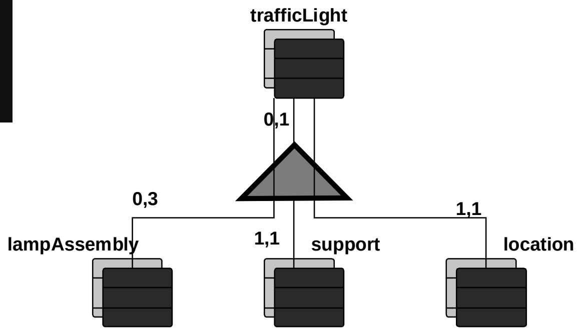
Structure: Subtyping or Generalization-Specialization

Generalization/specialization structures organize objects into taxonomies. **Patients** are either **in-patients** or **out-patients**. The **physician** attribute of **patients** is inherited by both **in-patients** and **out-patients**.



Structure: Whole-Part Structures

Whole-Part structures describe an object as an assembly of other objects. A **traffic light** consists of 0 to 3 **lampAssemblies**, a single **support** and a single **location**.

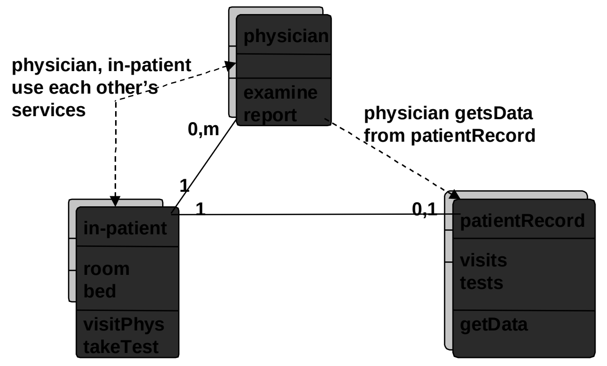


Services

- Objects provide **services** to other objects in their environments. For example, a physician object may provide services **examine**, **report.**
- Coad distinguished three types of services:
  - **Occurence services,** whereby objects are created, destroyed, changed,...; Coad suggests using a generic service **occur** and not mention it for any particular object;
  - **Calculate services**, where an object performs a calculation for some other object;
  - **Monitor services**, where an object is monitoring some process to see if some condition applies;
- A special notation is used (dashed-line arrow) to indicate that an object is using services from another object.

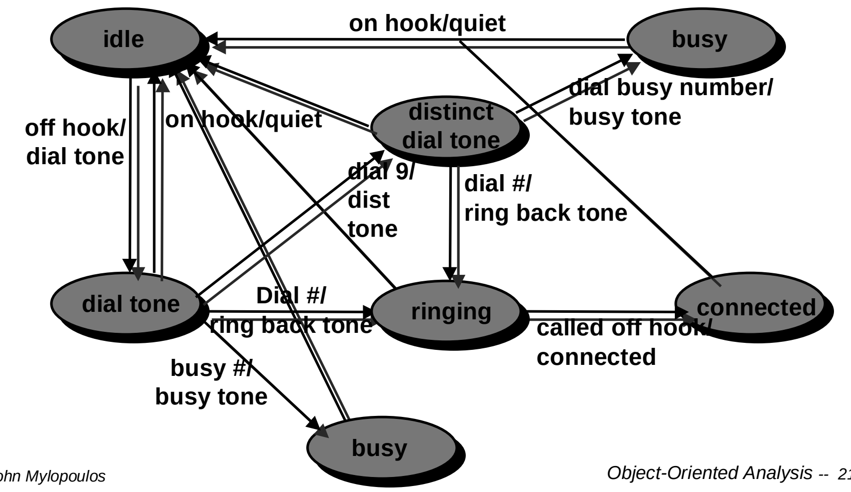
OOA views the world with Smalltalk glasses...

Services and Relationships



State Transition Diagrams

Legend:
input/output



Methodology

- Identify objects and **classes** (i.e., generic objects)
- Identify structures and build generalization, aggregation hierarchies.
- Define subjects. These partition all the objects and classes of an object model into subject layers, which represent the application from a particular perspective. Often whole Gen-Spec or Part-Whole structures are grouped under one subject.
- Identify information that should be associated with each object. Place attributes at the right structural level.
- Define services for each class.

Terminology

OOA	**OOSE**	**OOD**	**OMT**
	(Jacobson)	**(Booch)**	**(Rumbaugh)**
		Metaclass	
Object	Instance	Object	Object
Gen-Spec	Inheritance	inherits	Generalization
Whole-Part	Consists-of		Aggregation
Instance conn.	Acquaintance		Link
Message	Stimuli	Message	Event
Message conn.	Communication		
Attribute	Attribute		Attribute
Service	Operation		Operation
Subject	~View (subsystem)		Sheet
(Execution thread)	Use case		~Scenario
(User)	Actor		

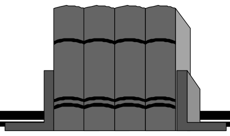
The Unified Modeling Language

- Booch and Rumbaugh started working towards a unified modeling language (UML) in 1994 under the auspices of Rational Inc.
- UML only offers a notation, not a methodology for modeling (as various OOA techniques do).
- UML will be proposed by Rational Inc. and by Hewlett-Packard as a standard for object-oriented analysis and design, to be adopted by the OMG.
- If adopted by OMG, it is expected that all vendors will modify their CASE tools to make them consistent with UML [UML97].

Remarks

- Object-oriented analysis techniques are supposed to make it easier to generate a design and subsequently code from a requirements specification.
- The introduction of semantic structuring mechanisms (generalization, aggregation) to requirements modeling is definitely a step in the right direction.
- Trouble is, in the process, OOA techniques straightjacket the modeller’s view of the world (...the guy with the hammer sees the world like a bunch of nails...)
- OOAD modellers often forget first few phases.
- OOAD doesn’t lend itself well to drastically different architectures

References



- [Booch86] Booch, G., “Object-Oriented Development”, *IEEE Transactions on Software Engineering 12(2)*, February 1986.
- [Booch94] Booch, G., *Object-Oriented Analysis and Design,* Benjamin-Cummings, 1994 (2nd edition).
- [Coad91] Coad, P. and Yourdon, E., *Object-Oriented Analysis*, Prentice Hall, 1991.
- [Jacobson92] Jacobson, I., Christerson, M., Jonsson, P., Overgaard, G., *Object-Oriented Software Engineering*, Addison-Wesley, 1992.
- [Martin93] Martin, J., *Object-Oriented Analysis and Design*, Prentice-Hall, 1993.
- [Rumbaugh91] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., Lorensen, W., *Object-Oriented Modeling and Design,* Prentice-Hall, 1991.
- [UML97] http://www.rational.com