

# On the Purpose of Object-Oriented Analysis

The What and How of *what*

**Geir Høydalsvik & Guttorm Sindre.**

**The Norwegian Institute of Technology**

# What is this about ?

A critical look at:

- The relationship between OOA and OOD, and the pre-requisites for a smooth transition to design.
- The relationship between Analysis, Conceptual Modeling, and Requirements Engineering.
- Suitability: The role of objects in analysis.

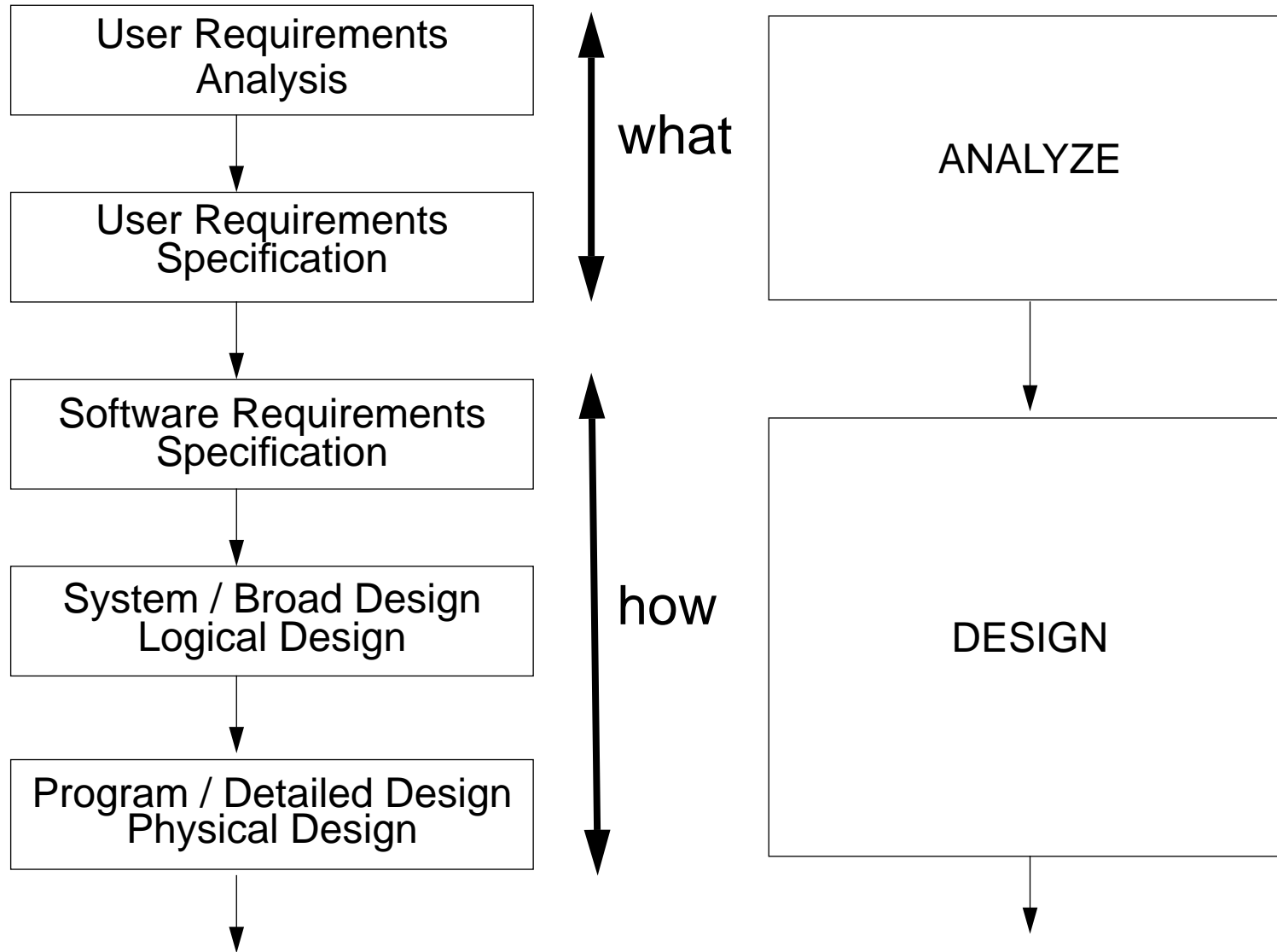
# What is analysis ?

[Rubin and Goldberg]:

“Analysis is the study and modeling of a given problem domain, within the context of stated goals and objectives. It focuses on *what* a system is supposed to do, rather than *how* it is supposed to do it [...] .The components of the problem domain can be described as anything that the end users of the system, both humans and machines, view as part of the problem context.”

- The What and How of *what*
- The *Problem* or the *Solution*

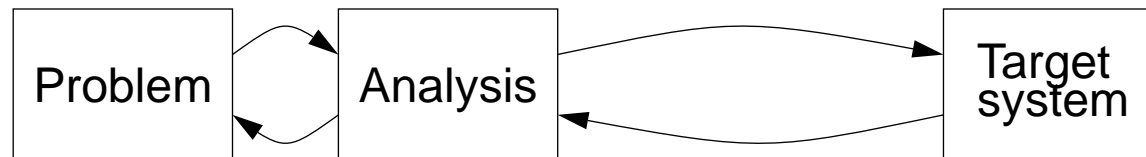
[Adopted from Henderson-Sellers / Edwards, CACM-9-90]



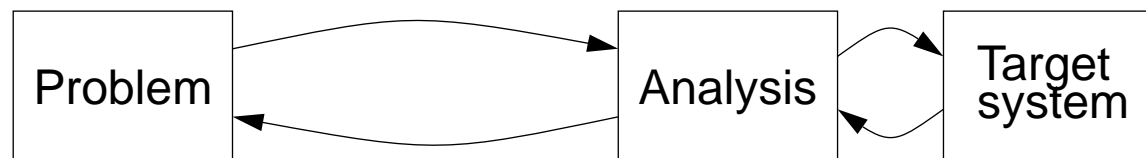
# Problem-orientation:

## Judgement criteria for analysis methods:

- Close to user concepts and terminology.
- Requirements, not solutions.
- Total system goals and objectives, not only computerized parts.



Problem-oriented



Target-oriented

[Hagelstein]

# Claims of OOA:

1. OOA fulfills the purpose of Analysis, and
2. OOA has a smooth transition to Design.

*Why:*

- Adequate representation of real-world concepts, and
- a uniform modeling language.

# Adequate Representation:

Depends on Domain, Goals, and Objectives.

- **Objects:**

- Interaction-oriented systems.
- Complex structures.

- **Processes:**

- Functionality.
- Flow of items.

- **Rules:**

- Goals.
- Constraints.

# Uniform Modeling Language:

- Does not imply a uniform *Model*,
- thus, *Guidelines* must be provided.

## “Uniform” does not mean “Good”

- Different objectives imply different languages.
- Analysis: Problem, Requirement Engineering.
- Design: Solution, Software Engineering.
- Uniform language: Which language to choose ?

# Does OOA fulfill the role of Analysis ?

No, because of these “soft” problems:

- Assume pre-existing requirements.
- Lack of goals, objectives, alternative solutions, and organizational aspects.
- Target-orientation: Early focus on the system to be implemented.
- Validation has not been properly addressed.
- Lack of guidelines for the transition to design.

No, because of limitations in the “paradigm”:

- OO modeling languages are not sufficient for representing real-world concepts.

# Does OOA have a smooth transition to Design?

Not necessarily, depends on goals and objectives:

- Yes, by considering the target system while doing analysis.
- Yes, at the cost of a possibly inferior analysis.

Our claim:

- Different objectives of analysis and design => different results.
- No smooth transition.
- Guidelines must be provided.

# Conclusion.

- “No free lunches”
  - OOA do not satisfy both adequate analysis and a smooth transition to design.
  - Guidelines must be provided.
- Adequate representation:
  - “Objects” not sufficient.
- We recommend:
  - Analysis should be judged by different criteria than design.
  - Synthesis of paradigms.
  - Opening the “Gap” - again.

# The scope of Analysis:

- Application domain knowledge.
- Objectives.
- Requirements on the environment.
- Requirements on the computer system.

# Requirements to the Analysis Language:

- Suitability:

“The ability of the language to easily describe the appropriate knowledge.”

- Understandability:

“How the appropriate knowledge is presented.”

- Formality:

“Well defined semantics.”

# What is OOA ?

“We define system analysis as the study of a specific domain of interacting objects for the purpose of understanding and documenting their essential characteristics.” [Embley et.al.]

Analysis with *objects* as the main building block:

- Object = identity + encapsulated state + interface

Characteristics of OOA:

- Models: Structure, Interaction, Behavior.
- Domain Analysis and Reuse.
- Prototyping and Iterative Development.

# OOA and the life-cycle:

## Iterative Development:

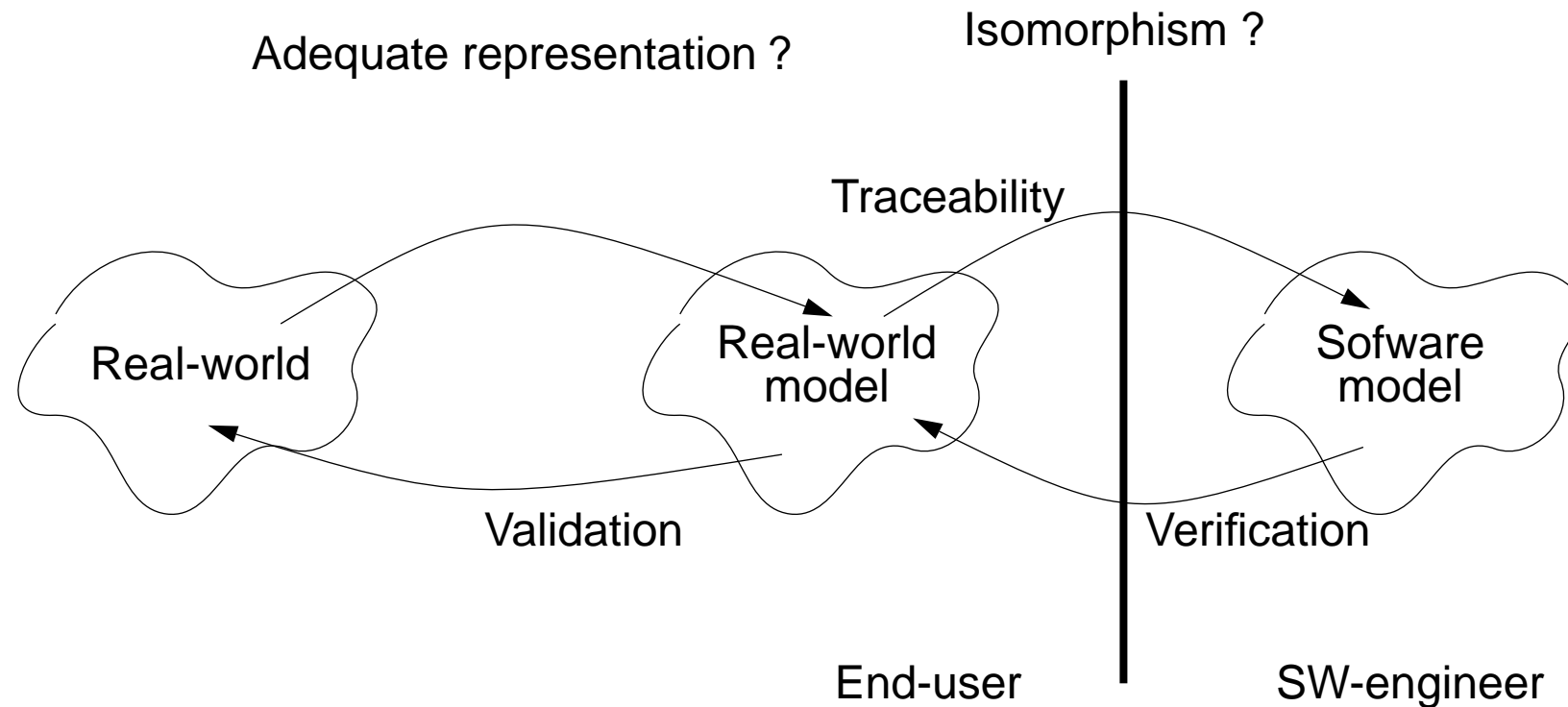
- Validation: Rapid prototyping.
- Conceptual modeling: Still needed.

## Domain analysis:

- Frameworks: Implementing a design.
- Domain-specific class hierarchies.
- Conceptual models.

<=> well known domains.

# The model perspective



## Guidelines must be provided:

- How to re-evaluate object properties.
- How and when to *add* and *remove* objects.
- How to deal with requirements not captured by the modeling language.
- Traceability of requirements.

# When to use OOA

- When the requirements are given.
- Well known problems or domains.
- Small systems.
- Systems not involving organizational change.
- Complex product structures.
- Modeling interacting components.

# Possible directions for OOA

- OOA incorporates requirements engineering, no implications for concepts and modeling language (OBA).
- Extend OOA concepts and modeling language (Embley et.al).
- “Kill the OO prefix”: Synthesis of paradigms.

# Representing reality:

- A house sending messages to its rooms ?
- Is a student a kind of person, or is it a person that play the role student for some time?
- Is a “purchase” an object, event, or activity ?
- Continuous flow as messages ?