


# **Class Diagrams and Use Cases – Experimental Examination of the Preferred Order of Modeling**



**Peretz Shoval, Avi Yampolsky & Mark Last**

**Dept. of Information Systems Engineering  
Ben-Gurion University, Beer-Sheva, Israel**

**EMMSAD-06, June 2006, Luxembourg**

# Introduction

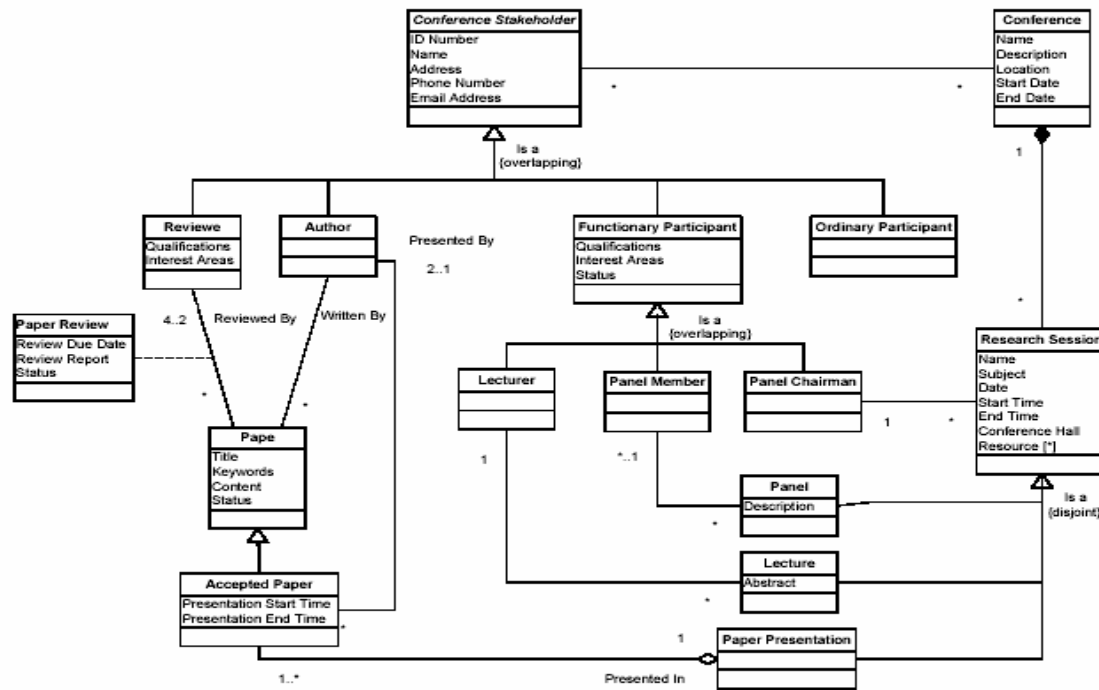
- Two major activities at the analysis stage: **Functional analysis** and **conceptual data analysis**
- **Traditional development methodologies** emphasize functional analysis (mainly using DFDs), while data analysis plays a secondary role only: after completing functional analysis, the data-stores defined in the DFDs are transformed to database relations - e.g. DeMarco (1978); Yourdon (1989).
- **OO development methodologies** tend to emphasize data analysis (mainly using class diagrams), while functionality (behavior) of the system is expressed mainly by methods encapsulated within the object classes.
- **Early OO development methodologies** (prior to UML) prescribed data modeling first (using class diagrams) - e.g.: OOA/OOD (Coad & Yourdon, 1991); OMT (Rumbaugh et al., 1991); Shlaer & Mellor, 1992); Booch (1994).

# Introduction (Cont.)

- **UML-based methodologies** usually prescribe **functional modeling first** – “Use case driven”: First create use cases; then create behavior charts (e.g. Sequence diagrams), leading to the creation of a class diagram
  - e.g. RUP (Jacobson, Booch, Rumbaugh, 1999); Mathiassen et al. (2000]; Larman (2002); Insfran, Pastor & Wieringa (2002).
- **But there are UML-base methodologies which prescribe conceptual modeling first** – creating a domain model (class diagram), then use cases
  - e.g. ICONIX (Rosenberg & Kendall, 2001).
- Some claim that there is **no particular order** for creating use cases and class diagram, as these two activities are done **simultaneously** and are feeding one another
  - e.g. Maciaszek (2001).

# Class Diagram

- Used in various stages of system development
- In the Analysis phase: used to identify the main domain entities, their attributes and relationships.
- Termed: **Domain Object Model**.

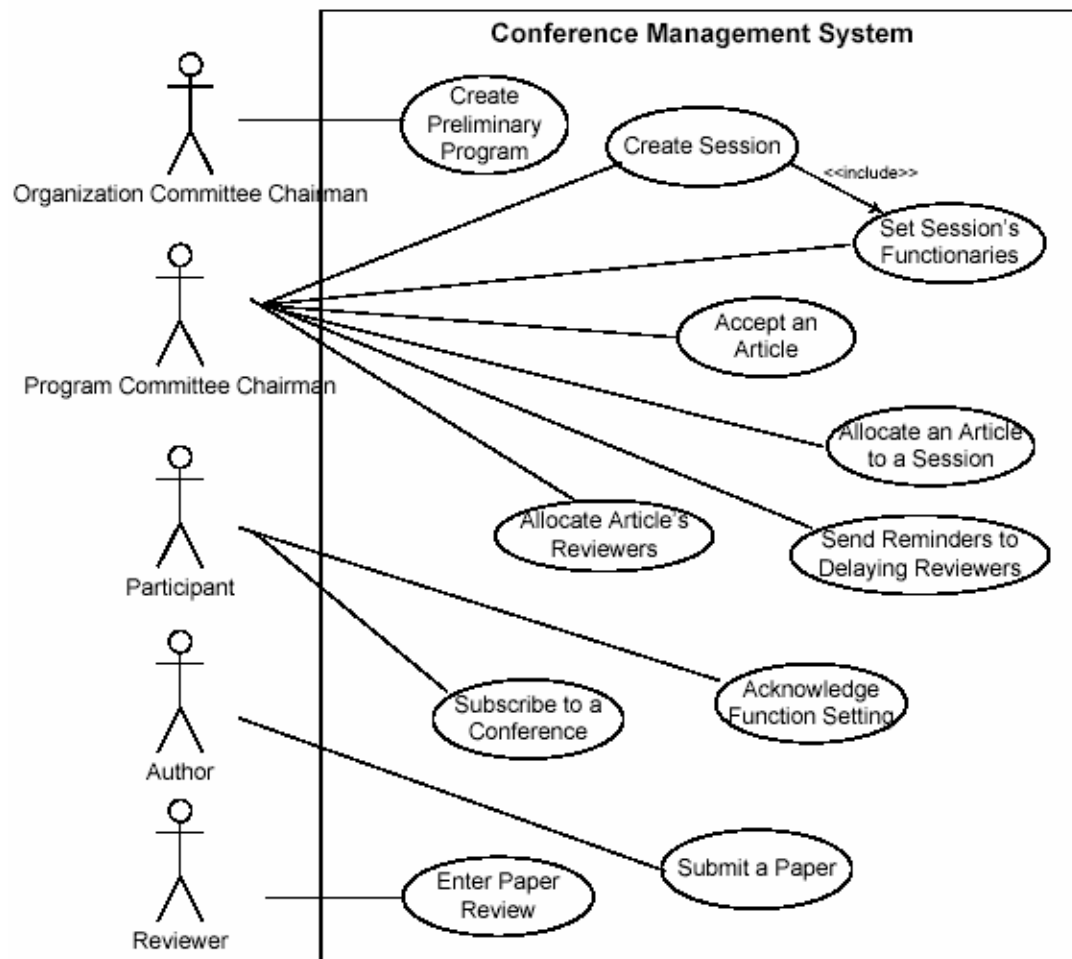


# Use-Cases

---

- Presented by Jacobson in 1987; adopted in UML
- Used to describe functional requirements – use scenarios
- Two main components: Use cases and Actors
- A Use-case describes the interaction b/w the “actor” and the system (a black-box) to achieve the actor’s goal
- **Mainly a textual tool** – the diagram is very limited

# Use-Case Diagram - an example



# Use-Case Description – an example

## UC1: Create Session

**Goal:** The Program Committee (PC) Chairman creates a session according to the conference subjects.

**Primary Actor:** PC Chairman.

**Pre-Condition:** A preliminary conference program exists.

**Post-Condition:** The new session's details are stored in the system.

### **Main Success Scenario:**

1. The PC Chairman requests to create a new conference session.  
*Steps 2-4 can be performed in any order:*
2. The PC Chairman enters the session's details: name, subject, date, start time, end time and conference hall.
3. The PC Chairman marks the session as a **Panel** and enters its description.
4. The PC Chairman enters the resources needed for the session.
5. The system saves the new session's details.

### **Extensions:**

- 3a. The PC Chairman marks the session as a Lecture.
  - 3a1. The PC Chairman enters the lecture's abstract.
- 3b. The PC Chairman marks the session as an Article Presentation.

# Research Motivation

---

- ❑ Many studies compare different methodologies; none deals with the **order of analysis activities**
- ❑ Different OO methodologies suggest different orders; no explanation/proof is provided for the chosen order
- ❑ Obviously, system analysis is an **iterative process** of refinement, not a linear sequence of activities. But still, the analysis must begin with some specific activity. Does it matter which?

# Research Motivation (Cont.)

- An earlier comparative experiment by **Shoval & Kabeli** (Comm. of AIS, 2005; initially presented at EMMSAD-03, Velden):
  - Utilized the **FOOM** (Functional & Object-Oriented Methodology):
    - **Initial Class diagram** for conceptual data modeling
    - **OO-DFDs** for functional modeling
  - Main results:
    - Starting with conceptual data modeling yield better data models
    - Analysts prefer starting with conceptual data modeling

# Hypotheses

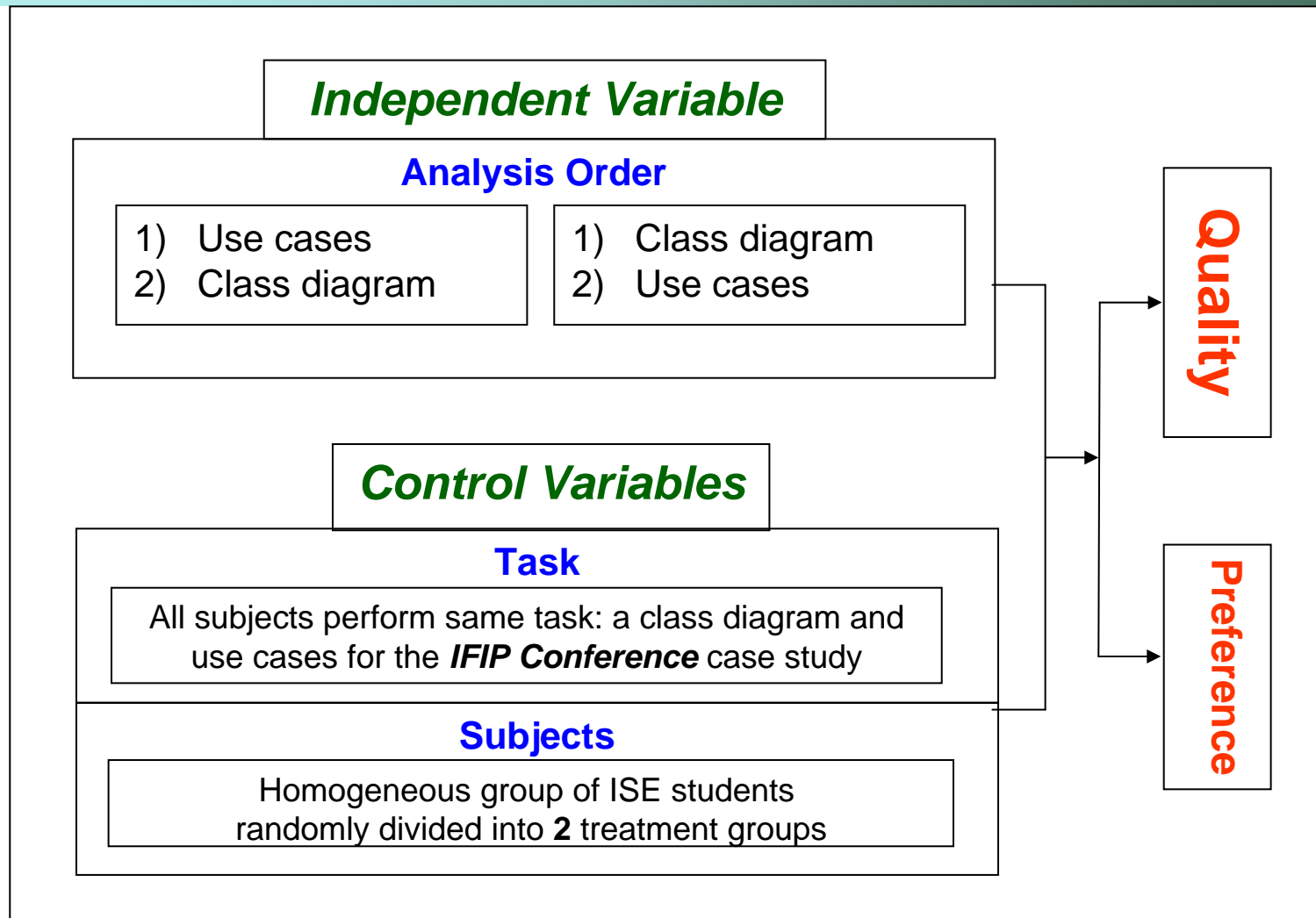
- Creating a functional model first seems to be **more difficult** because the identification and definition of functions is more complex; functions are not “concrete”/tangible“/”stable”; there are usually many use cases.
  - In contrast, a conceptual data model consists of more concrete/ tangible/ stable objects; more easy to identify and define; there are only a few constructs (mainly classes, attributes and relationships); usually there is just one diagram to create.
- Creating a functional model first seems to be more difficult because the analyst actually deals at the same time with the functional **and** the structural aspects (Use case descriptions include all kinds of interactions).
  - In contrast, when beginning with conceptual data modeling, the analyst deals only with the data (classes) aspects; then, when working on functional modeling, the classes are already defined, and therefore the more complex task becomes easier.

# Hypotheses (Cont.)

---

- ❑ However, in fact different methodologies advocate different orders of activities, and there is no scientific/empirical evidence which order is better.
- ❑ Therefore, we hypothesize that there is no difference in the quality of the analysis products created in the opposing orders of activities.
- ❑ The hypotheses will be tested using a controlled experiment.

# The Research Model



# Dependent variables

---

- **Quality:** correctness of the analysis models created by the analysts – based on the types and numbers of errors; measured using grading schemes
- **Preference:** which order of analysis analysts prefer – based on a subjective questionnaire; 1-7 point scale

# Independent variable

- **Analysis order:** the two opposing orders.

# Control variables

## □ Subjects:

- 121 Undergraduate students of Information Systems Engineering
- Same class of students; studied same courses, including UML
- Same training for the experiment

## □ Tasks:

- To create a class diagram and use cases for the *IFIP Conference* case-study (same as used in FOOM experiment)
  - Two version of the user requirements: one starting with functional-related requirements; the other starting with data-related requirements
  - Randomly distributed within the treatment groups – to avoid bias
  - Subjects trained using Cockburn's use case writing guidelines
  - An example given to subjects along with the requirements document
- Time: 2 hours allocated; ½ hour extension

# Treatment Groups

<b>Group</b>	<b>Analysis Order</b>
Group A	1 <sup>st</sup> task: Class Diagram 2 <sup>nd</sup> task: Use Cases
Group B	1 <sup>st</sup> task: Use Cases 2 <sup>nd</sup> task: Class Diagram

# Grading scheme – class diagram

Element	Error	points
Attribute	Missing attribute or attribute in the wrong class	-2
	Superfluous attribute	-1
Relationship multiplicity	Missing or incorrect multiplicity	-1
Relationship	Missing relationship	-4
	Erroneous relationship	-3
	Superfluous relationship	-3
	Incorrect relationship type	-2
Class	Missing class	-6
	Superfluous class	-2
	Incorrect class type	-1
Inheritance	Missing inheritance	-6
	Regular relationship instead of inheritance	-2
	Superfluous inheritance	-2

# Grading scheme – use cases

Element	Error	points
Actor	Incorrect actor	-4
	Inconsistent actor	-2
User goal	Missing goal	-10
	Goal appears in title only	-6
	Goal described as a part of other goal	-4
	Superfluous goal	-2
Sub-goal	Missing sub-goal	-3
	Superfluous or erroneous sub-goal	-2

In addition, we defined the following measures of quality of use description:

- Unreasonable solution (-3)
- Not clear (-1)
- No logical sequence (1)
- No consistency in level of detail (-1)

# Statistical tests

---

- **Quality of models** (grades – continuous variable): Two-way t-tests
- **Preferences** (ordinal 7-point scale): Wilcoxon (non parametric) test

# Results: Quality of class diagrams

Analysis Order	N	Mean grade (%)	t	p-value	Significance in favor of
1) Class Diagram; 2) Use Cases	57	<b>73.63</b>	4.386	.038	Starting with class diagram
1) Use Cases; 2) Class Diagram	64	<b>70.25</b>			

**Better to start with data modeling**

# Results: Quality of use cases

Analysis Order	N	Mean grade (%)	t	p-value	Significance in favor of
1) Class Diagram; 2) Use Cases	57	<b>63.72</b>	.192	.662	--
1) Use Cases; 2) Class Diagram	64	<b>65.03</b>			

**No significant difference**

# Analysts' preferences

Participants grouped by analysis order	N	Mean preference	Standard deviation
Class Diagram → Use Cases	22	2.91	1.54
Use Cases → Class Diagram	18	2.61	1.82
<b>All together</b>	<b>40</b>	<b>2.78</b>	<b>1.66</b>

1-7 point scale: 1 means total preference to start with class diagram;  
7 means total preference to start with use cases

**Significant preference to start with class diagram**  
Even stronger among those who started with use cases

# Summary of results

---

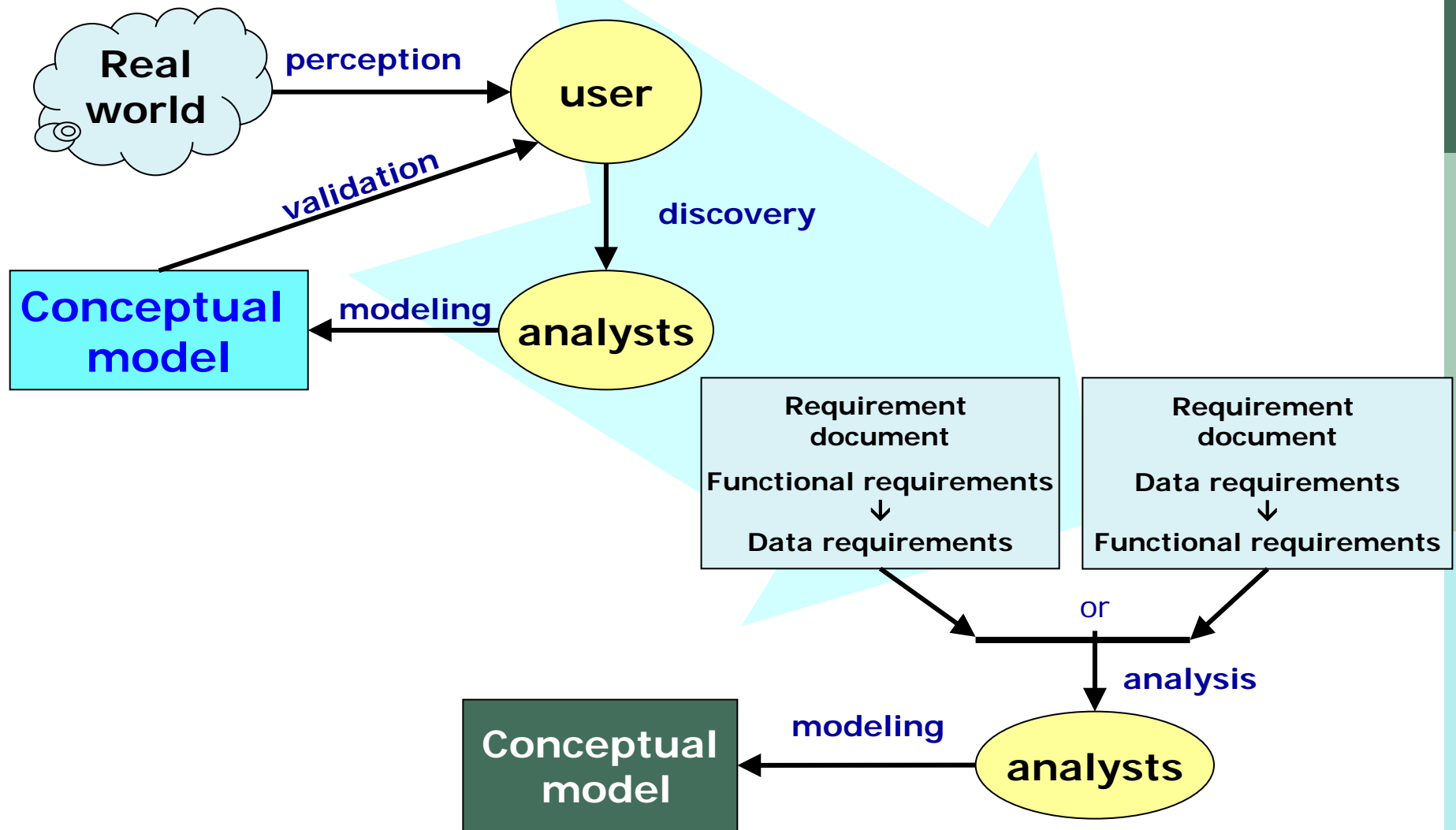
- ❑ **Class diagram is of better quality when starting the analysis with conceptual data modeling**
- ❑ **No significant differences in quality of use cases**
- ❑ **Analysts prefer starting with conceptual data modeling**
- ❑ **The results are consistent with the FOOM experiment results**

# Limitations of experiment

---

- ❑ One relatively small case study (part of the *IFIP Conference*)
- ❑ One domain of application (Data-driven / MIS)
- ❑ Participants were students – not experienced analysts
- ❑ Task was only to create analysis specification; not a real/working system
- ❑ Grading schemes are subjective
- ❑ Requirements document surrogate for real-world interaction b/w analysts and users

# Model of analysis in reality



# Further research issues

---

- Use experienced analysts
- Use real-world / large scale applications
- Applications from different domains
- Compare also to iterative (parallel) analysis
- Create/use a realistic analysis process

=====