Concept Identification in Object-Oriented Domain Analysis:

Why Some Students Just Don't Get It

D. Svetinovic D. M. Berry M. W. Godfrey

School of Computer Science
University of Waterloo
Waterloo, Canada

International Requirements Engineering Conference
Paris, 2005

Overview

- What is the real value of object-orientation (OO) and object-oriented domain analysis (OODA)?

  Not clear!

- Does OODA produce high-quality conceptual models?

  Case studies suggest NO!

Introduction

- Object-Oriented Analysis (OOA) is extremely popular.
- Object-Oriented Design (OOD) is extremely popular.
- Object-Oriented Programming (OOP) is extremely popular.
- Lots of people swear by OO.

Object-Oriented Domain Analysis (OODA) is OOA of the problem domain.

Need for Validation

Despite being widely used, OO has not been extensively validated.

- Hatton, Kaindl, and Kramer have indicated an urgent need for experimental validation of the effectiveness of not just OO but of all SE abstraction techniques and methods.
- **Our work attempts to explore the issues of incompleteness and non-predictability of conceptual models (CMs), important components of quality.**

Quality of CMs is Important

- We must ask:

  How complete and predictable are the CMs produced by the OODA efforts of groups of analysts?

- Our experience in evaluating CMs produced by student groups has shown that a good surrogate question is:

  How mutually consistent are the CMs of the same problem domain produced by OODA efforts of independent groups of analysts?

- Answer to both questions: "Not very", as we shall see!

Motivation

- Observations of the students' work in a Software Requirements Specification course at University of Waterloo.
- The term-long project for CS445 is to produce an SRS for
  1. a small telephone exchange or a VoIP telephone network and
  2. its related accounts management subsystem.
- Over 5 years, 740 students and 195 projects, 31 of which were examined closely.
- The incompleteness and variation in the CMs is breathtaking.

Relevant OODA Techniques

- Students relied upon a typical set of OODA techniques:
  - use case modeling,
  - noun phrase analysis,
  - category analysis,
  - group consolidation,
  - evaluation by customers (teaching assistants), and
  - general domain knowledge.
- They were instructed also in related project management techniques and UML.

Results: VoIP System Specifications

In the 31 closely examined group projects:

Total number of concepts:	527
After elimination of syntactic duplicates:	259
Number of semantically unique concepts:	134

Per group:

Minimum:	8
Maximum:	31
Average:	17
Median:	16

Results: VoIP System Specifications

Of the 134 semantically unique concepts:

- Only 51 appear in at least 2 SRSs
- Only 40 appear in at least 3 SRSs
- Only 8 appear in at least 50% of SRSs
- Only 6 appear in at least 80% of SRSs

The Fundamental Difficulty

The most frequently observed difficulty is that of *just doing* OODA, that is,

- identifying concepts of the system's domain and
- ascribing the system's functionality to these concepts.

We call this the **fundamental difficulty (FD)** of OODA.

FD in Small Systems?

Originally, we had thought that the FD came from the large size of the problem, but in another case study, we demonstrated that:

Also small systems suffer from the FD!

Even, for example, polished educational case studies of specifying elevator systems published by OODA experts.

Results: Elevator System Specifications

Look at how little overlap there is among three specifications!

Concept	*CS1 Status*	*CS2 Status*	*CS3 Status*	*Type*	*Purpose*
elevator system	I	D	I	CSE	to define the conceptual boundaries of the system to control and move the elevator cab
passenger	I	I	I	A	to use the elevator
elevator cab	D	D	D	PSE	transport passengers
building			I	PSE	physical system boundary
floor	I	D	I	PSE	to provide building's structure to define elevator travel destinations
top floor		I	I	PSE	special floor - different user interface to provide direction reference
bottom floor		I	I	PSE	special floor - different user interface to provide direction reference
button panel			I	PSE	container for buttons
elevator shaft			I	PSE	pathway for the elevator cab provide elevator access to the floors
button	D		I	PP	to unify all buttons
elevator button	D	D		PP	unify buttons inside the elevator cab
floor request button		D	D	PP	user interface for requesting floors
door button			D	PP	user interface for door opening
open door button		D	I	PP	request to open the doors when the elevator is not moving
close door button		D	I	PP	request to close the doors when the doors are open
floor button	D	D	D	PP	user interface for requesting elevator
stop button			I	PP	request immediate elevator stop at the next floor
door	D	D	D	PP	close the elevator cab
inner door		D		PP	close the elevator cab
outer door		D	I	PP	close the floor access to the elevator shaft
door opening device		I		PP	open the doors
floor number display			I	PP	user interface for indicating travel progress
floor sensor		D		PP	detect elevator position with respect to the floors
elevator engine		D		PP	move the elevator cab
elevator controller	D			CP	to delegate interface requests within the system to delegate internal responsibilities within the system
door timer		D	I	CP	constrain door opening time periods
current floor	I			IC	to define current location of the elevator
designated floor			I	IC	final travel destination
request		I	I	IC	unify all the request types
requested direction		D		IC	track user's traveling preference used for the elevator stopping scheduling purposes
elevator request		D		IC	track user's request for elevator services used for the elevator stopping scheduling purposes
elevator-up request		D		IC	*same as for elevator request*
elevator-down request		D		IC	*same as for elevator request*
pending queue		D		IC	keep track of unprocessed *elevator request button* and *floor request button* requests
summon			D	IC	to capture elevator request
stop request			I	IC	capture users request for stopping at a particular floor
time period		I	I	IC	constraint time allowed for various operations
stop			I	IC	unify different elevator stopping situations
immediate stop			I	IC	unplanned stop initiated by the passenger
planned stop			I	IC	stop at final travel destination
stop notification			I	IC	user interface for indicating elevator stops
button refusal notification			I	IC	user interface for invalid request notification
direction			I	IC	capture current traveling direction of the elevator
light		I		IC	user interface to indicate button status

Results: Elevator System Specifications

Very little commonality among the 3 specifications:

- Total concepts: 44

Case Study	Discovered	% Discovered	Ignored	% Ignored
CS1	6	14	4	9
CS2	19	43	7	15
CS3	6	14	25	57

Table: Numbers of Concepts

- In the first case study, the ratio of discovered to ignored concepts is 3:2; in the second, the ratio is 3:1; and in the third, the ratio is 1:4.

See paper for the detailed concepts table and evaluation.

Summary of Results

Thus, we have found in these two studies of large and small systems that:

- OODA models are underspecified.
- OODA models of the same system are drastically different.
- A typical specification has a large number of software concepts at inconsistent abstraction levels.

$\therefore$ **the FD is independent of problem size!**

Why is OODA Difficult?

- Not due to the size of the system
- Not due to the analysts' lack of OODA experience
- We believe that the OODA is difficult because it is poorly suited to deal with two inherent properties of complex business systems:
  1. Each of most concepts fulfills only a subactivity of a larger activity by interacting with other concepts.
  2. Each of most concepts participates in many different activities, each for different purposes.

What Analysts Actually Do

- Analysts try to assign responsibilities that are fulfilled through the collaboration of multiple concepts to only one concept.
- Analysts tend not to capture passive concepts, concepts that are produced or consumed by interaction of other concepts.

Solutions?

- Skipping OODA all together?
- Other paradigms?
- Improving OODA?
- Postponing and constraining OODA through additional refinement of RA artifacts?

From my observation of what works with students, I feel that the last option is the most promising solution.

Future Research

- Defining and constraining RA artifacts as a source of concepts beyond use cases and general domain knowledge.
- Exploiting goal theory and goal-based approaches.
- More precise functional modeling of the overall system before OODA.

Future Research

- We will pursue the last option, with a choice of detailed system high-level requirements specification using state-based modeling.
- Other options include activity modeling, aspect-oriented modeling, agent-oriented modeling, etc.
- We are pursuing state-based modeling due to its widespread use and simplicity, compared to other approaches.

Questions?

Concept Identification in Object-Oriented Domain Analysis:
Why Some Students Just Don't Get It

Questions?