Domain Models

Lecturer: Ngo Huy Bien
Software Engineering Department
Faculty of Information Technology
VNUHCM - University of Science
Ho Chi Minh City, Vietnam
nhbien@fit.hcmus.edu.vn

Objectives

- To *create* a domain model for a software system.

Contents

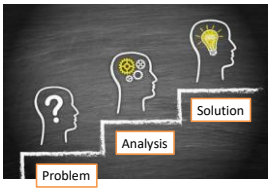
I. Domain Models

References

1. Craig Larman (2004). Applying UML And Patterns. 3rd Edition. Prentice Hall.
2. Colin Atkinson and Thomas Kuhne (2008). Reducing Accidental Complexity in Domain Models.
3. Suzanne Robertson and James Robertson (2012). Mastering the Requirements Process. Addison Wesley Professional.

Problem Analysis and Specification

- The purpose of software is to *solve problems*.
- How do we *analyze* and *document* a problem?



What is Domain Model? [1]

- Informally, a *conceptual class* is an idea, thing, or object that has symbol, intention and examples.
- A *domain model* is a visual representation of conceptual classes or real-situation objects in a domain.
- *Elements* of a domain model are domain object classes, and the relationships between them.

How to Create a Domain Model?

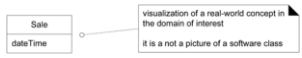
Find the conceptual classes

Reuse or modify existing models.	Listen to the core vocabulary and concepts that domain experts use.	Use noun phrase identification: Identify the nouns and noun phrases in textual descriptions of a domain

Draw them as classes in a UML class diagram

Add associations and attributes.

Domain Model vs. Software Business Objects

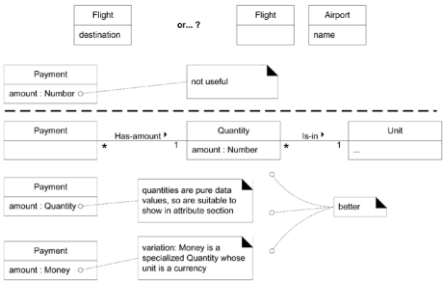


A DOMAIN MODEL SHOWS *REAL-SITUATION* CONCEPTUAL CLASSES, NOT SOFTWARE CLASSES.

A DOMAIN MODEL DOES *NOT* SHOW SOFTWARE ARTIFACTS OR CLASSES.

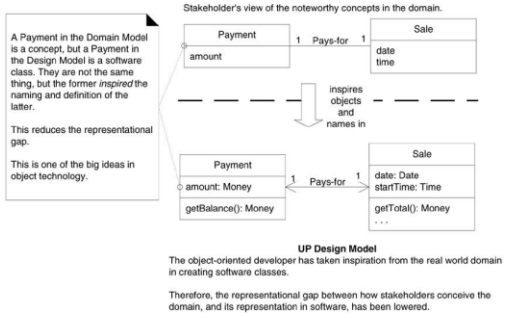
Attributes vs. Classes

If we do not think of some conceptual class X *as* a number or text or date/time in the real world, X is probably a conceptual class, not an attribute.



Why Domain Models?

This supports a *low representational gap* between our mental and software models.



The object-oriented developer has taken inspiration from the real world domain in creating software classes.

Therefore, the representational gap between how stakeholders conceive the domain, and its representation in software, has been lowered.

Domain Model? [2]

- "Domain model" or "analysis model": a solution-independent *description of the problem* to be solved.
- A domain model is assumed to *represent* a conceptualization of the entities of some problem domain.
- The most common choice for *creating* such domain models is the modeling language standard, the UML.
- Creating a domain model with the UML usually includes the creation of one or more *class diagrams* that capture the important domain concepts and their relationships.
- Often one also adds one or more *object diagrams* showing instances of the domain concepts to illustrate various configurations.

Why Domain Models?

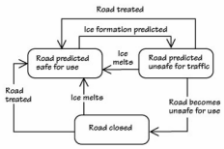
- To serve as a description of the problem that is *understandable* to the widest possible range of stakeholders.

Business Model [1]

- A *domain model* is defined as an abstract model that captures the most important types of objects in the context of the system.
- The *domain objects* represent the "things" that exist or events that transpire in the environment in which the system works.
- A *business model* comprises two models:
  - a *business use-case model* to describe the business actors and the business processes,
  - and a *business object model* to describe business entities used by the business use cases.

State Model [3]

- The object life history is a *state model* (also known as a state transition diagram) that shows which states the objects for a class (sometimes called entities) may take, and illustrates the transitions between those states.
- A *state* is a steady condition for the object, and each rectangle in the model identifies a different state that this object will be in at one time or another.
- The story is told not so much by the states, as by the *transitions between states*.



Thank You & See You Again