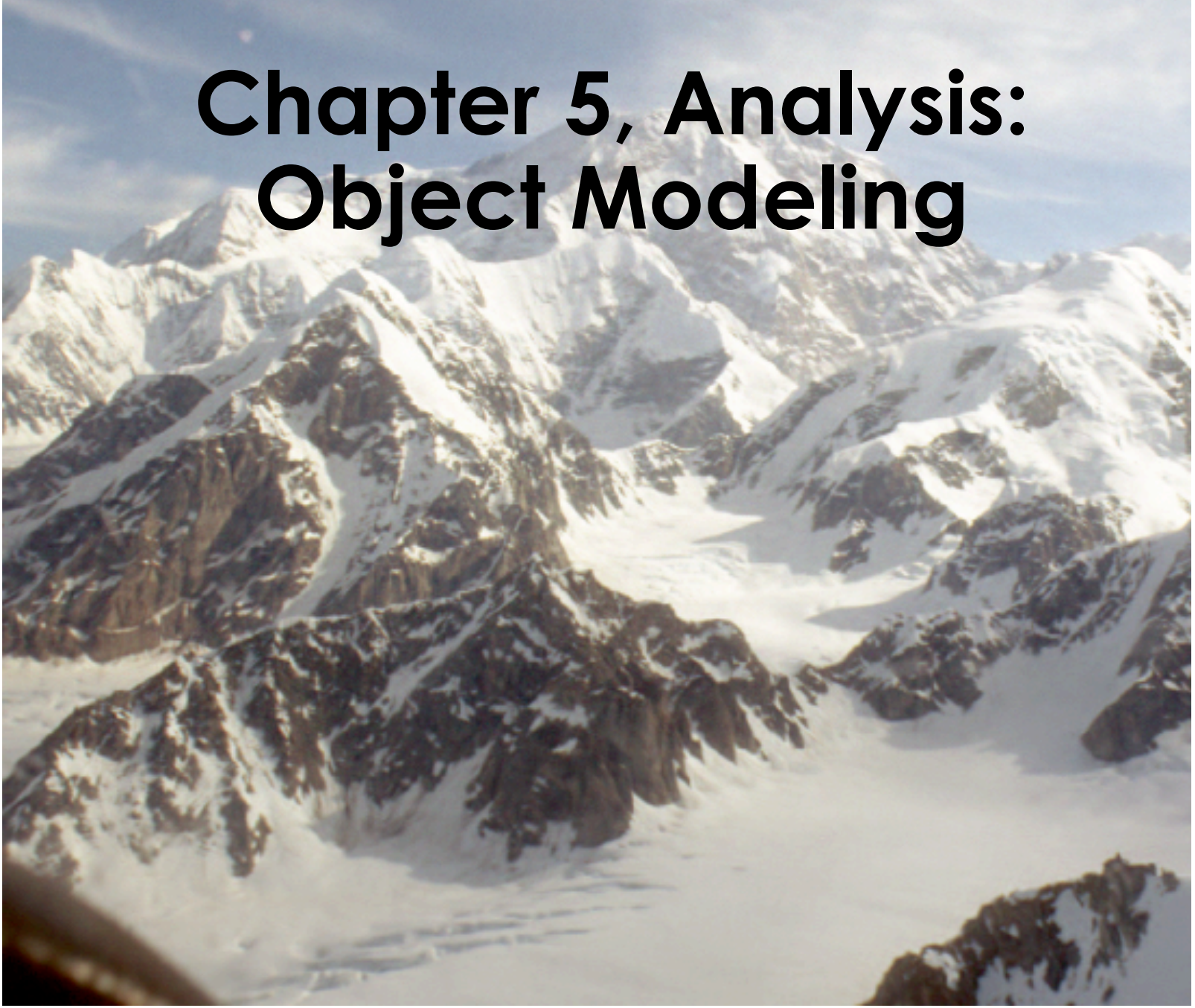


Chapter 5, Analysis: Object Modeling



Outline

Recall: System modeling = Functional modeling +
Object modeling + Dynamic modeling

✓ Last lecture: Functional modeling

• Now: Object modeling

- Activities during object modeling
- Object identification
- Object types
 - Entity, boundary and control objects
- Abott's technique
 - Helps in object identification.

Activities during Object Modeling

Main goal: Find the important abstractions

- Steps during object modeling



Class identification

- Based on the fundamental assumption that we can find abstractions

2. Find the attributes

3. Find the operations

4. Find the associations between classes

- Order of steps

- Goal: get the desired abstractions
- Order of steps is secondary

- What happens if we find the wrong abstractions?

- We iterate and revise the model.

Class Identification

Class identification is crucial to object-oriented modeling

- Helps to identify the important entities of a system
- Basic assumptions:
 1. We can find the classes for a new software system
(Forward Engineering)
 2. We can identify the classes in an existing system
(Reverse Engineering)
- Why can we do this?
 - Philosophy, science, experimental evidence.

Class Identification

- Approaches
 - Application domain approach
 - Ask application domain experts to identify relevant abstractions
 - Syntactic approach
 - Start with use cases
 - Analyze the text to identify the objects
 - Extract participating objects from flow of events
 - Design patterns approach
 - Identify relevant abstractions that can be reused (apply design knowledge)
 - Component-based approach
 - Identify existing solution classes.

Class identification is a Hard Problem

- One problem: Definition of the system boundary:
 - Which abstractions are outside, which abstractions are inside the system boundary?
 - Actors are outside the system
 - Classes/Objects are inside the system
- An other problem: Classes/Objects are not just found by taking a picture of a scene or domain
 - The application domain has to be analyzed
 - Depending on the purpose of the system different objects might be found
 - How can we identify the purpose of a system?
 - Scenarios and use cases => Functional model.

There are different types of Objects

- **Entity Objects**
 - Represent the persistent information tracked by the system (Application domain objects, also called “Business objects”)
- **Boundary Objects**
 - Represent the interaction between the user and the system
- **Control Objects**
 - Represent the control tasks performed by the system.

Example: 2BWatch Modeling

To distinguish different object types
in a model we use the
UML Stereotype mechanism

Year

Month

Day

ChangeDate

Button

LCDDisplay

Entity Objects

Control Object

Boundary Objects

Naming Object Types in UML

<<Entity>>
Year

<<Entity>>
Month

<<Entity>>
Day

<<Control>>
ChangeDate

<<Boundary>>
Button

<<Boundary>>
LCDDisplay

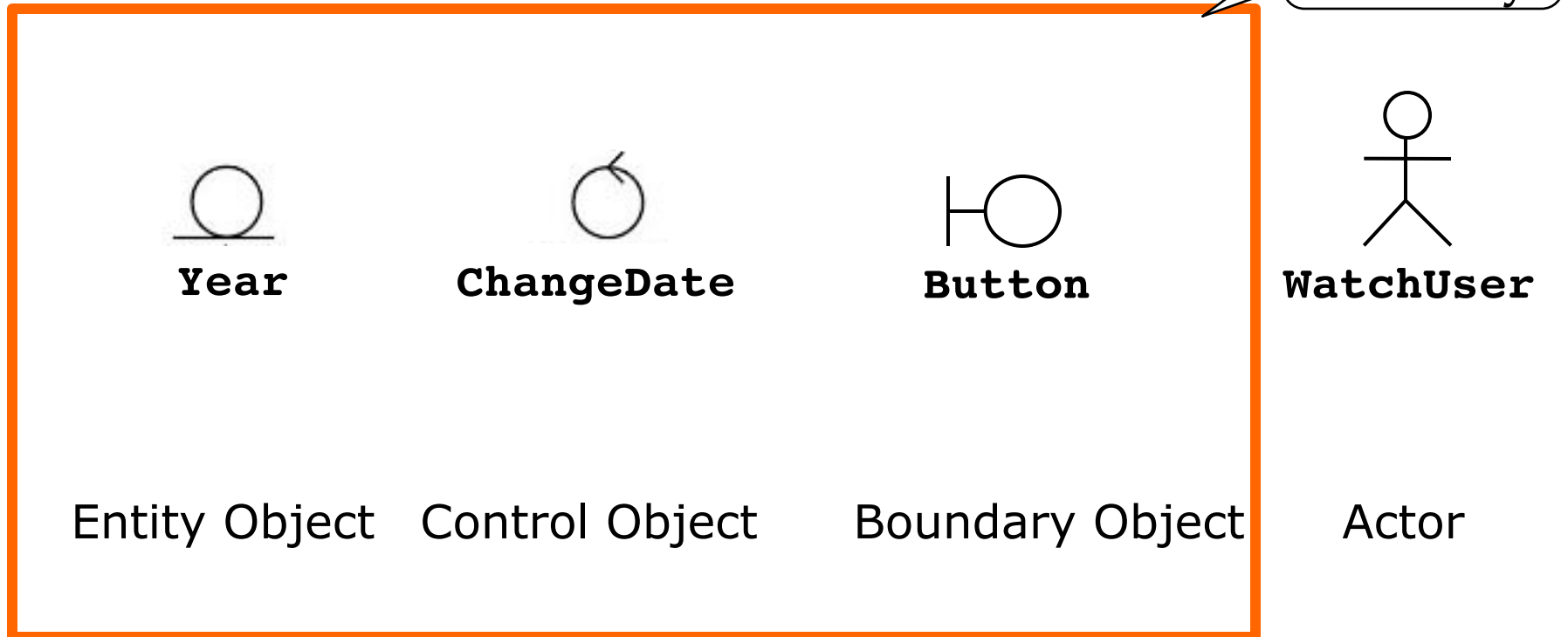
Entity Object

Control Object

Boundary Object

Icons for Object Types: Entity, Control and Boundary Object

- We can also use **icons** to identify a stereotype
 - When the stereotype is applied to a UML model element, the icon is displayed beside or above the name

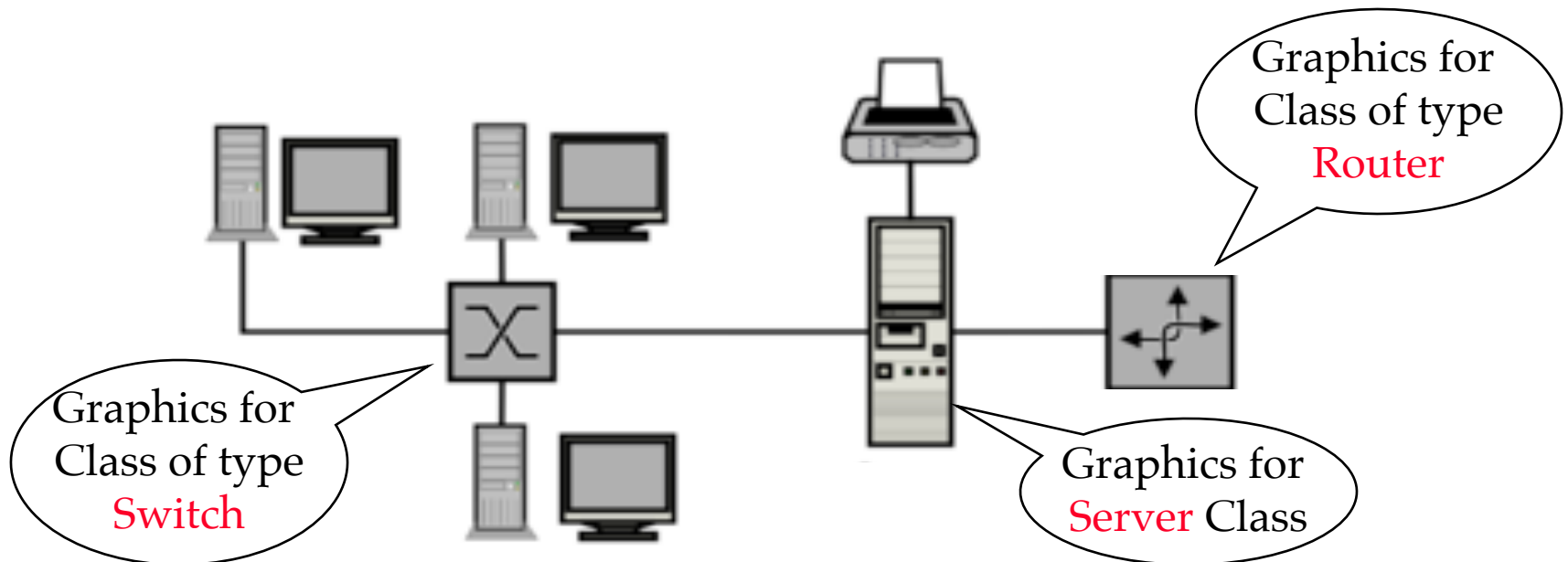


UML is an Extensible Language

- Stereotypes allow you to extend the vocabulary of the UML so that you can create new model elements, derived from existing ones
- Other Examples:
 - Stereotypes can also be used to classify method behavior such as <<constructor>>, <<getter>> or <<setter>>
 - To indicate the interface of a subsystem or system, one can use the stereotype <<interface>> (Lecture System Design)
- Stereotypes can be represented with icons as well as graphics:
 - This can increase the readability of UML diagrams.

Graphics for Stereotypes

- One can also use **graphical symbols** to identify a stereotype
 - When the stereotype is applied to a UML model element, the graphic replaces the default graphic for the diagram element.
- Example: When modeling a network, we can define graphical symbols to represent classes of type **Switch**, **Server**, **Router** and **Printer**.



Pros and Cons of Stereotype Graphics

- Advantages:
 - UML diagrams may be easier to understand if they contain graphics and icons for stereotypes
 - This can increase the readability of the diagram, especially if the client is not trained in UML
 - And they are still UML diagrams!
- Disadvantages:
 - If developers are unfamiliar with the symbols being used, it can become much harder to understand what is going on
 - Additional symbols add to the burden of learning to read the diagrams.

Object Types allow us to deal with Change

- Having three types of object leads to models that are more resilient to change
 - The interface of a system changes more likely than the control
 - The way the system is controlled changes more likely than entities in the application domain
- Object types originated in Smalltalk:
 - Model, View, Controller (MVC)
 - Model <-> Entity Object
 - View <-> Boundary Object
 - Controller <-> Control Object
- Next topic: Finding objects.

Finding Participating Objects in Use Cases

- Pick a use case and look at flow of events
- Do a textual analysis (noun-verb analysis)
 - Nouns are candidates for objects/classes
 - Verbs are candidates for operations
 - This is also called **Abbott's Technique**
- After objects/classes are found, identify their types
 - Identify **real world entities** that the system needs to keep track of (FieldOfficer → Entity Object)
 - Identify **real world procedures** that the system needs to keep track of (EmergencyPlan → Control Object)
 - Identify **interface artifacts** (PoliceStation → Boundary Object).

Example for using the Technique

Flow of Events:

The customer enters the store to buy a toy
It has to be a toy that his daughter likes
and it must cost less than 50 Euro

He tries a videogame, which uses a data
glove and a head-mounted display. He likes
it


An assistant helps him

The suitability of the game depends on the
age of the child

His daughter is 3 years old

The assistant recommends another type of
toy, the boardgame "Monopoly".

Mapping grammatical constructs to model components (Abbot's Technique)

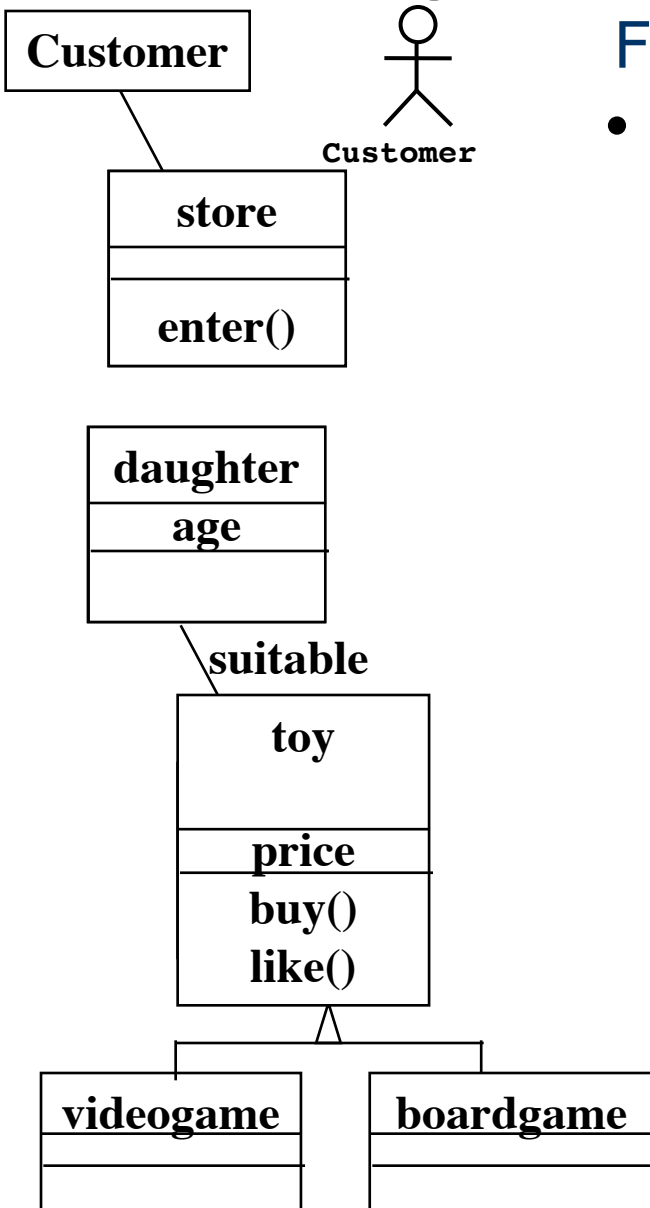
<i>Example</i>	<i>Grammatical construct</i> 	<i>UML model component</i>
“Monopoly”	Proper noun	object
Toy	Improper noun	class
Buy, recommend	Doing verb	operation
Is a	being verb	inheritance
has an	having verb	aggregation
must be	modal verb	constraint
dangerous	adjective	attribute
enter	transitive verb	operation
depends on	intransitive verb	Constraint, class, association

Generating a Class Diagram from Flow of Events

Flow of events:

- The **customer enters** the **store** to **buy** a **toy**. It has to be a toy that his **daughter** likes and it must cost **less than 50** Euro. He tries a **videogame**, which uses a data glove and a head-mounted display. He likes it.

An assistant helps him. The suitability of the game **depends** on the **age** of the child. His daughter is only 3 years old. The assistant recommends another **type of toy**, namely a **boardgame**. The customer buy the game and leaves the store



Ways to find Objects

- Syntactical investigation with Abbot's technique:
 - Flow of events in use cases
 - Problem statement
- Use other knowledge sources:
 - **Application knowledge:** End users and experts know the abstractions of the application domain
 - **Solution knowledge:** Abstractions in the solution domain
 - **General world knowledge:** Your generic knowledge and intuition

Order of Activities for Object Identification

1. Formulate a few scenarios with the help from an end user or application domain expert
2. Extract the use cases from the scenarios, with the help of an application domain expert
3. Then proceed in parallel with the following:
 - Analyse the flow of events in each use case using Abbot's textual analysis technique
 - Generate the UML class diagram.

Steps in Generating Class Diagrams

1. Class identification (textual analysis, domain expert)
2. Identification of attributes and operations (sometimes before the classes are found!)
3. Identification of associations between classes
4. Identification of multiplicities
5. Identification of roles
6. Identification of inheritance

Who uses Class Diagrams?

- Purpose of class diagrams
 - The description of the static properties of a system
- The main users of class diagrams:
 - The application domain expert
 - uses class diagrams to model the application domain (including taxonomies)
 - during requirements elicitation and analysis
 - The developer
 - uses class diagrams during the development of a system
 - during analysis, system design, object design and implementation.

Who does not use Class Diagrams?

- The **client** and the **end user** are usually not interested in class diagrams
 - Clients focus more on project management issues
 - End users are more interested in the functionality of the system.

Summary

- System modeling
 - Functional modeling+object modeling+dynamic modeling
- Functional modeling
 - From scenarios to use cases to objects
- Object modeling is the central activity
 - Class identification is a major activity of object modeling
 - Easy syntactic rules to find classes and objects
 - Abbot's Technique
- Class diagrams are the “center of the universe” for the object-oriented developer
 - The end user focuses more on the functional model and usability.