Introduction to Software Engineering
(2+1 SWS)
Winter Term 2009 / 2010

Dr. Michael Eichberg
Vertretungsprofessur Software Engineering
Department of Computer Science
Technische Universität Darmstadt
Domain Model and Domain Modeling
“Curtis’law: [...] Good designs require deep application knowledge.”
The **Domain Model** illustrates noteworthy concepts in a domain.

Domain Model = dt. Analysemödell (Konzeptmodell)

- It is the basis for the design of the software
- The domain model is created during object-oriented analysis to decompose the domain into concepts or objects in the real world
- The model should identify the set of conceptual classes
  (The domain model is iteratively completed.)

The domain model is also called **conceptual model**, **domain object models** or **analysis object models**.
Conceptual Classes

**Conceptual classes** are ideas, things or objects in the domain. A conceptual class has a symbol representing the class, an *intension* and an *extension* that defines the set of examples to which the conceptual class applies.

Domain concepts / conceptual classes are not software objects as e.g. in Java, C# or .NET!

intension = dt.(hier) Bedeutung
extension = dt.(hier) Ausprägung
To visualize domain models the UML class diagram notation is used.

- However, **no operations are defined** only:
  - domain objects or conceptual classes
  - associations between them and
  - attributes of conceptual classes are shown

The result is a conceptual perspective model.
A **class** describes a set of objects with the same semantics, properties and behavior. When used for domain modeling, it is a visualization of a real world concept.
Attributes are logical data values of an object.

It is useful to identify those attributes of conceptual classes that are needed to satisfy the information requirements of the current scenarios under development.
Attributes are logical data values of an object.

Full syntax for an attribute in the UML:
visibility name : type multiplicity = default {property string}
An **association** is a relationship between classes. The ends of an association are called roles. Roles optionally have a multiplicity, name and navigability.
The **multiplicity** defines how many instances of a class A can be associated with one instance of a class B at any particular moment.

**multiplicity =dt. Vielheit / Vielfachheit**

- 
  - `*` zero or more
  - `1..*` one or more
  - `5,10` exactly 5 or 10
Two classes can have multiple associations.

Multiplicity = dt. Vielheit / Vielfachheit
Which are noteworthy domain concepts / domain objects?
How to create the domain model?

1. Find the conceptual classes
   Strategies:
   a. *Reuse or modify an existing model*
   b. *Use a category list*
   c. *Identify noun phrases*

2. Draw them as classes in a UML class diagram

3. Add associations and attributes

Use the domain vocabulary; e.g. a model for a library should use names like “Borrower” instead of customer.
How to create the domain model? Find the conceptual classes -

**Strategy: reuse or modify an existing model.**

Example: existing model for bank accounts and associated entries.

\[
\text{balance} = \text{sum}(\text{entries.amount})
\]

A possible source: Analysis Patterns - Reusable Object Models; Martin Fowler; Addison-Wesley, 1997
How to create the domain model? Find the conceptual classes - Strategy: use a category list.

<table>
<thead>
<tr>
<th>Conceptual Class Category</th>
<th>Classes (for the POS system)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business transactions...</td>
<td>Sale, Payment</td>
</tr>
<tr>
<td>Transaction line items...</td>
<td>SalesLineItem</td>
</tr>
<tr>
<td>Product or service related to a transaction or transaction line item.</td>
<td>Item</td>
</tr>
<tr>
<td>Where is the transaction recorded?</td>
<td>Register</td>
</tr>
<tr>
<td>Roles of people or organizations related to the transaction; actors in use cases.</td>
<td>Cashier, Customer, Store</td>
</tr>
<tr>
<td>Place of transactions.</td>
<td>Store</td>
</tr>
<tr>
<td>Noteworthy events, often with a time or place that need to be remembered.</td>
<td>Sale, Payment</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

line item = dt. Einzelposten
transaction = dt. Abwicklung / Durchführung; Geschäftsvorfall
How to create the domain model?
Find the conceptual classes -
Strategy: identify noun phrases (linguistic analysis).

Identify the nouns and noun phrases in textual
descriptions of a domain and consider them as
candidate conceptual classes or attributes.

A mechanical noun-to-class mapping isn’t
possible; words in natural languages are
ambiguous; i.e. the same noun can mean
multiple things and multiple nouns can actually
mean the same thing.
How to create the domain model?
Find the conceptual classes -
Strategy: identify noun phrases.

Example:
Using the Process Sale use case as the source for identification.

Process Sale:
A customer arrives at a checkout with items to purchase. The cashier uses the POS system to record each item. The system present a running total and line-item details. The customer enters payment information, which the system validates and records. The system updates the inventory. The customer receives a receipt from the system and then leaves the store with the items.
How to create the domain model? Find the conceptual classes - Strategy: identify noun phrases.

Example: Using the Process Sale use case as the source for identification.

Process Sale:
A customer arrives at a checkout with items to purchase. The cashier uses the POS system to record each item. The system presents a running total and line-item details. The customer enters payment information, which the system validates and records. The system updates the inventory. The customer receives a receipt from the system and then leaves the store with the items.
Identified candidate conceptual classes: Customer, Item, Cashier, Store, Payment, Sales Line Item, Inventory, Receipt, Sale.

Process Sale:
A customer arrives at a checkout with items to purchase. The cashier uses the POS system to record each item. The system presents a running total and line-item details. The customer enters payment information, which the system validates and records. The system updates the inventory. The customer receives a receipt from the system and then leaves the store with the items.

Should Receipt be in the domain model?
Guidelines when to **include a candidate conceptual class that reports Information** into the domain model

- In general, it is not useful since all information is derived or duplicated from other sources.
- If it has a specific semantics w.r.t. the business, then it should be included.

**Should Receipt be in the domain model?**

A receipt is just a report of a sale and a payment…
Guidelines when to include a candidate conceptual class that reports information into the domain model

Should Receipt be in the domain model?
A receipt is just a report of a sale and a payment...

• Well, it depends....
  • if we just consider the Process Sale use case then receipt should not be part of the domain model; a receipt is just a report
  • if we consider a Handle Return use case then a receipt represents an important concept on its own
  • BTW, how about legal restrictions...?
Guidelines when to **include a description class** into the domain model

• A description class contains information that describes something else

Examples:

• a **product description** records the price, picture and text of an item

• a **flight description** contains information about the flight (number) and the source and target destinations.
Guidelines when to include a description class into the domain model

- A description class should be added to the domain model when:
  - There needs to be a description about an item or service, independent of the current existence of any examples of those items or services
  - Deleting instances of things they describe results in a loss of information that needs to be maintained, but was incorrectly associated with the deleted thing
  - It reduces redundant or duplicated information
When should I model something as an attribute or a class?

Rule of Thumb:

If we do not think of some conceptual class X as a number or text in the real world, X is probably a conceptual class, not an attribute.
When should I model something as an attribute or a class?

Let’s assume that we develop an airline reservation system. Should destination be an attribute of flight, or a conceptual class airport?

A destination airport is building at a specific place, it is not just a number or some text. Hence, it should be a conceptual class.

How about the name of the airport?
When should I **add an association to the domain model?**

**Rule of Thumb:**
Include associations in the domain model for which knowledge of the relationship needs to be preserved for some duration.

We are working on the conceptual model; we are not modeling associations at the software level.
When should I **add an association to the domain model?**

- When an association is among the common associations list:
  - A is a transaction related to another transaction B
  - A is a line item of a transaction B
  - A is a product or service for a transaction B
  - A is a role related to a transaction B
  - A is physical or logical part of B
  - ...

---

**Domain Modeling | 28**
When should I **add an association to the domain model**?

- E.g. the relation between a **Sale** and a **SalesLineItem** needs to be remembered
- However, it is not necessary to store the relation between a **Cashier** and a **ProductDescription** that he looks up
Name an association based on a ClassName-VerbPhrase-ClassName format where the verb phrase creates a sequence that is readable and meaningful.

- **Good examples:**
  - Player Is-on Square
  - Sale Paid-by CashPayment

- **Bad examples:**
  - Sale Uses CashPayment
    (Uses is usually generic and doesn’t tell us anything.)
  - Player Has Square
    (Has is usually generic and doesn’t tell us anything)
The **attributes** in a domain model should preferably be “primitive” data types.

- Very common data types include: Boolean, Date, Number, Character, String, Address, Color, Phone Number,…
- Consider modeling quantities as **classes** to associated units; e.g. the data type of the amount attribute of payment should indicate the currency

Use associations to model dependencies between conceptual classes; do not use attributes.
Consider defining a new data type class for something that is initially considered a string.

- If the string is composed of separate sections
e.g., phone number, name of person,...

- Different operations are associated with the string
e.g., social security number

- The string has other attributes

- The string is a quantity with a unit
e.g., money has a unit for currency

Avoid:

- ProductDescription
  itemId : ItemId

Recommended:

- ProductDescription
  ItemId
  id
  countryCode
Which are noteworthy domain concepts / domain objects?
The domain model serves as a source of inspiration for the design model.

By using the domain model as a direct inspiration for software classes, the representational gap between the domain concepts and the program is (relatively) small.