Enterprise System Integration (MTAT.03.229)

LECTURE 3: DOMAIN-DRIVEN DESIGN (DDD) APPROACH I - DOMAIN MODELING

MOHAMAD GHARIB
UNIVERSITY OF TARTU
Models and modeling
Domain models and domain modeling
Models and modeling

Models:

• A **model** is an abstraction of **reality**. Usually, **models** do not represent all aspects of **reality**.
• This allows for dealing with the real world in a simplified manner, avoiding the **complexity** and **danger** of reality.

Modeling:

• **Modeling** is a cost-effective use of **something** in place of **something** else for some purpose.
• **Modeling** allows for the use of something that is **simpler, safer**, and/or **cheaper** than **reality** instead of **reality** for some purpose.
Models and modeling

Modeling is supposed to represent a particular reference ("domain") for a particular purpose in a cost-effective (feasible) way [1].

Open questions:

- How to represent the reference ("domain")?
  - WHY we are modeling?
- How to define the purpose of the modeling?
  - WHAT are we modeling?
- How to guarantee it is cost-effective?
  - HOW the model can be of high-quality and still feasible to be used?

Rothenberg J, Widman LE, Loparo KA, Nielsen NR. The nature of modeling. in Artificial Intelligence, Simulation and Modeling. 1989. [Simple and very interesting paper about modeling].
Domain models

A domain model is a conceptual model for a specific domain.

A domain model aims at defining the key concepts of a specific domain as well as various relationships among these concepts.

A domain model uses high-level non-technical terms/concepts (e.g., almost every thing you can imagine).

A domain model is not a model of the software.
Domain modeling

A concept is an entity, a thing, an object or even an idea in the domain.

A domain concept can be represented as a conceptual class.

A domain concept/class is not a software object/class (e.g., Java classes).

This means, no operations can be in domain models, which only includes:

- conceptual classes and their attributes.
- relationships between these classes.
Domain models vs Software models

**Domain Model**

```
PlantCatalog
1 show 0..*
```

```
PlantEntry
Plant id:
Plant type:
Plant rent cost:
```
Domain models vs Software models

Domain Model

PlantCatalog

1 show 0..*

PlantEntry

Plant id:
Plant type:
Plant rent cost:

Software Model

<<interface>>
PlantCatalog

fetchAllPlantEntry()
findPlantById()
checkAvailability()

PlantEntry

Plant id: Number
Plant type: Type
Plant rent cost: Money

Derive
Domain modeling

Identifying concepts:
Analyze the *domain of interest* and try to identify *key terms*.

Usually, not all *key terms* are *concepts* but only a *subset* of them.

If the *system requirements* are available, they can be a *very good* source for identifying *key concepts*.

**How?**
Start by identifying all *nouns* and, then, determine which of them can be considered suitable *concepts*. 
Concepts identification – an example

RR1. The system should show available plants, availability status, price per day, etc.

RR2. The system should allow a customer to list the available plants and their prices.

RR3. The system should allow a customer to check the price for a given plant.

RR4. The system should allow a customer to check the availability of a given plant for a given time period.

RR5. The system should allow a customer to submit a PO for hiring a plant. The PO may be accepted or rejected depending on the plant’s availability.

RR6. The system should allow employees at Rentit to determine which plants need to be delivered on a given date.

RR7. The system should allow a customer to request modifying a PO request and accept/reject a modified PO depending on the plant’s availability.
Concepts identification – an example

RR8. The system should allow a customer to submit a request to cancel a PO. A cancellation request is normally accepted if the request arrives prior to the plant being delivered. If the plant has already been delivered, the cancellation request is rejected.

RR9. The system should allow employees at the plant depot to mark the plant as “rejected by customer”.

RR10. The system should allow employees at the plant depot to mark a plant as “returned”, meaning that the plant has been returned in due form and the rental period has expired.

RR11. The system should submit invoices for “returned” plants.

RR12. The system should submit payment reminders for unpaid invoices.
Concepts identification – an example

RR1. The **system** should show **available plants**, **availability status**, **price per day**, etc.

RR2. The **system** should allow a **customer** to list the **available plants** and their **prices**.

RR3. The **system** should allow a **customer** to check the **price** for a given **plant**.

RR4. The **system** should allow a **customer** to check the **availability** of a given **plant** for a given **time period**.

RR5. The **system** should allow a **customer** to submit a **PO** for hiring a **plant**. The **PO** may be accepted or rejected depending on the **plant’s availability**.

RR6. The **system** should allow **employees** at Rentit to determine which **plants** need to be delivered on a given **date**.

RR7. The **system** should allow a **customer** to request modifying a **PO request** and accept/reject a modified **PO** depending on the **plant’s availability**.
Concepts identification – an example

RR8. The system should allow a customer to submit a request to cancel a PO. A cancellation request is normally accepted if the request arrives prior to the plant being delivered. If the plant has already been delivered, the cancellation request is rejected.

RR9. The system should allow employees at the plant depot to mark the plant as “rejected by customer”.

RR10. The system should allow employees at the plant depot to mark a plant as “returned”, meaning that the plant has been returned in due form and the rental period has expired.

RR11. The system should submit invoices for “returned” plants.

RR12. The system should submit payment reminders for unpaid invoices.
Concepts identification – an example
Domain modeling

Identifying attributes:

• **Attributes** describe the *properties* of *concepts*.
• **Attributes** are needed to store *information* that an object may use.

Most **attributes** can be identified by asking:

• What *states* the *concept* can be in?
• What *information* do the *concept* need to store?
Attributes identification – an example
A concept or an attribute

An attributes describe the properties of concepts.
An attributes are needed to store information that an object may use.
An attribute should have values of a primitive type, e.g. integer, Boolean, enumeration, etc.
Customer is a potential attribute, but what about location?
**Customer** is a potential *attribute*, but what about *location*?

If *location* is represented as a string, it can be an attribute. If *location* is not a primitive type, it should be considered a concept.
Domain modeling

Identifying relationships:
When you define a *relationship*, you should keep in mind that such *relationship* should hold between the two *classes* all the time.
Relationships identification – example

Engineer
- Name:
- Works engineer

Site engineer
- modify
- approve/reject changeStatus

Plants
- Plant hire request
  - Request status:
    - Plant id:
    - Plant type:
    - Plant daily cost:
    - Project Name:
    - Project location:
    - Rent duration:
    - Start date:
    - End date:

Purchase Order
- Customer:
- PO status:
- Project location:
- Start date:
- End date:
- Plant rent cost:
- Plant id:

Plant catalog
- list
  - Plant item entry
    - Plant id:
    - Plant type:
    - Plant daily cost:
    - Availability:

Dispatch order
- Status:
  - Plant id:
  - Plant type:
  - Date:
  - Location:

Plant item
- represent
  - Plant item
    - Plant id:
    - Plant type:
    - Plant daily cost:
    - Availability:

Plant reservation
- Plant id:
- Start date:
- End date:

Pickup order
- Status:
  - Plant id:
  - Plant type:
  - Date:
  - Location:
Model refinement – an example
Thank You
for your attention

Mohamad Gharib
mohamad.gharib@ut.ee

unitartu

tartuuniversity