Modeling and metamodeling in Model Driven Development

On the difference between analysis and design models

Warsaw, May 14-15th 2009
Gonzalo Génova
ggenova@inf.uc3m.es
http://www.kr.inf.uc3m.es/ggenova/
Knowledge Reuse Group
Universidad Carlos III de Madrid

Structure of the seminar

What is a model:
syntax and semantics

On the difference between analysis and design models

What is a metamodel:
the OMG’s metamodeling infrastructure

Metamodeling directed relationships in UML
Sources

- Gonzalo Génova, Maricruz Valiente, Mónica Marrero
  - On the difference between analysis and design, and why it is relevant for the interpretation of models in Model Driven Engineering.
- Gonzalo Génova, Maricruz Valiente, Jaime Nubiola.
  - A Semiotic Approach to UML Models.

Purpose

- Clarify the confusions that lie around the widely used terms analysis model and design model in software engineering.
- Root of some difficulties that practitioners encounter in system modeling, and sometimes lead to bad engineering practices.
- Our approach consists of placing the duality of analysis and design within a three-dimensional modeling space.
- Models are classified according to:
  - First dimension: the reality they represent.
  - Second dimension: the purpose of the model.
  - Third dimension: the abstraction level expressed in the model.
- This classification facilitates:
  - interpretation of models, and
  - comprehension of model transformations as shiftings within this space.
Table of contents

1. Introduction
2. First dimension: Domain vs. System
3. Second dimension: Description vs. Specification
4. Third dimension: Abstract view vs. Concrete view
5. MDA concepts and the modeling space
6. Conclusions

Introduction
Introduction

- There are different kinds of models in software engineering:
  - Analysis and design models, structural and behavioral models, etc.
  - It is of major importance for MDE to understand the meaning of each kind of model, how they are related, and how they evolve.
- Roughly speaking…
  - **Analysis**: some kind of understanding of a problem or situation.
  - **Design**: creation of a solution for the analyzed problem.
  - **Model**: simplification that is used to better understand the problem (“analysis model”) or the solution (“design model”).
- Originally:
  - **Analysis/Decomposition**:
    - Breaking a whole into its component parts (in order to better understand it).
    - Opposed to **synthesis/composition**: “building a whole out of its parts”.
  - **Design**:
    - Drawing or making a blueprint of something before constructing it.
    - The design anticipates and guides the production process, the “synthesis”.
    - Design is part of (the preparatory phases of) synthesis.

Our initial concern

![Diagram](image)

Traditional structure of the software development process

- Which is the characteristic difference between AM and DM?
  - What does your boss mean when (s)he says: “you build the analysis model of the project, and you build the design model”.
- A single dimension is not enough:
  - Models are better characterized if placed in a 3D modeling space.
  - This traditional duality conveys really a triple difference that cannot be properly expressed through a single dimension, but requires three orthogonal dimensions.
  - Failing to acknowledge this triple difference leads to confuse the meaning of models, which has a practical relevance for the way models are interpreted and used in real software projects.
On the difference between analysis and design

The three dimensions

Orthogonal dimensions: independent ways to classify the models in the modeling space. Each one is related in its own way with the duality of analysis and design.

In each axis: two values corresponding to analysis (nearest to origin) and design (farthest).

Other (more or less orthogonal) dimensions are not necessarily related with the duality of analysis and design: “subject area”, “aspect”, “authorship”, “version”, etc.

First dimension: Domain vs. System
First dimension: Domain vs. System

- What does the model represent, i.e. which is the reality represented by it?
  - A software system, or some part of it.
    - “system model”, "software model", “machine model”
  - The application domain of the software system: that portion of reality that affects and is affected by the software system.
    - “model of the real world” (particularly inadequate), “model of the universe of discourse”, “business model" or even “domain model”.
- The contrast between the system and its application domain is often expressed with the terms “problem analysis” and “solution design”,
  - Related with the duality of analysis and design.
- The software system may include some degree of simulation of the application domain it provides a service to.
  - In general, no perfect correspondence between both models: the goal of the software system usually goes far beyond a simulation of its application domain, or even might not intend it at all.

When the System simulates the Domain

- A radar screen:
  - Simulates the positions and velocities of aircrafts in a certain aerial space.
  - The degree of accuracy in the simulation is extremely important.
- Role playing games:
  - Model imaginary worlds.
  - There is no proper simulation of any external world.
- An electronic voting system:
  - Contains some degree of simulation of its environment: voters, choices, etc.
  - However, the system is designed so that electronic votes do not simulate, but replace, manual votes, which will cease to exist once the system is deployed.
  - Besides, the new electronic votes will not be a mere simulation of the old manual votes; instead, they will have different features (for example, a different way to authenticate the voter, or the possibility to change the vote before some specified deadline, and so on).
- In general, the software system will create a new reality that was not present.
Shared concepts in System model and Domain model

- **Shared concepts:**
  - To be found in the part of the system that simulates the domain (rest unshared).
  - Represent different realities in each model: an entity in the domain / in the system.
  - Shared concepts do not constitute a full model of the domain, nor of the system.

Description of a voting system

1. Each poll has a **title** and a **date**, and it offers a number of different **questions**, each question having a **text** that expresses it.
2. A number of **voters**, each one with a **name**, can optionally participate in each poll.
3. In each poll, each voter selects one of the possible **answers** to each question.
4. Voters can have opinion exchanges among them in order to **influence** each other before selecting their answers.

- What is this description about?
  - Does it describe the application domain, or the software system, or both?
- **The description itself does not tell it.**
Description of a voting system: Domain vs. System

1. Each poll has a title and a date, and it offers a number of different questions, each question having a text that expresses it.
2. A number of voters, each one with a name, can optionally participate in each poll.
3. In each poll, each voter selects one of the possible answers to each question.
4. Voters can have opinion exchanges among them in order to influence each other before selecting their answers.
5. A voter can maintain provisional answers after the opening date and before the closing date of the poll; each answer becomes definitive when the voter marks it as such, otherwise it is discarded on the closing date of the poll.

- Opinion exchanges not implemented (req. 4, only in the domain).
- Provisional and definitive answers added (req. 5, only in the system).

Domain vs. System: Is this Analysis vs. Design?

- An important decision: which part of the domain will be automated (and which not).
- Modeling the domain before deciding the requirements for the new system can help.
Second dimension: Description vs. Specification

- What is the model used for? Two different ways:
  - **Forward engineering** - specification models
    - A specification of something that must exist.
    - An anticipation or specification of the software system to be built.
    - A template to guide the construction of the system, as a platform to simulate the behavior of the system before actually constructing it, even as a starting point to (semi)automatically generate the system, etc.
  - **Reverse engineering** - description models
    - A description of something that exists.
    - A conceptual tool or description to understand an existing system that has to be maintained or improved.
  - **Scale models** used in many branches of engineering:
    - Model-as-copy: reverse engineering.
    - Model-as-original: forward engineering.
- Compare to the use of electrical engineers (“circuit analysis”, “circuit design”).
  - This is close to the original or **classical sense** of “analysis” and “design”.
  - But it is not the **most usual sense among software engineers**!
Reverse and Forward Engineering: Is this Analysis vs. Design?

<table>
<thead>
<tr>
<th>Reverse engineering</th>
<th>Forward engineering</th>
</tr>
</thead>
<tbody>
<tr>
<td>system → model</td>
<td>model → system</td>
</tr>
<tr>
<td>description</td>
<td>specification</td>
</tr>
<tr>
<td>model-as-copy</td>
<td>model-as-original</td>
</tr>
<tr>
<td>analysis model</td>
<td>design model</td>
</tr>
<tr>
<td>process of analysis</td>
<td>process of synthesis</td>
</tr>
</tbody>
</table>

- Reverse engineering is a **process of analysis** where the existing system is understood by means of a **model-as-copy** (result: “**analysis model**”).
- Forward engineering is a **process of synthesis** in which the system is constructed starting from a **model-as-original** (start: “**design model**”).

Two ortogonal dimensions

<table>
<thead>
<tr>
<th>Specification</th>
<th>domain specification model</th>
<th>system specification model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>domain description model</td>
<td>system description model</td>
</tr>
<tr>
<td>Domain</td>
<td>System</td>
<td></td>
</tr>
</tbody>
</table>

- The distinction can be applied also to the **domain model**:
  - **Domain description models**: to understand the context of requirements, build a vocabulary.
  - **Domain specification models**: if the application domain is not an immutable reality.
- An application domain that will remain completely unaffected by the introduction of a new software system is **hardly conceivable**.
- However, specifying the new domain is **very often ignored** in software projects.
Description vs. Specification: Is this Analysis vs. Design?

- Consider both models referred to the Domain, or else to the System.
  - Description of the current domain, specification of the future domain.
  - Description of the current system, specification of the future system.

How to avoid a pernicious confusion

- A typical software project should include at least these two models:
  - **DDM** (domain description model) → **SSM** (system specification model).
  - Different models: different represented realities, and different purpose.
- What does your boss mean when (s)he says: “build the analysis model of the project”.
  - **DDM**: understanding the real world (“analysis model” in the classical sense).
  - **SSM**: high level system specification (“analysis model” in the SE sense).
- Too often the distinction between these two models is simply ignored.
  - A single “analysis model” where it is not very clear what it represents, nor its purpose.
  - “Include the model of the domain as a part of the model of the system”.
  - Uncritically assuming that the whole domain is simulated by the system is pernicious.
- **Understanding the real world**, i.e. analyzing it (in the classical sense of the word), is most useful to produce a good user requirements specification.
  - Often reflected in the fact that the SSM uses the concepts found in the DDM.
  - The difference is subtle, but real: the same vocabulary, two different realities represented, two different purposes. Maybe this is the source of the confusion we try to avoid.
- A **model-as-copy of the real world** (legitimate and useful) must be carefully distinguished from a **model-as-original of the future system**.
On the difference between analysis and design

Third dimension: Abstract view vs. Concrete view

- How is reality represented in the model, i.e. which abstraction level?
- In the context of software systems forward engineering:
  - **Analysis**: the what. Capture user requirements, omit technology-dependent details, use concepts from the problem domain.
  - **Design**: the how. Define a software solution that satisfies the requirements. New artifacts, concrete technological platform.
- Often expressed as:
  - Logical design vs. physical design.
  - Specification model vs. implementation model.
- Both kinds of models do represent the same system, with the same purpose:
  - Both are **system models**, not domain models.
  - Both are used in the context of a **forward engineering** process.
- But an important difference in **perspective** about the system-to-be-built.
  - Analysis model represents the external, higher level or logical view of the system (the more abstract **conceptual view, black box model**).
  - Design model represents the internal, lower level or physical view (the more concrete **implementation view, white box model**).
Analysis: a first step in synthesis

- Two senses of analysis:
  - The **classical sense**: something dually opposed to synthesis, different processes.
  - The **SE sense**: a first step in synthesis, followed by design, same process.

- Examples of models created during the **analysis phase**:
  - **Use case model**: model the system as it interacts with the outer world.
    - High level character, defined within the “analysis” activity of a project.
    - Specification of the expected functionality a software system must provide, described through typical user-system interactions
    - Does not represent interactions among actors, not a complete business model.
    - Use cases model a software system to be built, not at all the “real world” as it is before the system exists, or as it will be afterwards.
  - **User requirements**: a textual model-as-original of the required **system**.
    - Specifications of a desired computer information system, not aimed at describing the world outside of the computer.
    - Originate in the environment (the problem domain), but they are about the system (the solution domain).

---

Abstract vs. Concrete: Is this Analysis vs. Design?

- Both models express the same electronic voting system at different abstraction levels
  - Two associations have been replaced by intermediate classes.
  - A rather **gradual transformation**, many intermediate levels (gray boxes)
  - There is **no sharp distinction between analysis and design**, if they are understood as system specifications at different levels of abstraction.
The transition from analysis to design

- Do not confuse amount of detail with abstraction level.
  - The key is technology independence / dependence.
- The transition from AM (logical) to DM (physical) can be hard.
  - The design has to provide a creative solution for the problem specified in the analysis, and this rarely will be easy.
  - A good DM that takes into account non-functional requirements such as performance, reuse, maintainability, etc., might hardly resemble the AM specifying the same system.
- But the key point is:
  - The transition neither changes the represented reality (first dimension) nor the model's purpose (second dimension).
  - Both models are, within a synthesis process, specification models of the same software system, yet at a different abstraction level.

Three orthogonal dimensions

<table>
<thead>
<tr>
<th>Specification</th>
<th>Abstract view</th>
<th>Concrete view</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract view</td>
<td>domain specification abstract model</td>
<td>system specification abstract model</td>
</tr>
<tr>
<td>Concrete view</td>
<td>domain specification concrete model</td>
<td>system specification concrete model</td>
</tr>
</tbody>
</table>

- Domain models, either descriptive or specificative, can be abstract or concrete.
  - Example: getting a mortgage from a bank at different levels of abstraction.
  - “Concrete” does not mean “technology dependent”, but it is still meaningful.
- Any combination of “coordinates” makes sense, although actually some of them are less frequently used in typical software engineering processes.
  - system-description-concrete view: understanding details of the current system.
  - domain-specification-concrete view: modifying details of the business.
**MDA concepts and the modeling space**

Two typical trajectories:
- RW model (analysis model in the classical sense) to requirements analysis (analysis model in the SE sense), to design model.
- Legacy system concrete description to legacy system abstract description, to new system abstract description, to new system concrete description.
A trajectory through the three dimensional modeling space

First jump (1st and 2nd dim.): from DDAM to SSAM.

Second jump (3rd dim.): from SSAM to SSCM.

Both A and B are called “analysis model”, C is called “design model”.

On the difference between analysis and design

CIM-PIM-PSM vs. abstract / concrete dimension

- **CIM** (analysis, the What):
  - “A view of a system from the computation independent viewpoint”.
  - “A model of a system that shows the system in the environment in which it will operate”.
  - May include external agents, but aimed at modeling the system, not the external agents.
  - Unfortunately, nearly equated to “domain model” or “business model”.
  - Does it represent the domain, or else a very abstract view of the system?
  - What is a domain/business model? Shared concepts only?

- **PIM** and **PSM**:  
  - PIM and PSM implement the CIM: are PIM and PSM design models? 
  - PIM-PSM mapping is “refinement of designs”: PIM is also design. 
  - PIM-PSM parallel to A-D: PIM is analysis (first step in synthesis?). 
  - Fit in the third dimension: computation independence and platform independence are not orthogonal dimensions.
Conclusions

Results of the inquiry

<table>
<thead>
<tr>
<th>Answer</th>
<th>Questionnaires</th>
<th>Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>The analysis model represents the application domain, the design model represents the software system.</td>
<td>2  2  3  3  0  1  3  2  2  3</td>
<td>2,1</td>
</tr>
<tr>
<td>The analysis model describes something that exists, the design model specifies something we want to build.</td>
<td>3  3  3  1  3  2  2  1  1</td>
<td>2,2</td>
</tr>
<tr>
<td>The analysis model abstracts away low-level details, the design model gives all necessary details about the represented thing.</td>
<td>1  2  3  3  2  0  1  2  1  2</td>
<td>1,7</td>
</tr>
</tbody>
</table>

- Average is rather meaningless, do not look at it too much.
- Look at the differences in the questionnaires.
  - So few people, so different answers.
Ambiguity of the term “analysis model”

- **Two different meanings** of analysis models among software engineers:
  - Specifying the abstract or logical view of a software system (*the SE sense*).
  - Describing the RW context of the desired software system (*the classical sense*).
- Both views are very often confused:
  - **Moderate danger**: believe that we are modeling the real world, when we are really doing a high-level specification of the software system.
  - **Serious danger**: build a system that needlessly matches the structure of the RW.
    - The description of the RW cannot be used as a specification of the system.
    - The software system simulates the application domain in a limited way.
    - No machine can replace the stakeholders in the task of deciding which part of the domain must be simulated, and which part must not (no automatic).
    - Taking an analogy from civil engineering, a model of cars and rivers cannot be a model of the bridge the cars need to cross over the river.
- Analysis can be conceived as a task where **both models are built**, in a close relationship but properly distinguished at the time (avoid misunderstandings).

Can we use a model of the real world as a model of the system?
A better understanding of transformations

- **Analysis to design**: not a quasi-automatic, seamless transition.
- Advantages of unfolding the duality of A&D onto the 3D modeling space:
  - Avoid the **frequent misunderstandings** that hinder the use of models.
  - Better define model transformations as **shiftings within this space**.
- To automate model transformations we need...
  - A set of transformation rules (mappings) along each dimension.
    - First dimension: the **reality represented** in the model (domain vs. system).
    - Second dimension: the **purpose** of the model (description vs. specification).
    - Third dimension: the **abstraction level** of the model (abstract vs. concrete).
  - A good understanding of the dimension(s) involved in the transformation.
    - How can we judge the **correctness of a model** if we do not know what it is supposed to mean, what for, and how?
- Moving over the **first and second dimensions** is conceptually simple, but requires rules that generally will neither be simple nor automatable.
- Most of the efforts in **MDE tools** research have been devoted to the **shifting along the third dimension**, since automation seems to be a promising fruit at hand in this field.