Towards the Refinement of Topological Class Diagram as a Platform Independent Model

Uldis Donins, Janis Osis, Armands Slihte, Erika Asnina, and Bernards Gulbis

This work has been supported by the European Social Fund within the project «Support for the implementation of doctoral studies at Riga Technical University»

MDA&MDSD Workshop 2011, June 9-10, 2011 - Beijing, People’s Republic of China
Introduction (1)

- Transformation from model to model takes significant place in Model Driven Architecture (MDA)
- MDA considers system from three viewpoints: computation independent, platform independent, and platform specific
- Despite the fact that each viewpoint has its own representing model, the transformation between CIM and PIM is fuzzy
  - Reason of this might be that there is an opinion that requirements modeled in CIM often lack a good structure and therefore it is not possible to automate the CIM-to-PIM transformation
- Lack of traceability within the CIM and lack of transformation from CIM to PIM leads in manual CIM-to-PIM conversion
  - manual conversion depends much on designers’ personal experience and knowledge and therefore the quality of PIM cannot be well controlled
- The structure quality of CIM can be improved by using Topological Functioning Model (TFM) as model to represent the CIM
Topological modeling approach

Informal description of system functioning/
Software requirements/
Use cases

Definition of Topological functioning model

Topological functioning model

1-to-1 transformation

Problem domain object graph

Abstraction/Consolidation

Objects, relationships, operations, and attributes

Classes, relationships, operations, and attributes

Topological class diagram

CIM level

PIM level
Introduction (2)

- By following the guidelines of PIM creation by means of topological class diagrams, the initial topological class diagram gets developed.
- This topological class diagram can be considered as initial, because it shows only the classes and topological relations between them.
  - The topological relations between classes show the control flow within a system.
  - Topological relation is a strong relation because it exists between functional features, objects and even classes.
- In spite of having topological relations between classes, a refinement of topological class diagram should be performed in order to find and define generalized classes, structural relationships, enumerations, and provided and required interfaces.
- The main goal of this research is to provide guidelines for topological class diagram refinement.
Topological Functioning Model

- Topological Functioning Model (TFM) is a system functioning description embedded in topological space in form of a directed graph $G(\mathcal{X}, \Theta)$ taking into consideration topological and functional features
  - $\mathcal{X}$ – functional features
  - $\Theta$ – topology (cause and effect relations) between functional features

- TFM has a strong mathematical base
  - Topological characteristics of TFM – connectedness, closure, neighborhood, and continuous mapping
  - Functional characteristics of TFM – cause-effect relations, cycle structure, and inputs and outputs
Topological Class Diagram

- Class diagrams reflect the static structure of the system, and with the help of class diagrams it is possible to model objects and their operations involved in the system.

- Regardless of the opportunities provided by the class diagrams, it is not possible to reflect the cause and effect relation within a system because class diagram does not contain relation type to model topological relations.

- To allow to define cause and effect relations between classes a new type of relationship in class diagrams is introduced: **topological (or cause and effect) relationship**

- The class diagram which shows topological relationships among classes is called a **Topological Class Diagram**.

- Topological class diagrams also improves formalism level of class diagrams because between classes now are precisely defined relations.

- Notation used for representing topological relationship is directed line with filled triangular arrowhead pointing to effect class, the opposite end (without arrowhead) points to cause class:
Refinement Approach

- By reviewing and refining initial class diagram, associations, generalizations, dependencies, and other relationships defined in UML are added. Refinement process consists of following steps:
  - Identify generalizations (basing on topological relationships, attributes, operations, and responsibilities),
  - Define interfaces (both provided and required),
  - Identify structural relationships between classes (aggregations, compositions, and associations),
  - Identify enumerations,
  - Check for additional relationships (such as dependencies and realizations), and
  - Revise topological class structure

- These refinement process steps will be discussed in detail in following slides

- As a result of applying refinement process, a rich topological class diagram with lower abstraction level is obtained
Example: Laundry Business System

«(..) When a person **arrives** at laundry he **gets checked** by clerk. All laundry **clients** are **registered**. **Registration is done** by the clerk. Any person, who **is registered** in the laundry **client register** and who **has filled out client card** is considered as a client. If the person is not a registered client yet, the clerk **performs** the client registration. If the client does not have the **client card yet**, the clerk **makes** it anew. The **client card bears** the information about the client (**name, surname and address**) and every client card has its own **unique identification number**. Registered **clients with client card** have the **right to use** the services provided by laundry. (..)»

During informal description analysis nouns are denoted by *italic*, verbs are denoted by **bold**, and action preconditions (or postconditions) are **underlined**
Example: Topological Functioning Model

- Topological Functioning Model (TFM) is a system functioning description embedded in topological space in form of a directed graph $G(X, \Theta)$ taking into consideration topological and functional features
  - $X$ – functional features
  - $\Theta$ – topology (cause and effect relations) between functional features
- TFM has a strong mathematical base
  - Topological characteristics – connectedness, closure, neighborhood, and continuous mapping
  - Functional characteristics – cause-effect relations, cycle structure, and inputs and outputs
Example: Problem Domain Objects Graph

ClientRegister.
CheckPersonalData()
:ClientRegister.
RegisterClient()
:ClientRegister.
RegisterEMailAddress()
:ClientCard.
Create()

Invoice.
Create()
:Linen.
GiveToClient()
:ClientRegister.
MarkAsIssued()
:ClientRegister.
Authorize()
:ClientCard.
Issue()

:OrdersList.
SendNotification()
:Invoice.
Issue()
:Invoice.
Pay()
:Receipt.
ProvideToClerk()

:OrdersList.
CheckOrderStatus()
:OrdersList.
PrepareNotification()

:OrdersList.
TakeOutOrder()
:Linen.
Receive()
:Linen.
Transfer()

:LinenRegistrationForm.
RegisterPONumber()
:LinenRegistrationForm.
RegisterWeight()
:Linen.
MarkLaundryMachine()
:Order.
UseLaundryMachine()

:OrdersList.
Sort()
:PurchaseOrder.
Create()
:Receipt.
Issue()

:PurchaseOrder.
AssignNumber()
:Receipt.
Create()
:LinenRegistrationForm.
AnnounceNearestDueDate()

:OrdersList.
RegisterOrder()
:LinenRegistrationForm.
AssignUniqueID()
:LinenRegistrationForm.
RegisterOrder()
:LinenRegistrationForm.
CalculatePrice()
:LinenRegistrationForm.
CheckDueDate()
Example: Initial Topological Class Diagram
1st step: Identifying Generalizations

- The generalizations can be identified in two ways:
  - The first way is to review initial topological classes which are obtained from the TFM
    - To find a generalization you need to look for the same responsibilities, topological relationships, attributes, and operations that are common to two or more classes.
    - The set of common responsibilities, topological relationships, attributes, and operations can be elevated to a more general class (if this general class does not exist it can be created)
  - The second way is by doing additional interviews with stakeholders

- By applying both ways in generalization identification a more formal (by reviewing initial topological classes) and less formal (by making interviews) approaches are used
- By using together reviewing and interviewing an additional model checking gets performed
Example of Identified Generalizations

a) LinenRegistrationForm
   Invoice
     +UniqueID
     +Price
     +IssueDate
     +PaymentDate
     +Create()
     +Issue()
     +Pay()

   Receipt
     +UniqueID
     +DueDate
     +Price
     -Weight
     +IssueDate
     +Create()
     +Issue()
     +ProvideToClerk()

   OrdersList

b) Document
   +UniqueID
   +Price
   +IssueDate
   +Create()
   +Issue()

   Invoice
     +PaymentDate
     +Pay()

   Receipt
     +DueDate
     -Weight
     +ProvideToClerk()

   OrdersList

Topological relationships
2nd step: Defining Interfaces for Collaboration with Environment

- We can draw a line around the topological class diagram which is obtained by applying transformations on the TFM, thus showing the boundary of the system under consideration.
- The next step is to identify the operations and the signals that cross this boundary.
- These operations and signals can be found by analyzing both the TFM and the topological space of the system.
  - The TFM shows the functioning of the system.
  - Topological space shows the system within the (surrounding) environment.
- This analysis shows the inputs and outputs of the system.
  - **Input functional features** within TFM indicate **provided interfaces**.
  - **Output functional features** indicate the **required interfaces**.
Example of Identified Interfaces

- **Accounting Interface**
  - `+ReceiveInvoice()`
- **Person Interface**
  - `+CheckPersonalData()`
- **Client Register**
  - `+RegisterClient()`
  - `+Authorize()`
  - `ClientRegister`
- **Invoice Interface**
  - `+Create()`
  - `+Issue()`
  - `+Pay()`
3rd step: Identifying Structural Relationships

- At first it is needed to check and find the whole and part relationships – aggregations and compositions:
  - Aggregation can be placed between objects if a part object can belong to more than one whole object and the part continues to exist when the whole is destroyed
  - Composition can be placed between objects if a part is totally “owned” by the whole and the part is destroyed when the whole is destroyed

- The next step is identification of associations between classes:
  - Associations can be placed between objects if it is needed to navigate from objects of one type to objects of another
4th step: Identifying Enumerations

- The enumeration within a system can be found in two ways:
  - Review initial topological classes
    - To find enumerations at first you need to look for attributes which can contain only a restricted set of values. The second thing is to search for objects which can change its state value during its lifetime
  - Additional interviews with stakeholders
    - During the interviews the interviewees are asked if any of the attributes has only limited list of allowed values or if there exist a states of things involved into system
5th step: Checking for Additional Relationships

- The checking of additional relationships includes identification of dependencies and realizations.
  - Dependency relationship should be used to show that one thing is using another:
    - Most often dependencies between classes are used to show that one class uses operations from another class, or it uses variables or arguments typed by the other class, or to show required interfaces of a class.
  - Realization is used in two circumstances: in the context of provided interfaces and in the context of collaborations.
6th step: Revising Topological Class Structure

- The revising of topological class structure should be done using following guidelines:
  - Checking relationships between generalized classes and specialized classes.
  - Checking for additional generalizations
  - Every generalized class in the topological class diagram should be justified
  - As a final revising step of generalized classes is that each generalized class should have at least two specializations, with two exceptions:
    - The generalized class is concrete.
    - It is anticipated that more specializations will be added in the future.
  - Since the system is connected with the environment (through inputs and out-puts), at least one provided and one required interface should be identified

- After the revising process has been finished, the initial topological class diagram is refined

- Mainly initial topological class diagram should be refined in order to introduce generalized classes, structural relationships, and interfaces as seams between systems
Conclusions (1)

- Software development in topological modelling context begins with problem domain formalization in the form of TFM
- Once the TFM has been created, functional requirements can be validated against it
  - By doing this validation we get checked both TFM and functional requirements
- By applying transformations to the developed TFM we can obtain both dynamic and static representations of the system
- The most noticeable aspect is that classes and topological relations are identified in formal way by modelling problem domain with TFM
  - in contrast – in traditional software development scenario relations (mostly associations and generalizations) between classes are defined by the modeller’s discretion
- In addition this initial diagram can be refined in order to obtain associations, generalizations, dependencies and other artefacts included in UML class diagram
Conclusions (2)

- The benefit of applying topological modelling approach is that software development is done formal since the very beginning of its lifecycle.
  - Thus the quality level of software development process and software itself is elevated and traceability between different artefacts at different abstraction levels can be established.
- The largest drawback is that at the moment of implementing this case study there are no tool support for TopUML.
  - To eliminate this drawback one of the feature research and work directions is to create full specification of TopUML profile and to develop a tool which supports TopUML.