A Guide to Efficient Requirements Modeling

Introduction – Who Am I?

Name
- Dr. Darius Šilingas

Job
- Principal MagicDraw UML Trainer/Consultant

Contacts
- Skype: darius.silingas
- E-mail: darius.silingas@nomagic.com
- Mobile: +37068612748

Professional background
- Working in professional software development since 1998
- Started as Java/J2EE Programmer, gradually changed the role into System Architect, Requirements Analyst, and since 2004 into Trainer/Consultant focusing on modeling with UML
- Taught >100 industrial classes on modeling with UML, requirements analysis, business process modeling in 15 countries
- OMG-Certified UML Professional Advanced
Class Contents – Why Should You Attend this Class?

Mapping requirements artifacts 2 UML concepts

Requirements modeling workflow

Examples of UML requirements model artifacts
  - Based on MagicTest case study

How to simplify / customize UML modeling environment for requirements analysts

➔ You will learn some ideas how to represent your requirements models concisely in proper UML!

Some Facts About UML (*Unified Modeling Language*)

Developed by OMG (*Object Management Group*)
  ➔ [www.uml.org](http://www.uml.org)

Considered *de facto* standard in software modeling

UML 1.0 released in 1997
Current version – UML 2.1.2

Defines 248 interrelated metaclasses representing modeling concepts
Defines 13 “official” diagram types
UML 2.1.2 specification contains 945 pages

A very powerful language, but too complicated for a typical modeler!
Some Facts About Requirements Analysis

Considered to be one of the most important software development activities

State of the art – textual specifications with explanatory diagrams

Both customers and developers prefer graphical requirements
  - Based on professional experience
  - “A picture is worth a thousand words”

Most popular software requirements books use a lot of non-UML diagrams as requirements artifacts

Requirements analysts do not make use of the potential of UML models!
Some Myths About Applying UML to Requirements Analysis

Requirements analyst should apply UML only for use case modeling

UML is not suitable for user interface modeling

UML does not allow to model data flows

UML does not have capabilities for modeling business rules

Class diagram is too complicated for business or system analyst

Concept vocabularies should be captured in table-based text

Mapping Requirements Artifacts 2 UML Concepts (1)

<table>
<thead>
<tr>
<th>Requirements artifact</th>
<th>UML metaclass</th>
<th>Stereotype</th>
<th>Recommendations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business concept</td>
<td>Class</td>
<td>-</td>
<td>Attributes and operations should be hidden in diagrams.</td>
</tr>
<tr>
<td>Business concept</td>
<td>Association</td>
<td>-</td>
<td>Typically modelers specify association name and role multiplicities or roles names, multiplicities, and navigability.</td>
</tr>
<tr>
<td>Business object</td>
<td>State Machine</td>
<td>-</td>
<td>Should be nested within business concept.</td>
</tr>
<tr>
<td>Business object lifecycle</td>
<td>State Machine</td>
<td>-</td>
<td>Should be nested within business concept.</td>
</tr>
<tr>
<td>Business goal</td>
<td>Use Case</td>
<td>Goal</td>
<td>All of these elements should be placed in a specific Package or Model element, which is dedicated for business modeling.</td>
</tr>
<tr>
<td>Business role</td>
<td>Actor</td>
<td>Role</td>
<td>All of these elements should be placed in a specific Package or Model element, which is dedicated for business modeling.</td>
</tr>
<tr>
<td>Business process</td>
<td>Activity</td>
<td>Process</td>
<td>All of these elements should be placed in a specific Package or Model element, which is dedicated for business modeling.</td>
</tr>
<tr>
<td>Business task</td>
<td>Action</td>
<td>Task</td>
<td>All of these elements should be placed in a specific Package or Model element, which is dedicated for business modeling.</td>
</tr>
<tr>
<td>Business rule</td>
<td>Constraint, Guard</td>
<td>-</td>
<td>Might be specified formally using OCL or informally with simple text.</td>
</tr>
<tr>
<td>Business fact</td>
<td>Instance</td>
<td>Specification, Comment</td>
<td>Data facts can be defined using instance specifications in object diagram. Other types of business facts can be captured using simple comments.</td>
</tr>
</tbody>
</table>
### Mapping Requirements Artifacts 2 UML Concepts (2)

<table>
<thead>
<tr>
<th>Requirements artifact</th>
<th>UML metaclass</th>
<th>Stereotype</th>
<th>Recommendations</th>
</tr>
</thead>
<tbody>
<tr>
<td>System</td>
<td>Component</td>
<td>-</td>
<td>The component label may be hidden.</td>
</tr>
<tr>
<td>Document (information) form</td>
<td>Class</td>
<td>Document</td>
<td>Implementation specific information such as visibility, derivation should be hidden.</td>
</tr>
<tr>
<td>Document (information) sample</td>
<td>Instance Specification</td>
<td>Sample</td>
<td>Document should be created</td>
</tr>
<tr>
<td>Information flow</td>
<td>Information Flow</td>
<td>-</td>
<td>Information flow should be mapped on associations between classifiers, e.g. classes, components, activities</td>
</tr>
<tr>
<td>User group</td>
<td>Actor</td>
<td>-</td>
<td>Can be grouped into primary, secondary, system, and pseudo.</td>
</tr>
<tr>
<td>User task</td>
<td>Use Case</td>
<td>-</td>
<td>Extensions might be used for documentation.</td>
</tr>
<tr>
<td>Usage scenario</td>
<td>Activity, Interaction</td>
<td>-</td>
<td>Should be placed inside use case. Once use case may have multiple scenarios.</td>
</tr>
</tbody>
</table>

### Mapping Requirements Artifacts 2 UML Concepts (3)

<table>
<thead>
<tr>
<th>Requirements artifact</th>
<th>UML metaclass</th>
<th>Stereotype</th>
<th>Recommendations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-functional requirement</td>
<td>Class</td>
<td>Requirement</td>
<td>Stereotype tags or stereotypes should be used for specifying different categories of requirements.</td>
</tr>
<tr>
<td>Time constraint</td>
<td>Duration Constraint</td>
<td>-</td>
<td>Can be visualized in sequence or timing diagrams</td>
</tr>
<tr>
<td>GUI navigation schema</td>
<td>State Machine</td>
<td>-</td>
<td>State represents working in a specific screen, transition – navigation between screens, trigger – GUI event.</td>
</tr>
<tr>
<td>GUI prototype</td>
<td>Structured Class</td>
<td>-</td>
<td>Composite structure diagram can be used for abstract prototyping.</td>
</tr>
<tr>
<td>Refinement</td>
<td>Abstraction</td>
<td>-</td>
<td>Can be used for generating relationship matrices</td>
</tr>
<tr>
<td>Composition</td>
<td>-</td>
<td>-</td>
<td>Can be modeled by nesting requirements.</td>
</tr>
</tbody>
</table>
Using UML in Software Workflows

Domain concepts and relations
Domain object lifecycle
Business processes
Actors and use cases
Use cases scenarios

Package/component structure
Interaction scenarios
Data structure
Service API
GUI navigation schemas
GUI structure prototypes

Model transformations
Code generation from UML
Visualization of code structure

Requirements Levels: Adopted Cockburn Requirements Ship

Vision


Non-Functional Req.  Non-Functional Req.
Requirements Modeling Workflow (1): Domain Analysis

1. Identify Domain Concepts and Relations
   - Concepts Map (Class)
2. Model Business Object Lifecycle
   - Object Lifecycle (State Machine)
3. Identify Business Roles and Processes
   - Process Map
4. Model Workflow
   - Business Process (BPMN)
   - Define IT System Context

Requirements Modeling Workflow (2): End-User Requirements

1. Define IT System Context
   - System Context (Information Flows)
2. Identify Use Cases
   - Use Case Model (Use Case)
3. Model Use Case Scenarios
   - Use Case Behavior (Activity)
4. Define Information Structure
Case Study: Test Assessment System MagicTest

MagicTest is dedicated for automated students' skills evaluation. MagicTest provides functionality for managing questions, creating tests, and taking tests. MagicTest supports two types of users: teacher and student. MagicTest is integrated with university registry system MagicUniversity for getting available university courses, current classes, teachers’ and students’ profiles. A teacher can create and manage questions and tests. A question is assigned to the subject of one or more courses. A teacher can compose a test, which includes a selected subset of questions available for a course. A test is always bound to only one course, but can be assigned for multiple classes. Three types of questions are supported: single-choice, multiple-choice, and free-response questions. Both course questions and selectable answers can be presented as text and/or graphical image. A test has the following attributes: a title, the author (a teacher who created it), active period for taking it, a list of included questions, time limit, option if pausing a test taking is allowed, and instructions. Test cannot be activated for taking while it is under modification. The author should explicitly confirm a test as prepared. Only prepared test confirmed by the author are activated automatically when test-taking period starts. When test-taking period ends or all assigned students have already taken the test, the test is marked as expired and is no longer available for taking. When a test becomes active MagicTest notifies students of assigned classes by email. Students can take tests using MagicTest system. When student ends taking a test, MagicTest automatically calculates and displays test evaluation. If the test time runs out, test taking is automatically terminated. When a test expires, MagicTest automatically calculates test assessment statistics.
### Requirements Model Artifact: Domain Concepts Map

![Diagram showing relationships between Teacher, Test, Assessor, Question, Course, Class, Student, and Test Assessment.](image)

### Getting Textual Version of Domain Vocabulary

<table>
<thead>
<tr>
<th>Concept</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Answer</td>
<td>Student's input answering to a particular question included in the test.</td>
</tr>
<tr>
<td>Class</td>
<td>A particular running of a discipline, which is taught by a teacher to a number of students who have registered for it.</td>
</tr>
<tr>
<td>Course</td>
<td>A discipline, which could be taught multiple times.</td>
</tr>
<tr>
<td>Question</td>
<td>A statement asking for the answer in order to assess student's knowledge or skills based on the course contents. Each question should be applicable to at least one course.</td>
</tr>
<tr>
<td>Student</td>
<td>A member of university who aims to get a qualification degree by participating in a number of classes.</td>
</tr>
<tr>
<td>Teacher</td>
<td>An employee of the university who is responsible for supervising courses and teaching classes. A teacher can create questions and compose tests that are assigned for one or more of his classes.</td>
</tr>
<tr>
<td>Test</td>
<td>A setup of the test that includes the period in which the test is active. i.e. available for assessments, the classes for which it is assigned, a selection of questions (all of them should be applicable to the course that is instantiated by the assigned classes), instructions and some other properties.</td>
</tr>
<tr>
<td>Test Assessment</td>
<td>A particular trial of the student to take the test including the answers the questions defined in the test, start and end time, and the evaluation.</td>
</tr>
</tbody>
</table>
**Requirements Model Artifact: Domain Object Lifecycle (Test)**

- **Create**
- **Draft**
  - Edit
  - Confirm
- **Complete**
  - Entry / notify students
  - Exit / calculate statistics
  - Cancel [no test assessments]
  - At (active period start)
- **Active**
  - Entry / notify students
  - Exit / calculate statistics
  - At (active period end)
- **Expired**
  - After (10 years)

**Requirements Model Artifact: Business Process Map**

- **<<businessArea>>**
  - **Class Management**
    - **<<businessProcess>>** Schedule Class
    - **<<businessProcess>>** Register Students to Class
    - **<<businessProcess>>** Run Class
    - **<<businessProcess>>** Run Lecture
    - **<<businessProcess>>** Run Lab
    - **<<businessProcess>>** Cancel Class
    - **<<businessProcess>>** Evaluate Class
    - **<<businessProcess>>** Run Student Assessment
**Requirements Model Artifact: Business Process (Run Class)**

A man is flying in a hot air balloon and realizes he is lost. He reduces height and spots a man down below. He lowers the balloon further and shouts, "Excuse me, can you tell me where I am?"

The man below says, "Yes, you're in a hot air balloon, hovering 30 feet above this field. " "You must be an engineer", says the balloonist. "I am", replies the man. "How did you know?" "Well", says the balloonist, "everything you have told me is technically correct, but it's of no use to anyone."

The man below says, "You must be in management." "I am", replies the balloonist, "but how did you know?" "Well", says the man, "you don't know where you are, or where you're going, but you expect me to be able to help. You're in the same position you were before we met, but now it's my fault."
Requirements Model Artifact: System Context Information Flows

Teacher

Question, Test

MagicTest

Course, Student, Teacher, Class

Student

Answer, Test Assessment

Test Assessment

Teacher

Question, Test

Student

Test

Course, Student, Teacher, Class

MagicUniversity

Requirements Model Artifact: Use Case Model (Fragment)

Test

Create Question

Modify Question

Remove Question

Create Test

Copy Test

Modify Test

Remove Test

Activate Test

Take Test

Calculate Test Statistics

Teacher

Student

Time

Test Assessment

Send Announcement

<include>
Use Case Model in Actor-Use Case Association Matrix Form

<table>
<thead>
<tr>
<th>Actor-Use Case Association Matrix Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>Administration</td>
</tr>
<tr>
<td>Communication</td>
</tr>
<tr>
<td>Security</td>
</tr>
<tr>
<td>Test</td>
</tr>
</tbody>
</table>

- Administration
- Communication
- Security
- Test

### Requirements Model Artifact: Use Case Behavior (Take Test)

<table>
<thead>
<tr>
<th>Student</th>
<th>MagicTest</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Test</td>
<td>Give Instructions</td>
</tr>
<tr>
<td>Start Test Assessment</td>
<td>Show Question</td>
</tr>
<tr>
<td>Answer the Question</td>
<td>Register the Answer</td>
</tr>
<tr>
<td>[break]</td>
<td></td>
</tr>
<tr>
<td>Take a Break</td>
<td></td>
</tr>
<tr>
<td>Resume Test Assessment</td>
<td>[timeout]</td>
</tr>
<tr>
<td>[time left]</td>
<td>[more questions]</td>
</tr>
<tr>
<td>[last question]</td>
<td></td>
</tr>
<tr>
<td>End Test Assessment</td>
<td>Show Evaluation Results</td>
</tr>
</tbody>
</table>
Extending UML for Capturing Requirements?

A good example of extending UML for supporting more explicit requirements and their relationships modeling is SysML Requirements Diagram.
Simplifying / Customizing UML Modeling Environment

- **<wizard_Name>** Enter system name
- **<wizard_Capture>** Capture Actors
- **<wizard_Capture>** Capture Use Cases
- **<wizard_Relate>** Relate Use Cases and Actors
- **<wizard_Text>** Create views and move to further steps
- **<wizard_Capture>** Detail Use cases and describe flow of events
- **<wizard_Relate>** Structure Use Cases

MagicDraw UML also allows to define user perspectives, custom diagrams, template projects, element customizations, etc.
Useful Features of UML Modeling Environments

- Model Analysis Tools
- Building Relationship Matrices
- **Model Validation**
- Model Decomposition
- Support for Modeling in Team
- Model Version Comparison and Merge
- Diagram Layout Functionality
- Documentation and Report Engine

The Holy Grail for Requirements Modeler …

CHANGE

from requirements **DOCUMENTS** with supplementary **DIAGRAMS**
to
requirements **MODELS** with supplementary **DESCRIPTIONS**

from
Microsoft Word
to
**MagicDraw UML 😊**
The End

?? ?? ?? ?? ?? ??..