3rd Lecture – Languages for information modeling

Agenda

- Languages for information modeling - UML
- UML basic concepts
- Modeling by UML diagrams

- CASE tools: concepts, features and objectives
  - CASE toolset architecture
  - CASE tools for object-oriented analysis and design
Modeling in computer system development

- Modeling is one of the most important computer system design techniques.
- A *model* represents an abstraction of a part of the reality in order to better understand the software product that must be developed *before* starting the actual construction.
- Once built, the model serves to *communication* both within the development team and externally, with users.
- Characteristic features of modeling are: *simplification*, *subordination to a purpose*, *representation of a reality*, *division*, *hierarchy* and *communication*.
Modeling in computer system development

- Object oriented analysis and design uses **three types of models** for describing computer system:
  - *Static model* describes system objects and their relationships;
  - *Dynamic model* describes object interactions within the system;
  - *Functional model* describes data value transformation within the system.
Languages for information modeling

- Generally speaking, languages for information modeling enable **concise and accurate description** of system properties at various level of abstraction.
- They are an integrated part of the computer system development process and they can be used for:
  - linking the analysis phase with the requirements specification and implementation phases;
  - checking the critical properties of systems;
  - assisting in automatical generation of code and test cases.
Categories of languages for information modeling

Depending on the required formalization level, the languages for information modeling are divided in:

- **Informal languages:** natural language
- **Semi-formal languages:** UML, BPMN or SysML
- **Formal languages:** OCL (Object Constraint Language), Z

<table>
<thead>
<tr>
<th>Syntax rules</th>
<th>NO</th>
<th>YES</th>
<th>YES</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Semantic rules</strong></td>
<td>NO</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>Requirement specification language</td>
<td>Informal (natural language)</td>
<td>Semi-formal (UML)</td>
<td>Formal (Z, OCL)</td>
</tr>
</tbody>
</table>
UML language

- **UML (Unified Modeling Language)** It arose from the need to standardize the typology, semantics and representation of results.

- Currently, UML is a modeling standard recognized by OMG (Object Management Group). Standardization is conducted from November 1997, a continuous improvement being performed so far.

- UML can be defined as a visualization, specification, model building and documentation language. Its main value comes from being an open standard, covering the whole software development cycle and many application types. It is based on the experience of the team that developed it and can be implemented by many CASE tools
UML language

- UML has **standard notations** and a **semantic** that is appropriate for object-oriented system modeling.
- With the advent of this language, system designers can easier understand the **system documentation**.
- Before this standardization, an object-oriented project could be described using one of the multitude of available object-oriented methods. If a **revision** was needed, a lot of time would have been lost with analyzing **method notation** and **semantic**, before starting the design.
UML – basic elements

1. Metamodel for object oriented modeling
   - A coherent set of concepts and of the relationships between them;
   - Each element used for modeling is defined using a precise syntax (E.g.: a class definition);
   - It is a support language for transmitting visual models between various tools;
UML basic elements

2. Diagram types
UML basic elements

3. Extension mechanisms

- **Stereotypes** are attached to an element of the model or a relation between elements (there are predefined stereotypes).
- **Notes** offer a supplementary description of a model element.
- **Constraints** limit the use of a model element.
- **Tagged values** represent attributes that are defined for a specific stereotype.
- **Profiles** personalize the metamodel by constructions that are specific to a particular field of activity, platform or development method.
Modeling with UML diagrams - perspectives

A. *Business process modeling* is performed by *use case diagram*. This diagram conducts the whole development process in the case of *use case oriented methods*.

B. *Static structure modeling* is performed using *class diagram* (for modeling the static structure of system classes) and *object diagrams* (for modeling the static structure of system objects). *Package diagram* is a means of grouping diagram elements in packages.
Modeling with UML diagrams

C. Dynamic modeling is performed by:
   i. interaction diagrams
      • SEQUENCE DIAGRAM for modeling message flows between objects and
      • COMMUNICATION DIAGRAM for modeling object interactions.
   ii. diagrams that describe system behavior -
      • STATECHART DIAGRAM for modeling system object behavior and
      • ACTIVITY DIAGRAM for modeling the behavior of uses cases, objects or operations;

D. Implementation modeling is performed by two diagrams:
   • COMPONENT DIAGRAM – for modeling software components
   • DEPLOYMENT DIAGRAM for modeling system physical distribution
CASE tools are software packages based on a certain methodology/method that support designers in developing a software product.

CASE tools usefulness is:

- quantitative (reducing time and cost of implementation) and
- qualitative (rigorous application of a development methodology, reducing shortfalls in implementation, achieving standard documentation, ease of redesign).

CASE tools substantially reduce or eliminate many of the design and development problems of computer applications.
## Traditional development vs. CASE based development

<table>
<thead>
<tr>
<th>Computer system traditional development</th>
<th>CASE-based computer system development</th>
</tr>
</thead>
<tbody>
<tr>
<td>It focuses on coding and testing</td>
<td>It focuses on analysis and design.</td>
</tr>
<tr>
<td>The specifications are based on paper.</td>
<td>Quick interactive prototyping</td>
</tr>
<tr>
<td>Program manual coding</td>
<td>Automatic code generation</td>
</tr>
<tr>
<td>Manual documentation generation</td>
<td>Automatic documentation generation</td>
</tr>
<tr>
<td>Continuous testing of the software</td>
<td>Automatic validation</td>
</tr>
<tr>
<td>Maintenance of code and documentation</td>
<td>Maintenance of design specifications</td>
</tr>
</tbody>
</table>
Objectives

- correct and complete specification of system requirements;
- reducing time and cost of design and development;
- achieving accurate, up-to-date design specifications, with visual presentation;
- integrating design and development using common methodologies / methods;
- standardize the design and development of computer systems;
- simplifying and improving of testing process;
- development of quality documentations;
- improving project management;
- simplifying maintenance phase of the computer systems;
- reuse of applications modules and documentation;
- improving the portability of applications;
- flexibility
CASE features

- Assistance for project management (resource and version management)
- Generation of the computer system development documentation;
- Automatic generation of program code, based on design specifications;
- Use of Reverse Engineering technique that allows the return to a previous phase for changes.
- Support for one ore more analysis and design methods of computer systems. The main attended support are the diagram and text editors.
- Data storage and retrieval from the central data repository using specific utility software;
- Automatic check of data consistency and completeness using an Analyzer that contains specific rules for each methodology/method.
- Support for prototypes development, using high level programming languages and code generators.
Types of CASE tools

- By the scope of the application development cycle:
  - Front-end CASE tools (or upper CASE) offer assistance for the first development stages of computer systems (analysis and requirement specification, logical design).
  - Back-end CASE tools (or lower CASE) offer assistance for the last development stages of computer systems (physical design, program coding, testing, maintenance).
  - Cross life cycle CASE tools offer assistance for activities that take place in several stages of the design and development of computer system (e.g., tools for project management, documentation generators).
Types of CASE tools

- **By the consistence of the provided support:**
  - *Real CASE tools* provide support for a single activity of a development stage (e.g., diagram and text editors, tools for analyzing the consistency and completeness of system specification, debuggers etc.);
  - *CASE workbenches* provide support for a stage of the computer system development cycle;
  - *CASE environments* provide support for most (or all) stages of computer system development. This category includes: Oracle Designer/Oracle, IBM Rational Architect (Rational Rose)/IBM, Cradle/3SL, Corporate Modeler/CASEWise Inc etc.
Types of CASE tools

- **By the intended use:**
  - **CASE tools for analysis** that generally use structured and object oriented analysis methods.
  - **CASE tools for design** (automatic code generation, tools for interface generation, reverse engineering tools, data modeling tools).
  - **CASE tools for testing and debugging** (generators of data for testing, interactive debugging tools).
  - **Interface generators** (form generators, report generators, menu generators).
  - **Documentation generators** (image editors, page formatting tools (layout)) etc.
Architecture of the CASE toolset

- Coding Support
- Backup and Recovery
- Documentation tools
- Analysis and Design
- Export / Import
- Project management
- Prototyping tools
- Configuration management
- Modeling Tools
- Query and report Generator
- Information Security

CASE Tools
Architecture of the CASE toolset

- **Central data repository** (the core of an I-CASE) stores all the objects and information necessary for application design, modeling and automatic generation. It contains:
  - *Information repository* contains information about organization business and its portfolio of software applications.
  - *Data dictionary* manages and controls access to the information repository. It stores descriptions of data and data processing resources.
- **Diagram editors** support the visual representation of a system and of its components. Diagrams are very efficient for representation of process flows, data structures and program structures.
- **Transformation tools** convert elements resulted from the analysis stage into design stage elements.
Architecture of the CASE toolset

- **Forms and report generators** – they are intended for creating, modifying and testing prototypes of forms and reports and for identifying data that will be displayed and retrieved for each form and report.

- **Validation tools** – they generate reports that identify inconsistencies, duplications or gaps in diagrams, forms and reports.

- **Code generators** – they generate code based on design specifications contained in the central data repository (tools for generating database objects and application modules).
CASE tools for object-oriented system development

- They are among the newest types of CASE tools.
- They promote the iterative achievement of computer systems and applications. This enables a return to earlier stages for performing additions or changes as the information system architecture is completed.

Object oriented design encourages modularity, extensibility and reuse of code. Object oriented CASE tools include:

- Utility tools to describe objects, classes and their properties (diagrams for static and dynamic modeling) and working with them;
- Generators that are specialized in code generation, documentation generation, etc.
CASE tools for object-oriented system development

• Nowadays, there are many CASE tools that use object-oriented development methods, implement UML and the object-oriented languages are very popular
• Some examples of CASE tools that are UML-based:
  • Visual Paradigm for UML
  • Rational Software Modeler
  • MagicDraw
  • Microsoft Visio
  • Poseidon for UML
  • Enterprise Architect
  • BOUML
  • StarUML
  • UModel
Visual Paradigm

- **Visual Paradigm for UML (VP-UML)** is a next-generation CASE tool
- It focuses on three main areas:
  - Requirements identification
  - Model building
  - Code and database generation

- It offers **interoperability** with other CASE tools (Visio, Visual UML, Rational) and **integration** with IDE tools (Net Beans)
- It covers most of a computer system lifecycle
Visual Paradigm

Includes models of some standard languages:

- **UML modeling** - All the UML 2.x diagram types can be created, building use case models, behavior models, interaction models, structural models, deployment models.

- **BPMN modeling** – It can be created: business process diagram, data flow diagrams, process map diagrams, event-driven process chain diagram, organization chart. Business process diagram can be exported in BPEL.

- **SysML modeling** - SysML is a generic language for application and computer system engineering. VP-UML supports building the requirement diagram (specific to SysML).
Visual Paradigm

Requirement modeling:
It identifies requirements by several mechanisms:

- **SysML requirement diagrams** for identifying *functional or non-functional requirements* of the system.
- **Text analysis** It offers **text editor** whereby requirements are recorded in textual form. It enables the identification of important terms or objects (classes, use cases) for describing the problem.
- **CRC cards** contain information such as description of the class, its attributes and responsibilities. They have their own information display formats.
- **User interface editor** whereby screen layouts are designed.
- **Term glossary management** whereby project vocabulary is identified and described.
<requirement> Can use in VPN </requirement>
Text = "Can run in 1.5Mb Internet Connection speed"
ID = ""
source = ""
kind = "Performance"
verifyMethod = "Analysis"
risk = "Medium"
status = "Proposed"

<requirement> Less than 10s for Text Content </requirement>
Text = "If the page only contain text, the response time should be less than 10 seconds"
ID = ""
source = ""
kind = "Interface"
verifyMethod = "Test"
risk = "Medium"
status = "Proposed"

<requirement> Less than 30s for Image Content </requirement>
Text = "If the page contain less than 20 photos, the response time should be less than 30 seconds"
ID = ""
source = ""
kind = "Functional"
verifyMethod = "Test"
risk = "Medium"
status = "Deferred"

<requirement> Internet Explorer 6.0 </requirement>
Text = "Default client browser is Internet Explorer 6.0"
ID = ""
source = ""
kind = "Interface"
verifyMethod = "Test"
risk = "Low"
status = "Rejected"

<requirement> Window Mobile 6 </requirement>
Text = "The PDA module should run on Microsoft Windows Mobile 6 or above version"
ID = ""
source = ""
kind = "Interface"
verifyMethod = "Demonstration"
risk = "Low"
status = "Proposed"

<requirement> Support Camera in PDA </requirement>
Text = "Inspector and Inspector Assistant can take photos directly in PDA"
ID = ""
source = ""
kind = "Interface"
verifyMethod = "Inspection"
risk = "Low"
status = "Deferred"
User interface editor
Visual Paradigm

Database modeling

- two diagram types can be built:
  - Entity-Relationship Diagrams (ERD)
  - ORM Object Role Modeling diagrams (for visualization of the mapping between object model and data model).

- A ERD can model not only table elements and features but also stored procedures, triggers, sequences and views of the database.

- Diagrams are built by scratch or by reverse engineering, based on an existent database.

- Synchronization between class diagram and entity-relationship diagrams to ensure consistency between the two models.

- SQL code generation from models.
ERD diagram
Visual Paradigm

Code generation

- Code and reverse engineering generators offer support for model engineering. **Java Round-Trip engineering** offers continuous synchronization code and model for Java language.

<table>
<thead>
<tr>
<th>Model</th>
<th>Code generation</th>
<th>Reverse engineering</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>C++</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>XML Schema</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>PHP</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Python Source</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Objective-C</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>CORBA IDL Source</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>.NET dll sau fișiere .exe</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>CORBA IDL Source</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>XML (structure)</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>JDBC</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Hibernate</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>C#</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>VB.NET</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>ODL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>ActionScript</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Delphi</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Perl</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Ada95</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Ruby</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>
Visual Paradigm

Integration with IDE (Integrated Development Environments)

- It supports the entire development cycle of a computer system using for programming the following IDE products:
  - Eclipse
  - NetBeans/Sun ONE
  - IntelliJ IDEA

Documentation generation

- Documentation can be shared and designed together with computer system beneficiaries, using one of the following formats: HTML (report generation), HTML (project publisher), PDF, Word.
Trends in the development of CASE tools

In the last decade, CASE technologies and CASE market have become mature. One factor that stimulates the market of CASE tools is the organization's desire to extend the life of existing systems by using:

- i) CASE tools with **reverse engineering features** that support program adaptation to **new hardware configurations**. The analyst can restructure the code to match current business requirements;
- ii) CASE tools with **re-engineering facilities** that support existent computer **system change** in order to improve its quality and performance.

Also, object oriented and visual CASE tools will grow rapidly, and development environments will include **artificial intelligence** (by using **software agents**).
Trends in the development of CASE tools

- Increase of integration level of CASE tools depends largely on defining a standard for central data repository that is recognized by most tools.
- Although CASE technology is considered as being still immature, it provides an important support in system development.
- Although tend to cover the full system development cycle, CASE tools fail to replace systems implementation team.
- There will be some important changes in system development approach. Thus, the first stages of the system (analysis and design) will gain growing importance in spite of final stages of system development (software development, implementation and maintenance).