Current Trends and Perspectives in Ontology-Driven Software Development (ODSD)

Ontology-based Model Checking of Software Applications

Prof. Dr. U. Aßmann
Technische Universität Dresden, Lehrstuhl Softwaretechnologie
Aug 2013
Version 0.6


Lecture 1

1.1) Why Marrying Ontologies and Software Technology?

1.2) Model-Driven Integration of Technical Spaces
   - Model-Driven Software Development (MDSD)
   - Ontology-Driven Software Development (ODSD)
   - Bridging the technical spaces of system modeling and ontologies
   - Semantic-oriented modeling

1.3) Ontology-Integrated Modeling (OIM)

1.4) Ontology-Driven Software Development (ODSD) and beyond
1.1) Why Marrying Ontologes and Software Technology?

Syntactic and Semantic Modeling

First MDI workshop: http://mdi2010.lcc.uma.es/
Why is Logic not Integrated with Modeling and Programming?

- Prolog is untyped
- Ontologies are typed, but...
  - Open World Assumption vs. Closed World Assumption
  - Reasoners are separate tools
The MOF Metamodelling Hierarchy

- aka *metapyramid*

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>M0 Object level</td>
<td>Software objects describing world objects</td>
</tr>
<tr>
<td>M1 Model level</td>
<td>Types, programs, models, domain ontologies</td>
</tr>
<tr>
<td>M2 Metamodel level</td>
<td>Language descriptions</td>
</tr>
<tr>
<td>M3 Metametamodel level</td>
<td>Modelling concepts</td>
</tr>
<tr>
<td>M4 Level = M3</td>
<td>Metamodelling concepts</td>
</tr>
</tbody>
</table>

- `instanceOf` describes
- `describes`
## Metamodelling in Layers

<table>
<thead>
<tr>
<th>Layer</th>
<th>M3 Metalanguages Modeling Concepts</th>
<th>M2 Languages Language Concepts</th>
<th>M1 Models Application Concepts</th>
<th>M0 Running Instances</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>EMOF/Ecore</td>
<td>BPEL</td>
<td>Business Process Models</td>
<td>Running Workflows</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RDFS</td>
<td>OWL</td>
<td>Business Process Ontologies</td>
<td>Process Objects</td>
</tr>
</tbody>
</table>

- **Technical Space**: EclipseWare
- **Technical Space**: OWLWare
The MOST Project and the ODSD Book

- Project “Marrying Ontologies and Software Technology (MOST)”

- [ODSD] Ontology-Driven Software Development
  - Pan/Stab/Aßmann/Ebert/Zhao 2013

- Collection of techniques to marry ontologies and software technologies
- Treating the problem of technical spaces
Modern Software Tools Use Metamodels

Data manipulation and access interfaces (typed read/write; reflective access observing access)

Layer 4

Layer 3

Layer 2

Layer 1

Layer 0

External Database (File system, Web)

Document/-model-Repository (Material)

Loader

Exporter

Metadata- and metamodel repository

<<generates>>

Consistency checker

<<generates>>

Workflow Control „Automaton“

Query Engine

<<generates>>
From the metaclasses, the structure of types in M1 is generated.
Benefit: Generation of Access Layers from Metamodels

M2

<<instance-of>>

Metamodel A

<<generates>>

M1

Factory

Reflective access

write

read

authentificate

distribute

Models conforming to Metamodel A (Data, Programs, Specifications)
Benefit: Generation of Model Transformer from Metamodel Mapping

- Metamodel A <<instance-of>> Model 1
- Metamodel B <<instance-of>> Model 2
- Model Transformer A->B <<maps-to>> Metamodel B
- Tool A <<parses>> Model 1
- Tool B <<prints>> Model 2

M1

M2
Benefit: Generation of Parsers and Printers

- **Metamodel A**
  - Internal Representation Of Model 1
  - <<instance-of>>
  - <<generates>>
  - Exchange Metamodel B
    - <<instance-of>>

- **Leser (Lexer, Parser)**
  - <<import>>
  - <<instance-of>>

- **Schreiber (pretty printer)**
  - <<export>>
  - Internal Representation Of Model 1
    - <<instance-of>>

- **Control**
  - <<steuert>>

**Exchange Metamodel B**
- <<mapping>>

**External Representation of a Model 1**
- <<import>>

**Benefit: Generation of Parsers and Printers**
- M1
- M2
Benefit: Generation of Language-Specific Editors

- Syntax highlighting and completion
- Serialization and deserialization of data
Eclipse and its EMF Metahierarch depthy
(Metalanguage EMOF)

Metamodel repository (EMOF-based)

<<load>>

<<generates>>

Repository

Persistence Layer
Example: EMOF/Ecore based Metamodel of Statecharts

M3: Ecore is the metalanguage of Eclipse
An implementation of EMOF, provided by the Eclipse Modeling Framework (EMF).

M2: a metamodel of statecharts

M1: a set of states and their transitions
1.2) The Foundation: Model-Driven Integration of Technical Spaces (MDI)

Syntactic and Semantic Modeling

First MDI workshop: http://mdi2010.lcc.uma.es/
Different Technical Spaces

- Cuneiform Sumarian pictographic language (2800 bC, protowriting)
- Ugarit phonetic alphabet (1500 bC, writing)

Schøyen Collection MS 3029. USA-PD [Link to image]

PD [Link to image]
Why „Integration“?

- Multiple Technical Spaces (TS)

**TS Ontology-ware**

- **M3**
  - RDFS

- **M2**
  - OWL

- **M1**
  - Domain Ontologies

**TS Eclipse-ware**

- EMOF/Ecore
  - BPMN
  - Business Process Models

Ontology-Driven Software Development
Technical Spaces

- A **technological space** is a working context with a set of associated concepts, body of knowledge, tools, required skills, and possibilities.
- It is often associated to a given user community with shared know-how, educational support, common literature and even workshop and conference regular meetings.
  - Ex. compiler community, database community, semantic web community
- A **technical space** is a model management framework accompanied by a set of tools that operate on the models definable within the framework.
- [Model-based Technology Integration with the Technical Space Concept. Jean Bezivin and Ivan Kurtev. Metainformatics Symposium, 2005.]
Problem: Many Technical Spaces Exist!

- M3: RDFS
- M2: OWL
- M1: Domain Ontologies

- TS OWL-ware
- TS TGraph
  - grUML Schema
  - grUML
  - Graph Models

- TS Ecore
  - EMOF
  - UML-CD
  - Class Diagrams

- TS ADO
  - ADOxx
  - BPMN
  - Business Process Models

Ontology-Driven Software Development
Reasons for Multiple Technical Spaces

- Special tools shall be reused
  - Databases

- The system is heterogeneous and needs middleware
  - distributed
  - Language heterogeneous
  - Third party producer
  - Service-oriented architecture

- Syntax vs Semantics
  - The Logic / Software Engineering dichotomy is just an example for two different technical spaces

- Multi-Technical-Space-Development
Reason: “Syntactic” and “Semantic” Modeling

- In the UML world, syntactic modeling is dominant (structural models)
  - Structural modeling is needed most, semantics can wait..
  - Different forms of static semantics are done in OWL and other TS
- Dynamic semantics?
**Goal: Marrying Ontologies and Software Technology**

- Bring together domain knowledge and software know-how
- Bridge technical spaces
- [ODSD]

---

Ontology-Driven Software Development

26
Use Ontologies for Static Semantics!

- Investigate applicability of ontology technology (semantics reasoning)
- Ontology-based software development (Chap. 1.)
- Logic-based DSL (Chap. 2.)

Software Process
- generic process tools

Methodology
- generic modelling tools

Automation
- generic transformation and validation tools

Process Models + Ontologies

Meta-Models + Ontologies

Transform., Constraints + Ontologies
# MOST – Case Studies for TS Integration

<table>
<thead>
<tr>
<th></th>
<th>Comarch Network Devices</th>
<th>Comarch OSS</th>
<th>SAP Business Processes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Software Process</strong></td>
<td>network device design</td>
<td>data-centric development of telecommunication support systems</td>
<td>business processes refinement and grounding</td>
</tr>
<tr>
<td></td>
<td>and instantiation</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| **Methodology**     | MDSD using Physical Devices DSL and Physical Device Instances DSL | MDSD using Business Entities DSL, Managed Entities DSL, and Database DSL | MDSD using BPMN |
|                     |                                                              |                                                              |                |

| **Automation**      | network configuration validation | consistency checking, transformation between abstraction layers | business process refinement validation, business process grounding validation |
|                     |                                  |                                                              |                |
1.2.1) What is Model-Driven Software Development?
State of the Art Syntactic Modeling: Model-Driven Software Development (MDSD)

- Requirements Model (CIM)
- Platform-Independent Model (PIM)
- Platform-Specific Model (PSM)

Processes:
- CIM2PIM
- PIM Consistency checker
- PIM2PSM
- PSM2Code

Ontology-Driven Software Development
State of the Art Syntactic Modeling: Domain-Specific Languages (DSL)

Requirements Model (CIM) → Platform-Independent Model (PIM) → Platform-Specific Model (PSM)

- CIM2PIM
- PIM Consistency checker
- PIM2PSM
- PSM2Code

Ontology-Driven Software Development
State of the Art Usage of Ontologies: Querying on the Implementation Level

- Requirements Model (CIM)
- Platform-Independent Model (PIM)
- Platform-Specific Model (PSM)
- EclipseWare technical Space

Flow Diagram:
- CIM2PIM
- PIM Consistency checker
- PIM2PSM
- PSM2Code

Ontology-Driven Software Development

- Domain ontology
- Business Process ontology
- OntologyWare technical Space
What are „Integrated Ontology Services/Querying“?

Requirements Model (CIM) 
Platform-Independent Model (PIM) 
Platform-Specific Model (PSM) 
EclipseWare Techn. Space 

CIM2PIM 
PIM Consistency checker 
PIM2PSM 
PSM2Code 

Domain ontology 
Business Process ontology 
OntologyWare Techn. Space 

Requirements Model (CIM) 
Platform-Independent Model (PIM) 
Platform-Specific Model (PSM) 
EclipseWare Techn. Space 

Ontology-Driven Software Development
Advantage: Services Provided by Ontology Technology

- Precise documentation of services provided by ontology technology
  - Classification
  - Consistency Checking
  - Explanation
  - Consistency Guidance (Checking with Explanation)
  - Merging
  - Querying
  - Satisfiability
  - Subsumption (with Explanation)
  - Forget
1.2.2) What is Ontology-Driven Software Development?
Ontology-Driven Software Development (ODSD)

- Requirements Model (CIM)
- Platform-Independent Model (PIM)
- Platform-Specific Model (PSM)

- CIM2PIM
- PIM Consistency checker
- PIM2PSM
- PIM2PSM

- Domain Ontology
- Business Process Ontology

- Requirements Ontology
- Platform-Independent Ontology
- Platform-Specific Ontology

EclipseWare Techn. Space
Transformation Bridges enable „Model Transport“ between Technical Spaces

- Requirements Model (CIM)
- Platform-Independent Model (PIM)
- Platform-Specific Model (PSM)
- Software Modeling Techn. Space

Diagram:
1. PIM Consistency checker
2. Platform-Independent Ontology
3. Business Process Ontology

OntologyWare Techn. Space

Requirements Ontology

Domain Ontology

Platform-Independent Ontology

Platform-Specific Ontology
1.2.3) Bridging Technical Spaces
Bridging Technology for Integrated Ontology Services in ODSD

- **Model transformation bridge** (physical transport to the other space on M1)
- **Language integration bridge** integrates languages (mapping between metamodels on M2)
- **Metalanguage integration bridge** integrates metalanguage (mapping between modeling concepts on M3)
Bridging technologies in Detail

- **Model transformation bridge** (physical transport of models to the other space)
- **Language integration bridge** integrates languages
- **Metalanguage integration bridge** integrates metalanguages

![Diagram showing bridging technologies](image)

**Ontoware TS**
- OWL2 Metamodel
- Tools
- Querying Reasoning Technology
- Ontolog TBox
- ABox

**Modelware TS**
- Ecore Metametamodel
- DSL Metamodel
- Tools
- Validation Constraint Technology

- Model
- Model

**Tools**

- M1
- M2
- M3

Courtesy T. Walter, S. Staab, Uni Koblenz

Ontology-Driven Software Development
Example: Many Technical Spaces in MOST

Ontology-Driven Software Development
Language and Metalanguage Integration Bridges in MOST

- [ODSD]
  - Ontology Ware
  - Ontology Structure Definition Language
  - TwoUse
  - ADO-EMF
  - RDF-TGraphs
  - OntoDSL
  - Model Ware
  - Model Structure Definition Language
  - Query Language
  - SPARQL-GreQL
  - Query Language
  - Many...
A **model transformation bridge** moves all models (artefacts, specifications) from the software modeling tool to the reasoner
- Generated from a metamodel mapping, e.g., TwoUse
Language Bridge Uses a Mapping of Language Concepts

- For all metaclasses in metamodels, give a mapping (**language mapping**)
  - Used to generate model printer und parser

![Diagram showing the mapping from UML Models to Textual Representation of UML in XML through MOF and XSD]

**TS MOF**
- **MOF**
  - Type
  - Attribute
  - Association

**TS XML**
- **XSD**
  - Type
  - Attribute
  - Association

**M3**
- **MOF**
**M2**
- **UML**
- **Association**
**M1**
- **UML Models**

Textual Representation of UML in XML

Printer

Parser
### Ex.: Language Mapping Between EMOF and OWL

<table>
<thead>
<tr>
<th>EMOF</th>
<th>RDFS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>Axiom/Instance</td>
</tr>
<tr>
<td>Class</td>
<td>Class</td>
</tr>
<tr>
<td>Attribute</td>
<td>Role</td>
</tr>
<tr>
<td>Association</td>
<td>Property</td>
</tr>
<tr>
<td>Scalar</td>
<td>DataProperty</td>
</tr>
<tr>
<td>Type</td>
<td>ObjectProperty</td>
</tr>
<tr>
<td>Inheritance</td>
<td>Is-a</td>
</tr>
</tbody>
</table>

... workarounds ...

... workarounds ...

---

Ontology-Driven Software Development
Example: XMI, a Transformative TS-Bridge for UML

- XMI is a standardized textual TS-bridge of OMG, e.g., for UML
  - From MOF to XSD/XML
  - From EMOF to XSD/XML
Example: JSON, a Textual Transformative TS-Bridge

- JSON (Java Script Object Notation) is a lightweight exchange format between technical spaces

- [http://www.json.org/](http://www.json.org/)
Example: TwoUse-Based Transformation Bridge [U Koblenz]

- ADOxx → TwoUse Metamodel → TrOWL
- Metamodel mapping OWL/UML-CD

ADOxx as an integrated modelling toolkit

1. IsModelValid (model, query)
2. Transform ingr. model “model” into OWL only ontology and transform query into SparQL
3. IsOntologyValid (onto, query)
4. Perform Reasoning on Ontology “onto”
5. return ontology result
6. process result back As model result
7. return model result
8. display result on model “E.g. mark invalid objects”

TwoUse as bridge Or adapter

TrOWL as semantic reasoner

Modelling World

UML, BPMN, DSL...
Modelling world + Declarative Constraints Expressed in OWL

Ontology Reasoning World

UDOxx (BOC)

Integrated Modelling based on MOST Integrated Language

TwoUse

Terminals

OSH and Punea

 curtis singer

 Courtesy T. Walter, S. Staab, Uni Koblenz

Ontology-Driven Software Development
1.3 Integrated Modeling in ODSD

Ontology-Integrated Modeling with Ontologies and Software Models
Ontology-Integrated Modeling

- Tight integration enables *Ontology-Integrated Modeling* with Ontologies and Software Models
  - Using technical space bridges internally
- In the same specification, Users give:
  - Syntactic structure
  - Semantic constraints
- Models are checked by reasoner immediately
- Feedback is given interactively

Generation of Integrated Editors

Automated Generation Process

- Metamodel
- Textual Syntax in EBNF
- Ontology
- Integration Specification
- Manually Provided Specifications

Generation Results

- Generation of Syntax Services
- Generation of Semantics Services Integration
- ODSD Domain Editors
- Generation Results
OntoDSL: Domain-Specific Modeling with Hidden Transformation Bridge

- OntoDSL development environment for DSL for integrated modeling
- Based on Metalanguage bridge (KM3+OWL)
- As well as Language bridge (KM3+OWL)
- Transformation Bridge transports the integrated specifications to ontology space
- Feedback of reasoner is transported back
Network Device Specification (Graphic DSL Editing)

Courtesy K. Miksa, Comarch

Logic-Based DSL
Graphic DSLs in Device Modeling

PDDSL Device Types

PDDSL Devices

OWL Descriptions

OWL Frames
2-Dimensional Modeling Bridge (Example)

- [RW 2010]

M2 Layer Metamodel

- Device
- DeviceInstance
- Ontological metaHasType
- linguistic instanceof
- linguistic instanceof

M1 Layer Domain model

- Cisco7600
- Cisco7603
- ontological hasType

Language Engineering Ontology

- TBox
- ABox

- used to validate domain models wrt. metamodels using ontology technologies

Domain Engineering Ontology

- TBox
- ABox

- used to describe domain models with the expressiveness of ontology languages
1.3.2 Process Refinement Guided by Ontologies

- MOST developed Guidance Engine
  - To guide developers while developing software
- Development support:
  - Suggest continuation tasks
  - Invalid refinement of processes -> propose remodeling
  - Unbound tasks in processes -> propose refinement or remodeling
Screenshot: MOST Workbench on ADO Workbench (BOC)
Example: Suggest Continuation Tasks

- Game contests as processes

![Diagram showing available tasks]

- Remodel Process A
- Refine Process Process A
- Ground Process Process A
- Refine Activity Manage Contest
- Ground Activity Manage Contest
Example ctd.

Available Tasks

- Remodel Process B
- Refine Process Process B
- Ground Process Process B
- Refine Activity Select Contest Parameters
- Ground Activity Select Contest Parameters
- Refine Activity Manage Developer Relationship
- Ground Activity Manage Developer Relationship
- Refine Activity Manage Contest Start
- Ground Activity Manage Contest Start
Discover Refinement Clashes

Flow Violations
- Detected invalid flow from activity *Negotiate with Developer*
- Detected invalid flow from activity *Cancel Contract*
Screenshot: Marking Ill Refinements

The image shows a screenshot of a software tool, possibly a BPMN (Business Process Model and Notation) editor. The tool is used for modeling business processes. The screenshot includes a process flow diagram with various activities and decision points, indicated by shapes like circles and diamonds. The diagram appears to be part of a workflow management system, highlighting the steps involved in a business process.
Example Remedied
Example: Now Refinement Works!
Technology: Process Refinement Ontology

- Process ontology
  - Tasks as concepts
  - Ordering relations as *to* and *from* properties

- Refinement constraints:
  - Execution order after refinement must correspond to order before refinement
  - Pre-refinement process as constraints of post-refinement process
Accomplished ODSD Scenarios in MOST

Requirements Model (CIM)

Platform-Independent Model (PIM)

Platform-Specific Model (PSM)

Feature Model

Device model

BPMN model

Refinement

Traceability

CIM2PIM

Consistency checker

PIM2PSM

PIM2PSM

Cons. Check

Feature Ontology

Device Ontology

Business Process Ontology

Refined Business Process Ontology

Requirements Ontology

Traceability
1.4 Ontology-Driven Software Development
The Future of “Syntactic” and “Semantic” Modeling

- Standalone syntactical TS will remain
- How to bridge them to several semantic technical spaces?
Consistency needs Semantic Technical Spaces

Domain Expert

Domain world
- Ontologies
- Domain models

Domain Expert

Static Semantic World
- Ontologies
- Abstract Interpretation
- Model checking

Dynamic Semantic World
- Interpretation
- State systems
- Simulation
- Petri nets
- SOS
- Natural Semantics

Software Engineer

Structural (syntactic) Modeling World
- Structure Hierarchies
- Graphs

EMOF
- UML-CD
- MOF

OWL
- CTL
A new level in Software Engineering

- **Ontology-Driven Software Development (ODSD)**
  - Consistency-preserving development with ontology languages
  - Guided development with advising languages

- Constraint-safe development with constraint languages
- Type-safe development with typed languages
- Development with untyped languages

- Total Progress of MOST
What is Beyond ODSD?

Development with **multiple technical spaces**

Ontology-Driven Software Development (ODSD)

- Guided development
- Consistency-preserving development
- Constraint-safe development with constraint languages

Development with **two** technical spaces

Development in **one** technical space
Multi-Technical-Space Tools

A Multi-TS-Tool uses several technical spaces at the same time

- To engineer BIG software, many technical spaces are used at the same time (XML, Java, C, csv, OWL, UML, ...)
- Tools of the future need to handle several technical spaces
- Systematic engineering of technical space bridging is necessary

Model Engineering builds bridges between technical spaces

- Bezivin's Model Engineering Metaphor
Motivation: Multi-Quality Contracts in Embedded Software (Multi-TS Development)

- Embedded Software Component A
  - Real-time
    - Real-time contract checking (Technical Space 1)
  - Safety
    - Safety contract checking (Technical Space 2)
  - Dynamics
    - Dynamics contract checking (Technical Space 3)
  - Energy
    - Energy contract checking (Technical Space 4)

- Embedded Software Component B

Ontology-Driven Software Development
Motivation: Multi-Quality Contracts in Embedded Software (Multi-TS Development)

- MOST Bridge technology can be transferred
- Guided software development important for certification

Semantic Technical Space Real-Time

- Real-time contract + Real-time analysis

Semantic Technical Space Hybrid Automata

- Dynamics Contract + Dynamics language

Syntactic Technical Space Eclipse EMOF

System architecture model

Semantic Technical Space Safety

- Safety contracts + Safety Checker

Semantic Technical Space Energy

- Energy contract + Energy analysis

Ontology-Driven Software Development
The End