A Formal Specification of the Horus Modeling Language Using FDMM

Hans-Georg Fill, Susan Hickl, Dimitris Karagiannis, Andreas Oberweis, Andreas Schoknecht
Agenda

- Motivation and Foundations
- Formalizing the Horus Modeling Language
- Implementation on ADOxx
- Lessons learned
- Outlook and Future Work
Multitude of Modeling Methods

Sources: (Strecker et al., 2011); (Heise et al., 2010); (Fill et al., 2011); (Fill, 2011); (Ferstl and Sinz, 1995); (Breitling and Hofer, 2012)
IT based Modeling Tools

Tool-based support necessary for:

– Collaborative creation, modification and handling of large and numerous models
– Exchange of model information, e.g. for the configuration of systems, code generation, etc.
– Application of algorithmic analyses and simulations, e.g. for business process simulation, business impact analyses, etc.
– Visualization of model information
– …

How to realize a modeling tool?
Meta Modeling

Meta Modeling as a concept for realizing modeling methods:

– Specification of the syntax and semantics of modeling languages
– Assignment of static and dynamic graphical representations
– Definition of mechanisms for enhancing the user interaction, e.g. collaboration, concurrency, etc.
– Specification of algorithms for conducting analyses and simulations
– Definition of modeling procedures that state how models and algorithms are applied for problem solving
– Technical conceptualization and implementation
Aspects of Meta Modeling

Formalization
For exact meta modeling foundations

Conceptualization
For application of meta modeling concepts

Meta² Model
Philosophical Level, Basic Elements

Meta Model
Enabling Multiple Instantiation on Model Level

Model
Conceptual Representation of Instances

Instance
Implementation Level
Formalizing the Horus Modeling Language
The Open Models Horus Project


http://www.openmodels.at/web/adoxx-horus-method
The Horus Method

Phase 0: Preparation
- Strategy and Architecture

Phase 1: Strategy and Architecture
- Context analysis
- SWOT analysis
- Strategy analysis
- Enterprise architecture modeling
- System architecture design

Phase 2: Business Process Analysis
- Structure analysis
- Procedure analysis
- Process cluster
- Organization structure analysis
- Key figure analysis
- Risk analysis

Phase 3: Application
- Process management
- Process implementation
- Business performance management
- Process Evolution

Project Management | Quality Assurance | Documentation

(Schönthaler et al., 2012)
Model Types in Horus

- Goal model
- Context model
- Supply- & services model
- SWOT model
- Strategy model
- Risk model
- Key figure model
- Object model
- Rule model
- Business unit model
- Business process architecture model

- System architecture model
- Resource model
- Organization model
- Procedure model

Phase 0: Preparation
Phase 1: Strategy and Architecture
Phase 2: Business Process Analysis
Phase 3: Application
Chosen Approach: FDMM & ADOxx

Formalization
For exact meta modeling foundations

Meta² Model
Philosophical Level, Basic Elements

Meta Model
Enabling Multiple Instantiation on Model Level

Model
Conceptual Representation of Instances

Instance
Implementation Level

Conceptualization
For application of meta modeling concepts
The FDMM Formalism

- Designed to permit formal descriptions of ADOxx meta models and models
- Goal: Provide a simple and intuitive way for formal descriptions without specialized mathematical theories
- Based on set theory and first-order logic statements
- Provides constructs for:
  - The definition of meta models
  - Correctness criteria for meta models
  - The instantiation of meta models to models
  - Disjointness and partitioning constraints for the instantiation

(Fill, Redmond, Karagiannis, 2012)
Selected Examples of FDMM for Horus I

Model Types

\[
\begin{align*}
MT_{PM} &= \langle O_{PM}^T, D_{PM}^T, A_{PM} \rangle \\
MT_{EM} &= \langle O_{EM}^T, D_{EM}^T, A_{EM} \rangle \\
MT_{RM} &= \langle O_{RM}^T, D_{RM}^T, A_{RM} \rangle \\
MT_{OM} &= \langle O_{OM}^T, D_{OM}^T, A_{OM} \rangle
\end{align*}
\]

Object Types

\(O_{PM}^T = \{\text{Abstract-Procedure-Class, Object-store, Activity, Connection, Human-resource-requirements}\}\)

Data Types

\(D_{PM}^T = \{\text{String, Integer, Float, File, Enum}_{WF}, \text{Enum}_{AT}\}\)

Attributes

\(A_{PM} = \{\text{Name, Object-type, Object-number, Documents, connection-to, connection-from, sub-diagram, HR-req, Role-ref, Assignment-type, Quantity, Percentage, XQuery-transition-condition}\}\)

Subtypes of Object Types

Object-Store \(\leq\) Abstract-Procedure-Class

Activity \(\leq\) Abstract-Procedure-Class

Linking attributes to Object Types

\[
\begin{align*}
domain(\text{Object-type}) &= \{\text{Object-store}\} \\
range(\text{Object-type}) &= \{\text{Object, Object-Aggregation}\} \\
card(\text{Object-type}) &= \langle 0, 1 \rangle
\end{align*}
\]
Selected Examples of FDMM for Horus II

Model Instances

\[
\begin{align*}
\mu_{MT}(MT_{PM}) &= \{ mt_{pm1} \} \\
\mu_{MT}(MT_{OM}) &= \{ mt_{om1} \}
\end{align*}
\]

Objects

\[
\begin{align*}
\mu_O(Object-store, MT_{PM}) &= \{ os_1, os_2, \ldots, os_7 \} \\
\mu_O(Activity, MT_{PM}) &= \{ a_1, a_2, \ldots, a_5 \}
\end{align*}
\]

Data Objects

\[
\mu_D(String) = \{ 'Website opened', 'Buy products', \ldots \}
\]

Triple Statements attribute values

\[
\begin{align*}
(os_1 \text{ Name 'Website opened'}) &\in \beta(mt_{pm1}) \\
(a_1 \text{ Name 'Buy products'}) &\in \beta(mt_{pm1})
\end{align*}
\]

Instantiating relations

\[
\begin{align*}
\mu_O(Connection, MT_{PM}) &= \{ c_1, c_2, \ldots, c_{11} \} \\
(c_1 \text{ connection-from } os_1) &\in \beta(mt_{pm1}) \\
(c_1 \text{ connection-to } a_1) &\in \beta(mt_{pm1})
\end{align*}
\]

Step 2: Model Instantiation
Prototypical Implementation on ADOxx

- ADOxx as an industry proven meta modeling platform
- Used for several implementation projects on openmodels.at
- Freely available for academic purposes on www.adoxx.org
- Consideration of differences between FDMM and ADOxx, e.g. object types vs. classes, relations; domain/ranges vs. class/instance attributes, interref attributes, recordclass attributes etc.
- Some statistics: 17 model types, 44 classes, 9 relationclasses, 11 record classes, 18 abstract classes, 161 attributes for classes, 26 attributes for relationclasses + graphical representations for each class/relationclass, constraints and UI mechanisms in ADOscript
- Effort: ~60 person hours

Lessons learned

• Exact specification of modeling methods for ensuring common understanding between designers and developers
• Meta modeling oriented formal specification directly supports implementation
• FDMM & ADOxx were applicable to Horus method
• Limitations:
  – No specification of dynamic aspects in FDMM so far
  – Specification of graphics not included
  – Differences in FDMM and ADOxx require good knowledge of both approaches
• Advantages of the formal specification:
  – Unambiguous representation how modeling language is defined
  – Possibility of verifying requirements and actual implementation
Outlook and Future Work

• FDMM needs to be extended:
  – Mapping between FDMM constructs and ADOxx library language
  – Basis for automatic MM-platform code generation
  – Dynamic aspects for specifying mechanisms and algorithms

• Automatic assignment of graphical representations, e.g. based on semantic visualization

• Simulation functionalities for the Horus method, e.g. token games, XML net behavior
Thank you for your attention!

Download the tool:
http://www.openmodels.at/web/adoxx-horus-method