Toward Megamodels at Runtime

5th International Workshop on Models@run.time
Oslo, Norway, 5th October 2010

Thomas Vogel, Andreas Seibel, and Holger Giese
System Analysis and Modeling Group
Hasso Plattner Institute
University of Potsdam
Motivation

Multiple runtime models for monitoring and adaptation
(Model@run.time’09, SEAMS’10)
Motivation

Multiple runtime models for monitoring and adaptation
(Models@run.time’09, SEAMS’10)
Motivation

Multiple runtime models for monitoring and adaptation
(Models@run.time’09, SEAMS’10)

Relations between models: trade-offs, dependencies, …
Categories of Runtime Models

Simultaneous use of multiple runtime models? Conceivable relations between runtime models?

- Abstract categorization: purpose and content of a runtime model
- Based on literature, esp. the past Models@run.time workshops
- Categories: neither complete, nor a prerequisite for an approach
Implementation Models

Characteristics:
- Similar to models used in the field of reflection
- Causally connected to a running system
- Coupled to the system’s implementation and computation model (solution space)

Examples:
- Reflective programming languages [JBCG06, KV08]
- Platform-specific models, like for CORBA [CPV06] or EJB [VG10]
- Class and object diagrams [JBCG06, GIWO09, Mao09]
- Sequence diagrams [Mao09]
- Statecharts, state machines, automatons [GCZ08, Mao08, HDC09]
Configuration & Architectural Models

Characteristics:
- More abstract than *Implementation Models*
- Platform-independent, problem space
- Often causally connected to a running system
- Reflect the current configuration of a system
- Software architecture as an appropriate abstraction level

Examples:
- Component diagrams, often enhanced with non-functional properties [SXC$^{+}$10, MBJ$^{+}$09, OMT98, GCH$^{+}$04, VNH$^{+}$10, VG10]
- Process or workflow models [SBVD08]
- Abstract *Implementation Models*, like statecharts for components
Context & Resource Models

Characteristics:
- Describe the system’s operational environment
- Describe required or used resources (logical or physical)
- Context-aware systems

Examples:
- Some form of variables, like key value pairs [MBJ+09, SB08]
- Semi-structured tags and attributes, object-oriented or logic-based models [SB08]
- Feature models [ACF+09]
Configuration Space & Variability Models

Characteristics:
- Specify potential variants of a system
- Define the configuration space
- Used for finding adaptation options

Examples:
- Component type diagrams [GCH+04, GSV09, VG10]
- Feature models originating from software product lines [MBJ+09, CGFP09, EME09]
- Aspect models for Configuration & Architectural Models [MBJ+09, FHL+09]
Characteristics:

- Refer to models of the other categories
- Specify adaptations (rules, strategies, goals)
- Validation and verification (constraints, requirements, goals)

Examples:

- Event-Condition-Action (ECA) rules \([GCH^+04, ACF^+09, DM06]\)
- Goal-based models (utility functions) \([MBJ^+09, EME09, RC09]\)
- Constraints: OCL \([HRW07, VNH^+10]\), Linear Temporal Logic \([GCZ08]\)
- Goal models, like KAOS \([BWS^+10]\) \(\leadsto\) requirements@run.time
Relations between Runtime Models I

Goal Model
[WMYM09]

Architectural Model
Relations between Runtime Models I

Goal Model [WMYM09] ⇐⇒ Architectural Model

Thomas Vogel | MRT 2010 | 05.10.2010
Relations between Runtime Models I

Context and Resource Model

- send email
- load login form
- process send mail request
- send message
- get compose page
- report IMAP error
- process user login
- show compose page
- show form
- enter form
- start webmail

deployed on

Mail Server Connection

Web Session

User Interface

User Authentication
Feature Model \cite{CGFP09}

Configuration Space and Variability Model
Relations between Runtime Models II

Feature Model
Configuration Space and Variability Model

PervML Model
Configuration and Architectural Model

[CGFP09]
Failure Metamodel (PIM) refinement EJB Metamodel (PSM) abstraction [VG10]
Failure Metamodel
Configuration and Architectural Model

Performance Metamodel
Configuration and Architectural Model

overlaps

[VDN+10, VG10]
• Kind of models and relations depend on the concrete approach
• It’s likely that multiple models are used (vs. one model)
• Rather than isolated models, network of runtime models
• Explicitly considering relations between models
  • E.g., (impact) analysis across related models

• Existing approaches do not explicitly address these issues (ad-hoc and code-based solutions)
• Model-driven solution?
Similar Issues in MDSD

**Model-Driven Software Development (MDSD)**

- A multitude of models and relations
- A multitude of changes
- **Consistency** among different models
- Example: Model-Driven Architecture

![Diagram showing transformation between PIM and PSM with changes and consistency checks](image-url)
Megamodels

“Good enough” Definition (Megamodel)

A megamodel is a model that contains models and relations between those models or between elements of those models.

- Makes relations explicit
- Basis for model-driven management of models and relations
- Research by Favre [Fav05] and Bézivin et al. [BGMR03, BJV04, BFB07]
Megamodel Concepts

Organizational Purposes:
- Organizing and structuring models and relations
- Registry for models and their relations

Utilization Purposes:
- Navigation through different models in a model-driven manner
- Operational relations by means of executable units
Case Study: Self-Adaptive Software

Model Synchronization Engine

EJB Model

Causal Connection

Managed EJB System

Performance Model

Architectural Model

Failure Manager

Performance Manager

Architecture Manager

Failure Model
Case Study: Self-Adaptive Software
Case Study: Self-Adaptive Software

Megamodel

- Overlaps
- Performance Model
- Architectural Model
- Model Synchronization Engine
- EJB Model (Implementation Model)
- Causal Connection
- Managed EJB System

Performance Manager

Architectural Manager
Case Study: Self-Adaptive Software

Megamodel

Model Synchronization Engine

EJB Model (Implementation Model)

Causal Connection

Managed EJB System

Change Propagation

Performance Model

Performance Manager

Architectural Model

Architecture Manager
Conclusion

• Multiple runtime models for advanced self-adaptive systems

⇒ Categories of Runtime Models

• These models are not independent from each other

⇒ Relations between Runtime Models

• Explicitly considering models and relations

⇒ Megamodel concepts as a proposal

Future Work

• Elaborate categorization of models [FR07, Ben09, BBF09]

• Categorization of relations

• Applicability of our megamodel approach at runtime [SNG09]
Conclusion

- Multiple runtime models for advanced self-adaptive systems

⇒ Categories of Runtime Models
- These models are not independent from each other

⇒ Relations between Runtime Models
- Explicitly considering models and relations

⇒ Megamodel concepts as a proposal

Future Work

- Elaborate categorization of models [FR07, Ben09, BBF09]
- Categorization of relations
- Applicability of our megamodel approach at runtime [SNG09]
References I


References II


References III


