Multiple Runtime Models and their Relations for Self-Managing Systems

HPI/SAP Workshop
Walldorf, Germany, December 06, 2010

Thomas Vogel
HPI Research School
System Analysis & Modeling Group
Motivation

- **Continuous adaptation** of software to keep its value for the user (Laws of Software Evolution) [Lehman, 1996]
- (Increasing) **complexity** of software systems [Northrop et al., 2006]
- Maintenance & administration costs [Sterritt, 2005, Sommerville, 2007]
Motivation

- **Continuous adaptation** of software to keep its value for the user (Laws of Software Evolution) [Lehman, 1996]
- (Increasing) **complexity** of software systems [Northrop et al., 2006]
- Maintenance & administration costs [Sterritt, 2005, Sommerville, 2007]

**Self-Adaptive/-Managing Software** [Cheng et al., 2009]

*Systems that are able to adjust their behavior in response to their perception of the environment and the system itself.*

⇝ **Autonomic Computing**

[Kephart and Chess, 2003]
Self-Managing Systems

- Concepts originating from the control engineering discipline [Kokar et al., 1999, Diao et al., 2005]
- Self-healing/-optimization/-protection/-configuration [Lin et al., 2005]

Feedback Loop [Kephart and Chess, 2003]
The term Model-Driven Engineering (MDE) is typically used to describe software development approaches in which abstract models of software systems are created and systematically transformed to concrete implementations.

[France and Rumpe, 2007]
In our broad vision of MDE, models [...] are also the primary means by which developers and other systems understand, interact with, configure and modify the runtime behavior of software. [France and Rumpe, 2007]

Models@run.time [Blair et al., 2009]
In our broad vision of MDE, models [...] are also the primary means by which developers and other systems understand, interact with, configure and modify the runtime behavior of software. [France and Rumpe, 2007]

Models@run.time [Blair et al., 2009]
Self-Managing EJB Systems

- Component-based software systems
  - *Enterprise Java Beans 3* (EJB)

- Architectural monitoring and adaptation
  - components and connectors, above the level of *Java* objects

- Architectural models
  at runtime
EJB (Runtime) Metamodel
Abstract Runtime Metamodels

simplified

complex
detailed
multiple concerns
platform-specific
solution space
Abstract Runtime Metamodels

- Simplified vs. complex
- Abstract vs. detailed
- Platform-independent vs. platform-specific
- One concern vs. multiple concerns
- Problem space vs. solution space

Thomas Vogel | HPI/SAP Workshop | Dec 06, 2010
Abstract Runtime Models

Autonomic Manager

EJB Model

Causal Connection

Managed EJB System
Abstract Runtime Models

- Concern-specific, platform-independent models and managers
- Model Synchronization to maintain the runtime models
- Abstraction (monitoring) and refinement (adaptation)

Simplified EJB Metamodel
(Source Metamodel)

Failure Metamodel
(Target Metamodel)
- **Unidirectional vs. Bidirectional**
- **Transformation vs. Synchronization**
- **Bidirectional synchronization based on Triple Graph Grammars (TGG)** [Giese and Wagner, 2009, Giese and Hildebrandt, 2008]
- **Incremental** and event-driven solution applicable at runtime
Declarative rules
Automatic generation of operational rules
Abstraction gap between models: manually written code "extending" the rules for adaptation
→ MDE simplifies the development of maintaining multiple runtime models
Declarative rules
Automatic generation of operational rules
Abstraction gap between models: manually written code “extending” the rules for adaptation

→ MDE simplifies the development of maintaining multiple runtime models

Source Model | Target Model
Multiple Runtime Models...

...but how they are related?
Multiple Runtime Models...

...but how they are related? dependencies, trade-offs, ...?
Multiple Runtime Models...

...but how they are related? dependencies, trade-offs, ...?

...and how to describe and utilize relations?
“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 [Favre, 2005] and Bézivin et al. [Bézivin et al., 2003, Bézivin et al., 2004, Barbero et al., 2007]
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
Self-Managing EJB System

Megamodel

Performance Manager

Performance Model

Architectural Model

Architecture Manager

Model Synchronization Engine

EJB Model (Implementation Model)

Causal Connection

Managed EJB System
Self-Managing EJB System

Megamodel

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

Performance Manager

Architecture Manager
Self-Managing EJB System

Megamodel

- Performance Manager
- Change Propagation
- Performance Model
- Architectural Model
- Model Synchronization Engine
- EJB Model (Implementation Model)
- Causal Connection
- Managed EJB System
Implementation

Performance Manager

Performance Model

Architectural Model

Architecture Manager

Failure Manager

Failure Model

Model Synchronization Engine

EJB Model

Causal Connection

Managed EJB System
Implementation

Model Transformation Engine (MoTE)  
Story Diagram Modeling (SDM) Tools

mKernel  
[Bruhn et al., 2008]
Conclusion and Future Work

Conclusion

- Working approach using **multiple runtime models** for architectural monitoring and adaptation
- **Model Synchronization** techniques to maintain these models
- Initial ideas on megamodel concepts at runtime

Future Work

- Architected self-managing systems
- Semantics of models and model operations
- Describing and utilizing relations at runtime

Thank You!
Conclusion and Future Work

Conclusion

- Working approach using multiple runtime models for architectural monitoring and adaptation
- Model Synchronization techniques to maintain these models
- Initial ideas on megamodel concepts at runtime

Future Work

- Architecting self-managing systems
- Semantics of models and model operations
- Describing and utilizing relations at runtime
Conclusion and Future Work

Conclusion

- Working approach using **multiple runtime models** for architectural monitoring and adaptation
- **Model Synchronization** techniques to maintain these models
- Initial ideas on megamodelling concepts at runtime

Future Work

- Architecting self-managing systems
- Semantics of models and model operations
- Describing and utilizing relations at runtime

Thank You!
References

Traceability and Provenance Issues in Global Model Management.

MDA components: Challenges and Opportunities.
In 1st Intl. Workshop on Metamodelling for MDA, pages 23–41.

On the Need for Megamodels.

Models@run.time: Guest Editors’ Introduction.

Comprehensive support for management of Enterprise Applications.


In Language Engineering for Model-Driven Software Development, number 04101 in Dagstuhl Seminar Proceedings. IBFI, Schloss Dagstuhl.

Model-driven Development of Complex Software: A Research Roadmap.
In Proc. of the ICSE Workshop on Future of Software Engineering (FOSE), pages 37–54. IEEE.

Incremental Model Synchronization for Multiple Updates.
In Proc. of the 3rd Intl. Workshop on Graph and Model Transformation. ACM.

From Model Transformation to Incremental Bidirectional Model Synchronization.
Software and Systems Modeling, 8(1).

The Vision of Autonomic Computing.

Control Theory-Based Foundations of Self-Controlling Software.
References II


