HPI Hasso Plattner Institut

IT Systems Engineering | Universität Potsdam

Model-Based Self-Adaptation of Service-Oriented Software Systems

GK Workshop 2010

Schloss Dagstuhl, June 2, 2010

Thomas Vogel

Research School on Service-Oriented Systems Engineering

System Analysis and 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 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.

$\leadsto$ Autonomic Computing

[Kephart and Chess, 2003]

Self-Adaptive Software Systems

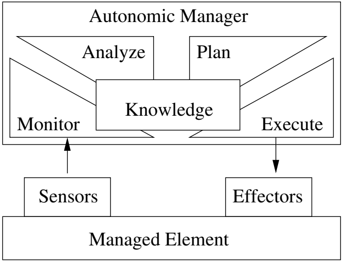


Figure: Feedback Loop [Kephart and Chess, 2003]

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

Service-Oriented Computing. . . [Papazoglou et al., 2007]

. . . promotes the idea of assembling application components into a network of services that can be loosely coupled to create flexible, dynamic business processes and agile applications.

- Composition of loosely-coupled services → modularity
- Self-containment of services (well-defined interfaces/contracts)
- Dynamic binding

→ **Basic support for architectural adaptation at runtime**

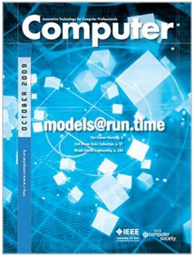
→ **Suitable abstraction mechanism for self-adaptation**

[Nitto et al., 2008]

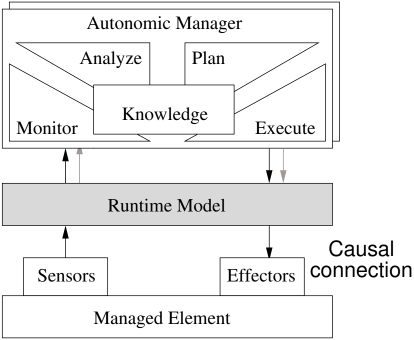
Model-Driven Engineering (MDE)

In our broad vision of MDE, models are not only the primary artifacts of development, they are also the ***primary means by which developers and other systems understand, interact with, configure and modify the runtime behavior of software***.

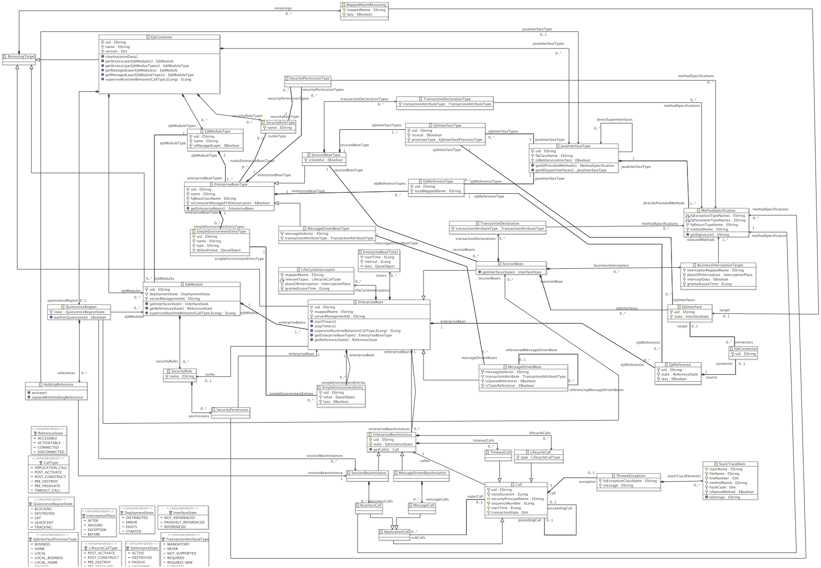
[France and Rumpe, 2007]



Special issue on models@run.time (Oct 2009)

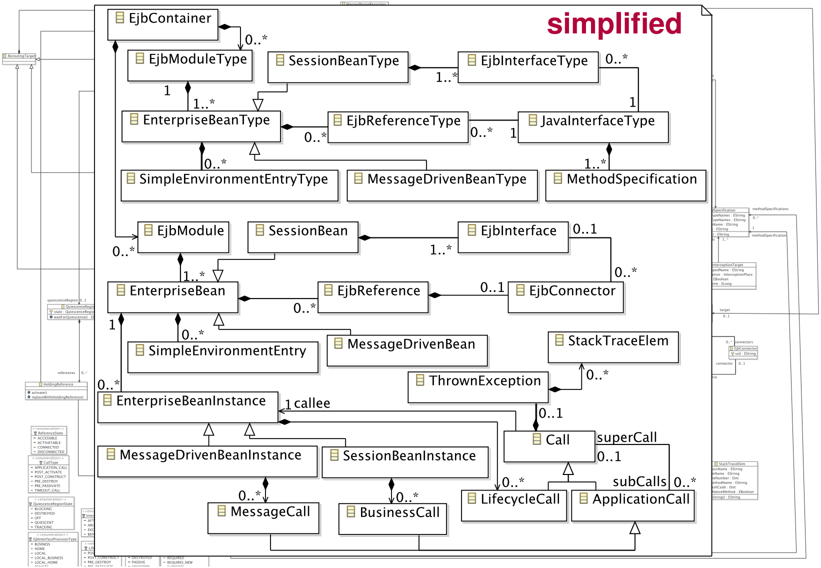


Managing EJB-based Services

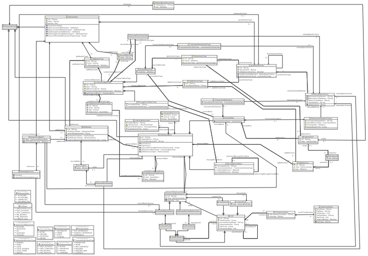


Managing EJB-based Services

Hasso Plattner Institut



Abstract Runtime Models



complex

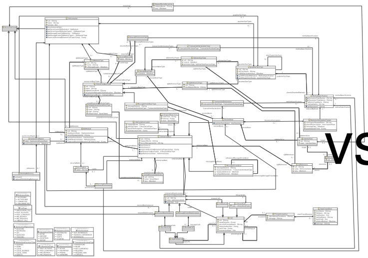
detailed

platform-specific

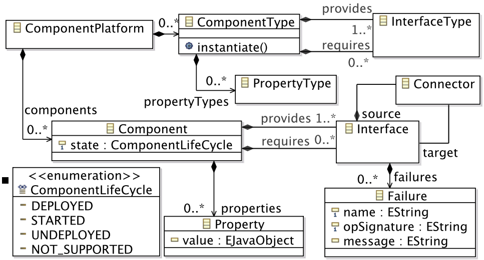
solution space

Abstract Runtime Models

Hasso Plattner Institut



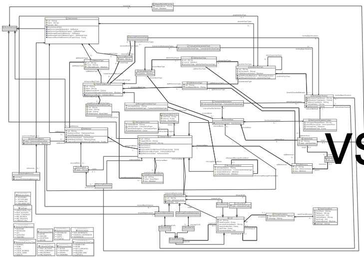
VS.



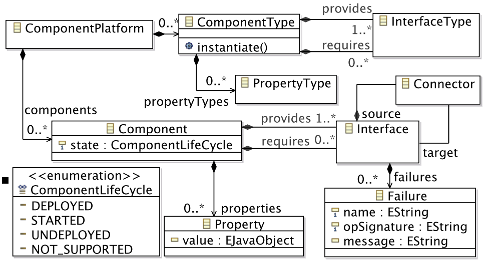
complex	*less complex*
detailed	*abstract*
platform-specific	*platform-independent*
solution space	*problem space*

Abstract Runtime Models

Hasso Plattner Institut



VS.



complex	*less complex*
detailed	*abstract*
platform-specific	*platform-independent*
solution space	*problem space*
Metamodel for a Source Model	**Metamodel for a Target Model**

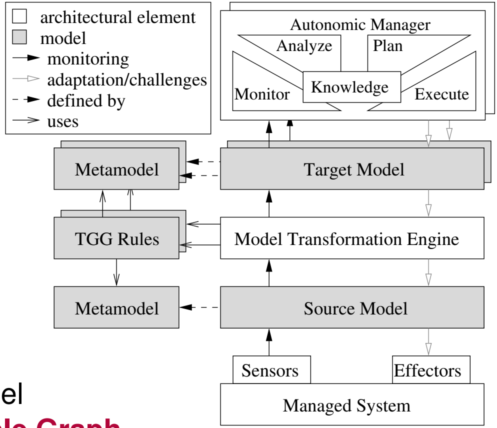
MDE for Self-Adaptive Systems

Hasso Plattner Institut

Different runtime models for **monitoring** [Vogel et al., 2010]

- performance,
- exceptions and
- architectural constraints,

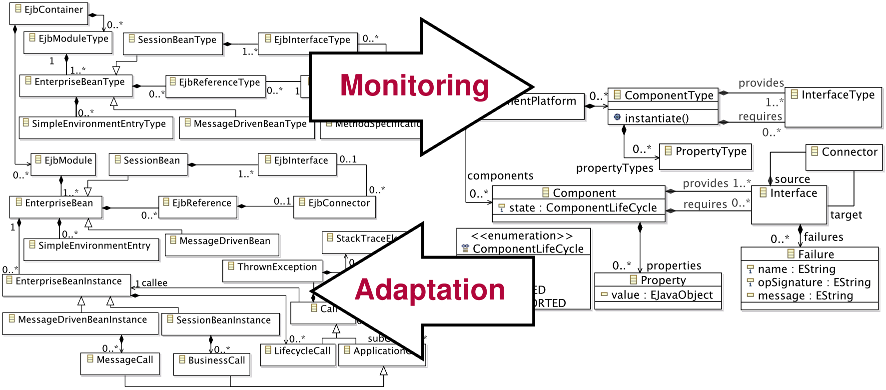
and for **adapting** [Vogel and Giese, 2010] service implementations.



[Vogel et al., 2009]

Incremental, bidirectional model synchronization based on **Triple Graph Grammars** (TGG).

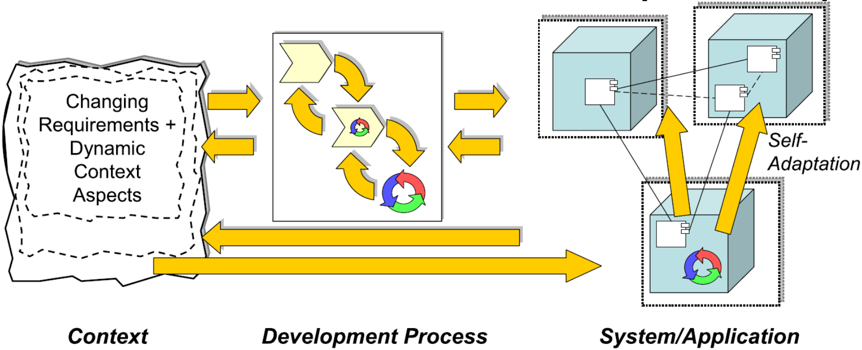
Runtime Model Synchronization



Current and Future Work

Hasso Plattner Institut

[Nitto et al., 2008]



- Model-driven development + runtime management
- Distributed managed and managing systems
  ⇝ distributing models and MDE techniques

Backup

Models at Runtime

model@run.time [Blair et al., 2009]

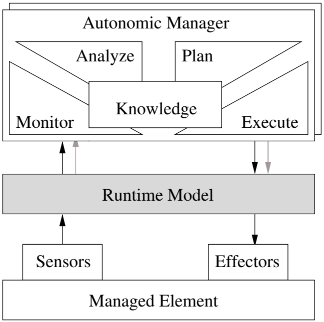
A model@run.time is a causally connected self-representation of the associated system that emphasizes the structure, behavior, or goals of the system from a problem space perspective.

- Causal connection $\rightsquigarrow$ reflection [Maes, 1987]
- Higher levels of abstraction and **problem space** perspective vs. low level models based on the **solution space** as in reflection
- Integrated into an MDE development approach: relation of runtime models to models from the development phase

Related Work

Architectural model as a runtime representation:

- One-to-one mapping between implementation classes and model elements [Oreizy et al., 1998]
- Focused on one concern of interest [Caporuscio et al., 2007, Dubus and Merle, 2006, Morin et al., 2009]
- All concerns of interests [Garlan et al., 2004]

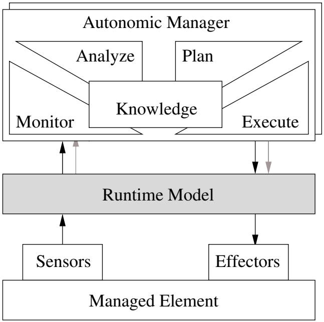


Is one runtime model enough?

Hasso Plattner Institut

Pros

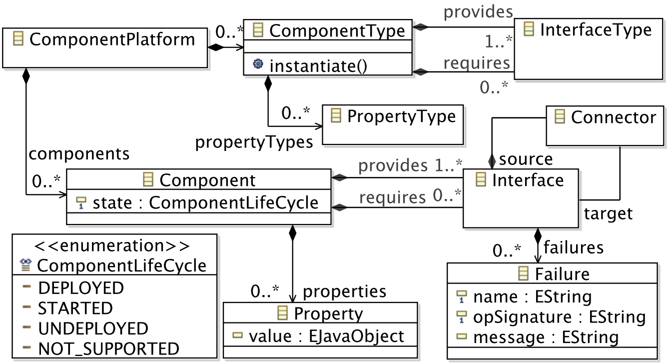
- Easing the connection between the model and the running system
- Avoiding the maintenance of several models



Cons

- Complexity of the model (all concerns + low level of abstraction)
- Platform- and implementation-specific model (solution space)
- Reusability of autonomic managers

Failure Target Metamodel



- Abstract and platform-independent model
- Architecture + occurred failures: self-healing
- Simplified as three associations are hidden

Model Transformation

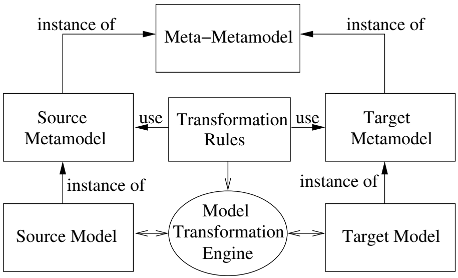
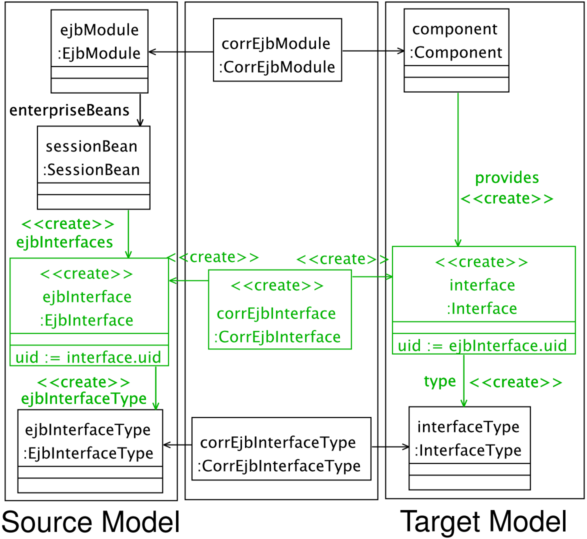


Figure: Generic Model Transformation System

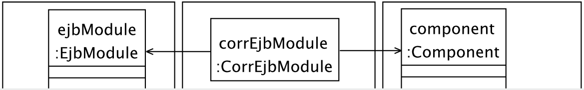
- Transformation vs. Synchronization
- Unidirectional vs. Bidirectional
- Bidirectional synchronization based on **Triple Graph Grammars** [Giese and Wagner, 2009, Giese and Hildebrandt, 2008]

Triple Graph Grammar Rule



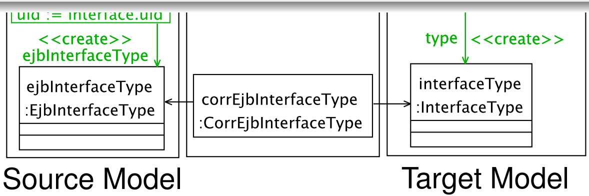
Triple Graph Grammar Rule

Hasso Plattner Institut



- 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 several runtime models**



Future Work

MDE for Self-Adaptive Systems

- Connect development phase with the runtime phase
- Development (requirements, design,. . .) & runtime models
- Elaborating on model-driven managing elements
- Operational environment/context

Large-scale, distributed system

- Distributed managed and managing elements
- Decentralized mgmt tasks [Papazoglou and Georgakopoulos, 2003]
- Distributing models and MDE techniques
- Local autonomy vs. global consistency/goals [Kramer and Magee, 2007]

References I

[Blair et al., 2009] Blair, G., Bencomo, N., and France, R. B. (2009).
Models@run.time.
Computer, 42(10):22–27.

[Caporuscio et al., 2007] Caporuscio, M., Marco, A. D., and Inverardi, P. (2007).
Model-based system reconfiguration for dynamic performance management.
Journal of Systems and Software, 80(4):455 – 473.

[Cheng et al., 2009] Cheng, B. H., de Lemos, R., Giese, H., Inverardi, P., Magee, J., and et al. (2009).
Software Engineering for Self-Adaptive Systems: A Research Road Map.
In *Software Engineering for Self-Adaptive Systems*, volume 5525 of *LNCS*, pages 1–26. Springer.

[Diao et al., 2005] Diao, Y., Hellerstein, J. L., Parekh, S., Griffith, R., Kaiser, G., and Phung, D. (2005).
Self-Managing Systems: A Control Theory Foundation.
In *ECBS '05: Proceedings of the 12th IEEE International Conference and Workshops on Engineering of Computer-Based Systems*, pages 441–448, Washington, DC, USA. IEEE Computer Society.

[Dubus and Merle, 2006] Dubus, J. and Merle, P. (2006).
Applying OMG D&C Specification and ECA Rules for Autonomous Distributed Component-based Systems.
In *Proc. of the 1st Intl. Workshop on Models@run.time*.

[France and Rumpe, 2007] France, R. and Rumpe, B. (2007).
Model-driven Development of Complex Software: A Research Roadmap.
In *FOSE '07: 2007 Future of Software Engineering*, pages 37–54, Washington, DC, USA. IEEE Computer Society.

[Garlan et al., 2004] Garlan, D., Cheng, S.-W., Huang, A.-C., Schmerl, B., and Steenkiste, P. (2004).
Rainbow: Architecture-Based Self-Adaptation with Reusable Infrastructure.
Computer, 37(10):46–54.

[Giese and Hildebrandt, 2008] Giese, H. and Hildebrandt, S. (2008).
Incremental Model Synchronization for Multiple Updates.
In *Proc. of the 3rd Intl. Workshop on Graph and Model Transformation*. ACM.

[Giese and Wagner, 2009] Giese, H. and Wagner, R. (2009).
From Model Transformation to Incremental Bidirectional Model Synchronization.
Software and Systems Modeling, 8(1).

[Kephart and Chess, 2003] Kephart, J. and Chess, D. (2003).
The Vision of Autonomic Computing.
IEEE Computer, 36(1):41–50.

References II

[Kokar et al., 1999] Kokar, M. M., Baclawski, K., and Eracar, Y. A. (1999).
Control Theory-Based Foundations of Self-Controlling Software.
Intelligent Systems and their Applications, IEEE, 14(3):37–45.

[Kramer and Magee, 2007] Kramer, J. and Magee, J. (2007).
Self-Managed Systems: an Architectural Challenge.
In *Proc. of the ICSE Workshop on Future of Software Engineering*, pages 259–268. IEEE.

[Lehman, 1996] Lehman, M. M. (1996).
Laws of Software Evolution Revisited.
In Montangero, C., editor, *Software Process Technology, 5th European Workshop, EWSPT'96, Nancy, France, October 9-11, 1996, Proceedings*, volume 1149 of *Lecture Notes in Computer Science*, pages 108–124. Springer.

[Lin et al., 2005] Lin, P., MacArthur, A., and Leaney, J. (2005).
Defining Autonomic Computing: A Software Engineering Perspective.
In *ASWEC '05: Proceedings of the 2005 Australian conference on Software Engineering*, pages 88–97, Washington, DC, USA. IEEE Computer Society.

[Maes, 1987] Maes, P. (1987).
Concepts and experiments in computational reflection.
SIGPLAN Not., 22(12):147–155.

[Morin et al., 2009] Morin, B., Barais, O., Jézéquel, J.-M., Fleurey, F., and Solberg, A. (2009).
Models@Run.time to Support Dynamic Adaptation.
Computer, 42(10):44–51.

[Nitto et al., 2008] Nitto, E. D., Ghezzi, C., Metzger, A., Papazoglou, M., and Pohl, K. (2008).
A journey to highly dynamic, self-adaptive service-based applications.
Automated Software Engineering, 15(3-4):313–341.

[Northrop et al., 2006] Northrop, L., Feiler, P. H., Gabriel, R. P., Linger, R., Longstaff, T., Kazman, R., Klein, M., and Schmidt, D. (2006).
Ultra-Large-Scale Systems: The Software Challenge of the Future.
Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA.

[Oreizy et al., 1998] Oreizy, P., Medvidovic, N., and Taylor, R. N. (1998).
Architecture-based Runtime Software Evolution.
In *Proc. of the 20th Intl. Conference on Software Engineering*, pages 177–186. IEEE.

[Papazoglou and Georgakopoulos, 2003] Papazoglou, M. P. and Georgakopoulos, D. (October 2003).
Service-oriented computing.
Commun. ACM, 46(10).

References III

[Papazoglou et al., 2007] Papazoglou, M. P., Traverso, P., Dustdar, S., and Leymann, F. (2007).
Service-Oriented Computing: State of the Art and Research Challenges.
Computer, 40(11):38–45.

[Sommerville, 2007] Sommerville, I. (2007).
Software Engineering.
Addison Wesley, 8 edition.

[Sterritt, 2005] Sterritt, R. (2005).
Autonomic computing.
Innovations in Systems and Software Engineering, 1(1):79–88.

[Vogel and Giese, 2010] Vogel, T. and Giese, H. (2010).
Adaptation and abstract runtime models.
In *Proceedings of the 5th Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2010) at the 32nd IEEE/ACM International Conference on Software Engineering (ICSE 2010), Cape Town, South Africa*, pages 39–48. ACM.

[Vogel et al., 2009] Vogel, T., Neumann, S., Hildebrandt, S., Giese, H., and Becker, B. (2009).
Model-Driven Architectural Monitoring and Adaptation for Autonomic Systems.
In *Proc. of the 6th Intl. Conference on Autonomic Computing and Communications*, pages 67–68. ACM.

[Vogel et al., 2010] Vogel, T., Neumann, S., Hildebrandt, S., Giese, H., and Becker, B. (2010).
Incremental Model Synchronization for Efficient Run-Time Monitoring.
In Ghosh, S., editor, *Models in Software Engineering, Workshops and Symposia at MODELS 2009, Reports and Revised Selected Papers*, volume 6002 of *LNCS*, pages 124–139. Springer.