Incremental Model Synchronization for Efficient Run-time Monitoring

4th International Workshop on Models@run.time

Denver, Colorado, USA, Oct 5, 2009

Thomas Vogel, Stefan Neumann, Stephan Hildebrandt,
Holger Giese, and Basil Becker
Hasso Plattner Institute
University of Potsdam

HPI

Motivation

- Self-adaptive software [Cheng et al., 2008] and autonomic computing [Kephart and Chess, 2003]
- Parameter & architectural adaptations [McKinley et al., 2004]

→ Monitoring parameters & architecture: **model of a running system**

- Capabilities: self-configuration, self-healing, self-optimization, self-protection [Lin et al., 2005]

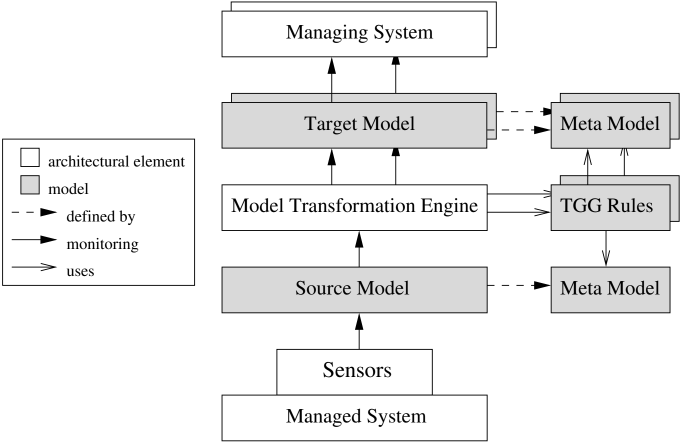
→ Monitoring with respect to different aspects: **different models**

- Runtime

→ Efficient solution: **incremental techniques**

Incremental Model Synchronization for Efficient Run-time Monitoring

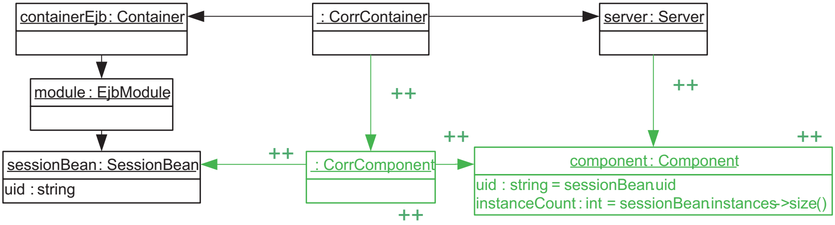
Generic Architecture



- Model Transformation Engine based on **Triple Graph Grammars (TGG)** [Giese and Hildebrandt, 2008, Giese and Wagner, 2009]

Incremental Model Synchronization

- Triple graph grammars: source, correspondence, and target model
- Example TGG Rule:

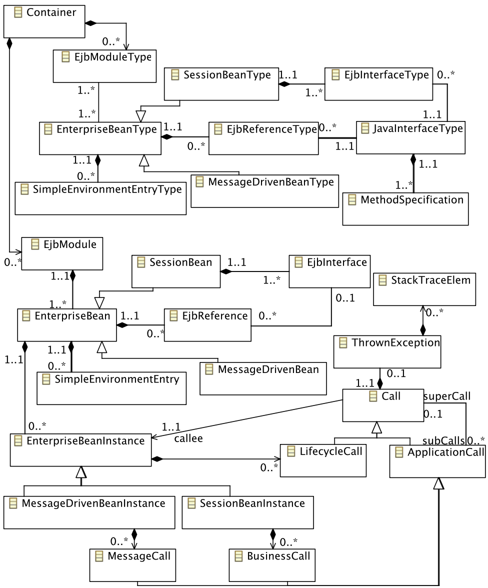


- Incremental synchronization: event-driven, local sync. strategies
- Automatic generation of operational rules from TGG rules

Implementation

- Engine and models are based on the **Eclipse Modeling Framework** (EMF) (decoupled from the Eclipse workbench)
- Models conform to EMF meta models
- Managed systems are **Enterprise Java Beans 3.0** (EJB) applications
- EJB infrastructure **mKernel** provides sensors as an API [Bruhn et al., 2008] for the **Glassfish** application server

Source Meta Model for EJB

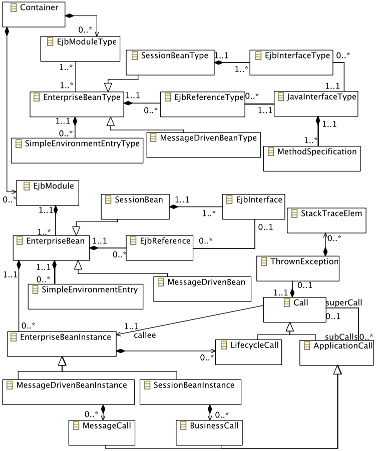


Complexity:

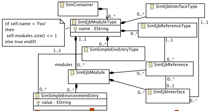
- Types of components
- Deployed components and their configurations
- Concrete instances and interactions

→ **Abstraction for different aspects**

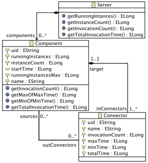
Multiple Run-time Models



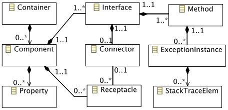
Source Meta Model



Architectural constraints for self-configuration



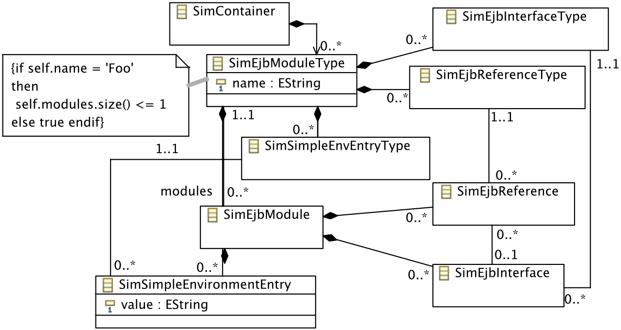
Performance for self-optimization



Failures for self-healing

HPI

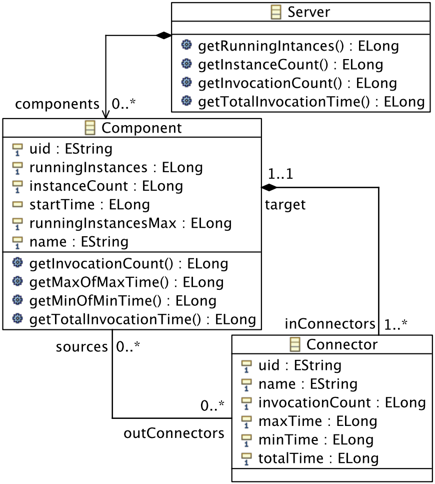
Architectural Target Meta Model



Self-Configuration

- Simplified run-time architectures of EJB-based applications
- Checking architectural constraints using OCL

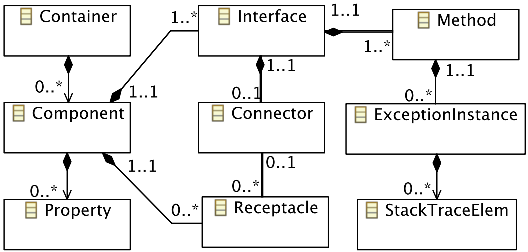
Performance Target Meta Model



Self-Optimization

- Architectural information enriched with performance data

Failure Target Meta Model



Self-Healing

- Architectural information enriched with occurred failures

Related Work

- Maintaining run-time models non-inrementally [Hein et al., 2007]
- Single view provided by run-time models [Dubus and Merle, 2006, Morin et al., 2008]
- No advanced model-driven techniques, like model transformation [Dubus and Merle, 2006, Hein et al., 2007, Morin et al., 2008]
- Model transformation at run-time [Song et al., 2008]
  - File-based synchronizations (*MediniQVT*)
  - Source model does not seem to be maintained at run-time and therefore non-incremental synchronizations seem to be involved

Conclusion & Future Work

Conclusion

- Multiple run-time models for monitoring
- Incremental model synchronization at run-time
- Efficient solution (for evaluation see paper)

Future Work

- Incremental model synchronization for adaptations
- Architectural adaptations

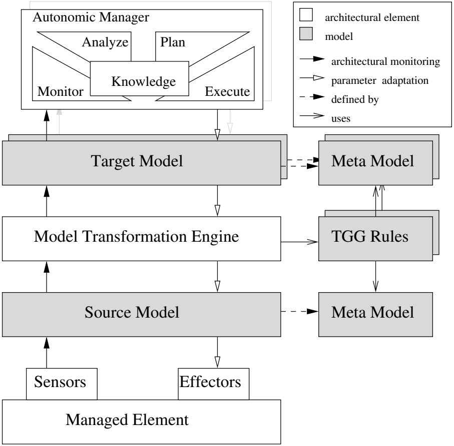


Figure: Monitoring and Adaptations [Vogel et al., 2009]

Literature

[Bruhn et al., 2008] Bruhn, J., Niklaus, C., Vogel, T., and Wirtz, G. (2008).
Comprehensive support for management of Enterprise Applications.
In *Proc. of the 6th ACS/IEEE Intl. Conference on Computer Systems and Applications*, pages 755–762. IEEE.

[Cheng et al., 2008] Cheng, B., de Lemos, R., Giese, H., Inverardi, P., Magee, J., and et al. (2008).
Software Engineering for Self-Adaptive Systems: A Research Road Map.
Number 08031 in Dagstuhl Seminar Proceedings.

[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 1st Intl. Workshop on Models@run.time*.

[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).

[Hein et al., 2007] Hein, C., Ritter, T., and Wagner, M. (2007).
System Monitoring using Constraint Checking as part of Model Based System Management.
In *Proc. of 2nd Intl. Workshop on Models@run.time*.

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

[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.

[McKinley et al., 2004] McKinley, P. K., Sadjadi, S. M., Kasten, E. P., and Cheng, B. H. C. (2004).
Composing Adaptive Software.
IEEE Computer, 37(7).

[Morin et al., 2008] Morin, B., Barais, O., and Jézéquel, J.-M. (2008).
K@RT: An Aspect-Oriented and Model-Oriented Framework for Dynamic Software Product Lines.
In *Proc. of the 3rd Intl. Workshop on Models@run.time*, pages 127–136.

[Song et al., 2008] Song, H., Xiong, Y., Hu, Z., Huang, G., and Mei, H. (2008).
A model-driven framework for constructing runtime architecture infrastructures.
Technical Report GRACE-TR-2008-05, GRACE Center, National Institute of Informatics, Japan.

[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.

Backup

Incremental Model Synchronization

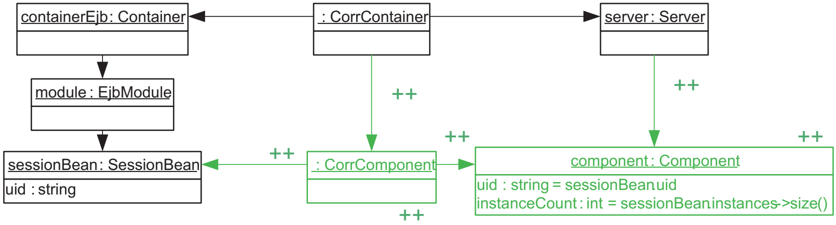
- Notification mechanism to efficiently detect modifications of source model elements
- Notification contains the relevant correspondence model element
- Correspondence model to efficiently navigate between source and target model
- Check consistency of source and target model elements
- Modification of the target model to reestablish consistency
    - Attribute values, links, nodes
- Queueing of notifications and on-demand synchronization

→ **Incremental synchronization of a source and a target model**

TGG Rules for Performance Meta Model

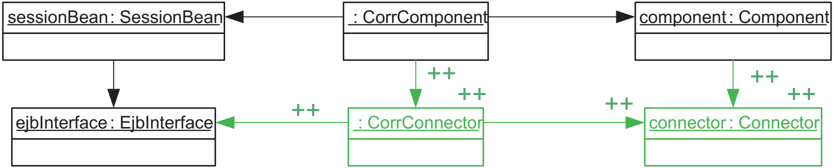
1st rule (axiom):

2nd rule:

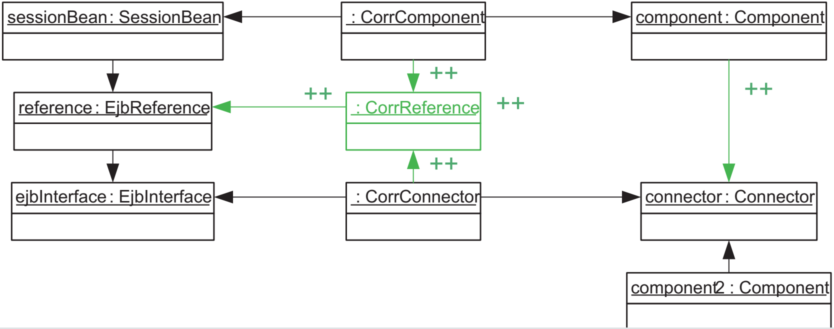


TGG Rules for Performance Meta Model (2)

3rd rule:



4th rule:



HPI

Evaluation

Comparing three approaches regarding development costs and performance:

- **Model-Driven Approach**
- **Non-Incremental Adapter (NIA)**
- **Incremental Adapter (IA)**

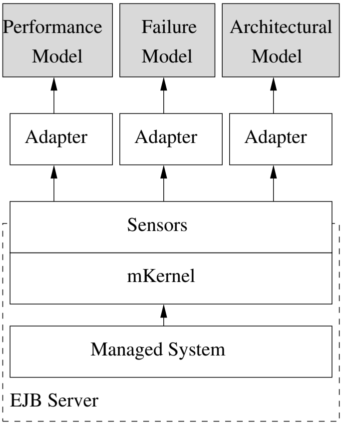


Figure: NIA/IA

Development Costs

Target Model	Model-Driven Approach Rules	Nodes/Rules	LOC	NIA LOC
Simpl. Architectural Model	9	7,44	15259	357
Performance Model	4	6,25	5979	253
Failure Model	7	7,14	12133	292
Sum	20		33371	902

- **Model-Driven Approach**: 2685 LOC for the EMF Adapter
- **Incremental Adapter**: 30k LOC?
- Declarative vs. imperative approaches
- **Non-Incremental Adapter (NIA)**: 20 TGG rules $\approx$ 902 LOC

Performance

Size	NIA		Model-Driven Approach						
	S	*B*	n=0	n=1	n=2	n=3	n=4	n=5	*B*
5	8037	20967	0	163	361	523	749	891	10733
10	9663	43054	0	152	272	457	585	790	23270
15	10811	72984	0	157	308	472	643	848	36488
20	12257	105671	0	170	325	481	623	820	55491
25	15311	142778	0	178	339	523	708	850	72531

[ms]

- **Size**: number of deployed beans
- Monitoring structural (**S**) and behavioral (**B**) aspects
- Pull-oriented (**S** & **B**) and push-oriented (**S**) monitoring
- **n**: number of events reflecting structural changes
- **Model-Driven Approach:** Depending on n, 3.7% to 7.2% of the average times for structural monitoring are used for model synchronization, the rest for event processing.

Performance for Architectural Monitoring

Size	Model-Driven Approach					
	n=0	n=1	n=2	n=3	n=4	n=5
5	0	163	361	523	749	891
10	0	152	272	457	585	790
15	0	157	308	472	643	848
20	0	170	325	481	623	820
25	0	178	339	523	708	850
Event processing	0%	92.8%	94.1%	95.6%	95.2%	96.3%
Synchronization	0%	7.2%	5.9%	4.4%	4.8%	3.7%

[ms]

- **Size**: number of deployed beans
- Architectural monitoring through event-driven sensors
- Processing **n** events and invoking **once** the transformation engine

HPI