Runtime Models as Interfaces for Adapting Software Systems

Seminar on Software Engineering for Self-Adaptive Systems
Schloss Dagstuhl, October 24-29, 2010
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.

...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]

- Runtime models as interfaces for adaptation
- Typical or new MDE concepts and techniques for self-adaptive software systems (online vs. offline)
One Runtime Model as an Interface

[Sensors Effectors
Managed Element
Monitor
Analyze Plan
Execute
Autonomic Manager
Knowledge
Kephart and Chess, 2003]
A Metamodel for a Runtime Model

simplified

EjbContainer
EjbModuleType
SessionBeanType
EjbInterfaceType
EjbReferenceType
JavaInterfaceType
SimpleEnvironmentEntryType
MessageDrivenBeanType
MethodSpecification
EjbModule
SessionBean
EjbInterface
EjbReference
EjbConnector
EnterpriseBean
SimpleEnvironmentEntry
MessageDrivenBean
ThrownException
EnterpriseBeanInstance
MessageDrivenBeanInstance
SessionBeanInstance
Call
MessageCall
BusinessCall
LifecycleCall
ApplicationCall

Thomas Vogel | Dagstuhl 10431 | October 24-29, 2010
Abstract Runtime Models

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

complex
detailed
multiple concerns
platform-specific
solution space

VS.

less complex
abstract
one concern
platform-independent
problem space
Abstract Runtime Models

complex
detailed
multiple concerns
platform-specific
solution space

Metamodel for a Source Model

vs.

less complex
abstract
one concern
platform-independent
problem space

Metamodel for a Target Model
Different runtime models for monitoring

- performance,
- failures, and
- architectural constraints,

and for parameter and structural adaptation.

Incremental, bidirectional model synchronization based on Triple Graph Grammars (TGG).
Runtime Model Synchronization

Further reading: ICAC’09, Models@run.time’09, SEAMS’10
Ongoing and Future Work

Which kind of models at which level of abstraction for which adaptation/management task?  

Architecting self-adaptive systems

Using models as interfaces for managed systems and within autonomic managers, how to specify the **semantics** of models and model operations?
Ongoing and Future Work cont’d

Distributed Self-Adaptive Systems ➞ Self-organizing systems

- Distributed managed *and* managing systems
- Distribution vs. consistency
- **Distributed models and MDE techniques**
- Specialized models for
  - monitoring and adapting managed systems
  - coordination between autonomic managers
- ...
Distributed Self-Adaptive Systems $\rightsquigarrow$ Self-organizing systems

- Distributed managed and managing systems
- Distribution vs. consistency
- Distributed models and MDE techniques
- Specialized models for
  - monitoring and adapting managed systems
  - coordination between autonomic managers
  - ...

Thomas Vogel | Dagstuhl 10431 | October 24-29, 2010
References


