Language and Framework Requirements for Adaptation Models

6th International Workshop on Models@run.time

Wellington, New Zealand, October 17, 2011

Thomas Vogel and Holger Giese

System Analysis and Modeling Group
Hasso Plattner Institute
University of Potsdam, Germany

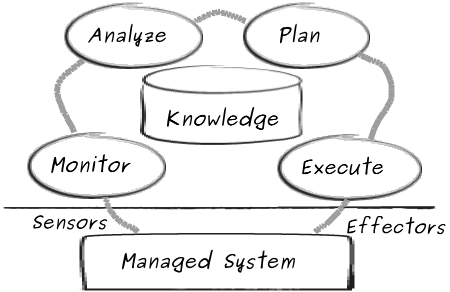
HPI

Introduction

Models@run.time for Self-adaptive Software

MDE & Models at Runtime for

- Knowledge
- Feedback Loop activities



Feedback Loop [Kephart and Chess, 2003]

Motivation

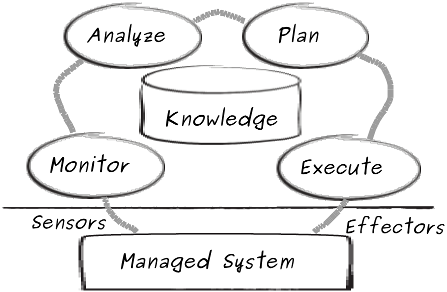
Models@run.time for Self-adaptive Software

- Focus on causal connection
  (e.g., discussions at MRT'09 and '10)

⇒ *Monitor* and *Execute*

- **Reusing** or **applying** existing techniques for decision-making
  (rule-based or search-based)

⇒ *Analyze* and *Plan*



Feedback Loop [Kephart and Chess, 2003]

Related Work

Example solutions:

- rule-based: ECA, policies
- search-based: Utility functions, goals

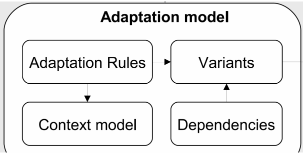
Characteristics (requirements):

- Performance
- Support for validation
- Scalability

Stitch [Cheng, 2008]

- Requirements!
- Policy-based language
- System administration tasks

```
RULE R_M
EVENT
   A new node N is detected onto the Platform
CONDITION
   N.profile == PDA
ACTION
   knowledge.domain.addNode(N)
```




```
rule BecomeDA : // Becomes a DA
  condition ElectedDA and not LowBatt and not DA
  effect DA
```

```
AdaptionPolicy ReplaceFiring
  (Description "Replaces firing component")
  (Observation energyReport (energy < 60))
  (Response RemoveComponent ReactiveFire)
  (Response A
    Distance
    Reactive
```

```
handsfree_util =
  if (context.handsfree AND STapp.handsfree) or
     (!context.handsfree AND !STapp.handsfree_offered) then 1 else 0
response_util =
  if (context.response >= STapp. response) then 1
  else 1 - ( (STapp.response - context.resonse) / STapp.response)
utility =
  if STapp.mem > context.mem then 0
  else weight_hf * handsfree_util + weight_rsp * response_util
```

[Dubus and Merle, 2006, Morin et al., 2008, Fleurey et al., 2009, Georgas et al., 2009, Floch et al., 2006]

Related Work

Example solutions:

- rule-based: ECA, policies
- search-based: Utility functions, goals

Cha

-
-
- Scalability

Stitch [Cheng, 2008]

- Requirements!
- Policy-based language
- System administration tasks

No systematic investigation of requirements for analysis and planning activities in conjunction with models@run.time

```
RULE R_M
EVENT
  A new node N is detected onto the Platform
CONDITION
  N.profile == PDA
ACTION
  knowledge.domain.addNode(N)
```

```
rule BecomeDA : // Becomes a DA
  condition ElectedDA and not LowBatt and not DA
  effect DA
```

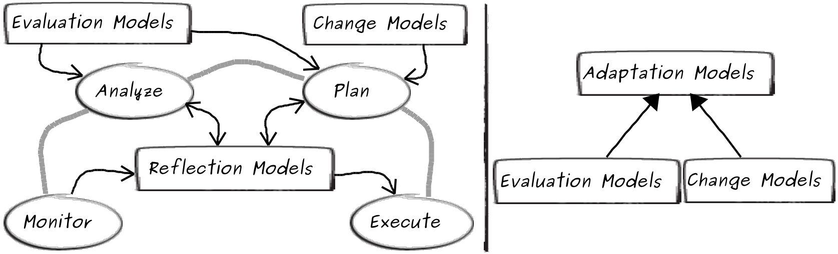
```
AdaptionPolicy ReplaceFiring
  (Description "Replaces firing component")
  (Observation energyReport (energy < 60))
  (Response RemoveComponent ReactiveFire)
  (Response 
    Distanc
    Reactiv
```

```
handsfree_util =
  if (context.handsfree AND STapp.handsfree) or
     (!context.handsfree AND !STapp.handsfree_offered) then 1 else 0
response_util =
  if (context.response >= STapp. response) then 1
  else 1 - ( (STapp.response - context.resonse) / STapp.response)
utility =
  if STapp.mem > context.mem then 0
  else weight_hf * handsfree_util + weight_rsp * response_util
```

[Dubus and Merle, 2006, Morin et al., 2008, Fleurey et al., 2009, Georgas et al., 2009, Floch et al., 2006]

Adaptation Models

MDE and models@run.time perspective (MODELS'10 Workshops)



Requirements for adaptation models concerning:

- **Languages** (meta-models, constraints, model operations etc.)
- **Frameworks** (execution environment)

Note: Not claiming a *complete* enumeration or *finalized* definitions

Language Requirements (LR)

Functional LR	
LR-1 *Functional Specification/Goals*	***LR-6*** *Evaluation Conditions*
LR-2 *Quality Dimensions*	***LR-7*** *Evaluation Results*
LR-3 *Preferences*	***LR-8*** *Adaptation Options*
LR-4 *Access to Reflection Models*	***LR-9*** *Adaptation Conditions*
LR-5 *Events*	***LR-10*** *Adaptation Costs/Benefits*
	LR-11 *History of Decisions*

⇒ Concepts contained or referenced by adaptation models
⇒ Expressiveness of the language

Non-functional LR	
LR-12 *Modularity, Abstractions, Scalability*	***LR-15*** *Formality*
LR-13 *Side Effects*	***LR-16*** *Reusability*
LR-14 *Parameters*	***LR-17*** *Ease of Use*

⇒ Quality of the language and adaptation models

Functional Language Requirements (I)

To-be specification of the running system (reference values)

LR-1 *Functional Specification/Goals*
Desired behavior, what the system should do

LR-2 *Quality Dimensions*
Desired QoS, how the system should be

LR-3 *Preferences*
Balancing competing quality dimensions or goals

Functional Language Requirements (I)

To-be specification of the running system (reference values)

LR-1 *Functional Specification/Goals*
Desired behavior, what the system should do

LR-2 *Quality Dimensions*
Desired QoS, how the system should be

LR-3 *Preferences*
Balancing competing quality dimensions or goals

As-Is situation of the running system

LR-4 *Access to Reflection Models*
Monitor & Execute changes through causally connected models

LR-5 *Events*
Trigger for analysis and planning; locating runtime phenomena

Functional Language Requirements (II)

Analysis of the running system

LR-6 *Evaluation Conditions*
Relate as-is (LR-4, 5) and to-be (LR-1, 2, 3) situations.

LR-7 *Evaluation Results*
Identify adaptation need, annotate reflection models (LR-4)

Functional Language Requirements (II)

Analysis of the running system

- ***LR-6*** *Evaluation Conditions*
  Relate as-is (LR-4, 5) and to-be (LR-1, 2, 3) situations.
- ***LR-7*** *Evaluation Results*
  Identify adaptation need, annotate reflection models (LR-4)

Planning of adaptation

- ***LR-8*** *Adaptation Options*
  Variability (config. space) and how to change reflection models
- ***LR-9*** *Adaptation Conditions*
  Applicability of adaptation options (by LR-4, 5, 7, 8)
- ***LR-10*** *Adaptation Costs and Benefits*
  Select options wrt goals, qualities and preferences (LR-1, 2, 3)
- ***LR-11*** *History of Decisions* wrt analysis and planning

Non-functional Language Requirements (I)

Characteristics and qualities of a language and models

LR-12 *Modularity, Abstractions and Scalability*
Composition of sub-models and different abstraction levels to promote scalability

LR-13 *Side Effects*
Explicit meta-information about side effects on reflection models $\rightsquigarrow$ consistency of the running system

LR-14 *Parameters*
Built-in mechanism to adjust adaptation models at runtime

Non-functional Language Requirements (II)

LR-15 *Formality*

How formal the modeling language should be?

⇝ Online or offline V&V of adaptation models

LR-16 *Reusability*

Degree of dependency between languages for adaptation models and reflection models

LR-17 *Ease of Use*

Modeling paradigm, notations, tools

⇝ Support engineers in creating, validating and verifying adaptation models

Framework Requirements (FR)

- Framework: Execution environment of adaptation models
- Specific requirements for executing/applying adaptation models

Framework Requirements	
FR-1 *Consistency*	***FR-4*** *Priorities*
FR-2 *Incrementality*	***FR-5*** *Time Scales*
FR-3 *Reversibility*	***FR-6*** *Flexibility*

Note: Typical non-functional requirements (reliability, security, etc.) of software are relevant for such frameworks as well, but left here.

Framework Requirements (I)

FR-1 *Consistency*

Preserve consistency of reflection models (running systems)

$\rightsquigarrow$ Conditions for performing adaptations (LR-9)

$\rightsquigarrow$ Transaction support for adaptation models

FR-2 *Incrementality*

For example,

- Locate need for analysis in reflection models by events
- Incremental planning
- Incrementally apply adaptation options on reflection models
- ...to avoid searching or copying potentially large models

FR-3 *Reversibility*

Reverse incremental operations (*do* and *undo* of operations)

Framework Requirements (II)

FR-4 *Priorities*

Organizing modular adaptation models by priorities, e.g., to order and analyze evaluation conditions based on criticality

FR-5 *Time Scales*

From exactly pre-defined adaptations for mission-critical situations to dynamically synthesizing adaptation plans

FR-6 *Flexibility*

Adapting adaptation models at runtime

$\rightsquigarrow$ Learning effects

$\rightsquigarrow$ Unanticipated scenarios

$\rightsquigarrow$ Hierarchical control

Adaptation Models and Feedback Loops

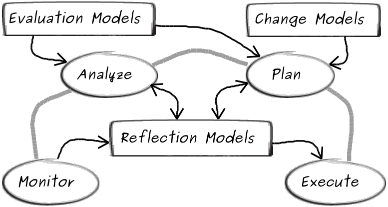
Language Requirements

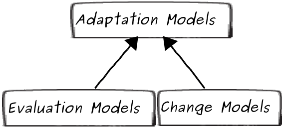
Functional LR	
LR-1 *Functional Specification/Goals*	***LR-6*** *Evaluation Conditions*
LR-2 *Quality Dimensions*	***LR-7*** *Evaluation Results*
LR-3 *Preferences*	***LR-8*** *Adaptation Options*
LR-4 *Access to Reflection Models*	***LR-9*** *Adaptation Conditions*
LR-5 *Events*	***LR-10*** *Adaptation Costs/Benefits*
	LR-11 *History of Decisions*

Non-functional LR	
LR-12 *Modularity, Abstractions, Scalability*	***LR-15*** *Formality*
LR-13 *Side Effects*	***LR-16*** *Reusability*
LR-14 *Parameters*	***LR-17*** *Ease of Use*

Framework Requirements

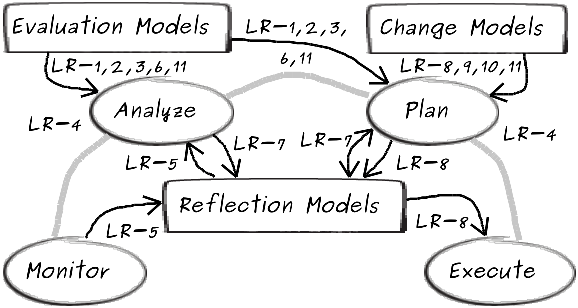
Framework Requirements	
FR-1 *Consistency*	***FR-4*** *Priorities*
FR-2 *Incrementality*	***FR-5*** *Time Scales*
FR-3 *Reversibility*	***FR-6*** *Flexibility*





Relationships between requirements and loops? ⇝ **loop “patterns”**

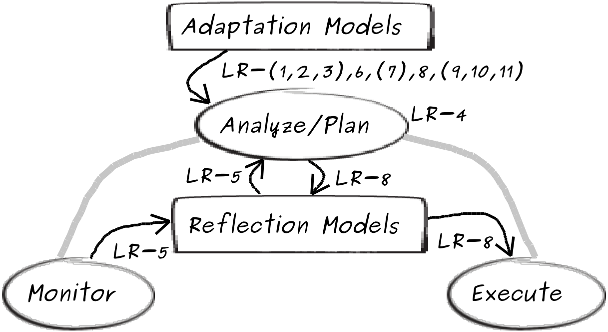
Decoupled Analysis and Planning



- Highlights LR where the corresponding concepts are relevant
- Explicitly covers all functional LR
- Rather sophisticated analysis and planning steps
- Rather longer time scales

⇝ **Search-based approaches**

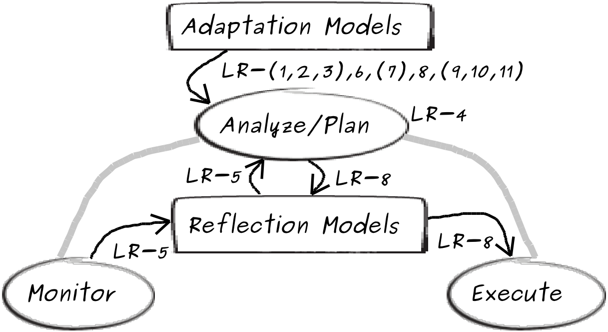
Coupled Analysis and Planning



- Highlights LR where the corresponding concepts are relevant
- LR written in brackets are only implicitly covered
- Precise specification of adaptation (like ECA $\approx$ LR-5, 6, 8)
- Rather short time scales

⇝ **Rule-based approaches**

Coupled Analysis and Planning



- Highlights LR where the corresponding concepts are relevant
- LR written in brackets are only implicitly covered
- Precise specification of adaptation (like ECA $\approx$ LR-5, 6, 8)
- Rather short time scales

⇝ **Rule-based approaches**

Extreme poles spanning a range of "patterns".

Conclusion and Future Work

Conclusion

- Adaptation models for self-adaptive software using MRT
- Language and framework requirements for adaptation models
- Adaptation models and feedback loops

Future Work

- Analyze existing approaches with respect to the requirements
- Engineer a language and framework for our approach
  (ICAC'09, MODELS'09 Workshops, SEAMS'10)
- Integration of multiple languages in a framework

References I

[Bencomo et al., 2011] Bencomo, N., Blair, G., Fleurey, F., and Jeanneret, C. (2011).
Summary of the 5th International Workshop on Models@run.time.
In Dingel, J. and Solberg, A., editors, *MODELS'10 Workshops*, volume 6627 of *LNCS*, pages 204–208. Springer.

[Bencomo et al., 2010] Bencomo, N., Blair, G., France, R., Munoz, F., and Jeanneret, C. (2010).
4th International Workshop on Models@run.time.
In Ghosh, S., editor, *MODELS'09 Workshops*, volume 6002 of *LNCS*, pages 119–123. Springer.

[Chauvel and Barais, 2007] Chauvel, F. and Barais, O. (2007).
Modelling Adaptation Policies for Self-Adaptive Component Architectures.
In *M-ADAPT'07*, pages 61–68.

[Cheng, 2008] Cheng, S.-W. (2008).
Rainbow: Cost-Effective Software Architecture-Based Self-Adaptation.
PhD thesis, Carnegie Mellon University, Pittsburgh, USA.

[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 *Models@run.time'06*.

[Fleurey et al., 2009] Fleurey, F., Dehlen, V., Bencomo, N., Morin, B., and Jézéquel, J.-M. (2009).
Modeling and Validating Dynamic Adaptation.
In Chaudron, M., editor, *MODELS'08 Workshops*, volume 5421 of *LNCS*, pages 97–108. Springer.

[Floch et al., 2006] Floch, J., Hallsteinsen, S., Stav, E., Eliassen, F., Lund, K., and Gjorven, E. (2006).
Using Architecture Models for Runtime Adaptability.
Software, 23(2):62–70.

[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.

[Georgas et al., 2009] Georgas, J. C., Hoek, A., and Taylor, R. N. (2009).
Using Architectural Models to Manage and Visualize Runtime Adaptation.
Computer, 42(10):52–60.

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

[Morin et al., 2008] Morin, B., Fleurey, F., Bencomo, N., Jézéquel, J.-M., Solberg, A., Dehlen, V., and Blair, G. (2008).
An Aspect-Oriented and Model-Driven Approach for Managing Dynamic Variability.
In Czarnecki, K., Ober, I., Bruel, J.-M., Uhl, A., and Völter, M., editors, *MODELS'08*, volume 5301 of *LNCS*, pages 782–796. Springer.

[Ramirez and Cheng, 2009] Ramirez, A. J. and Cheng, B. H. (2009).
Evolving Models at Run Time to Address Functional and Non-Functional Adaptation Requirements.
In *Models@run.time'09*, volume 509 of *CEUR-WS.org*, pages 31–40.

References II

[Song et al., 2010] Song, H., Xiong, Y., Chauvel, F., Huang, G., Hu, Z., and Mei, H. (2010).
Generating Synchronization Engines between Running Systems and Their Model-Based Views.
In Ghosh, S., editor, *MODELS'09 Workshops*, volume 6002 of *LNCS*, pages 140–154. Springer.

[Vogel and Giese, 2010] Vogel, T. and Giese, H. (2010).
Adaptation and Abstract Runtime Models.
In *SEAMS'10*, 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 *ICAC'09*, 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'09 Workshops*, volume 6002 of *LNCS*, pages 124–139. Springer.

[Vogel et al., 2011] Vogel, T., Seibel, A., and Giese, H. (2011).
The Role of Models and Megamodels at Runtime.
In Dingel, J. and Solberg, A., editors, *MODELS'10 Workshops*, volume 6627 of *LNCS*, pages 224–238. Springer.