Model-Driven Engineering of Self-Adaptive Software with EUREMA

Seminar 14382: Control Theory meets Software Engineering
Schloss Dagstuhl, Germany, September 16, 2014

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


2. Requirements for Modeling Feedback Loops

3. EUREMA
   - Single Feedback Loop
   - Multiple Feedback Loops
   - Layered Feedback Loops
   - Off-line Adaptation
   - Discussion

4. Conclusion
Models reflecting the adaptable software
Models specifying individual MAPE activities
Reflection Model (Architectural Model)
Setting

- Adaptable Software (Domain Logic)
- Sensing
  - Effecting
  - Adaptation Engine (Adaptation Logic)
- Self-Adaptive Software

**External Approach**

- Monitor
- Knowledge
- Analyze Plan
- Execute

**Adaptable Software**

- Sensors
  - Effectors
  - Adaptation Engine

**MAPE-K**

- Monitor
- Reflection
  - Models
  - Analyze Plan
  - Execute
  - Evaluation Models
  - Change Models
  - Monitoring Models
  - Execution Models

- Runtime Models in Feedback Loops

- How to capture the interplay of such models?
- Specifying and executing feedback loops
- Modeling language and interpreter

---

**Monitoring|Execution Model (TGG rule)**

```
    ejbModule : EjbModule
    enterpriseBeans
    sessionBean : SessionBean
    <<create>> ejbInterfaces
    <<create>> ejbInterface : EjbInterface
    uid := interface.uid
    <<create>> ejbInterfaceType : EjbInterfaceType

    corrEjbModule : CorrEjbModule
    <<create>> <<create>>
    corrEjbInterface : CorrEjbInterface
    corrEjbInterfaceType : CorrEjbInterfaceType

    component : Component
    provides <<create>>
    interface : Interface
    uid := ejbInterface.uid
    type <<create>>

    interfaceType : InterfaceType
```
1 Setting

Evaluation Model: find crashed components

- Model reflecting the adaptable software
- Models specifying individual MAPE activities
- How to capture the interplay of such models?
- Specifying and executing feedback loops
- Modeling language and interpreter
Change Model: redepoly crashed components

- Deploy the component
- Start the component
- Undeploy the component
- Delete the failure

Find critical failures:
- Errors: FailureRoot
- Component: CF1
- CriticalFailure1

In compdia: ComponentPlatform
in errors: FailureRoot
Models reflecting the adaptable software
Models specifying individual MAPE activities
How to capture the interplay of such models?
**Specifying and executing feedback loops**
**Modeling language and interpreter**
Requirements for Modeling Feedback Loops

- **Feedback Loops** (MAPE)
  - *Explicit* feedback loops in design and analysis
  - Intra-loop and inter-loop coordination
  - Triggering condition
  - Distribution
  - Concurrency

- **Runtime Models** (K)
  - Arbitrary runtime models and their usage by MAPE

- **Sensor and Effectors**
  - Connecting reflection models to the adaptable software
  - Triggering of feedback loops based on sensor values

- **Layered Feedback Loops**
  - Adaptable feedback loops
  - Reflection on feedback loops

- **Off-line Adaptation**
  - Interface for manual maintenance
EUREMA: Executable Runtime Megamodels

- Model-driven engineering approach to specify, execute, and adapt feedback loops in layered architectures

Feedback Loop Diagram (FLD)

- FLD: activities + control flow, runtime models + their usage (behavior)
EUREMA: Executable Runtime Megamodels

- Model-driven engineering approach to specify, execute, and adapt feedback loops in layered architectures

**Feedback Loop Diagram (FLD)**

- **FLD:** activities + control flow, runtime models + their usage (*behavior*)
- **LD:** layers, white/black-box modules + their relationships (*structure*)
  - Triggering of modules: <events>; <period>; <initialState>;

---

**Layer Diagram (LD)**
EUREMA: Executable Runtime Megamodels

- Model-driven engineering approach to specify, execute, and adapt feedback loops in layered architectures

**Feedback Loop Diagram (FLD)**

- FLD: activities + control flow, runtime models + their usage (behavior)
- LD: layers, white/black-box modules + their relationships (structure)
  - Triggering of modules: <events>; <period>; <initialState>;
- FLDs and LD are kept alive at runtime and executed by an interpreter

**Layer Diagram (LD)**
3 Modularity

- Multiple FLDs for one feedback loop
- **Complex model operation** to invoke an FLD (entries and exists)
- Binding in the LD
Modularity

- Multiple FLDs for one feedback loop
- **Complex model operation** to invoke an FLD (entries and exists)
- Binding in the LD

**Diagram**

Self-repair-A

- <<EvaluationModel>>
  - Failure analysis rules
  - [ELSE]
  - <<EvaluationModel>>
    - Deep analysis rules

- <<ReflectionModel>>
  - Architectural Model

- [C_SINCE(no failures) > 5]

Deep check for failures

- <<Analyze>>
  - no failures

Failure analysis rules

- <<EvaluationModel>>
  - Failures

- [ELSE]

Start

Self-repair-A

- <<Analyze>>
  - Complex model operation
  - Repair strategies

- <<Plan>>
  - Repair
  - repaired

- <<Execute>>
  - Effect
  - done

Layer-0

:Self-repair

<<CausalConnectionModel>>

TGG Rules

Layer-1

:Self-repair-A

<<Plan>>

Monitor

Effect

Done

Monitor

<<ChangeModel>>

Repair strategies

<<ReflectionModel>>

Architectural Model

<<Analyze>>

 Failures

OK

Analyzed

Analyzed

Repair

 repairing

Plan

Layer-1

:Self-repair

M..PE

:mRUBis

RtException;

10s; Monitor;

:Self-repair-A

<<Analyze>>

AAnalyze

r w

Layer-0

:mRUBis

<<CausalConnectionModel>>

TGG Rules

<<Plan>>

Monitor

Effect

Done

Monitor

<<ChangeModel>>

Repair strategies

<<ReflectionModel>>

Architectural Model

<<Analyze>>

 Failures

OK

Analyzed

Analyzed

Repair

 repairing

Plan

Layer-1

:Self-repair-A

<<Analyze>>

AAnalyze

r w

Layer-0

:mRUBis

<<CausalConnectionModel>>

TGG Rules

<<Plan>>

Monitor

Effect

Done

Monitor

<<ChangeModel>>

Repair strategies
Variability

- Alternative modules as **variants** of FLDs
- Rebinding to switch between alternatives
- Design-time and runtime
- Example: different analysis techniques

The same applies to implementations (black-box modules) of basic model operations
- Example: different monitoring techniques
3 Multiple Feedback Loops

- Multiple concerns to be managed
- Competing concerns and interferences → coordination

EUREMA

- Modeling the synchronized execution of feedback loops
- Model operation implementation realizes the coordination mechanism (e.g., utility functions or voting)
Independent execution

- Individual trigger for each feedback loop
- Potentially, concurrent execution of different feedback loops
- Possibility to implicitly synchronize the execution by triggers (e.g., appropriate frequencies of execution runs)
Explicitly modeling the synchronized execution

MAPE for self-repair → MAPE for self-optimization
Sequencing Adaptation Activities of Feedback Loops

- More fine-grained synchronization (activities vs. whole feedback loop)
- Interleaved execution of different feedback loops
- $M \rightarrow A+P$ for self-repair $\rightarrow A+P$ for self-optimization $\rightarrow E$
Layered Feedback Loops

- A higher-layer feedback loop adapts a feedback loop at the layer directly below

- **Adaptable feedback loops** by dynamic changes of FLDs
  - Flexibility due to runtime models and the interpreter
  - Adaptable with respect to EUREMA concepts: operations, models (externalized state of a feedback loop), control flow, and model usage

- **Reflection** of feedback loops
  - **Procedural**: one representation to specify, execute, and adapt the feedback loop
  - **Declarative**: representation to adapt a feedback loop is different from the representation to specify and execute the loop

- Example: replace or enhance repair strategies of the self-repair loop
3 Layered Feedback Loops II

Procedural Reflection

Self-repair-strategies

<<EvaluationModel>>
Repair strategies analysis rules

<<Monitor>>
Check success rate

<<ReflectionModel>>
Adapted

<<ChangeModel>>
Repair strategies synthesis rules

<<Plan>>
Synthesize new repair strategies

<<ChangeModel>>
Adapted

<<Execute>>
Repair strategies repaired

<<CausalConnectionModel>>
Trigger to intercept the lower-layer feedback loop

Layer-0 Layer-1
:Self-repair
M..PE
:mRUBiS
RtException;
10s; Monitor;
:Self-repair-A

Layer-2
After [Deep check for failures]; Adapt;

Layer-0
:Self-repair-strategies
MAPE
feedbackLoopModel

Layer-1
:Self-repair-A
Analyze

Layer-0
:mRUBiS

Thomas Vogel | MDE of Self-Adaptive Software with EUREMA | Seminar 14382 | September 16, 2014
Layered Feedback Loops II

Procedural Reflection

- **feedbackLoopModel** = Self-repair FLD
- Directly use the FLD runtime model as a reflection model: no M+E at layer 2.

Thomas Vogel | MDE of Self-Adaptive Software with EUREMA | Seminar 14382 | September 16, 2014
Declarative Reflection

Self-repair-strategies-2

<<EvaluationModel>>
Repair strategies
analysis rules

<<Analyze>>
Check
success rate

<<Plan>>
Synthesize new
repair strategies

<<ChangeModel>>
Repair strategies
synthesis rules

<<Monitor>>
Observe
updated model

<<ReflectionModel>>
Self-repair Model

<<Execute>>
Replace
strategies

<<EvaluationModel>>
Repair strategies
analysis rules

<<Analyze>>
Check
success rate

<<Plan>>
Synthesize new
repair strategies

<<ChangeModel>>
Repair strategies
synthesis rules

<<Monitor>>
Observe
updated model

<<ReflectionModel>>
Self-repair Model

<<Execute>>
Replace
strategies
Layered Feedback Loops III

Declarative Reflection

- **Self-repair Model** \(!=\) **Self-repair FLD**
- M+E at layer-2 to build/sync. a representation of the self-repair FLD (MDE!)
### Off-line Adaptation

- Model and develop a patch/update/feedback loop off-line → FLD
- Upload the FLD describing the patch/update/feedback loop
- Specify how an instance of the FLD should be integrated in the LD

**LD as a procedural reflection model of the self-adaptive software.**

- **Current architecture**
- **Integration rule**
- **Resulting architecture**
Discussion of Requirements

- **Feedback Loops** (MAPE)
  - *Explicit* feedback loops in design and analysis ✓
  - Intra-loop and inter-loop coordination ✓ wrt execution
  - Triggering condition ✓
  - Distribution x
  - Concurrency X except of whole feedback loops

- **Runtime Models** (K)
  - Arbitrary runtime models and their usage by MAPE ✓

- **Sensor and Effectors**
  - Connecting reflection models to the adaptable software ✓
  - Triggering of feedback loops based on sensor values ✓

- **Layered Feedback Loops**
  - Adaptable feedback loops ✓
  - Reflection on feedback loops ✓

- **Off-line Adaptation**
  - Interface for manual maintenance ✓
4 Conclusion

- **EUREMA**
  - Modeling language + interpreter to specify + execute feedback loops
  - Adaptation and maintenance of feedback loops
  - Flexibility due to runtime models and interpreters/engines
    - of feedback loops (EUREMA)
    - within feedback loops
  - Reuse of MDE tools for implementation of adaptation activities

- **Topics**
  - Modeling feedback loops ↔ Control specification
  - Runtime models ↔ System identification
  - Layered architectures ↔ Adaptive or hierarchical control
  - Assurances ↔ Constraints, properties
    - Stability, accuracy, overshoot, settling time for software?
  - Structural (architectural) and parameter adaptation

- **self-adaptive.org**
Further Reading


