Self-Adaptation: MAPE-K and Control Theory

Danny Weyns & Stepan Shevtsov
www.lnu.se/adaptwise

Grenoble, France
November 4, 2014
Linnaeus University Sweden
Växjö Campus

AdaptWise research team focusing on engineering self-adaptive systems
Research setting

Software engineers have to deal with uncertainties resulting from incomplete knowledge at design time, e.g.
- Dynamic operating conditions that are difficult to predict
- Parts enter and leave the system at will
- Changing goals

How to engineer such systems and guarantee the required system goals?
Self-adaptation

System maintains a model of itself at runtime
System reasons about itself and adapts to realize system goals

Architecture-based self-adaptation
- Separation between domain concerns & adaptation concerns
- Adaptation concerns are handled by a feedback loop
- Feedback loop $\approx$ set of components that share knowledge
Model architecture-based self-adaptation

Software system

Environment
Non-controllable software, hardware, network, physical context
Model architecture-based self-adaptation

Self-adaptive software system

Managing system

Managed system
Controllable software

Environment
Non-controllable software, hardware, network, physical context
Model architecture-based self-adaptation

- **MAPE-K**
  - M: Monitor
  - A: Analyze
  - P: Plan
  - E: Execute
  - K: Knowledge

Managed system

Controllable software

Environment

Non-controllable software, hardware, network, physical context
Research focus

Principles of architecture-based self-adaptation
- FORMS: Unifying reference model for self-adaptation
- Do External Feedback Loops Improve the Design of Self-Adaptive Systems?

Rigorous modeling and verification
- ActivFORMS: Active formal models for self-adaptation
- Unifying approach for self-adaptation that integrates architecture-based approach with control-theoretic approach
Research focus

Principles of architecture-based self-adaptation
- FORMS: Unifying reference model for self-adaptation
- Do External Feedback Loops Improve the Design of Self-Adaptive Systems?

Assurances to handle uncertainty
- ActivFORMS: Active formal models for self-adaptation
- Unifying approach for self-adaptation that integrates architecture-based approach with control-theoretic approach
ActivFORMS: Starting points

Assurances for self-adaption
- One prominent approach is using formal models
- Main focus on K of MAPE-K
- Main focus on uncertainty wrt model parameters

Assurance of adaptation capabilities by MAPE functions?
- E.g., does analysis detect errors correctly?
- Are adaptations performed in order of selected plan?

How to provide assurances with changing/new goals?
- Requires support for adaptations of adaptation functions
ActivFORMS: Active formal models for self-adaptation

Formal model of complete MAPE-K loop

Model is directly executed to adapt the managed system

Model directly supports online verification of goal satisfaction/violation

Goals and MAPE-K models can be changed at runtime to support changing/new goals
Active Model

MAPE-K behaviors modeled as network of timed automata
Adaptation requirements formulated in timed computation
Model can be verified offline (e.g., using Uppaal tool)
Virtual Machine

Can load and execute active model

Interaction with managed system and environment through probes and effectors

Allows verification of goal satisfaction/violation

Goals and MAPE-K models can be changed at runtime to support changing/new goals
Goal management

- Goal model with associated adaptation models

- Goal monitor monitors goal satisfaction

- Goal adaptor satisfies current goals and adapt active model when needed

- Goal manager supports inspection active model and updating models
Case study
Goal Management Interface

![Image of the Goal Management Interface](image_url)

The interface shows two robots:
- **Robot 1** connected to `localhost:9000`
- **Robot 2** connected to `localhost:9002`

The goals for **Robot 2** include:
- Disable lane
  - `DISABLED_LANE == true`
- Not on disabled lane
  - `knowledge[0].currentLane != knowledge[0].disabledLane && DISABLED_LANE == true`

The goals for **Robot 1** include:
- Disable lane
  - `DISABLED_LANE == true`
- Not on disabled lane
  - `knowledge[0].currentLane != knowledge[0].disabledLane && DISABLED_LANE == true`
Goal Management Interface
Show executing model
Goal Management Interface
Updating goals and models
Goal Management Interface
Update model
Discussion

Active formal model paves the way for “handling continuous change”

Goal model and associated adaptation models
- Support changing goals and adding new goals
- Results in small models allowing efficient verification

We have applied the approach successfully to different case studies

Various opportunities for future research remain, including:
- Support for probabilistic models
- Runtime support for model checking
- Coordination between active models
Research focus

Principles of architecture-based self-adaptation
- FORMS: Unifying reference model for self-adaptation
- Do External Feedback Loops Improve the Design of Self-Adaptive Systems?

Rigorous modeling and verification
- ActivFORMS: Active formal models for self-adaptation
- Unifying approach for self-adaptation that integrates architecture-based approach with control-theoretic approach
Unifying approach for self-adaptation that integrates architecture-based approach with control-theoretic approach

Control theory provides a mature mathematical framework to design and analyze feedback loops

Has been used to control computing systems

Focus primarily on controlling resources of computing systems (CPU, memory, bandwidth, etc.)

Our aim is to control software
Motivation for controlling software

How do control-based solutions relate to MAPE-K based solutions?

Are they complementary?

Or does control theory provide a foundational theoretical framework to software adaptation?
Unifying approach for self-adaptation that integrates architecture-based approach with control-theoretic approach

Control in service-based system: e.g.,
- Dynamically replace service that goes down with alternative
- Change service with better quality properties
- Change the workflow at runtime to improve QoS, etc.
Variety of challenges

How does the (mathematical) model of software look like?

For which types of adaptations can we control software and for which not? How is a setpoint/input defined?

What are the “knobs” to adapt software?

What guarantees can be provided? How do typical software adaptation properties relate to typical properties of controllers?

What are the benefits and limitations of applying control theory to software systems?
Current work

Systematic literature review of using control theory to adapt software

Focus on application layer and supporting service layer (not 1 to 1 mapping to resources (e.g. a driver or virtual machine)

Research questions we plan to address

- RQ1: What is the focus of research on control-based adaptation in software systems?
- RQ2: What qualities are affected by adapting software using principles from control theory?
- RQ3: How are the software systems with feedback controllers modeled and constructed?

In parallel, we are studying how to apply a control-based approach to adapt the architecture of a service based system
Bibliography


• Papers available via: http://homepage.lnu.se/staff/daweaa/publications/publications.htm