Integrating Adaptation Mechanisms Using Control Theory Centric Architecture Models: A Case Study

Filip Kříkava
University Lille 1 / LIFL
INRIA Lille
France

Philippe Collet
Université Nice Sophia Antipolis I3S - CNRS UMR 7271
France

Romain Rouvoy
University Lille 1 / LIFL
INRIA Lille
France
SELF-ADAPTIVE SOFTWARE SYSTEMS, FEEDBACK CONTROL

- Increasing complexity of software systems, uncertain operating environment
- A shift towards self-adaptive software systems
  - systems adjusting themselves in accordance with higher-level goals
- Feedback control loops (FCLs) provide a generic mechanism for engineering self-adaptive software systems

---

**Monitoring**
- measures
- system outputs
  - (response time, utilization, throughput)

**Decision Making**
- decisions

**Reconfiguration**
- actions
- system control inputs
  - (queue size, scheduling policy, concurrency level)

**Target System**
- events
- sensors
- effectors

achieve externally specified goals
CHALLENGES

- Engineering self-adaptive software systems is challenging
- Example: web server self-optimization (MC based on MEM)

FCL control challenges

- Prepare experimental environment
  - identify system outputs (sensors)
  - identify system control inputs (effectors)

- Design decision mechanism
  - data collection
  - model construction and evaluation
  - controller design

- Implementation
  - integration into target system
  - consistent monitoring
  - coordinated reconfiguration
INTEGRATING ADAPTATION MECHANISMS

- **Ad hoc Implementations**
  - Extensive handcrafting of non-trivial code
  - Low level of FCL abstraction
  - Low FCL visibility
  - Limited verification and reasoning
  - Giving raise to accidental complexities

- **Reusing and adapting existing work**
  - Often target specific types of adaptation problems [Bertran’12]
  - Require the use of certain adaptation mechanism [Garlan’04]
  - Applicable to single domain [Rouvoy’08] or technology [Asadollahi’09]
  - Do not support remoting or complex control schemes [Adamczyk’08, Gorton’06]
  - Overall lack of supporting control theory based controller
  - Limiting their applicability
OVERVIEW

Systematic integration of self-adaptive mechanisms into software systems through control theory centric architecture models

1 Feedback Control Definition Language

2 The ZNN.COM case study
Feedback Control Definition Language
FEEDBACK CONTROL DEFINITION LANGUAGE - IN A NUTSHELL

Apache adaptation example - *adjusts the maximum number of connections to be processed simultaneously (MC) based on the difference between reference (MEM*) and measured memory usage (MEM) [Hellerstain’04].*
Apache adaptation example - adjusts the maximum number of connections to be processed simultaneously (MC) based on the difference between reference (MEM*) and measured memory usage (MEM) [Hellerstain'04].

- Domain-specific modeling language
- Network of Adaptive Elements (AEs)
  - actor-like element
  - push/pull communication
  - properties
  - specifies implementations
  - supports reflection
  - defines interaction contracts
2
THE ZNN.COM CASE STUDY
THE ZNN.COM CASE STUDY

- A news service model
- Web-based N-tier client-server
- Originally used for Rainbow evaluation [Cheng’09]
- Acknowledged SEAMS case-study
- Partially used by others (e.g., DYNAMICO, Zenshin, AdaptCases)

Objective

- “To serve the content within reasonable response time and quality in the event of traffic spikes caused by highly popular news.”

Challenges

- content adaptation (e.g. serving reduced content quality),
- service differentiation (e.g. prioritizing premium customers),
- infrastructure adaptation (e.g. adapting the size of the server pool)

http://www.hpi.uni-potsdam.de/giese/public/selfadapt/exemplars/model-problem-znn-com/
ZNN.COM - LOCAL CONTENT DELIVERY ADAPTATION

QoS management control of web servers by content delivery adaptation

Goal: maintain server load around some pre-set value

Idea: service time = fixed overhead + data-size dependent overhead

Prerequisite: preprocessed content (different quality and size)

\[ U = aR + bW \]
\[ G = G + k(U^* - U) \]

[Abdelzaher et al., 1999, 2002]
1. Compute the number of requests ($r$) and size of responses ($w$)

2. Compute the requests rate ($R$), bandwidth ($W$) and utilization ($U$)

3. Compute severity of adaptation ($G$)
1. Compute the number of requests ($r$) and size of responses ($w$)

```
requestCounter : Accumulator
responseSizeCounter : Accumulator

accessLogParser : AccessLogParser

accessLog : FileTailer
file=/var/log/apache2/access.log

access_log
```
ZNN.COM - LOCAL CONTENT DELIVERY ADAPTATION

2 Compute the requests rate \((R)\), bandwidth \((W)\) and utilization \((U)\)

\[
U = aR + bW = a\frac{\sum r_j}{t} + b\frac{\sum w_i}{t}
\]

active processor PeriodTrigger\(<T>\) {
    pull in port input: \(T\)
    push out port output: \(T\)
    property initialPeriod = 10.seconds
}
Compute severity of adaptation ($G$)

$G = G + k(U^* - U)$

controller IController {
    in push port input: double
    out push port output: double
    property KI: double
    property reference: double
    property loBnd: double
    property upBnd: double
}

Generality

Visibility

access_log

content_tree
Integrating Adaptation Mechanisms Using Control Theory Centric Architecture Models: A Case Study

ZNN.COM - LOCAL CONTENT DELIVERY ADAPTATION

Complete model

```
composite ApacheQOS {
    feature accessLog = new FileTailer {
        file = "~/var/log/apache2/access_log"
    }
    feature accessLogParser = new AccessLogParser
    feature requestCounter = new Accumulator
    feature responseSizeCounter = new Accumulator
    feature loadMonitor = new LoadMonitor
    feature scheduler = new PeriodTrigger<Double>
    feature utilController = new IController {
        reference = 0.8
    }
    feature adaptor = new ContentAdaptor

    connect accessLog.lines to accessLogParser.lines
    connect accessLogParser.size to responseSizeCounter.input
    connect accessLogParser.requests to requestCounter.input
    connect requestCounter.output to loadMonitor.requests
    connect responseSizeCounter.output to loadMonitor.size
    connect loadMonitor.utilization to scheduler.input
    connect scheduler.output to utilController.utilization
    connect utilController.contentTree to adaptor.contentTree
}
```

Generality  Visibility  Composition
ICAC’14, 18.6.2014

Integrating Adaptation Mechanisms Using Control Theory Centric Architecture Models: A Case Study

ZNN.COM - LOCAL CONTENT DELIVERY ADAPTATION

Diagram of QOSControl, ApacheQOS, and ApacheWebServer components with control theory models.
• Using the reflection support for adaptive control
ILLUSTRATION - DISTRIBUTED CONTENT DELIVERY ADAPTATION

- QoS management control of a pool of web servers using content delivery adaptation.
- Load balancer schedules requests to a server with highest QoS.

\[ s \in S \text{ such that } G_s = \max (G_1, \ldots, G_n) \]
ILLUSTRATION - DISTRIBUTED CONTENT DELIVERY ADAPTATION

ApacheQOS

LocalApacheQOS

Load Balancer

LoadBalancerControl
ILLUSTRATION - RESOURCE MANAGEMENT

\[ n = n + K_I E = n + K_I (G^* - \overline{G}) \]
• Support for FCL design - black-box modeling
• Open control loops for data collection
IMPLEMENTATION

- Reference implementation of FCDL based on Eclipse Modeling Framework
- Eclipse IDE-based prototype to facilitate the use of FCDL - ACTRESS

- xFCDL (Extended FCDL)
  - Textual DSL for authoring FCDL models
  - Modularity, Java interoperability, Xbase
  - Eclipse IDE support

- ACTRESS runtime
  - based on Java/AKKA

- Model well-formedness
- User-defined structural constraints (OCL)
- Architecture verification (completeness, consistency, determinacy)
- Temporal properties (PROMELA/SPIN)
3

Conclusions
SUMMARY

• Combining self-adaptive software systems with principles of MDE to provide systematic and tooled approach for integrating control mechanisms into software systems.

• A proof of concept implementation and tools facilitating the language use including modeling, code synthesis and verification support.
FUTURE WORK

- Address ACTRESS limitations - a new MPS-based implementation
- More robust FCDL - support data units, input output assertions
- ZNN.COM challenge
  - From a case study to a benchmark
  - Easily reusable (docker packaging)
  - INRIA Lille Non-A team in charge of a new controller
- Evaluation against two base lines
  - Apache mod_proxy
  - Amazon elastic load balancer
Thank you

Filip Křikava
filip.krikava@inria.fr