Software Engineering for Self-Aware Computing

Samuel Kounev

University of Würzburg
http://se.informatik.uni-wuerzburg.de/

Jan 19, 2015
Dagstuhl Seminar 15041:
“Model-driven Algorithms and Architectures for Self-Aware Computing Systems”
Agenda

- Self-aware computing and related terms
- Models in software engineering
- Modeling examples for self-aware computing
- Open issues and challenges
- Vision
Self-Aware Computing Systems?

- Intelligent systems
- Autonomic computing systems
- Self-* systems
- Organic computer systems
- Adaptive systems
- Self-managing systems
- Cognitive systems
- Self-optimizing
- Self-protecting
- Self-describing
- Self-configuring systems
- Self-tuning
- Self-healing systems
- Self-expressiveness
- Self-explaining

Introduction
S. Kounev
Examples
Models in SE
Challenges
Def (Self-Aware):

- **Introspective**: can observe and optimise their own behaviour,
- **Adaptive**: can adapt to changing needs of applications running on them,
- **Self-healing**: can take corrective action if faults appear whilst monitoring resources,
- **Goal-oriented**: attempt to meet user application goals,
- **Approximate**: can automatically choose the level of precision needed for a task to be accomplished.

SEIf-awarE Computing (SEEC) Project

- **Def (self-aware):** Understand high-level goals and automatically adapt to meet those goals online

- Presence of **observe-decide-act (ODA) loops** in all system layers – hardware, compilers, OS, and applications

- Applications specify goals, system software specifies possible actions, and the SEEC framework decides how to use the available actions to meet the goals

---

**SEEC: A General and Extensible Framework for Self-Aware Computing**

by H. Hoffmann, M. Maggio, M. Santambrogio, A. Leva, and A. Agarwal

MIT CSAIL Technical Report, MIT-CSAIL-TR-2011-046, November 2011. ([doi](https://doi.org/))

*Project was named one of ten "World Changing Ideas" by Scientific American*
Individual components and ensembles of components that are

- **self-adaptive**: able to properly react on need by self-tuning their internal behavior and/or structure in an autonomic way –
- **self-aware**: able to recognize the situations of their current operational context requiring self-adaptive actions

Awareness of

- not simply “what I am and what is happening in the world”, but also
- “what I could become and how the world could change accordingly”

---

**On Self-adaptation, Self-expression, and Self-awareness in Autonomic Service Component Ensembles** by F. Zambonelli, N. Bicocchi, G. Cabri, L. Leonardi, M. Puviani

EPiCS EU Project

- Engineering Proprioception in Computing Systems
  - collect and maintain information about their state and progress, which enables them to reason about their behaviour (self-awareness)
  - and utilise this knowledge to effectively and autonomously adapt their behaviour to changing conditions (self-expression)
SEAMS Community

- „Software Engineering for Self-Adaptive Systems“ Community

- **Def (self-adaptive systems):**
  - adapt at run-time to changing user needs, system intrusions or faults, changing operational environment, and resource variability
  - can configure and reconfigure themselves, augment their functionality, continually optimize themselves, protect themselves, and recover themselves, while keeping most of their complexity hidden from the user and administrator

- **Generic Control Loop Model**
Def (self-aware): possess, and/or are able to acquire at run-time, three properties, ideally to an increasing degree the longer they are in operation:

1. **Self-reflective**: Aware of their operational goals and of the aspects of their architecture and environment relevant to achieving these goals,

2. **Self-predictive**: Able to predict the effect of dynamic changes, as well as predict the effect of possible adaptation actions,

3. **Self-adaptive**: Proactively adapting as the environment evolves in order to ensure that their operational goals are continuously met.

http://descartes.tools
„The indispensable first step to getting the things you want out of life is this: decide what you want“.

-- Ben Stein

[Neshan Naltchayan, Wikipedia]
Stresses Explicit Awareness of

- What are my high-level (application) goals?
- What aspects of my architecture and my environment are relevant for achieving my goals?
- How well am I currently meeting my goals?
- What changes are anticipated that will have impact on my goals?
- What possible adaptation actions can I undertake? What would be the impact of an adaptation on my goals?
- How can I find a suitable adaptation tactic in time and proactively adapt to continue fulfilling my goals?
Examples of Modern Systems

Introduction

S. Kounev

Challenges

Models in SE

Examples
Semantic Gap Problem

Applications
- Multiple tiers
- Multiple resource types

Complex Software Stacks
- Multiple layers
- Heterogeneous

Resource Allocation

High-level Application Goals (e.g., SLOs)

Configuration of System Components, Layers & Tiers

Introduction

S. Kounev

Examples

Challenges
Semantic Gap Problem

- **Performance**
  - # requests that can be processed per sec > 1000
  - Response time of service x < 20 ms
  - Server utilization > 60% on average
  - ...
- **Availability / Reliability**
  - Time to recover after a server failure < 1 min
  - ...
- **Security**
  - Intrusion attempts are detected on time and prevented
  - ...

On which server to deploy software component y?

- How many vCPUs to allocate to VM n?
- How much memory to allocate to VM n?
- When exactly should a reconfiguration be triggered
- Which particular resources to scale / replicate / migrate?
- How quickly and at what granularity?

High-level Application Goals (e.g., SLOs)
Models in Software Engineering

Descriptive Models

- Capture relevant knowledge about the system and the environment in which it is running
- Describe selected aspects that have influence on the goal fulfilment

(Predictive) Analysis Models

- Allow to reason about the system behavior
- Predict the impact of changes on the goal fulfilment
Descriptive Models

Meta-Model

Node : Class

name : String
numCPUs : Integer
numVMs : Integer

Model

Server 1 : Node

name = “Server 1”
numCPUs = 3
numVMs = 3

“Real world”

Resource Allocation

VMM

Server 1

VM
VM
VM

instance of

instance of

describes

describes
A **meta-model** is a model precisely defining the parts and rules needed to create valid models. It covers an **abstract syntax**, at least one **concrete syntax**, and **static and dynamic semantics**.

- **Parts** → model elements
- **Abstract syntax**: elements and their relations independently of representation
- **Static semantics** → semantics evaluable without executing the model
- **Rules** → well-formed rules - when is a model valid?
- **Concrete syntax** → representation of model-instances, e.g., in a file
- **Dynamic semantics** → what does the model mean/express?
Meta-Model Levels

(M₀)
Instances

(M₁)
Model

(M₂)
Meta-model

(M₃)
Meta-meta-model

InstanceOf

Instances

Models in SE
Meta-Object Facility (MOF)

- Abstract language and framework for specifying, constructing, and managing technology neutral meta-models

- MOF is self-describing and has two parts
  - EMOF (Essential MOF, lightweight, subset of CMOF)
  - CMOF (Complete MOF, heavyweight)

- EMF (Eclipse Modelling Framework) can be seen as an implementation of EMOF
  - using the Ecore meta-model

- Example of a MOF-based meta-model → UML
### Abstract vs. Concrete Syntax

#### Abstract

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Server1 : Node</td>
<td></td>
</tr>
<tr>
<td>name = “Server 1”</td>
<td></td>
</tr>
<tr>
<td>numCPUs = 3</td>
<td></td>
</tr>
<tr>
<td>numVMs = 3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Concrete

```java
Node Server1 {
    String name = “Server 1”
    int numCPUs = 3;
    int numVMs = 3;
    ...
}
```

```xml
<Node
    nodeName="Server1">
    <attribute
        attributeName="name" value="Server 1"/>
    <attribute
        attributeName="numCPUs" value="3"/>
    <attribute
        attributeName="numVMs" value="3"/>
</Node>
```
Object Constraint Language (OCL)

- A declarative language for describing rules that apply to valid model instances
  
  "A constraint is a restriction on one or more values of an object-oriented model or system" [Warmer & Kleppe]

- Example:

  ```
  Node : Class
  name : String
  numCPUs : Integer
  numVMs : Integer
  ...
  
  context Node
  inv CPUs: numCPUs > 0
  ```

  the attribute numCPUs of every Node must be greater than 0
Model-2-Model Transformations

- Transformations
  - Input: A model instance of meta-model A
  - Output: A model instance of meta-model B
  - Rules: How to transform meta-model elements of meta-model A into elements of meta-model B
  - Rule Engine: A system capable to execute the rules

![Diagram of Model-2-Model Transformations]

- Meta-Model A
- Transformation-Rules-Meta-Model
- Meta-Model B
- Model Instance A
- Transformation Rules
- Model Instance B
-InstanceOf
-InstanceOf
-InstanceOf
-Input
-Output
Transformation Languages

imperative style
- Xtend, QVT-O, Kermeta, XSLT…

declarative style
- QVT-R, TGG…

supporting both paradigms
- ATL, RubyTL, VIATRA…

general purpose
- pragmatic

problem specific
- formal and sometimes academic
Feature Models

- A feature is a choice you have
  - e.g. in a transformation
  - i.e. they model variation points that can be influenced via transformation parameters

Legend:
- Common attribute
- Variable attribute
- Multiplicity
- Requires-relation
- Excludes-relation
- Formula-relation
Concept Formation

- Domain
- Meta Model
  - Meta Meta Model
    - describes relevant concepts of
      - Subdomain
      - specified based on
        - <<instanceof>>
  - Abstract Syntax
    - specified based on
      - Static Semantics
      - DSL
        - <<synonym>>
      - Concrete Syntax
        - <<instanceof>>
      - Semantics
      - Modelling Language
        - respects
          - Formal Model
          - Semantics

cf. [Voe05]
Descartes Modeling Language (DML)

- Architecture-level modeling language for modeling QoS and resource management related aspects of IT systems and infrastructures
  - Prediction of the impact of dynamic changes at run-time
  - Current version focused on performance including capacity, responsiveness and resource efficiency aspects

http://descartes.tools/dml
Descartes Modeling Language (DML)

Adaptation Process Model
- Strategies
- Tactics
- Actions

Adaptation Points Model

Architecture-level Performance Model
- Application Architecture Model
- Resource Landscape Model
  - Usage Profile
  - Degrees-of-Freedom

S. Kounev

Introduction ➤ Models in SE ➤ Examples ➤ Challenges
Big Picture

Adaptation Process Model
- Strategies
- Tactics
- Actions

Describes

Adaptation Process
Adapts

Adaptation Points Model

Managed System

Managed System

Architecture-Level Performance Model

Degree of Freedom

Models
Parameterizes
References


- S. Kounev, F. Brosig, and N. Huber. **The Descartes Modeling Language.** Technical report, Department of Computer Science, University of Wuerzburg, October 2014. [http](http) [pdf](http://example.com/)


References


Example (Resource Landscape)

Weblogic Application Server hosting the SPECjEnterprise2010 benchmark

WebLogic Server® 11g

XenServer 5.5 Virtual Machines

GBit LAN
Example
(Resource Landscape Model)

<<ComputingInfrastructure>>
ComputeNode1

<<RuntimeEnvironment>>
XenServer1

<<RuntimeEnvironment>>
VM_1

<<ActiveResourceSpecification>>
processingResourceType = vCPU
processingRate = 2.66 GHz
schedulingPolicy = PROCESSOR_SHARING
numberOfCores = 2

<<ActiveResourceSpecification>>
processingResourceType = CPU
processingRate = 2.66 GHz
schedulingPolicy = PROCESSOR_SHARING
numberOfCores = 4

<<ActiveResourceSpecification>>
processingResourceType = CPU
processingRate = 2.66 GHz
schedulingPolicy = PROCESSOR_SHARING
numberOfCores = 4

<<ComputingInfrastructure>>
DatabaseServer

<<ActiveResourceSpecification>>
processingResourceType = CPU
processingRate = 2.66 GHz
schedulingPolicy = PROCESSOR_SHARING
numberOfCores = 4
Example
(Resource Landscape Model) + (Adaptation Points Model)

<<ComputingInfrastructure>>
ComputeNode1

<<RuntimeEnvironment>>
XenServer1

<<RuntimeEnvironment>>
VM1

<<ActiveResourceSpecification>>
processingResourceType = vCPU
processingRate = 2.66 GHz
schedulingPolicy = PROCESSOR_SHARING
numberOfCores = 4

<<ModelEntityConfigurationRange>>
VmHost
variationType = SetOfConfigurations
possibleValues = "XenServer1, XenServer2, ..."

<<ComputingInfrastructure>>
DatabaseServer

<<ActiveResourceSpecification>>
processingResourceType = CPU
processingRate = 2.66 GHz
schedulingPolicy = PROCESSOR_SHARING
numberOfCores = 4

<<ModelEntityConfigurationRange>>
VmInstances
variationType = PropertyRange
minValueConstraint = "minVmInstances"
maxValueConstraint = "maxVmInstances"

<<ModelVariableConfigurationRange>>
NrOfVcpus
minValue = 2
maxValue = 4
Example
(Application Architecture Model)
Example
(Coarse-Grained Service Behavior Model)

CallFrequency = IntPMF[(0;0.5)(1;0.5)]

<<CoarseGrainedBehavior>>

<<ResourceDemand>>

<<ExplicitDescription>>
Exp (1/25)

<<ExternalCallFrequency>>

<<ExternalCall>>
callDBS
Example
(Fine-Grained Service Behavior Model)

BranchingProbabilities = 
EnumPMF[('Branch1'; 0.5)('Branch2'; 0.5)
Adaptation Process

Events / Objectives

trigger / guide

Strategy\textsubscript{X}\quad \text{use}\quad \text{execute}\quad \text{reconfigure}

Strategy\textsubscript{Y}

Tactic\textsubscript{A}\quad \text{Tactic}\textsubscript{B}\quad \text{Tactic}\textsubscript{C}

Action\textsubscript{1}\quad \text{Action}\textsubscript{2}\quad \text{Action}\textsubscript{3}\quad \text{Action}\textsubscript{n}

System Model / Real System

Goal-Oriented (Logical)

System-Specific (Technical)
Example (Tactics)

**AddResources**

**Adaptation Plan**

**Loop**
iterationCount = iterations

- **Action**
  - AddVCPU
  - AddVM

**RemoveResources**

**Adaptation Plan**

- **Action**
  - RemoveVCPU
  - RemoveVM

**MigrateVM**

**Adaptation Plan**

**Action**
MigrateVM

---

Examples (Tactics)

Introduction

Models in SE

Examples

Challenges
Example (S/T/A Model Instance)

<<OverallGoal>>
"Maintain SLAs of all services using resources efficiently"

<<Specification>> < 500ms
<<MetricType>> 90%_Quantile_of_rt_x
<<Objectives>> hasObjectives
<<Objective>> MaintainSLAs
objective
<<Strategy>> ResolveBottleneck
<<Event>> SlAViolated
<<uses>>

<<Specification>> > 60%
<<MetricType>> OverallUtilization
<<Objectives>> hasObjectives
<<Objective>> OptimizeResourceEfficiency
objective
<<Strategy>> ReduceResources
<<Event>> ScheduledOptimization
<<uses>>

<<WeightedTactic>> AddResources
weight = 1.0
<<InputParameter>> name = "Iterations" type = Integer

<<WeightedTactic>> RemoveResources
weight = 1.0

<<WeightedTactic>> MigrateVM
weight = 0.5

<<Adaptation Plan>>

<<Loop>>
iterationCount = iterations
<<Action>> AddVCPU
allServersAtMaxCap
FALSE
<<Action>> AddVM
serverAtMinCapExists
TRUE

<<Action>> MigrateVM
Self-Predictive Property

![Diagram showing online prediction of SLA violation and reconfiguration impact.]

- **Online prediction of SLA violation**
- **Online prediction of reconfiguration impact**
Transformations to Predictive Models

Queueing Petri Net

Bounds Analysis Model

Layered Queueing Network
Example: Automatically Generated QPN Model

Tailored Model Solution

Analytical Analysis

\[ R \geq \max \left[ N \times \max \{D_i\}, \sum_{i=1}^{K} D_i \right], \quad X_0 \leq \min \left[ \frac{1}{\max \{D_i\}}, \frac{N}{\sum_{i=1}^{K} D_i} \right] \]

\[ \frac{N}{\max \{D_i\} [K + N - 1]} \leq X_0 \leq \frac{N}{\text{avg}(D_i) [K + N - 1]} \]

Analysis Results

Simulative Analysis

Analysis Results

Overview of Applied Modeling Techniques

Descriptive Architecture-level Models
- OMG Meta Object Facility (MOF)
- MOF-based meta-models
- (UML MARTE)
- (UML SPT)

Predictive Performance Models
- Bounding techniques
- Operational analysis
- Statistical regression models
- Stochastic process algebras
- (Extended) queueing networks
- Layered queueing networks
- Queueing Petri nets
- Reinforcement learning models
- Detailed simulation models

Workload Forecasting
- AR(I)MA
- Extended exp. smoothing
- tBATS
- Croston's method
- Cubic smoothing splines
- Neural network-based

Resource Demand Estimation
- Regression-based techniques
- Kalman filter
- Nonlinear optimization
- Maximum likelihood estimation
- Independent component analysis

Regression Analysis
- MARS
- CART
- M5 trees
- Cubist forests
- Quantile regression forests
- Support vector machines

Introduction
Examples
Challenges
Example Statistical Regression Models

\[ f(x_1, x_2) = c_1 x_1 x_2 + c_2 x_1 + c_3 x_2 + c_4 \]

interaction term

Parameters: # of terms, …

**LRM**

Tree structure
Step function

**MARS**

Piecewise linearity

Combination

Boosting Inst. - based

**Cubist**

Parameters: # of trees, # of nodes, …

**LRM - Linear Regression Models**

**MARS - Multivariate Adaptive Regression Splines**

**CART - Classification and Regression Trees**

Parameters: # of nodes, …
Challenges

- **Interoperability** of modeling languages

- **Automatic model extraction**, maintenance, refinement, and calibration during operation

- Supporting **flexible analysis** (accuracy vs. overhead)

- Scalable and **efficient algorithms** for system adaptation

- **Separation of responsibilities** in virtualized infrastructures

- Lack of **representative benchmarks** for evaluating self-awareness
Lack of Benchmarks

Reliable Metrics
- What exactly should be measured and computed?

Representative Workloads
- For which scenarios and under which conditions?

Sound Measurement Methodology
- How should measurements be conducted?

“To measure is to know.” -- Clerk Maxwell, 1831-1879

“It is much easier to make measurements than to know exactly what you are measuring.” -- J.W.N.Sullivan (1928)
Standard-Performance-Evaluation-Corporation

- **Open-Systems-Group (OSG)**
  - Processor and computer architectures
  - Virtualization platforms
  - Java (JVM, Java EE)
  - Message-based systems
  - Storage systems (SFS)
  - Web-, email- and file server
  - SIP server (VoIP)
  - Cloud computing

- **High-Performance-Group (HPG)**
  - Symmetric multiprocessor systems
  - Workstation clusters
  - Parallel and distributed systems
  - Vector (parallel) supercomputers

- **“Graphics and Workstation Performance Group” (GWPG)**
  - CAD/CAM, visualization
  - OpenGL

http://www.spec.org
SPEC Research Group (RG)

- Founded in March 2011: http://research.spec.org
  - Transfer of knowledge btw. academia and industry

Activities
- Methods and techniques for experimental system analysis
- Standard metrics and measurement methodologies
- Benchmarking and certification
- Evaluation of academic research results

Member organizations (Feb 2014)
Questions?