Design-Time vs. Run-Time Models for Quality-of-Service Prediction

Samuel Kounev

RELATE Open Excellence Workshop, Karlsruhe, December 4th, 2012
Motivation

Traffic Management System

Induction Loops
GPS Sensors
Traffic Cameras
Traffic Light Sensors

http://www.cl.cam.ac.uk/research/time/
Motivation: Traffic Management System

Event Bus

- Cam
- Cam
- License Plate Recognition
- Toll
- Bus Proximity
- Speeding
- Traffic Control
- Location
- Bus Sensors

Motivation:
Traffic Management System

Run-time Models vs. Descartes Meta-Model
Case Study
Summary & Outlook
Motivation: Inventory Management System

Event Bus

Source: UpdateStockData

Sink: UpdateStockData

RFID Scanner

Cashdesk Service

Order Management Service

Inventory Management Service

Logging Service

Prov. Interface: CreateOrder

Req. Interface: CreateOrder
Motivation

Run-time Models

Descartes Meta-Model

Case Study

Summary & Outlook

Traffic Management System

Inventory Management System

© Samuel Kounev

Design-Time vs. Run-Time Models for Quality-of-Service Prediction
Motivation

Traffic Management System

Varying Workloads

Inventory Management System

Varying Workloads

Run-time Models

Descartes Meta-Model

Case Study

Summary & Outlook
Motivation

- Software systems increasingly complex and dynamic
- Must be reconfigured at run-time more and more frequently
  - Resource allocations / system configuration
  - Dynamic deployment of new services & applications
  - Changes of existing components / addition of new components
- Problem: When and how exactly should the system be reconfigured?
State-of-the-Art

- Hard to predict the effect of dynamic changes on the system performance and resource demands

- Minimize risks by avoiding the need for reconfiguration
  - Over-provisioning of IT resources
  - Simple rule-based adaptation techniques ("best effort")
  - Manual adaptation in more complex scenarios

- Consequences: Poor resource efficiency
  - Rising energy costs for IT systems
    - 1600% increase by 2025 [Gartner]
  - Rising global CO2 emissions of ICT sector
    - Today: ca 3%, Increase to 10% expected in 10 years [EU]
Descartes Research Group

- Modeling methods for predicting at run-time the effect of dynamic changes on the system Quality-of-Service (QoS)
  - Current focus: availability and performance (response time, throughput and resource/energy efficiency)

- Model-based algorithms and techniques for autonomic system adaptation during operation

Goal:
- End-to-end QoS guarantees
- High resource/energy efficiency
- Low operating costs
PHASE 1
Online QoS Prediction for Problem Anticipation
Online reconfiguration impact prediction for trade-off analysis

Service A
VM replication/cloning

Scaling up/Improving dependability
Dependability/Responsiveness OK

Online prediction

Service A
VM replication/cloning

Service A
VM replication/cloning

Dynamic server consolidation
Efficiency OK

LiveVM migration

Service B
Service C

PHASE 2
Online QoS Prediction for Reconfiguration Impact Analysis

Motivation ➤ RUN-TIME MODELS ➤ Descartes Meta-Model ➤ Case Study ➤ Summary & Outlook

© Samuel Kounev
Design-Time vs. Run-Time Models for Quality-of-Service Prediction
Proactive Self-Adaptive Systems Management

PHASE 3
Autonomic System Adaptation

Motivation

RUN-TIME MODELS

Descartes Meta-Model

Case Study

Summary & Outlook
Proactive Self-Adaptive Systems Management

PHASE 1

Motivation

PHASE 2

Run-time Models

PHASE 3

Descartes Meta-Model

Summary & Outlook

© Samuel Kounev
Examples of Performance-Influencing Factors

System workload and usage profile
- Number and type of clients
- Input parameters and input data
- Data formats used
- Service workflow

Software architecture
- Connections between components
- Flow of control and data
- Component resource demands
- Component usage profiles

Execution environment
- Number of component instances
- Server execution threads
- Amount of Java heap memory
- Size of database connection pools

Virtualization layer
- Physical resources allocated to VMs
  - number of physical CPUs
  - amount of physical memory
  - secondary storage devices

Network bandwidth between system nodes
High-Level Research Questions

- What models of the system architecture are appropriate to enable the prediction of the impact of dynamic changes at run-time?
  - Resource allocations and configuration parameters in each system layer should be explicitly taken into account
  - How do changes in service workloads and resource allocations impact the system QoS?

- How to deal with the large state space of possible reconfigurations?

- Which model analysis methods and optimization techniques are appropriate for a given adaptation scenario at run-time?

…
State-of-the-Art: Summary

1. Modeling Approaches for Design-time Analysis
   - UML SPT, UML MARTE, CBML, SPE-MM, KLAPER, CSM, PCM, SAMM, ...
   - Models assume static system architecture
   - Dynamic aspects not considered
   - Maintaining models at run-time prohibitively expensive

   [M. Woodside et al], [D. Petriu et al], [R. Reussner et al], [C. Smith et al], [R. Mirandola et al],
   [K. Trivedi et al], [V. Cortellessa et al], [I. Gorton et al], [D. Menasce et al], [E. Eskenazi et al], ...

2. Modeling Approaches for Run-time Analysis
   - Queueing networks, „Reinforcement Learning“-Models, LPV-Models, ...
   - Models at a high level of abstraction: Components as „Black-Box“
   - Architecture layers and configuration parameters not modeled explicitly

   [G. Pacifici et al], [A. D' Ambrogio et al], [G. Tesauro et al], [D. Menasce et al], [C. Adam et al],
   [Rashid A. Ali et al], [I. Foster er al], [S. Bleul et al], [A. Othman et al], [P. Shivam et al], ...

© Samuel Kounev
Design-time vs. Run-time Models

- Two orthogonal dimensions
  - Modeling of design-time vs. run-time aspects
  - Use of models at design-time vs. run-time

- Fine granular differentiating factors
  1. Model purpose
  2. Model target users / consumers
  3. Degrees of freedom in model use case scenarios
  4. Model structure & parameterization
  5. Possibilities for model calibration
  6. Required model flexibility
1. Model Purpose

- **Design-time**
  - Evaluate and compare different design alternatives
  - Optimize system architecture
  - Sizing and capacity planning

- **Run-time**
  - Anticipate QoS issues resulting from
    - E.g., changing workloads, deployment of new services
  - Predict impact of possible dynamic reconfiguration
  - Adapt system configuration in a predictable manner
    - Elastic resource provisioning
    - Intrusion prevention
    - Failover after a server crash
2. Model Target Users / Consumers

- **Design-time**
  - System architect / performance engineer
  - Use by humans in an offline setting
  - Could also serve as architecture documentation

- **Run-time**
  - System administrator and/or the system itself
  - Use by humans and/or the system itself in an online setting
3. Degrees-of-Freedom

- **Design-time**
  - Theoretically every single aspect of the system can be varied
  - Degrees of freedom focused on
    - Software and system architecture
    - Deployment platforms
    - System configuration

- **Run-time**
  - Software architecture is relatively stable
  - Degrees of freedom focused on
    - Workloads / usage profiles
    - System deployment and configuration (incl. resource allocations)
    - Deployment of new services and/or change of service providers
4. Model Structure & Parameterization

- **Design-time**
  - Aligned with software development processes
    - Development phases and developer roles
    - Component: Unit of composition at design-time
  - Assumption: clear separation of concerns
  - Sub-models parameterized to capture their context dependencies

- **Run-time**
  - Aligned with system layers
    - Component: Unit of composition at run-time
  - Sub-models parameterized according to their dynamic reconfiguration aspects
  - Explicit distinction between static and dynamic aspects
5. Possibilities for Model Calibration

- **Design-time**
  - Flexibility to run experiments in a controlled environment
  - Possible lack of complete implementations of system components
  - Possible lack of a realistic production-like testing environment

- **Run-time**
  - All system components implemented and deployed
  - Monitoring in the production environment possible
  - Less control over the system to run experiments
  - Monitoring in a non-intrusive manner
6. Required Model Flexibility

- **Design-time**
  - Plenty of time to analyze the model
  - Can run detailed time-intensive simulations
  - Generally accuracy more important than analysis overhead

- **Run-time**
  - Model may have to be solved in seconds, minutes, hours, or days
  - Trading-off btw. accuracy and overhead critically important
  - Generally more flexibility required
    - Support for multiple abstraction levels, parameter granularities
    - Support for different analysis techniques
PCM and DMM

Palladio Component Model (PCM)  Descartes Meta-Model (DMM)

Design-time aspects  Run-time aspects

Motivation  Run-time Models  Descartes Meta-Model  Case Study  Summary & Outlook

© Samuel Kounev
**Descartes Meta-Model (DMM)**

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

---

**Adaptation Points**

- Application Architecture
- Resource Environment

**Adaptation Process**

---

**Run-time Models**

- [IEEE ICEBE 2012]
- [ACM CBSE 2012]
- [IEEE/ACM ASE 2011]

**Summary & Outlook**

- Design-Time vs. Run-Time Models for Quality-of-Service Prediction
Example: Resource Environment
Influence Factors of the Virtualization Layer

Virtualization Platform

Virtualization Type
- Full Virtualization
- Para-Virtualization

Binary Translation

VMM Architecture
- Dom0
- Monolitic

Resource Management Configuration

CPU Scheduling

CPU Allocation
- e.g. vcpu=4
- e.g. mask=1,2

CPU Priority

Core Affinity

Resource Overcommitment

Number of VMs

Memory Allocation

Memory

Network

Disk

I/O

Workload Profile

Legend
- exclusive OR
- inclusive OR

Example: Application Architecture

- Control flow and data flow
- Service resource demands
- Parameter and context dependencies

Prediction Method: Step 1: Dynamic Model Composition
**Prediction Method:**
Step 2: Tailored Model-to-Model Transformation

Dynamically Composed Model Instance

- **Usage Sub-model**
- **Soft. Arch. Sub-model**
- **Middleware Sub-model**
- **Virtualization Sub-model**
- **Infrastructure Sub-model**

**Operational Analysis**
- Analytical Sol.

**Queueing Network Models**
- Analytical Sol.

**Queueing Petri Nets**
- Analytical Sol.

**Stochastic Process Alg.**
- Analytical Sol.

**Full-Blown Simulation**
- Simulation

---

Motivation ➔ Run-time Models ➔ **DESCARTES META-MODEL** ➔ Case Study ➔ Summary & Outlook
Example Transformations

Simple Bounds Analysis

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

Queueing Network Model (Product Form)

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

Queueing Petri Net (QPN) Model

Layered Queueing Network (LQN) Model

Motivation → Run-time Models → DESCARTES META-MODEL → Case Study → Summary & Outlook
Modeling with Queueing Petri Nets

- Modeling methodology [TSE 2006]
- Efficient discrete event simulation [PerfEval 2006]
- Modeling tool
  - “Queueing Petri net Modeling Environment” (QPME)
  - “Eclipse Public License (EPL) v1.0”
  - Distributed under 130 organizations worldwide
  - Website: http://qpme.sourceforge.net/
  - Further details:
    - [Petri Nets 2012] [LNCS 6462] [PER 2009] [QEST 2006]


Case Studies (Selection)

- **Java EE-based systems**
  - [IEEE Trans. on SE 2006] [Elsevier PerfEval 2006]
  - [IEEE ISPASS]

- **Enterprise data fabrics**
  - [ICST SIMUTools 2011]

- **Enterprise Grid Environments**
  - [Elsevier JSS 2009] [VALUETOOLS 2007]

- **Message-oriented systems**
  - [Springer SoSyM 2012]

- **Distributed event-based systems**
  - [IEEE ISORC 2008] [Springer SoSyM 2012]

- **Component-based software architectures**
  - [IEEE MASCOTS 2012] [Elsevier SciCo 2012]
Empirical Validation ("Proof-of-Concept")


DESIGN TIME vs. RUN-TIME MODELS FOR QUALITY-OF-SERVICE PREDICTION

PHASE 1
Online QoS prediction for problem anticipation

PHASE 2
Online reconfiguration impact prediction for trade-off analysis

PHASE 3
Autonomic system adaptation

Motivation ➔ Run-time Models ➔ Descartes Meta-Model ➔ CASE STUDY ➔ Summary & Outlook
Case Study: SPECjEnterprise2010

Business Logic

- Dealers
- Dealer Domain
- Customer Domain
- Corporate Domain
- Suppliers
- Supplier Domain
- Manufacturing Domain

Example Deployment (Oracle)

- Customer Relationship Management (CRM)
- Manufacturing
- Supply Chain Management (SCM)
- SPARC T4-4 Server + Sun Fire X4270 M2
- 444 CPU-Cores @ 3 GHz
- Oracle WebLogic + Database Server 11g

Motivation ➔ Run-time Models ➔ Descartes Meta-Model ➔ Case Study ➔ Summary & Outlook

© Samuel Kounev
Scenario

Experimental environment at KIT

High-level architecture model overview

- AppServer up to 20 nodes
  - 8 CPU cores per server
- Database server
  - 24 CPU cores
- 28 software components
- 63 behavior specifications
  - Control flow and data flow
  - Service resource demands
  - Parameteric dependencies
System Control Loop

Phase 1
- Refine/Calibrate Model(s)
- Forecast Workload

Phase 2
- Collect
  - Monitor System and Workload
    - *Service Workloads*
    - *Resource Utilization*
    - *SLAs*
  - Analyze
    - *SLA Violations*
    - *Inefficient Resource Usage*
    - Anticipate/Detect Problem

Phase 3
- Act
  - Reconfigure System

Problem resolved
- Decide
  - Predict Reconfiguration Effect(s)
  - Generate Reconfiguration Scenario

Problem persists
- Run-time Models
  - Design-Time vs. Run-Time Models for Quality-of-Service Prediction

© Samuel Kounev
PUSH

while $\exists c \in C : \neg P_R(c)$ do
  for all $t \in V(c[s]) : \neg P_U(t)$ do
    while $cap(c, t) \leq cap(c, t)$ do
      if $\exists i \in F(c[s], t) : i[\kappa] < i[\kappa]$ then
        $i[\kappa] \leftarrow i[\kappa] + 1$
      else
        $F(c[s], t) \leftarrow F(c[s], t) \cup \{i\}$
    end if
  end for
end while

PULL

for all $c \in C$
  while $\exists t \in V(c[s]) : U(t) - U(t) \geq \epsilon$ do
    if $\exists i \in F(c[s], t) : i[\kappa] > 0$ then
      $i[\kappa] \leftarrow i[\kappa] - 1$
    if $\neg P_R(c)$ then
      $i[\kappa] \leftarrow i[\kappa] + 1$
    end if
  if $i[\kappa] = 0$ then
    $F(c[s], t) \leftarrow F(c[s], t) \setminus \{i\}$
  end if
end while

PULL

- Add resources till SLAs are fulfilled
- vCPUs & AppServer cluster nodes

PUSH

- Release resources as long as no SLAs are violated

System Control Loop
Evaluation

Comparison of the model predictions with measurements on the real system

Prediction error: for utilization/throughput: < 5%, for response time: up to 30%

Example scenario: Deployment of a new service
Cooperation with VMware, Inc.

- Market leader in virtualization technology
- Cooperation since 2009
- “VMware Academic Research Award 2012”

- 3 year project aiming at
  - Model-based performance and resource management
  - Integration into virtualization platforms
Self-Aware Software Systems

- **Self-Reflective**
  - Aware of their software architecture, execution environment and hardware infrastructure, as well as of their operational goals (e.g., QoS and efficiency)

- **Self-Predictive**
  - Able to anticipate and predict the effect of dynamic changes in the environment, as well as the effect of possible adaptation actions

- **Self-Adaptive**
  - Proactively adapting as the environment evolves to ensure that their operational goals are continuously met

---

"I think, therefore I am…"

-- René Descartes
“Self-Aware Complex Systems Engineering”

- Service-oriented architectures & modeling techniques

Motivation

- Descartes Meta-Model

Case Study

- Software & Systems Engineering
- Computer Systems Modeling
- Cluster, Grid & Cloud Computing
- Distributed Systems & Autonomic Computing

- Stochastic models for QoS prediction
- Dynamic virtualized data center infrastructures
- Control theory and self-adaptation techniques

SUMMARY & OUTLOOK
**Motivation**  
Run-time Models  
Descartes Meta-Model  
Case Study  
**SUMMARY & OUTLOOK**

Design-Time vs. Run-Time Models for Quality-of-Service Prediction
Vielen Dank!

http://www.descartes-research.net