Combining fUML and Profiles for Non-Functional Analysis Based on Model Execution Traces

Tanja Mayerhofer, Philip Langer

Business Informatics Group
Institute of Software Technology and Interactive Systems
Vienna University of Technology
Favoritenstraße 911/1883, 1040 Vienna, Austria
phone: 43 (1) 5880118804 (secretary), fax: 43 (1) 5880118896
office@big.tuwien.ac.at, www.big.tuwien.ac.at

Luca Berardinelli

Sealab Quality Group
Department of Information Engineering, Computer Science and Mathematics
University of L’Aquila
Via Vetoio, Coppito 67100 L’Aquila, Italy
Outline

- Motivations
- UML-Based Analysis Methodologies:
  - translational approaches
  - fUML-based approaches
- The proposed framework
  - Model Execution
  - UML/fUML integration
  - UML-based Analysis
  - Reusable and Analysis-specific components
- Case Study: Early Performance Analysis
- Future work
Motivations

• **Problems**
  • fUML ignores UML Profiles: *stereotypes* and *tags* don’t have a running counterpart at the instance level.
  • fUML cannot be exploited by model-based methodologies that relies on model annotations
  • Semantic anchoring mechanisms (e.g., model transformations) are always required that link external semantics (and notations) to (a subset of )UML modeling elements

• **Solution (contribution)**
  • a reusable framework for performing model-based analyses leveraging execution traces of UML models and integrating UML profiles heretofore unsupported by fUML.

• **Example given**
  • Modeling solution for Early Performance Analysis
    • **notation**: fUML, MARTE Profile
    • **methodology**: Software Performance Engineering (SPE, by C. Smith)
    • **tool**: framework tailored for early model-based performance analysis
Motivations

Combining fUML and Profiles for Non-Functional Analysis Based on Model Execution Traces

Luca Berardinielli
Department of Information Engineering, Computer Science and Mathematics
University of Lecce
Lecce, Italy
luca.berardinielli@unile.it

Philip Langer
Business Informatics Group
Vienna University of Technology
Vienna, Austria
langer@bg.fh-wien.ac.at

Tanja Mayrhofer
Business Informatics Group
Vienna University of Technology
Vienna, Austria
mayrhofer@bg.fh-wien.ac.at

ABSTRACT
The development of software systems is crucial to understand non-functional properties already in an early development stage to guarantee that the system will satisfy its non-functional requirements. Following the model-based engineering paradigm, non-functional properties of the system being developed are described and verified during design. Although UML is widely used in model-based engineering, it is not suitable for non-functional analysis directly due to its lack of formal semantics. Therefore, various model-based analysis approaches transform UML models into formal languages (e.g., LTL, SPIN) for analysis purposes, which may involve accidental complexity in implementing the required model transformations.

Recently introduced fUML standard provides a formal semantics of a subset of UML enabling the execution of UML models. In this paper, we show how fUML can be utilized for analyzing UML models directly without having to translate them. We present a reusable framework for performing model-based analysis leveraging execution traces of UML models and recognizing UML profile violations supported by fUML. A case study in the performance analysis domain is used to illustrate the benefits of our framework.

Categories and Subject Descriptors
D.4 (Computing Methodologies); F.3.1 (Software Engineering): Design Tools and Techniques; D.2.9 (Computer Systems Organization): Performance of Systems

General Terms
Design, Performance

Keywords
Model-Based Analysis, fUML, UML Profiles

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Motivations

1. INTRODUCTION
To guarantee that non-functional requirements are fulfilled by a software system, it is of utmost importance to consider the system’s non-functional properties right from an early stage of the development. When applying the model-based engineering paradigm in developing software systems, models are used throughout the whole development process serving as the main specification of the system to be developed. Early in the development process, models are defined at a high level of abstraction and continuously refined in the next steps during the development process until an executable system can be derived (e.g., automatically using model transformations and code generation) [1]. This model-based engineering emphasizes the importance of architectural design by means of models that act as the primary artifact in the development process. This also facilitates model-based analysis and verification of functional and non-functional properties of the modeled system already in an early development stage to ensure the quality of the system and to consider costly time in later phases of the development process.

Evaluating the effects of architectural adaptations at runtime

Ninth International ACM Sigsoft Conference on the Quality of Software Architectures
QoSA 2013
Vancouver, Canada, June 17-21, 2013
The conference is a part of CompArch 2013
http://comparch2013.org

Notification: March, 25th

Architecture Evaluation
• quality assessment of legacy systems and third party applications
• empirical validation, testing, prototyping and simulation for assessing architecture quality
different attributes of software architectures
• methods and processes for evaluating architecture quality
• model-driven evaluation of software architecture quality
• evaluating the effects of architectural adaptations at run-time
• lessons learned and empirical validation of theories and frameworks on architecture quality
UML-Based Analysis Methodologies: translational approaches

- **UML-based methodologies** prescribe model annotations on input UML Models added through user-defined or standard UML Profiles.
- **Integration Tools** produce Analysis Models in different (textual/graphical) (formal) notations for sake of semantics anchoring.
- **Model Analysis Reports** are produced by Analysis Tools to assess functional/non-functional properties of the modeled system.

- Three “metamodels” combined (UML, S-PMIF, PMIF);
- Four unidirectional m2m transformations (UML->SPMIF, UML->PMIF, SPMIF+PMIF->SPMIF, SPMIF+PMIF->PMIF);
- Three different technical spaces (MOF, Ecore, XML);
- Toolset: modeling tools (UML, S-PMIF, PMIF), analysis tools (Execution Graph Analysis, QN Solvers)

UML-Based Analysis Methodologies: translational approaches

**Pros**
- **Existing analysis techniques and tools** for the target language can be directly exploited.

**Cons**
- Additional level(s) of indirection
- Deep knowledge of source/target(s) metamodels and semantics, model transformation techniques.
- **Implementation and maintenance of model transformations (as Integration Tools)**
- Academic tools less stable than industrial-strength ones

**Model transformations for semantic anchoring**
UML-Based Analysis Methodologies: fUML-based approaches - Overview

**Cons**

- (not) additional level of indirection
- (not) implementation and maintenance of model transformations
- (not) deep knowledge of source/target(s) metamodels and semantics, model transformation techniques.
UML-Based Analysis Methodologies: fUML-based approaches - Overview

Pros → Contribution of paper
- Integration of the analysis algorithms directly with the design modeling language
- Integration of Model Analysis Report(s) directly with the design modeling language
The proposed framework: fUML-based Model Execution

- **fUML View**: a UML Model whose elements belong to the fUML language units (*Classes, Common Behaviors, Activities, and Actions*)

- **Extended fUML VM**: standard fUML VM + moliz extensions for

1. **analyzing** the execution  
   - Chronological execution order  
   - Input / output  
   - Token flow  
   - Call hierarchy  

2. **observing** the execution state  

3. **controlling** the execution  

   ➔  Trace model

   ➔  Event model

   ➔  Command API
The proposed framework: fUML-based Model Execution

- **fUML View**: a UML Model whose elements belong to the fUML language units (*Classes, Common Behaviors, Activities, and Actions*).

- **Extended fUML VM**: standard fUML VM + **moliz extensions for**

1. **analyzing** the execution ➔ **Trace model**
2. **observing** the execution state ➔ **Event model**
   - Start / end of activity execution
   - Start / end of activity node execution
   - Suspension of activity execution
   - Modification of extensional values
3. **controlling** the execution ➔ **Command API**
The proposed framework: fUML-based Model Execution

- **fUML View**: a UML Model whose elements belong to the fUML language units (*Classes, Common Behaviors, Activities, and Actions*)
- **Extended fUML VM**: standard fUML VM + **moliz extensions for**

1. **analyzing** the execution ➔ **Trace model**
2. **observing** the execution state ➔ **Event model**
3. **controlling** the execution ➔ **Command API**
   - Start / stop execution
   - Resume execution until certain activity node
   - Resume execution stepwise
   - Retrieve runtime information
The proposed framework: UML/fUML Integration

- **fUML Adapter**: it truncates all unsupported UML elements, as well as the profile applications, and generates the in-memory representation of fUML models processable by the fUML virtual machine;

- **UML Trace Adapter**: It maps the fUML Execution Traces with the corresponding modeling elements on the source UML Model by exploiting mapping information saved by the fUML Adapter. integrated UML execution trace is created

- **UML Execution Trace**: it refers to the original elements of the UML model instead of the fUML view.
The proposed framework: UML-based Analysis

Model-based Analyzer: A model-based analysis is typically performed to answer certain questions about the expected behavior of the modeled system. The expected behavior is directly reflected by the trace of the model's execution and can now be analyzed using a model-based analyzer, which computes certain metrics by interpreting the execution trace and the additional information captured using profile applications.
The proposed framework: Reusable and specific to analysis components

fUML-based Modeling Solution for Software Performance Engineering (SPE)
Case Study: Early performance analysis based on fUML (1)

Case Study
- e-health system
- Retrieval of patient information
- Analyze end-to-end execution time

Software View: UML Classes (structure)
- software components
- component operations

Hardware View: UML Classes (structure)
- software allocation on hardware
- hardware platform resources: processing, storage, communication

Marte Annotations: hardware resources details like
- frequencies of CPUs
- bandwidth of comm.networks
- disk access time
- ...

Ignored by fUML
Case Study: Early performance analysis based on fUML (2)

- **Software View:** UML Activities (behavior)
  - associated to each component operation
  - MARTE Annotations: resource usages (a.k.a. demand vectors in SPE) for any atomic behavioral unit. The usedResources correspond to those equipping the execution host where the software components are allocated.
Case Study: Early performance analysis based on fUML (3)

Analysis of execution time based on **Execution Graph** (**graph reduction algorithm**)

EGs are particularly suitable during the early stages of a software development process because an EG "[...] characterizes the resource requirements of proposed software alone, in the absence of other workloads, multiple users or delays due to contention for resources." [10]
Case Study: Early performance analysis based on fUML (3)

Analysis of execution time based on **Execution Trace** *(graph reduction algorithm)*

- **t**: Trace
  - **ae1**: ActivityExecution
    - activity = eHS::invokeServices()  
    - execTime = 0.2888 sec

- **c1**: CallActionExecution
  - callee = Client::requestPatientInfoPages()  
  - execTime = 0.0767 sec

- **ae2**: ActivityExecution
  - activity = Client::requestPatientInfoPages()
  - execTime = 0.0722 sec

- **c2**: CallActionExecution
  - callee = AppServer::getPatientData()  
  - execTime = 0.0736 sec

- **ae3**: ActivityExecution
  - activity = AppServer::getPatientData()
  - execTime = 0.2888 sec

- **c3**: CallActionExecution
  - callee = displayResults
  - execTime = 0.0736 sec

- **c4**: CallActionExecution
  - callee = Database::loginInteraction()
  - execTime = 0.0767 sec

- **c5**: CallActionExecution
  - callee = Database::getPatientData()
  - execTime = 0.0736 sec

- **c6**: CallActionExecution
  - callee = Database::getMedicalHistories()
  - execTime = 18.5570 sec

- **c7**: CallActionExecution
  - callee = Database::getDiseaseData()
  - execTime = 9.2473 sec

- **c8**: CallActionExecution
  - callee = ImageServer::getXrayImages()
  - execTime = 18.5570 sec

- **c9**: CallActionExecution
  - callee = ImageServer::getXrayImages()
  - execTime = 28.3156 sec

- **c10**: CallActionExecution
  - callee = ImageServer::getXrayImages()
  - execTime = 184.3489 sec
Case Study: Early performance analysis based on fUML (3)

Analysis of execution time based on **Execution Trace** *(graph reduction algorithm)*

Table 2: Execution times of the behavioral units of the *eHealthSys* RPD service.

<table>
<thead>
<tr>
<th>Component</th>
<th>Behavioral Unit</th>
<th>Model Element</th>
<th>Remote/Local</th>
<th>execTime</th>
<th>Input/Calculated</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Client</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RPD</td>
<td>Activity</td>
<td></td>
<td></td>
<td>185,7795</td>
<td>calculated</td>
</tr>
<tr>
<td>RPD_service</td>
<td>CallOperationAction</td>
<td>remote</td>
<td></td>
<td>185,7073</td>
<td>calculated</td>
</tr>
<tr>
<td>elaborate_display_results</td>
<td>CallBehaviorAction</td>
<td>local</td>
<td></td>
<td>0,0722</td>
<td>input</td>
</tr>
<tr>
<td><strong>Application Server</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RPD_service</td>
<td>Activity</td>
<td></td>
<td></td>
<td>185,7073</td>
<td>calculated</td>
</tr>
<tr>
<td>login</td>
<td>CallOperationAction</td>
<td>remote</td>
<td></td>
<td>0,0722</td>
<td>calculated</td>
</tr>
<tr>
<td>getPatientData</td>
<td>CallOperationAction</td>
<td>remote</td>
<td></td>
<td>0,2888</td>
<td>calculated</td>
</tr>
<tr>
<td>getMedicalHistories</td>
<td>CallOperationAction</td>
<td>remote</td>
<td></td>
<td>2,3067</td>
<td>calculated</td>
</tr>
<tr>
<td>getXrayImages</td>
<td>CallOperationAction</td>
<td>remote</td>
<td></td>
<td>18,5570</td>
<td>calculated</td>
</tr>
<tr>
<td>getDiseaseImages</td>
<td>CallOperationAction</td>
<td>remote</td>
<td></td>
<td>9,2473</td>
<td>calculated</td>
</tr>
<tr>
<td>elaborate_send_results</td>
<td>CallBehaviorAction</td>
<td>local</td>
<td></td>
<td>155,2333</td>
<td>input</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>login</td>
<td>Activity</td>
<td></td>
<td></td>
<td>0,0722</td>
<td>calculated</td>
</tr>
<tr>
<td>elaborate_send_results</td>
<td>CallBehaviorAction</td>
<td>local</td>
<td></td>
<td>0,0722</td>
<td>input</td>
</tr>
<tr>
<td>getPatientData</td>
<td>CallOperationAction</td>
<td></td>
<td></td>
<td>0,2888</td>
<td>calculated</td>
</tr>
<tr>
<td>getMedicalHistories</td>
<td>Activity</td>
<td>local</td>
<td></td>
<td>0,2888</td>
<td>input</td>
</tr>
<tr>
<td>elaborate_send_results</td>
<td>CallBehaviorAction</td>
<td>local</td>
<td></td>
<td>2,3087</td>
<td>input</td>
</tr>
<tr>
<td><strong>Image Database</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>getXrayImages</td>
<td>Activity</td>
<td></td>
<td></td>
<td>18,5570</td>
<td>input</td>
</tr>
<tr>
<td>elaborate_send_results</td>
<td>CallBehaviorAction</td>
<td>local</td>
<td></td>
<td>18,5570</td>
<td>input</td>
</tr>
<tr>
<td>getDiseaseImages</td>
<td>Activity</td>
<td></td>
<td></td>
<td>9,2473</td>
<td>calculated</td>
</tr>
</tbody>
</table>
Future Work: overview

Different Types of possible future work:

• **Modeling Solutions**
  • for different Functional Analyses
  • for Non-Functional Analyses
  • for combined Functional/Non-Functional Analyses

• **Extending the Model Execution component**

• **Extending the applicability of the framework @runtime**
Future Work: Modeling Solutions

fUML-based Modeling Solutions for Non-Functional Analysis

1. **reliability analysis**

2. **trade-off analysis**.
   - **Methodologies**: EG Analysis (available) + Reliability (1)

3. **context-aware analysis**
   - **Possible Combinations**:
     - Performance (current)
     - Reliability (1)
     - Trade-off (2)
As software becomes increasingly pervasive, the need for quality and reliability in software systems continues to increase. Model-based methods have been an important development in helping organizations build software with higher quality. Prior to the development of the UML Testing Profile (UTP), UML (which is the most popular modeling language) did not have support for testing-related activities.

The UTP provides extensions to UML to support the design, visualization, specification, analysis, construction, and documentation of the artifacts involved in testing.

It is independent of implementation languages and technologies, and can be applied in a variety of domains of development.
Future Work: Extension of the fUML-based Model Execution component

5. Introducing the concept of Time in the fUML VM.

References:
http://link.springer.com/chapter/10.1007%2F978-3-642-15234-4_6

In this paper (cited in the work submitted to QoSA) the authors proposed the coupling of fUML with TimeSquare (http://timesquare.inria.fr/).

TimeSquare has been developed by a partner of the PRESTO project (http://www.presto-embedded.eu/) where I’m currently involved.

We can contact them to establish collaboration on this possible extension.
Future Work: Extension of the fUML-based Model Execution component

6. **Runtime Model for UML StateMachines**
   - replicate the fUML-based Model Execution for UML StateMachines;
   - an open source implementation of a SM VM is available at [http://commons.apache.org/proper/commons-scxml/](http://commons.apache.org/proper/commons-scxml/)

Both fUML VM and SM VM have been integrated in **Cameo Simulation Toolkit**
without any support for Execution Traces

then...

7. **Implementing the current framework as a plugin for Cameo Simulation Toolkit**
   - still based on Eclipse UML (it seems possible)
   - collaboration with MD *(conflict of interest?)*
Future Work: Extension of the fUML-based Model Execution component

8. fUML Semantics vs. QN Semantics vs. PN Semantics

Is fUML Semantics suitable for supporting QN-based and/or PN-based Analysis Techniques?

+ Time
+ Queuing Networks
  (M/M/1)

an M/M/1 queue represents the queue length in a system
• having a single server,
• where arrivals are determined by a Poisson process and
• job service times have an exponential distribution.

+ Petri Nets
+ Time
+ Petri Nets
= Stochastic Petri Net
Future Work: WSN, new application domain

8. Applying MDE fUML-based approaches to WSNs
   • still based on Eclipse UML (it seems possible)
   • collaboration with MD (any conflict of interest?)

http://mobilab.wustl.edu/projects/agilla/
Future Work: Extending the applicability of the framework @runtime

All the proposed extensions are applicable

Future Work:
Extending the applicability of the framework @runtime
Question Time
Event model

**Event**
- timestamp : long
  + parent 0..1

**TraceEvent**
- executionID : int

**ExtensionalValueEvent**
- type : ExtensionalValueEventType

**ActivityEvent**

**ActivityEntryEvent**

**ActivityExitEvent**

**ActivityNodeEvent**

**ActivityNodeEntryEvent**

**ActivityNodeExitEvent**

**SuspendEvent**

**BreakpointEvent**

**Breakpoint**

**fUML::Semantics::Classes::Kernel::ExtensionalValue**
+ extensionalValue

**fUML::Semantics::Classes::Kernel::FeatureValue**
+ featureValue

**fUML::Syntax::Activities::IntermediateActivities::Activity**

**fUML::Syntax::Activities::IntermediateActivities::ActivityNode**

**fUML::Semantics::Classes::Kernel::Element**
+ location 1

**ExtensionalValueEventType**
- «enumeration»
  - CREATION
  - DESTRUCTION
  - TYPE_ADDED
  - TYPE_REMOVED
  - VALUE_CREATION
  - VALUE_DESTRUCTION
  - VALUE_CHANGED
Command API

void execute(Behavior activity, Object context, ParameterValueList input)
void executeStepwise(Behavior activity, Object context, ParameterValueList input)
void nextStep(int executionID, ActivityNode node)
void resume(int executionID)
void terminate(int executionID)
void addBreakpoint(Breakpoint breakpoint)