**Abstract**

Various methodologies exist to annotate software models with data related to performance, and to translate the annotated models into performance models. A relevant objective is now at hand in this direction, that is to make these methodologies acceptable from the software engineering community. We present here the XPRIT tool that allows to annotate UML diagrams and translate them into models ready to performance validation, that are Execution Graphs and Queueing Networks. The translation is based on an existing methodology (PRIMA-UML), and the output models represent the inputs to a Software Performance Engineering approach.

1. Introduction

In the last few years many efforts have been devoted to fill the gap between software development and performance validation. Primary sources of this gap are the short time to market as well as the special skills and knowledge that are required for software developers dealing with performance validation.

We introduce here the XPRIT (XML-based PRIMA-UML Tool) tool that allows to annotate UML diagrams with data related to performance issues, and to automatically generate a performance model from the annotated diagrams. This work represents an implementation of the PRIMA-UML methodology [1]. Following PRIMA-UML, an Use Case Diagram and several Sequence Diagrams can be annotated with the operational profile of the software system (i.e., user profile and stochastic distributions over nondeterministic choices), and a Deployment Diagram can be annotated with the internal configuration of each platform site (i.e., number and characteristics of devices on the site).

XPRIT is made of two components: (i) UML2EG, that allows to annotate Use Case and Sequence Diagrams, and to generate from the annotated diagrams an Execution Graph, i.e. a graph representing the software dynamics and the amount of resources that each software step requires for its execution [2]; (ii) UML2QN, a tool that allows to annotate a Deployment Diagram, and to generate from the annotated diagram a Queueing Network representing the hardware platform where the software shall run. The outputs of UML2EG and UML2QN merge into the Software Performance Engineering (SPE) approach [2][4], in that an Execution Graph represents the software workload used to parameterize a Queueing Network. The solution of the parameterized Queueing Network gives the values of the performance indices of interest.

Both tools are based on exporting an XMI representation of diagrams from UML modeling environments. All the annotations and transformations are based on XML representations of: additional information, intermediate models and final models (i.e. Execution Graphs and Queueing Networks). The tool implementation has been driven from transparency, usability and scalability criteria.

2. **XPRIT**: the tool architecture

In Figure 1 a view of the tool and its relationships with other tools is given. In order to gain experience on UML-based design tools, we have chosen different environments to draw the UML diagrams, among the ones that allow the exporting of the diagrams in the XMI representation. UML2EG takes as input an XMI representation of Use Case and Sequence Diagrams, which has been obtained by exporting the diagrams from the Poseidon tool [9]. On the other side, UML2QN takes as input an XMI representation of a Deployment Diagram, which has been obtained by exporting it from the ArgoUML tool [6]. Note that this choice does not limit the scope of the tool components, as both the representations fulfill the XMI specifications [10] even though they slightly differ from each other.

For sake of effectiveness, the first action performed from each tool component (i.e., UML2EG and UML2QN) is to
filter the entering XMI representation in order to trash all
the information needless to the translation step. Two XML
Schemas have been defined for the outputs of the filtering
steps (i.e., Use Case and Sequence Diagram on one side,
and Deployment Diagram on the other side). Hence any
UML modeling environment may be made compliant to our
tool upon providing two filters from its exported XMI rep-
resentation of UML diagrams to the XML Schemas that we
have defined.

In Figure 1 the annotations needed to perform the trans-
lation have been split into software annotations, entering the
UML2EG component, and platform annotations, entering the
UML2QN component. Two XML Schemas have been
defined for both types of annotations, and XPRIT provides
a user-friendly GUI to allow users entering the annotation
data (such as the operational profile or the number and type
of devices on each platform site).

The outputs of XPRIT (i.e. Execution Graph and Queue-
ing Network topology represented in XML) represent the
inputs of a tool based on the Software Performance En-
ingineering approach [2] (e.g. SPE-ED [8]). The scope of
XPRIT is clearly bounded to the production of an Execu-
tion Graph and a Queueing Network topology in XML. The
integration of the XPRIT outputs with existing QN builders
and solvers could be achieved through XSLT transforma-
tions from our XML Schemas to the internal format of ex-
isting tools, analogously to what we have done for graphi-
cal purposes.

Indeed, as shown at the bottom of Figure 1, the outputs
of XPRIT also feed two viewer tools. The GraphViz tool [7]
has been chosen for the graphical representation of an Exec-
ution Graph. An algorithm to translate the XML represent-
ation of an Execution Graph to the internal notation of
GraphViz has been implemented in XSLT in order to fully
automate the visualization of an Execution Graph starting
from UML Use Case and Sequence Diagrams. Again to dif-
ferentiate the experiences between the XPRIT components,
GraphViz has not been adopted to represent Queueing Net-
works. An XSLT code has been instead implemented to
translate our XML representation of a Queueing Network
into SVG (Scalable Vector Graphics), which is a standard
data format imported from most browsers for graphical vi-
ualization.

As a final consideration, we like to remark that no XML
standard format for performance model representation has
been yet proposed and accepted. The model-to-model trans-
formations, in this area, usually have the proprietary nota-
tions of performance model solvers as targets. A very in-
teresting work to introduce a standard language for per-
formance model representation has been done in [5], and
later in [3]. A meta-model for a Performance Model Inter-
change Format (PMIF) has been first proposed in [5], with
the aim of defining information to be exchanged between
CASE and performance tools. Classes, attributes and rela-
tionships have been introduced to describe a software work-
load in terms of basic elements of an Execution Graph. A
similar job has been done in [3] to describe a Queueing Net-
work model.

References

Validation Incremental Methodology on Early UML Dia-
grams", Science of Computer Programming, 44 (2002), 101-
Addison-Wesley, 1990.
tical guide to creating responsive, scalable software", Addison-
[5] L.G. Williams, C. Smith, "Information Requirements for Soft-
[10] OMG, "XML Metadata Interchange (XMI) 2.0", formal/03-
05-02.