Martin Gogolla
Cris Kobryn (Eds.)

«UML» 2001 – The Unified Modeling Language

Modeling Languages, Concepts, and Tools

LNCS 2185

4th International Conference
Toronto, Canada, October 2001
Proceedings

Springer

Martin Gogolla Cris Kobryn (Ed.)

«UML» 2001 – The Unified Modeling Language

Modeling Languages, Concepts, and Tools

4th International Conference
Toronto, Canada, October 1-5, 2001
Proceedings

Springer

Series Editors

Gerhard Goos, Karlsruhe University, Germany
Juris Hartmanis, Cornell University, NY, USA
Jan van Leeuwen, Utrecht University, The Netherlands

Volume Editors

Martin Gogolla
University of Bremen, Department of Mathematics and Computer Science
Database Systems Group
P.O. Box 33 04 40, 28334 Bremen, Germany
E-mail: gogolla@informatik.uni-bremen.de

Cris Kobryn
Telelogic Technologies
P.O. Box 23 20, Fallbrook, CA 92088, USA
E-mail: cris.kobryn@telelogic.com

Cataloging-in-Publication Data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

The unified modeling language : modeling languages, concepts, and tools ;
4th international conference ; proceedings / "UML" 2001, Toronto, Canada,
October 1- 5, 2001. Martin Gogolla ; Cris Kobryn (ed.). - Berlin ;
Heidelberg ; New York ; Barcelona ; Hong Kong ; London ; Milan ; Paris ;
Tokyo : Springer, 2001
(Lecture notes in computer science ; Vol. 2185)
ISBN 3-540-42667-1

CR Subject Classification (1998): D.2, D.3, K.6

ISSN 0302-9743
ISBN 3-540-42667-1 Springer-Verlag Berlin Heidelberg New York

Springer-Verlag Berlin Heidelberg New York
a member of BertelsmannSpringer Science+Business Media GmbH

http://www.springer.de

Typesetting: Camera-ready by author, data conversion by PTP- Berlin, Stefan Sossna
Printed on acid-free paper SPIN 10840541 06/3142 5 4 3 2 1 0

Preface

In the four years since the Object Management Group adopted the Unified Modeling Language (UML) in 1997, it has become widely accepted throughout the software industry and successfully applied to diverse domains. During this time it has become the de facto standard for specifying software blueprints, which continue to increase in value as we evolve from analysis and design models to multi-view architectures. Indeed, it is becoming difficult to find a software project with more than ten developers who don't use UML in some way to specify part of their architecture.

Despite its rapid and widespread acceptance, however, the UML 1.x series of revisions has not been without its problems. Some of the major issues commonly cited include: excessive size, gratuitous complexity, limited customizability, non-standard implementations, and lack of support for diagram interchange. Such substantive problems can only be addressed by major revisions to UML. Fortunately, the Object Management Group realizes this and has issued four Requests for Proposals for UML 2.0.

UML 2.0 represents both a wonderful opportunity and a serious responsibility for the UML community. It is an opportunity to resolve the serious shortcomings listed above; it is also a responsibility to ensure that the second version of the language does not suffer from "second system syndrome." This conference, whose objective is to bring together researchers and practitioners to share their visions for the future of UML, is an ideal place to explore how we can exploit the opportunity and share the responsibility for UML 2.0. Now in its fourth year, the ≪UML≫ conference series remains the premier forum for presenting and discussing innovative ideas that will make UML easier to learn, apply, and implement.

In total 122 abstracts and 102 papers were submitted to this year's conference, of which 32 were selected by the program committee for presentation. As in 2000, this year's conference included a two-day tutorial and workshop session, in which nine tutorials and five workshops were scheduled. The primary purpose of these sessions was to provide a more informal forum for discussing state-of-the-art research in UML. Topics included: Agile modeling, teaching UML, concurrency, rigorous development methods, OCL, software architecture, concurrent, distributed, and real-time applications, tools, requirements, time-critical systems, meta-modeling, quality assurance, effective diagrammatic languages and executable UML. A short description of the workshops and tutorials can be found in these proceedings and details at the conference web site: `http://www.cs.toronto.edu/uml2001/`.

We would like to express our deepest appreciation to the authors of submitted papers, tutorials, workshops, and panels, and the program committee members

and the additional referees. Jaelson Castro together with Manuel Kolp did an excellent job of managing all matters of the conference organization. Heinrich Hußmann chaired the workshop and tutorial submissions. We would also like to thank Werner Damm, John Mylopoulos and James Rumbaugh for agreeing to present invited talks at the conference. Mark Richters and Oliver Radfelder at the University of Bremen are thanked for their contribution to setting up the conference web site and in organizing and handling the electronic submission process. The ConfMan program (`http://confman.unik.no/~confman/ConfMan/`) was used to gather and organize submitted papers and reviews, and Mark Richters extended it to deal with an online preference selection process for the PC members. Ralf Kollmann at the University of Bremen organized the preparation of the final version of the conference proceedings. We would also like to thank the ≪UML≫ steering committee for their advice, Jean-Michel Bruel and Robert France for maintaining the mailing list, and last year's program chair, Andy Evans, for lots of helpful emails and hints.

July 2001

Martin Gogolla
Cris Kobryn

Organization

Executive Committee

General Chair:	Cris Kobryn (Telelogic Technologies, USA)
Conference Chair:	Jaelson Castro (Universidade Federal de Pernambuco, Brazil)
Program Chair:	Martin Gogolla (Universität Bremen, Germany)
Tutorial/Workshop Chair:	Heinrich Hußmann (Technische Universität Dresden, Germany)

Organizing Team

Publicity Chair (Europe):	Jean-Michel Bruel (University of Pau, France)
Publicity Chair (Americas):	Robert France (Colorado State University, USA)
Program Organization:	Ralf Kollmann (Universität Bremen, Germany) Oliver Radfelder (Universität Bremen, Germany) Mark Richters (Universität Bremen, Germany)
Local Organization:	Manuel Kolp (University of Toronto, Canada)

Program Committee

Colin Atkinson (DE)
Jean Bezivin (FR)
Marco Boger (DE)
Grady Booch (US)
Jean-Michel Bruel (FR)
David Bustard (UK)
Betty Cheng (US)
Derek Coleman (US)
Steve Cook (UK)
Desmond D'Souza (US)
John Daniels (UK)
Bruce Douglas (US)
Gregor Engels (DE)
Andy Evans (UK)
Robert France (US)
Brian Henderson-Sellers (AU)
Pavel Hruby (DK)
Peter Hruschka (DE)
Heinrich Hußmann (DE)
Jean-Marc Jezequel (FR)
Stuart Kent (UK)
Haim Kilov (US)
Steve Mellor (US)
Richard Mitchell (UK)
Ana Maria Dinis Moreira (PT)
Pierre-Alain Muller (FR)
Gunnar Övergaard (SE)
James Rumbaugh (US)
Bernhard Rumpe (DE)
Andy Schürr (DE)
Bran Selic (CA)
Keng Siau (US)
Perdita Stevens (UK)
Alfred Strohmeier (CH)
Jos Warmer (NL)
Alan Wills (UK)

Additional Referees

João Araújo
Toby Baier
Julian Bradfield
Benoit Baudry
Didier Buchs
Olivier Burgard
Benoit Caillaud
R. G. Clark
Birgit Demuth
Massimo Felici
Frederic Fondement
Falk Fünfstück
Sudipto Ghosh
Nabil Hameurlain
J.H. Hausmann
Reiko Heckel
Annig Lacayrelle
Katharina Mehner
Manfred Muench
Thierry Nodenot
Francois Pennaneach
Noël Plouzeau
Mohamed Kandé
Anneke Kleppe
Thomas Kühne
Jochen Küster
Juliana Kuester-Filipe
Frank-Ulrich Kumichel
Stefan Sauer
Ansgar Schleicher
Lothar Schmitz
João Costa Seco
Shane Sendall
Zixing Shen
Gerson Sunyè
Anne Thomas
Yuhong Tian
Yves Le Traon
Guido Wimmel

Sponsors

IEEE Computer Society
http://www.computer.org

IEEE-CS Technical Committee
on Complexity in Computing (TCCX)
http://www.elet.polimi.it/tccx/

Corporate Donors

Telelogic Technologies
http://www.telelogic.com

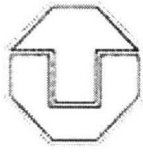
Rational Software Corporation
http://www.rational.com

Academic Supporters

Universidade Federal Pernambuco
http://www.ufpe.br

Universität Bremen
http://www.uni-bremen.de

University of Toronto
http://www.toronto.edu



Technische Universität Dresden
http://www.uni-dresden.de

Table of Contents

Invited Talk

Metamodeling

Activity Diagrams

OCL

Architecture and Patterns

Analysis and Testing

Performance and Databases

Invited Talk

Graph Transformations

Real-Time and Embedded Systems

Associations and Ontology

Statecharts

Invited Talk

Components

Use Cases

Workshops and Tutorials

The Preacher at Arrakeen

Jim Rumbaugh

Rational Software Corporation

Abstract. In the Dune novels, Paul Atreides fights a battle for survival against nefarious forces and succeeds in uniting the Universe under his control. Eventually, however, a bureaucratic and militaristic religion grows up around his legend. Disillusioned by the atrocities committed in his name, Paul abandons his throne and returns in disguise as the mysterious Preacher at Arrakeen to denounce the bureaucracy, fanaticism, and tyranny of his out-of-control followers. Sometimes that's how I feel about UML. This talk (sermon?) will denounce the excesses of the UML cult and see if it can be saved from its friends.

M. Gogolla and C. Kobryn (Eds.): UML 2001, LNCS 2185, p. 1, 2001.

An Action Semantics for MML

José M. Álvarez[1], Tony Clark[2], Andy Evans[3], and Paul Sammut[3]

[1] Dpto. de Lenguajes y Ciencias de la Computación.
University of Málaga, Málaga, 29071, Spain
alvarezp@lcc.uma.es
[2] Dpt. of Computer Science, King's College,
Strand, London, WC2R 2LS, United Kingdom
anclark@dcs.kcl.ac.uk
[3] Dpt. of Computer Science, University of York,
Heslington, York, YO1 5DD, United Kingdom
andye@cs.york.ac.uk, pauls@cs.york.ac.uk

Abstract. This paper describes an action semantics for UML based on the Meta-Modelling Language (MML) - a precise meta-modelling language designed for developing families of UML languages. Actions are defined as computational procedures with side-effects. The action semantics are described in the MML style, with model, instance and semantic packages. Different actions are described as specializations of the basic action in their own package. The aim is to show that by using a Catalysis like package extension mechanism, with precise mappings to a simple semantic domain, a well-structured and extensible model for an action language can be obtained.

1 Introduction

The UML actions semantics has been submitted by the action semantics consortium to "extend the UML with a compatible mechanism for specifying action semantics in a software-independent manner" [1]. The submission defines an extension to the UML 1.4 meta-model which includes an abstract syntax and semantic domain for an action language. This language provides a collection of simple action constructs, for example write actions, conditional actions and composite actions, which can be used to describe computational behaviours in a UML model. A key part of the proposal is a description of the semantics of object behaviour, based on a history model of object executions.

Unfortunately, the action semantics proposal suffers from a problem commonly met when developing large meta-models in UML - how to structure the model so as to clearly separate its different components. Failure to achieve this results in a meta-model that is difficult to understand and to modify, particularly, to specialize and extend. In addition, meta-models based on the current UML semantics suffer from a lack of a precisely defined semantic core upon which to construct the meta-model. This means that it is often hard to ascertain the correctness of the model, and to overcome this, significant work must

M. Gogolla and C. Kobryn (Eds.): UML 2001, LNCS 2185, pp. 2–18, 2001.

be invested in clarifying the semantics before any progress can be made. On the positive side, the basic semantic model used in the action semantics, with its notion of snapshots and history and changes seems quite appropriate to define the different changing values of a system. In addition, the actions defined in the submission thoroughly cover the wide range of actions necessary for a useful action language. Thus, if a way can be found to better restructure what is a significant piece of work, then clearly there will benefits to all users and implementors of the language.

The purpose of this paper is to show how the definition of a precise semantic core and the use of a Catalysis [2] like package extension mechanism can result in a better structured and adaptable definition of the action semantics. The work is based on an extension of the Meta-modelling Language (MML) [6], a precise meta-modelling language developed to enable UML to be re-architected as a family of modelling languages. However, it must be clear that this is not an intent to solve the general problem of model executability, but only a proposal to describe executability features in a MML context.

1.1 The Basics of the MML Model

MML is a metamodeling language intended to rearchitect the core of UML so it can be defined in a much simpler way and it can be easily extended and specialised for different types of systems. Among other basic concepts, MML establishes two orthogonal distinctions, the first one being a mapping between model concepts (abstract syntax) and instances (semantic domain). The second is the distinction between model concepts and concrete syntax and applies both to models and instances. The syntax is the way concepts are rendered. Models and instances are related by a semantic package that establishes the satisfaction relationship between instances and models. A similar relationship is defined between between model concepts and concrete syntax.

These distinction are described in terms of a language definition pattern, see Figure 1. Each component in the pattern is defined as a package. As in UML, a package is a container of classes and/or other packages. In addition, packages in MML can be specialised. This is the key mechanism by which modular, extensible definitions of languages are defined in terms of fundamental patterns, and is similar to the package extension mechanism defined in [2]. Here, specialization of packages is shown by placing a UML specialization arrow between the packages. The child package specializes all (and therefore contains all) of the contents of the parent package.

Another important component of the MML is its core package, which defines the based modelling concepts used in UML: packages, classes and attributes. Currently, the MML model concepts package does not provide a dynamic model of behaviour. Thus, in order to define a semantic model for the action semantics in the MML, an extension must be made to the core MML package. This is described in the following sections.

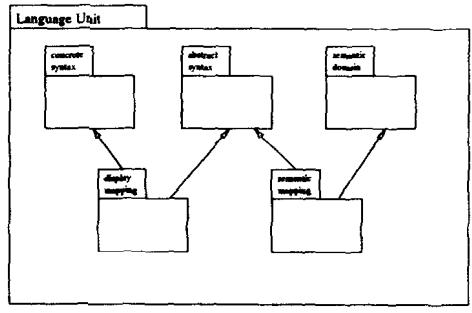


Fig. 1. The MMF Method

2 Principles of the New Action Semantics

Two basic goals have led to the redefinition of the action semantics. The first one is to include the action semantics as the dynamic core of MML, and possibly substitute the static core. This implies the definition of model and instances views and separation between concepts and syntax. The second goal is to have this action semantics as simple as possible and as easy to extend and specialize as possible. For the goal of simplicity, it is necessary to define as few new concepts as possible. One of the ways to do this is to reuse whenever possible the concepts already defined in the other packages of MML. It is also important to abstract out the fundamental concepts common to all actions and to be as removed as possible from the implementational aspects of actions. MML is designed to be easily extensible, as will the action semantics if we include it as another part of MML. As the actions semantics will be another package in MML, it has to follow the structure of the rest of packages, that is, the package should be composed of model, instance and semantics packages, with the model and instance packages further divided in packages for concepts, syntax and the mapping between concepts and syntax.

2.1 New Basic Concepts

The dynamic core tries to model the evolution of the values of the objects in the system with time. This is in contrast with the view of the static model that considers instances to be attached to a single value. The approach taken to define the dynamic model considers a history as a sequence of values, often called snapshots, being the execution of the actions responsible for the progression from one value to the next. A snapshot can be related to the whole system, as it is in the first approach to actions in the MMF document, or to a single object as it is in this approach. Only those acts causing the change of a value will be considered to be actions. For example, to write a new value in an instance slot will be an action as the value of the instance slot is different before and after the execution of the action. However, the reading of the value of a variable will not be

considered as an action as no element in the system changes its value. These kind of acts will be considered to be expressions. In the current definition of MML, expressions are defined to model the OCL. There is also a subclass of expression called method, which is used to model the static methods of classes. Thus, the basic notion of an action is of a model element that relates a series of values with a series of elements. These input values will be used in the action execution to update some of the values of the elements associated with the action. The specific semantics of every action will be described in every subclass action in terms of its class diagram and well-formedness rules. Unlike in the previous proposal for action semantics, there is no concept of action execution history with a step for every state in the execution. In this model, the action execution is simply the occurrence of the action. If it is a compound action, it can be decomposed into simpler actions.

2.2 Time

Time was introduced in the previous action semantics definition to define a timing order in the dynamic model. In our point of view, time is not a concept general to every type of systems, so will not be used in the basic dynamic model. However, particular systems as real-time systems can easily extend this model to cope with the notion of time as best fitted to its purposes.

3 Actions

An Action represents a computational procedure that changes the state of an element in the system. In order to execute, an action requires some input values that will be used to compute the new values for other elements in the system.

Methods in MML are also used to define computational procedures. Methods are side-effect free - they simply evaluate a set of parameters against an OCL expression to obtain a result. Any method can be defined just by changing the body expression.

Actions are not side-effect free since they change the value of an element. Actions will not have a body expression that specifies what the action does. However, new action classes that specialize the basic Action class will be defined. Their particular behaviour will be described by means of well-formedness rules.

With this approach, a new action cannot be defined by just changing the expression that defines it, but there is a set of standard basic actions on top of which new actions are constructed.

The actions package specializes the staticCore package.

3.1 Concepts

The abstract class Action specializes Classifier. Every action has a set of input parameters and can produce output values. As the order of the parameters and results is significant, these associations are ordered. An action will be executed