
ELSEVIER

Available online at www.sciencedirect.com
SCIENCE DIRECT®

Electronic Notes in Theoretical Computer Science 94 (2004) 1–5

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Preface

This volume contains papers presented at International Workshop on Meta-Models and Schemas for Reverse Engineering (ateM 2003) that was held in Victoria BC, Canada on November 13, 2003. This workshop was held as part of the 10th International Working Conference on Reverse Engineering (WCRE-03).

Motivation

The aim of this workshop was to explore the use of meta technology — that is, meta-models and meta-programming techniques — for the development of reverse engineering toolsets. In reverse engineering, meta-models, or schemas, are at the core of any supporting tool; the schemas effectively define the internal (data) structure and specify the underlying semantic model of various analysis services that the tool might provide. However, schemas exist at different levels of detail (from AST to the architectural), model different programming languages and paradigms (procedural, object-oriented), support different kinds of concerns (user interface, business logic, web services) and lifecycle stages (requirements, design, maintenance). The lack of generally accepted standards for schemas in reverse engineering has resulted in a plethora of difficult and interesting research problems.

Workshop Issues

The ateM-2003 workshop aimed to build on previous work on themes related to the issues above, such as the 2000 Workshop on a Standard Exchange Format (WoSEF), Dagstuhl Seminar #01041 on Interoperability of Re-engineering Tools, as well as recent research within the model-driven architecture (MDA) community.

The program for ateM-2003 consisted of three sessions that concentrated on paper presentations and ensuing discussions, plus a final discussion session

doi:10.1016/j.entcs.2004.01.009

that elaborated on the various themes of interest that arose during the first three sessions. The paper sessions were organized around three themes:

(i) *Meta-models:* talks by Tim Lethbridge (University of Ottawa) on *The Dagstuhl Middle Model: An Overview*, Jens Knodel (IESE, Germany) on *A Meta-Model for Fact Extraction from Delphi Source Code*, and Kenny Wong (University of Alberta) on *A GXL Metaschema for Story Diagrams*.

(ii) *Frameworks and exchange formats:* talks by Ravindra Naik (Tata Consulting, India) on *A Programmable Analysis and Transformation Framework for Reverse Engineering*, Michael Blaha (OMT Associates) on *Data Store Models are Different than Data Interchange Models*, and Abdelwahab Hamou-Lhadj on *The Compact Trace Format*.

(iii) *Meta-modelling technologies:* Anthony Cox (Dalhousie University, Canada) on *Multi-Layered Data-Modelling*, Kenny Wong on *Issues in Integrating Schemas for Reverse Engineering*, and Yuan Lin (University of Waterloo, Canada) on *Formalizing Fact Extraction*.

Each presenter was encouraged to contribute a question that they would like the community to consider. These questions were collected, and used in the design of the final discussion session.

Discussion session

The final discussion session, which was moderated by Andreas Winter (University of Koblenz, Germany), elaborated on four themes of common interest that arose during the preceding discussions: exchange formats; levels of schemas; variants of schemas; and representation and definition of schemas.

1. Exchange Formats

Questions considered:

What should an exchange format look like? How useful is GXL? Given that we have been considering this problem for a while, why has the it been so hard to exchange schemas and analysis information?

Discussion:

It was generally agreed that program analysis data is often graph-like, and that GXL is a good solution for storing XML-ised graph data. However, there were several remarks that questioned and/or criticized the use of GXL as an exchange format for schemas and analysis data. GXL was portrayed by some as being too bloated and complex for easy use (largely because of its basis in XML); GXL has been around for a few years now, but most of its use has been to exchange simple data (in a single schema) rather than

allow data in different schemas to be *exchanged*. It was suggested that the TA language or a MOF-based model might be preferable sometimes.

2. Levels of Schemas

Questions considered:

What kinds of schemas and meta-models are actually needed? What are the relationships between schemas (sub- and supersets, variants, versions)? How can tools accommodate, view, and integrate multiple schemas?

Discussion:

Tim Lethbridge (University of Ottawa, Canada) made several remarks about the model of *three plus one* schema levels that came out of Dagstuhl Seminar #01041 on Interoperability of Re-engineering Tools. The three-plus-one levels are: *architectural* (high level design components and connectors), *middle* (classes, files, functions, methods, fields, and their inter-relationships), and *code* (AST information) plus data.

Lethbridge (who previously had led the effort to develop the Dagstuhl middle-level model, or DMM, into a concrete meta-model) argued that the DMM is probably the *right* taxonomy for reverse engineering schemas, but that there are other kinds of useful schemas that do not easily fit into this model, such as requirements models (e.g., business processes), specialized analysis techniques (e.g., program dependence graphs, or PDGs, used in program slicing analysis), and visualization languages. After some discussion, it was concluded that these are more like *orthogonal dimensions* than basic design layers (abstraction levels); each of them addressed a different kind of concern about a software system and might indeed incorporate its own hierarchy of levels. The *dimension* that the workshop was intended to discuss was the *design*, or *static structural* dimension. There was some discussion over the best term for this idea: *dimension*, *domain*, *view*, *facet*, *aspect* were all suggested.

Ravinder Naik (Tata Consulting, India) was asked what kinds of models they use for requirements; he replied that his group is usually concerned with modelling business rules and business processes, but these are *just* documents rather than formal and machine processable artifacts.

3. Variants of Schemas

Questions considered:

How to deal with *slightly different* schemas (dialects)? How to deal with *core schemas* vs. *variants of core schemas*? How to create and represent schema variants?

Discussion:

A variant of a schema is one that is very close to a core schema, but differs in some important way, such as how the GCC dialect of C might differ from the ANSI standard. Andreas Winter (University of Koblenz, Germany) suggested that a core model with broad appeal can be identified, and then variants can be defined in terms of their differences to it. Several others commented that it is important to have a clear and explicit model of such variants, such as a formal model using UML's Object Constraint Language (OCL) or some other machine-checkable mechanism. Winter made the concluding remark that our community has learned the hard lesson that extraction and analysis tools need to be based on explicitly defined schemas.

4. Representation and Definition of Schemas

Questions considered:

How to represent or store schemas for multiple view use? How to validate schemas?

Discussion:

Andreas Winter (University of Koblenz, Germany) stated that our community needs a reliable, prescriptive methodology for schema definition; he wondered if the database community could help us. Michael Blaha (OMT Associates) stated that this plus schema migration or transformation are well known problems in the database community; they are well known and well studied, but there are no general purpose overriding solutions.

Winter asked about how to visualize the schemas, GXL uses a subset of UML object diagrams, stored in XMI. It's a challenging question without an obvious easy solution; complexity and detail-level are problems. Blaha strongly recommended the investigation of OCL as being possibly useful here. We said that relative to UML (*designed by committee by putting everyone's favourite ideas into one big basket*), the OCL is relatively coherent and well thought out.

Workshop Organization and Acknowledgments

The workshop was organized by Andreas Winter (University of Koblenz, Germany), Jean-Marie Favre (University of Grenoble, France), and Michael Godfrey (University of Waterloo, Canada).

We, the organizers, would like to thank the program committee who reviewed the submissions and provided useful feedback to the authors:

- Stphane Ducasse, University of Berne, Switzerland
- Jens Jahnke, University of Victoria, Canada

- Timothy C. Lethbridge, University of Ottawa, Canada
- Mark Minas, Univ. of the Federal Armed Forces, Munich, Germany
- Leon Moonen, Delft University of Technology, CWI, The Netherlands
- Juha-Pekka Tolvanen, MetaCase, Jyvaskyla, Finland
- Susan Elliott Sim, University of California, Irvine, U.S.A.
- Tarja Systa, Tampere University of Technology, Finland

We would also like to thank the organizers and program chairs of WCRE-03: Elliot Chikovsky, Peggy Storey, Ari van Deursen, Eleni Stroulia, Ladan Tahvildari, Victor Chong, and Ian Bull. We would also like to thank Michael Mislove for agreeing to publish the ateM 2003 post-proceedings in Elsevier Electronic Notes in Theoretical Computer Science.

Jean-Marie Favre
University of Grenoble, France
Mike Godfrey
University of Waterloo, Canada
Andreas Winter
University of Koblenz-Landau, Germany
January, 2004