Enhancing UML Models: A Domain Analysis Approach

Iris Reinhartz-Berger, University of Haifa, Israel
Arnon Sturm, Ben-Gurion University of the Negev, Israel

ABSTRACT

UML has been largely adopted as a standard modeling language. The emergence of UML from different modeling languages that refer to various system aspects causes a wide variety of completeness and correctness problems in UML models. Several methods have been proposed for dealing with correctness issues, mainly providing internal consistency rules but ignoring correctness and completeness with respect to the system requirements and the domain constraints. In this article, we propose addressing both completeness and correctness problems of UML models by adopting a domain analysis approach called application-based domain modeling (ADOM). We present experimental results from our study which checks the quality of application models when utilizing ADOM on UML. The results advocate that the availability of the domain model helps achieve more complete models without reducing the comprehension of these models.

Keywords: domain analysis; model completeness; model correctness; object-oriented analysis; object-oriented design; software development methodologies; UML

INTRODUCTION

Conceptual modeling is fundamental to any area where one has to cope with complex real-world systems. The most popular, de-facto modeling language today is UML, which is used for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems (OMG-UML, 2003; OMG-UML, 2006). Although UML provides convenient, standard mechanisms for software engineers to represent high-level system designs, as well as low-level implementation details (Tilley & Huang, 2003), it also introduces a variety of correctness and completeness problems.

According to Major and McGregor (1999), correctness is measured as how accurately the model represents the information specified within the requirements. For defining the correctness of a model, a source that is assumed to be (nearly) infallible is identified. This source, termed a “test oracle,” is usually a human expert whose personal knowledge is judged to be sufficiently reliable to be used as a reference. The accuracy of the model representation is measured relatively to the results expected by the oracle. Completeness, on the other hand, deals with the necessity and usefulness of the model to represent the real life application, as well as the lack of required elements within the model (Major & McGregor, 1999). In other
words, completeness is judged as to whether
the information being modeled is described
in sufficient details for the established goals.
This judgment is based on the model’s ability
to represent the required situations, as well as
on the knowledge of experts.

Different studies concluded that it is dif-
ficult to model a correct and consistent applica-
tion using UML and even to understand such
a specification (Dori, 2001; Kabeli & Shoval,
2001; Peleg & Dori, 2000; Reinhartz-Berger &
Dori, 2005; Siau & Cao; 2001). Several methods
have been suggested for checking the correct-
ness of UML models. However, these mainly
deal with syntactic issues directly derived from
the modeling language metamodel, neglecting
the correctness and completeness of the models
with respect to the domain constraints and the
system requirements.

In this research we utilize the application-
based domain modeling (ADOM) approach
(Reinhartz-Berger & Sturm, 2004; Sturm &
Reinhartz-Berger, 2004), whose roots are in
the area of domain engineering, for enhancing
UML models. ADOM enables specifying and
modeling domain artifacts that capture the
common knowledge and the allowed variability
in specific areas, guiding the development of
particular applications in the area, and validating
the correctness and completeness of applications
with respect to their relevant domains. ADOM
does these with regular application and software
engineering techniques and languages, bridging
the gap between the different abstraction levels
at which application and domain models reside
and reducing learning and training times. We
present initial results from our study which
checks the comprehension and quality of UML
models when applying ADOM.

Following the introduction we review
relevant works from related areas and briefly
introduce the ADOM approach, emphasizing
its usage for developing correct and complete
UML models. We then elaborate on the experi-
ment we conducted, its hypotheses, settings, and
results. Finally, we summarize the advantages
and limitations of the proposed approach, rais-
ing topics for future research.

LITERATURE REVIEW

Shull, Russ and Basili (2000) defined six
types of software defects that can be found in
object-oriented designs: missing information,
incorrect facts, inconsistent information, am-
biguous information, extraneous information,
and miscellaneous defects. Incorrect facts,
inconsistent information, ambiguous informa-
tion, and extraneous information refer to the
model correctness, while missing information
refers to completeness.

Several solutions have been proposed over
the years for handling these defects, mainly con-
cerning consistency and integration problems.
These solutions can be roughly divided into
translation and verification approaches. Transla-
tion approaches, such as Bowman et al. (2002),
Rasch & Wehrheim (2002), Mens, Van Der
Straeten and Simmonds (2003), Große-Rhode
(2001), and Baresi & Pezze (2001), translate
multi-view models into formal languages that
can be analyzed by model checkers. After
detecting inconsistencies or mistakes a back-
ward process should be applied, translating the
locations where the defects were found back to
the multi-view models in order to enable the
developers to fix them. Whittle (2000) surveyed
some of the attempts to formalize the semantics
of UML by applying formal methods for analyz-
ing UML models. His main conclusion was that
UML semantics is largely informal and, hence,
more effort should be directed towards making
the semantics precise. Verification approaches,
on the other hand, such as Chiorean et al. (2003),
Bodeveix et al. (2002), Engels et al. (2002),
and Nentwich et al. (2003), present testing or
validation algorithms which check inconsistenci-
es and contradictions between various views.
They require sophisticated environments which
include test drivers, interpreters, controllers,
and so on. Reinhartz-Berger (2005) suggests
a top-level approach that glues the different
UML views into one coherent system through-
out the entire development process life-cycle.
However, all these works refer to the syntax
of the models only. Moreover, none of them
deals with completeness issues and errors that
Database Pointers in Navigational and Object–Oriented Database Management Systems: A Comparison
www.igi-global.com/article/database-pointers-navigational-object-oriented/51155?camid=4v1a

Database Integrity Checking
www.igi-global.com/chapter/database-integrity-checking/7913?camid=4v1a