Component-based Methodology and Environment OMAR

Jan Kopinec¹

¹ Department of Cybernetics, Faculty of Electrical Engineering, CTU Prague,
Karlovo namesti 13, Prague 2, 121 35,
Czech Republic, Europe, phone: +420-606-600882
kopinec@ips-ag.cz

Research Area: The subject of PhD thesis is to propose the new software development methodology based on exploring of a developed universal model, which is used during all process phases. The development according to the proposed methodology is supported by related runtime environment.

Key Words: software development methodologies, conceptual modeling, impedance problem, analysis, design, component-based approaches, object visualization, Catalysis Approach, UML, RUP.

1 Problem Definition

Majority of contemporary software A/D (analysis & design) methodologies are based on UML (Unified Modeling Language) [1,2,9], one of the best known and used is RUP (Rational Unified Process) [2]. Current methodologies are capable of producing very good application design, so a very nice conceptual schema of the application is generated. From it a static skeleton of developed application can be generated into a programming language, which is selected for the implementation. Then programmers continue with the application implementation. However during the refinement processes and implementation a part of information and coupling between entities defined in conceptual schema are lost. The most difficult transition is between design and development phases [12] and the transition between analysis and design phases need not be trivial for some kinds of applications [3]. That has several reasons: one of them is that results of A/D process are only loosely coupled to subsequent application programming. The second reason is that expressive power of tools used during A/D (UML, for example) has another abstraction and qualitative level then currently used programming languages [8]. The second reason is called the impedance problem between A/D (Analysis & Design) on one side and D/D (Design & Deployment) on the other side. The proposed OMAR Methodology tries to overcome these two problems above all, which are present in the core of current software development [3,8,12].

2 State of the Art

Currently used methodologies based on UML do not integrate A/D and D/D processes so deeply [1,2,5,6,8]. UML models cannot be directly executed, especially because of its low ability to express behavioral aspects unambiguously and incomplete semantic definition for their execution. This is the motivation for endeavors to define executable UML by excessive use of OCL and its enhancing by SDL features [8]. But executable UML tends to be used as a base for code generation rather then for direct interpretation and its meta-model is rather complex. Model
Driver Architecture initiative introduced by OMG [7] proposes to develop centralized UML based models used for automatic generation of application components. Generated applications suffer from more difficult changing application itself and application data structures during runtime then applications containing meta-data. Introduction of plug and play concept requires handling meta-data in runtime as well.

There are several other approaches to integrate of a meta-model with a model in runtime, for example, Common Lisp Object System (CLOS) and Eiffel language exploring similar concepts as OCL can be mentioned, ADL (Architecture Definition Languages) approaches, which are used for definition of component assembly, and declarative programming in RAD (Rapid Application Development) oriented more towards quick development of database applications. Currently used component-based technologies EJB (Enterprise Java Beans), Windows.NET, CCM (CORBA Component Model) also define at least partially meta-models. These implementation centric meta-models are targeted to the application code mutation during runtime and they are not well suited to be used for analysis and design. They are also not supported by any integrated development methodology [6,7,11].

3 Proposed Experimental Solution

The proposed solution is based on introduction of a universal model, which is used unmodified in all phases of software development without applying any model transformations. All other methodologies are based on more or less complex transformations required during the software development process.

OMAR Methodology is based on a meta-model, called OMAR Application Schema, which enables developers to model structural as well as behavioral aspects of the developed application in one universal model [4]. This meta-model is explored in the same form during all phases of software development life cycle. The OMAR Methodology covers all steps of software development process. The unchanged Application Schema resulting from design step is used for development and deployment phases as well. This feature of OMAR Methodology requires implementation of strong runtime support to be able to process the meta-model during the application execution. The Application Schema resulting from analysis and design steps is inserted into OMAR Environment, where it is interpreted during application execution [4,5].

![Diagram](image.png)

Fig. 1: Comparing of UML- and OMAR-based approaches.

Figure 1 compares UML-based software development life cycle with OMAR-based approach. In OMAR-based development, the transition between results of A&D and D&D
phases is not subjected to code generation in forward and reengineering processes in backward
directions because the unique OMAR Application Schema model is used for all steps without
any transformations.

4 Details of OMAR Approach

The proposed solution is based on three building blocks: Design of meta-model suitable to
be used for all phases of software development process, design of methodology defining how to
develop application by exploring of proposed meta-model, and design and experimental
implementation of related runtime environment capable of interpretation of the meta-model.

4.1 OMAR Meta-Model

The requirement that the unique OMAR Application Schema has to be used in all phases
of software development life cycle requires its properties to be designed with maximal care
because very different requirements of all development phases have to be balanced inside one
commonly used meta-model. OMAR Application Schema meta-model must allow for
development of detail-rich structural description of a designed application. The structural
description can be realized by means of class diagrams [1] and there are no big obstacles why
such a class diagram cannot be used for all phases of software development process. In contrast
to structural description, a more crucial problem is to propose the suitable behavioral description
of the developed application. The Application Schema must be simpler then final
implementation or closer related to the problem domain then to the implementation domain
otherwise it brings no advantage to the development process. One reasonable solution is to
derscribe only the top-level behavior of the application in Application Schema and to let the local
behavior to be defined later during the forthcoming development. For description of behavioral
aspects OMAR meta-model offers integrated apparatus of final automatons. OMAR meta-model
consist of about 20 entities for modeling of classes, which can have several kinds of methods,
modeling of arbitrary customizable relations used for execution flow control during runtime, and
apparatus enabling to model state transitions. The algorithm how to create well-defined
Application Schemas has been designed as well as its detail definition [5].

4.2 OMAR Methodology

OMAR Methodology defines what should be exactly done in different A/D/D/D/D
steps. Phases, which are very similar in almost all currently up-to-date methodologies, e.g.
requirements assembly [2] have not been designed again from scratch. The Catalysis Approach
[9] has been selected as a base for the OMAR Methodology. Only steps, which really differ from
Catalysis Approach, have been redefined in OMAR. The Catalysis Approach has been selected
mainly because of two reasons: it is up to date methodology for component-based software
development and it is fully modular based on patterns that enable easy extensions. The core
features of OMAR Methodology are its support for very quick repeatable iterative software
development processes due to elimination of model transformations and a part of coding and its
support for component-based development in that way that groups of requirements are mapped
to components by means of Application Schemas. Partitioning of complete Application Schema
to one Application Schema Skeleton and several optional Application Schema Extensions is used
for definition of deployment components. Each component can contain also definitions of
Application Sub-Schemas.
4.3 OMAR Environment

OMAR Environment is composed from several subsystems. The central subsystem consists of Persistent Services equipped by query and schema manipulation language enabling to handle all OMAR meta-model and model entities during all software development steps and especially during intensive manipulations in runtime. Schema Engine is the interpreter of Application Schemas. It is based on several algorithms exploring different graph search techniques. Visualization Services enable automatic projection of all objects from Application Schema and its extent into GUIs. All core components of OMAR Environment can be implemented with the help of already existing component-bases environments as J2EE and/or CORBA [11]. In this sense the OMAR Environment should be considered to be the additional abstraction layer between existing techniques for conceptual modeling and currently explored component-based technologies.

5 Conclusions

5.1 Research State

Definition of universal model, which can be used during all phases of software development process, without being subjected to any transformations, seems to be pretty unique feature of the proposed approach. The methodology of using this meta-model for the development of real applications has been proposed. The prototype of OMAR Environment including simple file-based repository is being implemented in Java language to prove the concept of Application Schema interpretation.

5.2 Research Contributions

The proposed approach leads to planned goals: no information and coupling between entities is lost during the complete process and impedance problems have been significantly reduced. Other positive effect of proposed approach is the enabling of very rapid iterative development due to elimination of model transformations and especially due to reduction of required amount of work in the development phase. Full traceability among groups of requirements and deployed components has been reached, mainly because the fact that proposed meta-model is used also for definition of components and their assembly. The integration of Application Schema into runtime enables the easier introduction of application mutation and plug and play concepts.

5.3 Research Evaluation

As the main disadvantages of proposed approach can be considered the fact that proposed OMAR methodology cannot be used for development of all kinds of applications. This is caused mainly by two limitations: 1st the definition of unique meta-model, which must be used for all development phases, has only the expressive power of final automatons, 2nd there is the restriction to use the predefined application architecture given by OMAR Environment. This is a logical conclusion, because concurrent approaches based on exploring of rather different models for all steps bring higher flexibility and more freedom to design arbitrary applications. That is the reason why extensions for behavioral description of proposed meta-model should be considered, for example, in the direction towards Petri Nets apparatus. But these limitations are not hard restrictions for Information System development because final automatons can describe majority
of business logic and their implementation does not require necessity to redefine architecture. It is also more difficult to work with different levels of abstraction resp. with transfer from the problem domain to the implementation domain in OMAR because of lack of intermediate transformations. Handling of incomplete and inconsistent models is also restricted. Possible technical obstacle to use this methodology can be also the fact that large Application Schemas are difficult to be treated without some special graphical software support.

Properties of the proposed approach are best suited for development of flexible distributed workflow systems. The practical usability of proposed approach has been proved on a case study dealing with development of distributed workflow application for drawing management called CADMAN. To justify general usability of the proposed approach would require development of a comparative study of more application cases developed parallel according to different methodologies. Such a complex and credible comparison of different component-based methodologies and environments is a complex task requiring large amount of work.

References