Multiview: Requirements Modeling and Formalisation*

Jaelson F. B. Castro,
Marco A. Toranzo and
Christian J. Gautreau
Marcio A. S. Bueno

Departamento de Informática, Universidade Federal de Pernambuco
Recife, PE, CEP 50732-970, Caixa Postal 7851, Brazil
E-mail: {jbc, matc, cg}@di.ufpe.br

Keywords : Software Engineering, Requirements, Case

May 1997

Abstract

The sucess of the requirements engineering process often depends on the ability to proceed from informal, fuzzy individual statement of requirement to a formal specification that is understood and agreed by all stakeholders. *Multiview environment* supports the process of requirements modeling and formalisation.

The purpose of this paper is to provide a general overview of the implementation and current status of the *Multiview environment*.

1 Introduction

Software engineering (SE) is a very broad discipline. It is refers to an integrated set of methods, procedures, and tools for specifying, desiging, developing, and maintaining software [10]. Requirements engineering is a sub-field of Software engineering [12]. It can be defined as the systematic process of developing requirements through an iterative co-operative process of analysing the problem, documenting the resulting observations in a variety of representatation formats, and checking the accuracy of the understanding gained. The sucess of the requirements engineering process often depends on the ability to proceed from informal, fuzzy individual statement of requirement to a formal specification that is understood and agreed by all stakeholders.

We have been working on the *Multiview environment*[5, 6, 7, 8] which supports the process of requirements modelling and formalisation. *Multiview* is part of the Formlab research projetc which aims at building an integrated process centred software engineering environment to support formal methods[3].

Our aim is to be able to formalise requirements in Modal Action Logic (MAL). The logic is used to describe the behaviour of objects of the system. Objects can interact by sharing attributes or by sharing actions. An *attribute* is part of the state of the system. The actions of an object should only update attributes of that object, and vice versa.

Multiview supports the VSCS method [4] that provides an organised collection of representation schemes (Object Diagrams, Textual descriptions, Event Traces, Statecharts, Causal Diagrams, MAL, etc) that are closely related and provide guidance, integrated with a work plan, for moving between these schemes [3].

Some details of the metamodels of the representation schemes are described in [8]. The notion of incremental formalisation is stressed and we argue that most formal declarations can be written at an early stage of the method. The actions, attributes, functions, etc. produced at earlier stages are, of course, informal, but they are produced in a sufficiently constrained way that the transformation into the formal language of structured Modal Action Logic is possible.

We begin in Section 2 gives an overview of the Multiview environment. The current status of multiview is addressed in Section 3.

*This work was sponsored by CNPq/Protem II grant Proc. 680062/94-7. It is part of FORMLAB Research Project which is a collaborative work among UFPE, PUC-Rio (Prof. A. Haeberer), UEM (Prof. I. Gimenes) and Equitel.

2 An Overview of the Multiview Environment

This section presents an overview of the environment in terms of domains as well as some graphical editors.

In [7] we provided a description of the analysis and design phases of *Multiview*. It relied heavilly on object oriented techniques. The *analysis phase* used Shlaer and Mellor technique [15] to divide *Multiview* into domains, which are further refined using OMT method [13]. Several representation schemes of VSCS w ere also described in OMT. In the *design phase* of the *Multiview environment*, OMT was also use ful for refinement of the various metamodels (Object, Statecharts, MAL) supported by the environment. The Booch notation [1, 2] was very convenient for expressing the message exchange among objetcs, to model the existence of objects and classes (with their reponsabilities), and to express their dynamic and static relationships. The Fusion notation [9] was appropriate for presenting the general overview of *Multiview* system interface context. It is defined in terms of the set of operations (messages) that the interface can receive and events that it can generate.

2.1 Domains

Multiview was structured in terms of *domains* (objects with strong relationships) and client/server relationships (figure 1).

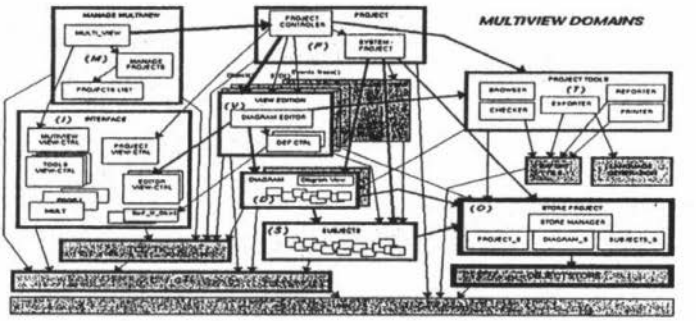

Figure 1: Multiview Domains

The top-level domain of the system is *Manage Multiview*. It is responsible for system initialisation and management of projects. This domain relies on services provided by three other domains: *Interface*, *Project* and *Store Projects*.

The *Project* domain is responsible for managing current project informations and the activation of specific editors to create/modify a diagram (in the *View Edition* domain). It can also initiate tools associated with the project (from *Project Tools* domain) and relies on services from *Interface* domain for its own communication. It uses the *Store Project* domain to retrieve/save project and data dictionary (*Project_S* and *Subjects_S* sub-domains).

The *View Edition* domain is responsible for the construction of each specification ViewPoint. It uses services from the *Diagram* domain to structure the representations and record associated in formations, as well as, services from *Store Project* to retrieve/save diagram informations (*Diagrams_S*). It also can activate tools in the *Project Tools* domain. The *Diagram* domain contains the structure of the graph of the current diagram as well as the visual representation of each diagram's element. For each representation, an associated dictionary element is maintained (in *Subjects* domain).

2.2 User interface

In Figure 2, the menus represent services and requirements associated with the main window of the *Multiview Environment*. An user can define projects, edit and create its associated diagrams. In this particular case,

the analyst 'Marco Toranzo' is creating a project called 'Demo' (version '1.0') using the method 'VSCS'. According to the VSCS method, several descriptions will be associated with this project (Object ViewPoint, Text ViewPoint, Statechart ViewPoint, etc). In the example, a new diagram (of type Object) called 'Car-System' is defined.

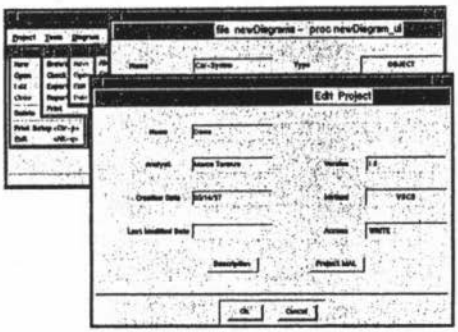

Figure 2: Defining Project and Model

2.3 Object Model Editor

The interface of the Object Model Editor can be find in Figure 3. A stakeholder can define classes and relationships (association, inheritance, aggregation, client-server) to model the static view of the problem domain. Classes contain operations and attributes.

In the example, 'Car' is composed of four components 'Brake', 'Accelerator', 'Ignition' and 'Transmission'. The component 'Brake' has an attribute ('brake-pos') and two operations ('depress-brake' and 'release-brake'). The attribute 'brake-pos' is of type 'On/Off'.

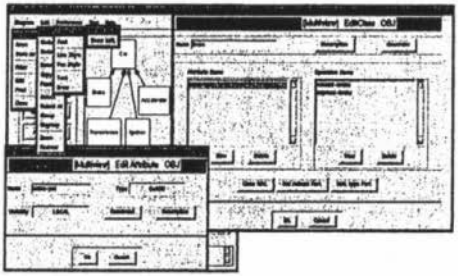

Figure 3: Object Model Editor

2.4 Statechart Model Editor

In Figure 4 the interface of the Statechart Model Graphical Editor is presented. States (normal, initial and final) and transitions can be created. Optional services, similar to the ones available in Object Model Graphical Editor (previous section), are also available. The state 'off' and transition 'release-brake' are defined. The behaviour of the 'Brake' component is described as a sequence of transitions ('depress-brake', 'release-break') that leads the system from the 'off' to the 'on' state. The formal

specification, in MAL [14, 11], of the 'Brake' component is also provided.

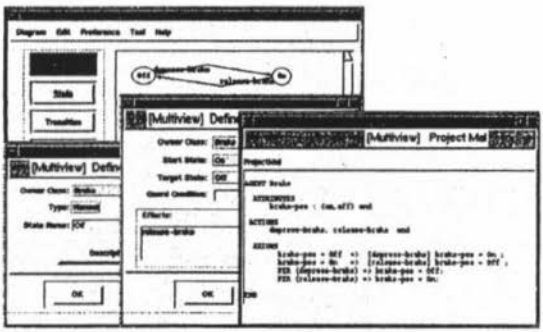

Figure 4: Statechart Editor

3 Current Status and Conclusions

Multiview currently supports two graphical editors and MAL specification generator. Other editors (Event life-cycle Editor, Casual Editor) are under development. Also, a MAL parser and animator will be coupled into system.

Our system is implemented with the language Tcl/Tk and OODBMS ObjectStore C++[7]. Tcl/Tk was used to implement the user interfaces. ObjectStore C++ stores and manages the users's definitions (class, relationship, state).

In this paper we presented an overview of the Multiview Environment, including its domains, user interface and graphical tools.

Multiview is part of a more ambitious *Formlab* project which is building an integrated process centered software engineering environment to support formal methods. Hence, work is under progress to define suitable integration mechanisms.

Acknowledgment We are particularly indebted to Marcelo P. Barbosa, Pedro J. R. Saunders, and Gilberto A. C. Filho.

References

[1] G. Booch. *Object-Oriented Analysis and Design With Applications.* The Benjamin/Cunnings Publishing Company, Inc., Redwood City, CA (USA), 1994.

[2] G. Booch and J. Rumbaugh. *Unified Method For Object Oriented Development - Documentation Set, Version 0.9.* Rational Software Coorporation, 1996.

[3] J. Castro. The process of requirements formalisation : The formlab project. In *Proceedings of Information Systems Analysis and Synthesis - ISAS'95, Focus Symposium, 5th International Symposium on SYstems Research, Informatics and Cybernetics*, pages 01–05, August 1995. Baden-Baden, Germany.

[4] J. Castro and A. Finkelstein. VSCS: An Object Oriented Method for Requirements Elicitation and Formalisation. Technical Report Deliverable NFR/WP2.2/IC/R/002/A, Imperial College, Department of Computing, Formal Requirement Specification TECHNIQUEs (FOREST) Research Project, October 1991. 107 pages.

[5] J. Castro, C. Gautreau, and M. Toranzo. Multiview: An environment for requirements elicitation and formalisation. In *Proceedings of International Conference on Information Systems Analysis and Synthesis - ISAS'96*, pages 161–168, July 1996. Orlando, USA.

[6] J. Castro, C. Gautreau, and M. Toranzo. Multiview: An integrated environment to support requirements elicitation and formalisation. In *Proceedings of XVI Congress of the Brazilian Computing Society, XXIII Seminar on Integrated Hardware and Software (SEMISH'96)*, pages 445–456, August 1996. Recife, Brazil.

[7] J. Castro, C. Gautreau, and M. Toranzo. Tool suport for requirements formalisation. In *Proceedings of the ACM SIGSOFT Viewpoint 96: International Workshop on Multiple Perspective Software Developments*, pages 202–206, October 1996. San Francisco, USA.

[8] J. Castro, C. Gautreau, and M. Toranzo. Towards an environment to support requirements formalisation. In *Proceedings of X Brazilian Symposium on Software Engineering*, pages 189–206, October 1996. São Carlos, Brazil.

[9] D. Coleman, P. Arnold, S. Bodoff, and C. Dollin. *Object-Oriented Development: The FUSION Method*. Prentice-Hall, 1994.

[10] T. Flecher and J. Hunt. *Software Enginnering and CASE*. McGraw-Hill, Inc, 1993.

[11] S. Kent, T. Maibaum, and W. Quirk. Formally specifying temporal constraints and error recovery. In *Proceedings of IEEE International Symposium on Requirements Engineering - RE93*, pages 208–215, January 1993.

[12] P. Loucopoulos and V. Karakostas. *System Requirements Engineering*. McGraw-Hill, Inc, 1995.

[13] J. Rumbaugh, M. Blaha, F. Eddy, and W. Lorensen. *Object-Oriented Modeling and Design*. Prentice-Hall International, Englewood Cliffs, NJ(USA), 1991.

[14] M. D. Ryan, J. Fiadeiro, and T. Maibaum. Sharing actions and attributes in modal action logic. In T. Ito and A. Meyer, editors, *Theoretical Aspects of Computer Software*. Springer Verlag, 1991.

[15] S. Shlaer and S. J. Mellor. *Object-Oriented Systems Analysis: Modeling the World in data*. Yourdon Press, Englewood Cliffs, N.J., 1988.