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 7881, Brasil
E-mail: {jbc, matc, cg}@di.ufpe.br
Keywords: Software Engineering, Requirements, Case

May 1997

Abstract

The success 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, designing, 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 representation formats, and checking the accuracy of the understanding gained. The success 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 project which aims at building an integrated process centered 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. Haebeler), 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 heavily 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 were also described in OMT. In the design phase of the Multiview environment, OMT was also used 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 objects, to model the existence of objects and classes (with their responsibilities), 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).

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 View Point, Text View Point, Statechart View Point, etc). In the example, a new diagram (of type Object) called 'Car-System' is defined.

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'.

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](image)

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.

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

References


