The Macrotec Toolset for CASE-based Business Modelling

Rudolf K. Keller* Richard Lajoie** Marianne Ozkan* Fayez Saba*
Xijin Shen** Tao Tao** Gregor v. Bochmann***

* Centre de recherche informatique de Montréal (CRIM)
1801, McGill College, Suite 800, Montréal, PQ H3A 2N4, Canada, keller@crim.ca
** McGill University, Montréal, *** Université de Montréal

In Proceedings of the 6th International Workshop on Computer-Aided Software Engineering, pages 114-118, Singapore, July 1993.

Abstract

We have developed a new methodology for the architectural modelling and high-level requirements specification of business processes and information systems. To support and validate our methodology, we have engineered the Macrotec toolset. Macrotec currently allows for graphical model specification, automatic graphical layout, logical and performance analysis, and hierarchical decomposition. To support this functionality, various tools were built or integrated into Macrotec. Externally, integration is achieved through a seamless user interface, and internally, integration is furthered by one single data representation scheme and a simple yet effective and extensible mechanism for tool interaction. In this paper, we present our methodology and focus on the CASE tool development effort that led to Macrotec. Specifically, we discuss Macrotec's requirements, design and implementation, and we evaluate our development effort.

Keywords: Requirements Engineering, Business Modelling, Information System, CASE Tool.

1 Introduction

In a joint research project, we have developed, in cooperation with DMR Group Inc., a new methodology for business modelling. Our approach combines several concepts that have originally been developed in separate contexts, such as entity-relationship modelling of information, specialization and inheritance in the sense of object-oriented languages, event analysis, and analysis of data (product) flow as well as resource utilization. We have integrated these concepts into a uniform modelling framework with a precise semantics for the dynamic aspects, which has been defined through the formalism of Petri nets.

The resulting modelling approach [2] supports facilities such as architectural views at different levels of abstraction, and performance analysis, based on the dynamic semantics mentioned above. In many ways, however, it goes beyond current approaches. For instance, it explicitly supports inheritance and specialization, and it includes an expressive set of relationships, yet small enough to make them easy to use. Moreover, our approach lends itself to automation, e.g., semi-automatic substitution of model parts, various consistency checks, and flexible animation (forward and backward).

In order to support and validate our approach, we have developed the *Macrotec*[1] toolset, a CASE tool which will eventually support all facets of our methodology. In this paper, we discuss its distinguishing features, the major stages in its development and validation, and our experience which seems to be typical for CASE tool development efforts in research environments.

First, we discuss the requirements for systems supporting our methodology. Then, we detail the design considerations behind Macrotec and provide a scenario of its usage. Next, the implementation of Macrotec is described, together with an evaluation of our development effort. Finally, we present the current status and future research directions of our project.

2 Toolset Requirements and Existing Tools

In this section, we describe the functional requirements of the Macrotec toolset. We then report on our evaluation of existing tools against these requirements and detail the tools we have integrated into Macrotec.

The toolset must include:

- A tool for the graphical editing and validation of models (i.e., their representation as annotated graphs or networks), complemented with facili-

This research is part supported by research grants from NSERC. Canada and by Macroscope Information INC. (project managed by DMR Group Inc., Montréal).

1. ''Macrotec'' is a contraction of the words *Macro*scope, our project's name, and archi*tec*ture.

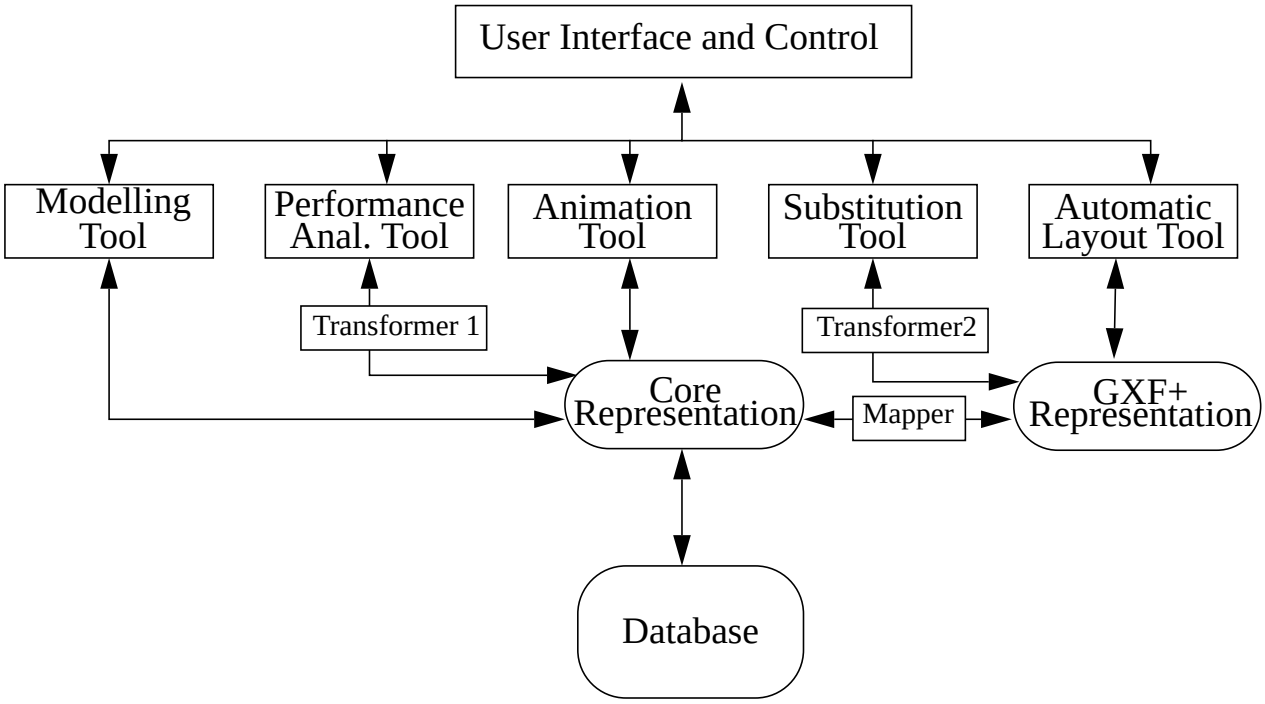

Figure 1 :Macrotec Toolset: Overall architecture

ties for automatic graph layout.

- A dynamic analysis tool supporting both timed animation and performance analysis. The animation engine (we use the terms ‘‘engine’’ and ‘‘tool’’ interchangeably) must graphically and interactively simulate the execution of the model, thus enabling visualization of the model components’ interactions. Furthermore, it must support forward and backward execution in order to answer questions such as: What are the actions that consume given inputs and in what order do they occur? What are the outputs obtained? What are the necessary intermediate actions to be performed? What inputs are required for a specified output? The performance analysis engine must analytically generate quantitative results such as action throughput, bottlenecks, resource utilization and waiting time for actions. This way, potential problems (for example, loops) may be revealed, and the impact of modifications on the system design may be more easily understood.
- A substitution tool supporting multi-level modelling. This tool must provide mechanisms for the decomposition and abstraction of network parts into sub-nets and super-nets, describing, respectively, lower and higher levels of abstraction. Obviously, these mechanisms should preserve visual and behavioral consistency between different-level nets in respect to their adjacent nodes.
- Facilities for data exchange and evolutionary design, namely, a state-of-the-art database, and a standard data exchange format.

Given this extensive list of requirements, we strove for using existing tools and thus conducted an evaluation. In what follows, we describe our principal findings.

Most existing modelling tools are based on one of the following models: data-flow, entity-relationship, object-oriented or Petri net model. Our methodology does not exclusively support these underlying models but rather merges their main concepts into one coherent approach.

The dynamic analysis and substitution tools we evaluated include *DesignCPN*, *Eval*, *GSPN*, *MetaDesign*, *RDD100, SPNP, Voltaire* [6] and Franck’s system [3]. None of them did meet the above requirements, providing only insufficient support for one or many of the following criteria: timed dynamic analysis, backward animation, automatic performance analysis results generation, integrated view of static and dynamic modelling, concurrency

and parallelism, data exchange and model evolution, hierarchical models, and finally multi-level validation.

These tools offer, at best, partial solutions to our requirements. Thus, we decided to develop an integrated toolset based on existing tools and on tools built from scratch. Figure 1 depicts the different tools involved in the resulting Macrotec toolset. We have integrated the *SPNP* performance analysis tool [10] and an automatic graphic layout package being developed at the University of Toronto [7]. The modelling, animation and substitution tools [4] were all developed by our group.

3 Design of the Macrotec Toolset

The design of the Macrotec toolset was driven by three major considerations: internal and external integration, and extensibility. We felt that these guiding principles would be essential to our design, if Macrotec was to support all of the functionality mentioned in the previous section, and possibly more in the future.

Internally, the *core representation* is the heart of Macrotec (see figure 1). All information to and from the user interface is, after manipulation by the various tools, managed in the core representation. Internal integration in Macrotec is furthered by the underlying storage facility, the Gemstone system [2], an object-oriented database allowing the storage and retrieval of the core representation, and supporting simple versioning.

In Macrotec, there are two categories of tools. The first category consists of tools that manipulate graph layout data. Such tools store their data in the *GXF+ representation*. GXF+ is our customized version of *GXF*, a standardized graph exchange format [7]. Supporting GXF+/GXF allows us to easily exchange data with other, special-purpose, GXF-based systems such as the automatic layout tools being developed at the University of Toronto. Non-GXF+-based systems require data transformation programs. For instance, integrating our substitution tool (implemented before adopting the GXF+ standard) required the development of the *Transformer2* program.

Tools belonging to the second category manipulate the model data unrelated to the graph representation. In case these tools are part of the Macrotec process, e.g., the animation tool, they interact with the core directly. Otherwise, a data transfer program to and from the core is required. For instance, the performance analysis tool, running as a separate process, interacts with the main Macrotec process via *Transformer1*. Mapping of the core into the GXF+ representation and vice versa is carried out by the *Mapper* component.

2. Gemstone is a registered trademark of Servio Corporation.

By external integration we mean that the user's interactions with all the tools and facilities of Macrotec are as uniform and comfortable as possible. This is achieved by having the user interact with the system via one single base window, giving him or her access to the complete functionality of the system and allowing for easy switching between the different tools. User interface prototyping and software reuse at the design level were instrumental in accomplishing this type of integration.

Extensibility in Macrotec is furthered by a loosely coupled yet efficient architecture. Since the different tools, while running in parallel, do not interact with each other, Macrotec's control component can be kept quite simple. It synchronizes tools, transmits external events to the tools, and controls access to the core representation, using the inter-process communication mechanisms provided by Unix. In terms of Meyer's categorization [8], our approach to integration and extensibility is a blend of the shared file system and the simple database approach.

4 Using Macrotec

One typical application domain of Macrotec is the modelling of enterprise information systems for prescriptive usage. In this section, we provide a scenario of this kind of modelling. We modelled, as an example, a delivery system with products being ordered, transported and finally delivered to their destination.

Figure 2 shows the base window through which the user has access to the full functionality of Macrotec. The window consists of three distinct areas: the menu bar which gives access to the animation, performance analysis, substitution and automatic layout tools; the palette in which the user may select an icon for editing actions, places, relations and their attribute values (our models' building blocks); and the drawing area in which the user may display and edit the network. In figure 2, the user has loaded a network, possibly reusing a template or parts of an existing model. The user may edit and refine the network, adopting a top-down or bottom-up approach by respectively decomposing or abstracting parts of the network. The built-in validation component of the modelling tool makes sure that the different levels of the network are consistent.

The animation engine is triggered through the base window via the pull-down menu shown in figure 2. The graphical execution of the model may be performed at user-specified hierarchical levels, allowing for partial model assessments and permitting the user to define a configuration that corresponds to his or her mental model of the system.

Performance analysis generates quantitative results on model behavior. For example, a comparatively low

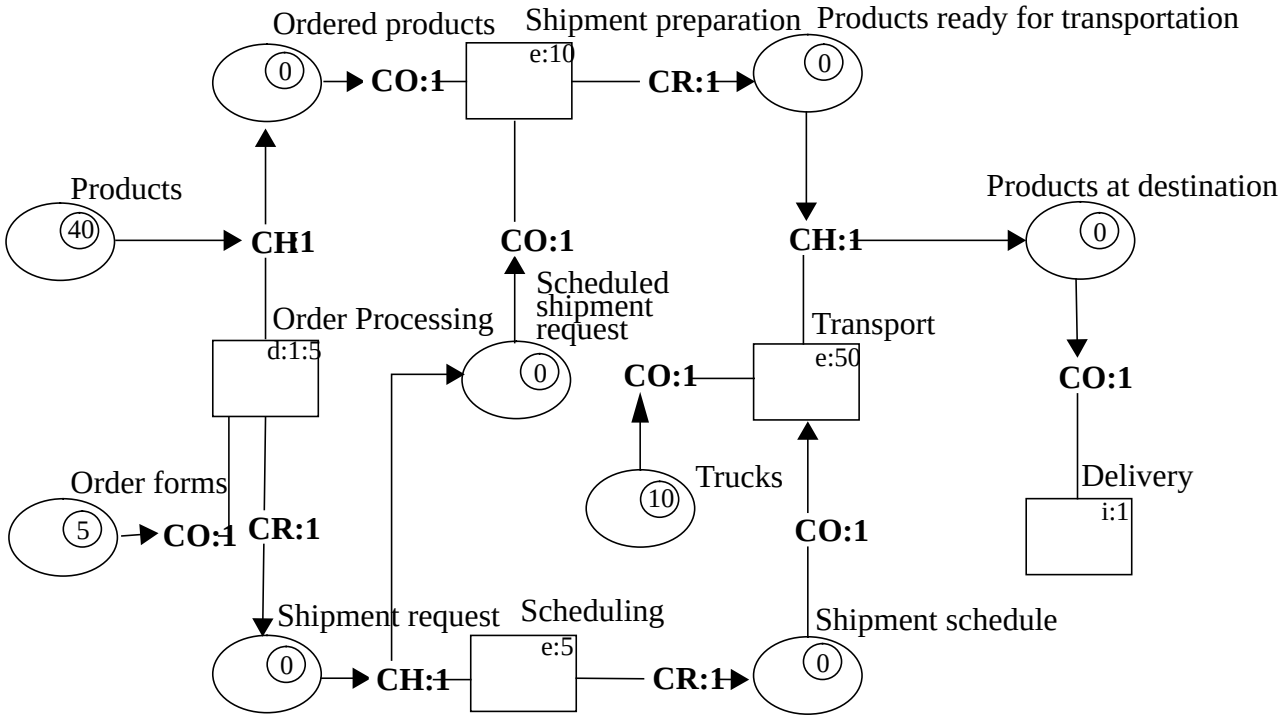

Figure 2 :Sample model *Delivery System* with animation and performance analysis attributes

action throughput and average number of entities in an action's input place (''low'' might have a different significance depending on the network architecture) may confirm a bottleneck that has already become apparent during animation. In light of these results, the user may increase the number of over-strained resources to smoothen execution (in our example, we could increase the number of ''Trucks'', if the action ''Transport'' had low throughput). Similar to methodologies and tools in related domains [9], the user may prototype the model. He or she might run, in an iterative way, animation and performance analysis, until the appropriate attribute values and configuration for a desired model behavior are found.

5 Implementation and Evaluation of Development Effort

Macrotec is a Unix-based system developed mostly in C++ (minor parts were written in C), running under SunWindows, NeWS, and the X11 window system. Its user interface has been implemented with the *ET++* application framework [11]; as database we are using Gemstone. To date, we have invested more than four person years in its development.

The data transformation programs required to integrate external tools into Macrotec are an indicator for the extensibility of the system. According to our experience, transformation programs dealing with the GXF+ representation tend to be considerably longer (4:1 ratio in lines of code) and more complicated than the ones interacting with the core representation. However, this will not be a severe limitation, since future Macrotec extensions will most likely require a transformation of type Transformer1. We do not see the need for further graph manipulation tools, and hence transformations of type Transformer2 will not be required.

We have adopted an object-oriented development approach that has provided significant productivity and quality gains. For instance, our user interface software is highly flexible and reusable, in part due to the use of ET++. ET++ is a powerful, object-oriented class library integrating user interface building blocks with high-level application framework components. The usually steep learning curve for such application frameworks has been alleviated by the use of some powerful C++ development tools, most notably, the *Sniff* tool [1]. Another benefit of the object-oriented approach in Macrotec is the use of Gemstone together with its C++ interface which has led to an efficient database interface.

To prototype the user interface of Macrotec, we ran several user interface development cycles, using tools such as *HyperCard* and *Chiron-1* [5]. We tried to integrate into the Macrotec user interface the ‘‘best’’ user interface features we found in other advanced tools (cf.[6]).

As our system evolves, with new external tools requiring integration, we will be in a better position to determine how easily our system can be adapted and hence whether or not our design and guiding principles are indeed sound.

6 Status and Future Work

A prototype version of the Macrotec toolset has recently been completed. Preliminary experience indicates that the prototype efficiently and effectively supports our methodology. In the current version, backward animation and model substitution are not yet fully supported. We intend to further validate our methodology and toolset by applying it to large examples and by using it with evolving systems. Further plans include customizability at the presentation level, the addition of a model documentation and elicitation component, and navigation aids for substitution hierarchies.

Acknowledgment: We appreciate the contributions and suggestions provided by our other colleagues in the Macroscope project.

References

[1] Walter R. Bischofberger. Sniff - a pragmatic approach to a C++ programming environment. In *Usenix C++ Conference*, Portland, OR, August 1992.

[2] G. v. Bochmann, A. Debaque, R. Dssouli, A. Jaoua, R. Keller, N. Rico, and F. Saba. A method for architectural modelling and dynamic analysis of information systems and business processes. Technical Report CRIM-92/12/10, Centre de recherche informatique de Montréal (CRIM), Montreal, December 1992.

[3] Ulrich Frank. Designing procedures within an object-oriented enterprise model. In *Proceedings of the Third International Working Conference on Dynamic Modelling of Information Systems*, pages 365–385, Noordwijkerhout, The Netherlands, June 1992.

[4] A. Jaoua, J.M. Beaulieu, N. Belkiter, A.-C. Debaque, J. Desharnais, R. Lelouche, T. Monkam, and M. Reging. Rectangular decomposition of object-oriented software architectures. Technical report, Laval University, Québec, June 1992.

[5] Rudolf K. Keller, Mary Cameron, Richard N. Taylor, and Dennis B. Troup. User interface development and software environments: The Chiron-1 system. In *Proceedings of the Thirteenth International Conference on Software Engineering*, pages 208–218, Austin, TX, May 1991.

[6] Rudolf K. Keller, Marianne Ozkan, and Nathalie Rico. Comparaison fonctionelle des outils de simulation. Technical Report Tig-92-4, Centre de recherche informatique de Montréal (CRIM), Montreal, July 1992.

[7] Alberto O. Mendelzon, Frank Ch. Eigler, and Emmanuel G. Noik. GXF: A graph exchange format. Technical report, University of Toronto, Toronto, July 1992.

[8] Scott Meyers. Difficulties in integrating multiview development systems. *IEEE Software*, 8(1):49–57, January 1991.

[9] Bran Selic, Garth Gullekson, Jim McGee, and Ian Engelberg. ROOM: An object-oriented methodology for developing real-time systems. In *Proceedings of the Fifth International Workshop on Computer-Aided Software Engineering*, pages 230–240, Montreal, Canada, July 1992.

[10] K. S. Trivedi, J. K. Muppula, S. P. Woolet, and B.R. Haverkort. Composite performance and dependability analysis. *Performance Evaluation*, 14(3-1):197–215, 1992.

[11] Andre Weinand, Erich Gamma, and Rudolf Marty. Design and implementation of ET++, a seamless object-oriented application framework. *Structured Programming*, 10(2):63–87, 1989.