REAL-IT: MODEL-BASED INTERFACE DEVELOPMENT ENVIRONMENT[1]

Alexander N. Ivanov and Dmitrij V. Koznov
Saint-Petersburg State University

ABSTRACT

We present REAL-IT – model-based interface development environment with powerful support of source code generation from declarative interface models. REAL-IT provides some special modeling technique based on UML collaboration diagrams to specify data dependencies. This information is actively used for generation of advanced user interface features. REAL-IT supports iterative development process and provides guarantied consistency between different models and also between models and source code. REAL-IT development environment is integrated with UML CASE-tools (IBM Rational Rose and REAL) and development environment Microsoft Visual Studio/Visual Basic. It provides code generation for MS Visual Basic/Java and MS Access/MS SQL Server/Oracle. REAL-IT is intensively used in industry, and we analyze two industrial projects where it was successfully used.

INTRODUCTION

The model-based user interface development environment (MBUIDE) aims to provide a context in which developers can design and implement user interfaces (UIs) constructing declarative models (ref. 1). These models give the following advantages:

- They can provide more high-level description of the UI than UI descriptions provided by the other UI development tools (ref.2).
- They support the infrastructure for automating the processes of design and implementation of the UI (ref. 3).
- They facilitate the creation of methods to design and implement the UI in a systematic way (ref.1).

There are numerous MBUIDEs, for example, GENIUS (ref. 4), TEALLACH (ref. 5), MASTERMIND (ref.3)[2]. Model-based user interface development facilities are embedded into various CASE-tools, for example, ERwin[3]. Data Base Management Systems (DBMSs) – MS Access, Oracle, etc. – also have a partial support for model-based development of the UI. The model-based user interface development approach is adopted for Web-applications (ref. 7, 8).

However, MBUIDEs are not widely used since some essential problems that are related to this approach are not completely solved (ref. 1):

- There is no consensus about the set of models that is the most suitable for describing user interfaces.
- It is not easy to integrate generated UI code together with other software components.
- It is hard to estimate the quality of UI models without demonstrating the execution of UI code.

In this paper we present REAL-IT, which is a MBUIDE with powerful support of source code generation from declarative interface models. REAL-IT provides some special modeling technique, which is based on UML collaboration diagrams. The diagrams are used to specify data dependencies for future generation of advanced UI features. REAL-IT supports iterative development process and provides consistency between different models and between models and source code. REAL-IT toolset is integrated together with UML CASE-tools (IBM Rational Rose and REAL (ref. 9)) and Microsoft Visual Studio/Visual Basic. It provides code generation for MS Visual Basic/Java and MS Access/MS SQL Server/Oracle. We discuss also perspectives of applying REAL-IT for Web-applications development. Lastly, we analyze two industrial projects where REAL-IT was successfully used.

---

[1] This work is partially supported by RFBR grant 05-0951/07.
[2] Detailed surveys on MBUIDEs can be found in references 1 and 6.
[3] AllFusion® ERwin Data Modeler 4.1.1, http://www3.ca.com/

REAL LIFE CHALLENGES FOR MBUIDEs

In this section we highlight the technologic challenges for MBUIDEs. They are the following:

- Support of the UI iterative development process. The development of the UI is an iterative process. It means that the models should be intensively refined and MBUIDE must support their consistency through the whole refinement process. Moreover if MBUIDE provides code generation it should also provide the support of consistency for models and source code. This is because, first, models can be changed after generation has been performed and we have to regenerate the UI. Second, the generated code can be modified manually and since these modifications are not described in the models they are obviously not regenerated in the new version of the UI. This leads to the classical roundtrip problem as described in reference 10. If the model-based approach does not solve this problem it is hard to apply such approach in real practice (ref. 11).
- Balance between poor/rich models and simple/complex development process. There are two wide MBUIDEs groups. MBUIDEs of the first group implement simple process development but provide poor modeling facilities. The UI that has been modeled in these environments can be used only as a prototype. Examples of the first group of MBUIDEs are GENIUS (ref. 4), MS Access and ERwin. MBUIDEs of the second group provide rich modeling facilities but their development processes are extremely complicated. The cost of constructing model descriptions can be higher than the benefits provided by such environments. Examples of the second group of MBUIDEs are TEALLACH (ref. 5) and MASTERMIND (ref. 3).
- Integration between modeling toolset and development environment. It is not easy to split real software development process into design and implementation, as well as into designs of the UI and other software components (database, business logic, etc.). All process activities are tightly integrated together because of the iterative nature of software development process. So it is not effective isolating one of the process activity (in this case, the development of the UI) providing it with any standalone tool. For UI modeling tools it is necessary to be deeply integrated together with the development environment.

Many researches ignore the above mentioned aspects and concentrate their attention only on models (structure, notation, etc.) and modeling techniques.

REAL-IT OVERVIEW

Today there is no general software development process (ref. 12). Therefore, if we want to shift from general principles of the software development to the real approaches and toolsets we have to introduce some constraints and restrictions for our development process as well as the target software.

REAL-IT is oriented on the development of data intensive systems with database business logic – such operations as input, modify, browse of data, etc. These systems should not support complicated business functionality. The structure of the system's user actions is typical and very simple. Therefore, the corresponding part of the UI has also a typical structure.

There is a large number of industrial applications that are such systems or have corresponding subsystems. The size of code for such applications is significantly large and their development process is an unwanted routine.

We introduce the following types of windows for data intensive systems:

- Card. It is for editing information about one database object. Figure 1 (a) shows an example for this window type.

- Data browser. It is for browsing database objects. Figure 2 (c) is an example of this window type. Data browser can contain a number of different browse filters that can depend on each other.
- Relation. It is for linking database objects, which are instances of two classes connected by n:n association[4]. An example is shown in Figure 1 (b): Classes District and Street connected by n:n association, i.e. one district may be linked with many streets and one streets may be linked with many districts. If we select one district from the combo box named "District" we will see all the available streets in the left list (streets that can be linked with the district). In the right list we will see those streets that have been already linked to this district. We are able to link or unlink streets selected for this district.

Windows of these types can be combined together with a different manner: card can be invoked from the data browser and can contain other data browsers and relation windows, etc.

For these types of windows REAL-IT provides a mature UI development framework: model-based design of the user interface, generation of source code, supporting iterative development process, toolset integration with other development tools.

We can summarize three dimensions of REAL-IT following reference 1:

- Declarative models that are used for development of the UI;
- Development process of the UI;
- Environment that supports the development process.

DECLARATIVE MODELS

In terms of reference 6 REAL-IT supports domain and dialog models. We don't support user model and task model because we want to reduce the problem of model consistency. REAL-IT does not have separate presentation model because MS Visual Studio, Delphi, Power Builder and other software environments have it. We think MBUIDE has to integrate with such software environments but not duplicate them. REAL-IT is integrated with MS Developer Studio/Visual Basic.

Domain model

We use UML for creating domain model, i.e. class diagrams to model data base schema, and collaboration diagrams to capture data dependencies. We apply UML class diagrams for modeling rational database schema. Classes denote tables, attributes are columns, associations express foreign keys, and objects are table records. We will name classes and objects in the context of data modeling as "data class" and "data object".

There are a lot of dependencies in data that can not be specified in terms of class diagrams. But these dependencies are very impotent because they capture some essential properties of UI, for example, connected combo box filters in data browsers, connected combo boxes in card windows for restriction of user inputs and so on. Collaboration diagrams are very useful to specify the data dependencies.

Figure 2 (a) shows a fragment of a database schema of a business company. But this fragment does not answer to the following question: is it possible for employee to be connected with computer of another department or not? Figure 2 (b) shows a collaboration diagram, which is expressed the negative answer to this question. For Employee-card two related combo boxes will be generated to specify department and computer. Computers will be presented for selection depending on department.

REAL-IT provides the ability for more complicated modeling of data dependencies as it is presented in reference 13.

[4] Actually REAL-IT is able to work not only with binary n:n associations but also with k-ary ones where $k > 2$.

Dialog model

This model provides a conceptual description of the structure and behavior of the visual parts of the UI in terms of abstract objects while omitting the visual details (ref. 1).

The structure of each window designed in REAL-IT is typical. The developer is able to do one of the following:

- In case if window is Card, developer selects data class which properties (attributes and associates) she/he want to place on the window. Class attributes selected will be presented as input controls, its associates selected will be presented as list/combo box controls. The developer creates property pages and arranges window elements on them.
- In case if window is Data browser, developer selects data class or view[5], which objects will be presented on the window, creates and tunes browse filters.
- In case if window is Relations, developer selects two data classes, which are connected by n:n association; defines one of them to be presented as combo box; creates and tunes browse filters.

The developer can also connect designed windows with each other.

The dialog model is saved in terms of UML class diagrams, and developer can analyze these diagrams in CASE-tool. Actually, the developer very rear works with these UML diagrams. The main aim of UML representation of dialog model is to provide the storage for all REAL-IT models in one UML-repository. It simplifies the support of model consistency.

PROCESS

In this section we present the user interface development process that is supported by REAL-IT. We briefly describe the baseline of the process emphasizing on the code generation and iterative development additionally provided by REAL-IT.

Baseline

REAL-IT user interface development process starts from creating a domain model. The developer models a database schema by means of class diagrams and specifies data dependencies using UML collaboration diagrams. The domain model is developed using UML CASE-tool.

The next step is creating a dialog model. It should be stressed that the firstly created domain model serves as a foundation for building the dialog model. During the development of the dialog model the developer creates views, elaborates windows for separated data-classes and views, and links these windows with each other. He/she does this work using a special dialog wizard (not UML CASE-tool) provided by REAL-IT. From this model REAL-IT generates UML class diagrams and system source code. Generated code is later refined and integrated with other system components. This can be performed using any conventional and commonly used development environment such as MS Visual Studio, IntelliJ IDEA, etc. The next step is linking system code together with REAL-IT runtime library resulting in the target application. Figure 3 illustrates the user interface development process with REAL-IT.

Code generation

Beside the ability to generate the programming code for the user interface, REAL-IT can also produce a database schema (SQL/DDL), code for the connection of the database and the UI, and the main application window.

[5] Besides data classes we used SQL-views to as a skeleton for data browser windows.

REAL-IT bases its source code generation on a set of manually predefined templates. Such templates can be easily modified in order to meet the needs of a particular software project.

Supporting iterative development process

Because REAL-IT has only a few types of UML diagrams and stores all models in a one repository it can easily guarantee the model consistency. However, the synchronization between models and system code is more complicated.

It is impossible that a MBUIDE addresses all numerous specific features for every software project. These features should be implemented manually, in particular, by means of making modifications of generated code. REAL-IT allows save such modifications during regeneration process by providing support for the following roundtrip approaches:

- Inheritance with polymorphism. Code modifications are specified as subclasses and original classes containing generated code become superclasses. The subclasses redefine properties of superclasses using polymorphism. During the code regeneration REAL-IT replaces only superclasses, and subclasses remain unchanged that insures safety of previously made code modifications. The approach is quite suitable for Java since this language has a built in inheritance. However, if we apply this approach to Visual Basic (VB) we have to emulate inheritance. Moreover, the user interface controls in VB are not represented as VB-code but specified in some special format. Therefore, a totally different mechanism needs to support manual changes of controls' properties.
- Supporting mechanism of window controls' changes. The manual changes of a window control properties are possible to localize automatically in the generated code. In REAL-IT a special algorithm analyzes the previous version of the generated code and insures the presence of the found manual changes in the newly regenerated code.
- Registration modifications of the user in the log. When the developer modifies generated code than all his actions are saved in a special log. After regeneration these actions are automatically reproduced for the regenerated code.

The above described mechanisms are not fully automated. The developer still has to do some additional work. For example he/she provides for REAL-IT toolset an extra data that can not be extracted automatically. It should be noted that the above described approaches do not work when the volume of manual code modifications is comparable with the volume of the automatically generated code. Otherwise, synchronization algorithms may work incorrectly or add a great computational overhead to the synchronization.

ENVIRONMENT

REAL-IT is integrated with UML CASE-tool (IBM Rational Rose, REAL (ref. 9)) and MS Visual Studio/Visual Basic, as Figure 4 shows.

REAL-IT toolset is deeply integrated with MS Visual Studio/VB because this development environment has a special format for storing information related to the UI. MS Visual Studio/VB provides a special program interface to support the integration with user applications. On the contrary, when we use Java environments all we need to support integration is to generate a plain Java-code. For REAL-IT/Java we use IntelliJ IDEA. However, it is possible to use any other Java development environment.

INDUSTRIAL APPLICATIONS

REAL-IT was successfully used in the development of two industrial systems.

The first one was the information system STUDENT designed to support a faculty level business processes of Saint-Petersburg State University. This project was developed by Information Institute of Saint-Petersburg State University. The system was developed on the base of MS VB/Access/MS SQL Server by 10 employees during 3 years.

The second system was the pilot application of a government system for network management of telecommunication equipment. This project was developed by LANIT-TERCOM Ltd. (Russia) on the base of Java/CORBA/Oracle by 5 employees during 1.5 years.

Results of applying REAL-IT in these projects are presented in tables 1 and 2. The rows of these tables indicate different window types of the UI, the columns specify the manner in which the code for these widows was developed. Captions of the tables can be clarified as follows:

- "Fully generated" means that windows were fully generated by REAL-IT.
- "Manually modified" means that windows were generated by REAL-IT, changed manually and supported by roundtrip REAL-IT facilities.
- "Generated, but manually supported" means that generated windows were manually changed, but after that were developed outside of REAL-IT.
- "Manually developed" means that windows from the very beginning were developed outside of REAL-IT.

For the first project 95% of the UI has been developed in REAL-IT, about 60% has been fully generated. For the second project 87% of the UI has been developed in REAL-IT and 20% has been fully generated. In the second case REAL-IT was not so effective because this system had specific business logic (for network monitoring). This specificity makes the volume of manual changes to the generated code higher. Many windows were unique and were developed outside REAL-IT.

FOCUS ON WEB-APPLICATIONS

Widely spreading Web-applications stimulate creation of advance development environments. There are several model-based approaches in this area (ref. 7, 8).

REAL-IT is quite suitable for Web-application development because a large number of Web-systems are data intensive systems. It should be noted that WebML approach (ref. 8) and WebRatio toolset[6] are very similar to REAL-IT, but they do not support iterative development process, usage of collaboration diagrams, and other REAL-IT advantages.

At the present moment we have done the following work in order to orient REAL-IT for Web-application modeling:

- We have designed the REAL-IT runtime library for J2EE.
- We have partially ported the generator and the runtime into J2EE platform.
- We have developed the web-prototype of STUDENT system using REAL-IT.

CONCLUSIONS

The orientation of REAL-IT to the defined type of information systems allows us to find out the balance between poor/rich models and simple/complex development process. REAL-IT provides generation of non-trivial target code and quite simple development process. In STUDENT project the level of professional skills of REAL-IT/VB developers needed was significantly less than the usual VB developers.

[6] http://www.webratio.com

Industrial projects that are presented in this paper were developed in collaboration with authors of REAL-IT. In present days, the independent development team of Information Technologies Institute of Saint-Petersburg State University successfully uses REAL-IT to develop and support various information systems.

TABLES

Table 1: REAL-IT in STUDENT-project.

Window type	Fully generated	Manually modified	Generated, but manually supported	Manually developed	Total
Data browser	69	16	6	0	91
Card	37	21	16	0	74
Relation	17	1	4	0	22
Others	0	0	0	6	6
Total	123	38	26	6	193

Table 2: REAL-IT in development of the pilot application of a government system for network management.

Window type	Fully generated	Manually modified	Generated, but manually supported	Manually developed	Total
Data browser	12	27	5	0	44
Card	9	22	7	0	38
Relation	0	0	1	0	1
Others	0	0	0	13	13
Total	21	49	13	13	96

PICTURES

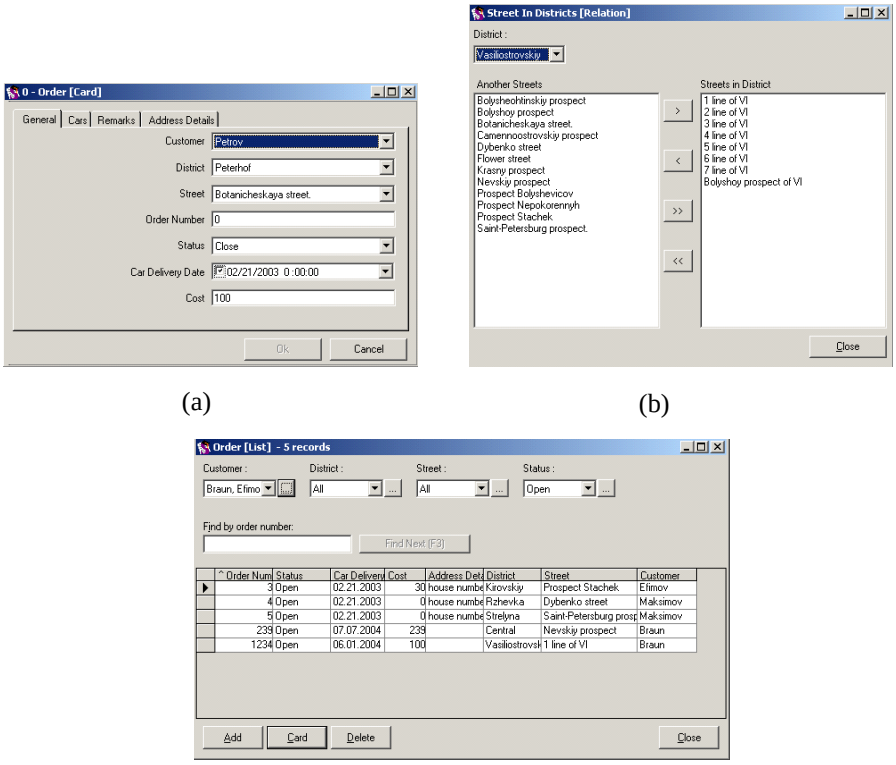

Figure. 1: REAL-IT window types: (a) card; (b) relation; (c) data browser.

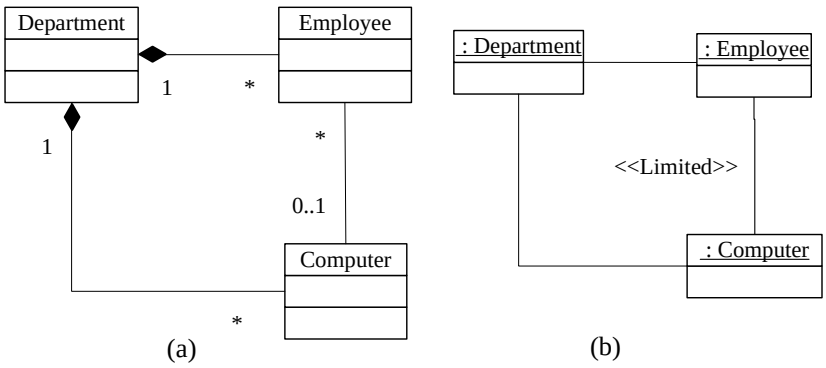

Figure 2: (a) a fragment of a database schema of a business company;
(b) a data dependence for this fragment.

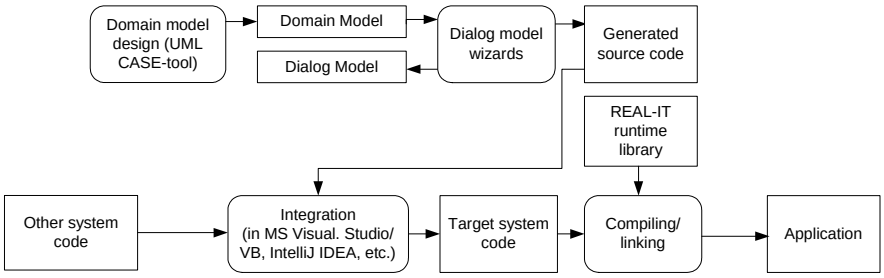

Figure 3: Baseline of user interface development process with REAL-IT.

Figure 4: REAL-IT and its environment.

REFERENCES

1. Da Silva P. P., "User Interface Declarative Models and Development Environments: A Survey", *Lecture Notes in Computer Science*. Vol. 1946, 2000, pp. 207-226.

2. Wiecha C. and Boies S., "Generating user interfaces: principles and use of ITS style rules", *Proceedings of the 3rd annual ACM SIGGRAPH symposium on User interface software and technology*, ACM Press, 1990, pp. 21-30.

3. Szekely P., Sukaviriya P., Castells P., Muthukumarasamy J., and Salcher E., "Declarative Interface Models for User Interface Construction Tools: the MASTERMIND Approach", *IFIP Conference Proceedings*, Vol. 45, Chapman & Hall, London, UK, 1996, pp. 120-150.

4. Janssen C., Weisbecker A., and Ziegler J., "Generating User Interfaces from Data Models and Dialogue Net Specifications", *Proceedings of the SIGCHI conference on Human factors in computing systems*, New York, NY, ACM Press, 1993, pp. 418-423.

5. Grffths T., Barclay P., McKirdy J., Paton N., Gray P., Kennedy J., Cooper R., Goble C., West A., and Smyth M., "Teallach: A Model-Based User Interface Development Environment for Object Databases", *Proceedings of User Interfaces to Data Intensive Systems*, September 1999, Edinburgh, UK, IEEE Press, pp. 86-96.

6. Schlungbaum E., "Model-based User Interface Software Tools: Current state of declarative models", *GIT-GVU-96-30*, November 1996.

7. Schattkowsky T., Lohmann M., "Rapid Development of Modular Dynamic Web Sites Using UML", *Lecture Notes in Computer Science*, Vol. 2460, 2002, pp. 336-350.

8. Ceri S., Fraternali P., Bongio A., "Web Modeling Language (WebML): a modeling language for designing Web sites", *Computer Networks*, 33 (1-6), 2000, pp. 137-157.

9. Terekhov A. N., Romanovskii K. Yu., Koznov D. V., Dolgov P. S., and Ivanov A. N., “RTST++: Methodology and a CASE Tool for the Development of Information Systems and Software For Real-Time Systems”, *Programming and Computer Software*, Vol. 25, No. 5, 1999, pp. 276-281.

10. Aßmann U., “Automatic Roundtrip Engineering”, *Electronic Notes in Theoretical Computer Sci*ence. 82(5), 2003.

11. Koznov D., Kartachev M., Zvereva V., Gagarsky R., Barsov A., “Roundtrip Engineering of Reactive Systems”, *Proceedings of 1st International Symposium on Leveraging Applications of Formal Methods (ISoLA)*, Paphous, Cyprous, 2004, pp.343-347.

12. Sommerville I., “Software Engineering”, *Addison-Wesley*, 6th edition, 2001.

13. Ivanov A. N., “Graphic Language for Describing Constraints on Diagrams of UML Classes”, *Programming and Computer Software*, Vol. 30, No. 4, 2004, pp. 204-208.