An approach to the platform independent specification of a business application

Ivan Luković, Sonja Ristić
University of Novi Sad
Faculty of Technical Sciences
Trg D. Obradovića 6, Novi Sad, Serbia
{ivan, sdristic}@uns.ac.rs

Aleksandar Popović, Pavle Mogin
{University of Montenegro, University of Wellington}
{Faculty of Science, School of Mathematics, Statistics and Computer Science}
{Džordža Vašingtona bb, Podgorica, MN, NZ}
{aleksandarp@rc.pmf.ac.me, Pavle.Mogin@mcs.vuw.ac.nz}

Abstract. *IIS*Studio is a software tool aimed at the information system design and generating executable application prototypes. By means of IIS*Studio, a modeling is performed at the high abstraction level, because a designer creates a system model without specifying any implementation details. Such a model may be classified as a platform independent model (PIM) with respect to the model driven approach to the software development. A goal of this paper is to present PIM concepts and tools embedded into IIS*Studio, that are aimed at specifying business applications. The purpose of such improvements of IIS*Studio is to provide model-to-code transformations, i.e. the development of a code generator for transaction programs and applications that are executed over a database. In the paper we present the IIS*Studio concept named business application, as well as the Business Application Designer tool, by means of a designer can create PIM specifications of business applications. We discuss the expressiveness of new concepts and present Business Application Designer features. We also outline the related concepts concerning visual properties of program forms and functionality of transaction programs.*

Keywords. Software engineering; Model driven development; Form type; Business application.

1 Introduction

Numerous information system (IS) development approaches have been formulated in the last few decades. Nowadays, most of the widely used methods in IS development may be characterized as model-driven. One of the particularly prominent phases in a typical IS development is conceptual modeling which is expected to produce a representation of a real world system using selected platform independent concepts. Platform independent modeling (PIM) of information systems and generation of application prototypes play an important role in software development process. One of its advantages is that it makes the collaboration between developers and users smoother since this approach allows the creation of an environment where end-users can better understand the system being developed and even contribute by doing some of the modeling themselves.

Through a number of research projects lasting for several years, we developed the IIS*Studio development environment (IIS*Studio DE, current version 7.1). It is aimed to provide the IS design and generating executable application prototypes. Our approach is mainly based on the usage of model driven software development (MDSD) [6] and DSL paradigms [14]. The main idea was to provide the necessary PIM meta-level concepts to IS designers, so that they can easily model semantics in an application domain. Afterward, a number of formal methods and complex algorithms may be utilized to produce database schema specifications and IS executable code, without any considerable expert knowledge.

The software development process provided by IIS*Studio is, in general, evaluative and incremental. It enables an efficient and continuous development of an IS, as well as an early delivery of software prototypes that can be easily upgraded or amended according to the new or changed users' requirements. In our approach we strictly differentiate between the specification of a system and its implementation on a particular platform. Detailed information about IIS*Studio may be found in several authors' references. A case study illustrating main features of IIS*Studio and the methodological aspects of its usage is given in [12]. IIS*Studio currently provides the following functionalities:

- Conceptual modeling of database schemas, transaction programs, and business applications of an IS;
- Automated design of relational database subschemas in the 3rd normal form (3NF);
- Automated integration of subschemas into a unified database schema in the 3NF;

- Automated generation of SQL/DDL code for various database management systems (DBMSs);
- Conceptual design of common user-interface (UI) models; and
- Automated generation of executable prototypes of business applications.

A goal of this paper is to present some research results concerning PIM concepts and tools embedded into the IIS*Studio V.7.1 that are aimed at specifying the structures of business applications. These concepts and features provide the specifications of: (i) visual properties of transaction program screen forms; (ii) functionality of transaction programs; and (iii) business applications comprising calling structures.

A business application handles the interaction between an end-user and a database by means of transaction programs that are executed over a database. In the paper we present a general structure of business applications in IIS*Studio, as well as ways for their formal specification. Besides, we present a tool of IIS*Studio named *Business Application Designer* that provides corresponding design activities. *Business Application Designer* enables a designer to specify a business application structure in a unified and visually oriented way, i.e. it provides selecting form types that will participate in the business application, as well as their mutual linkage so as to create the appropriate call specifications.

The paper is organized as follows. In Section 2 we position our approach to business application design. In Section 3 we briefly describe the form type concept that is used in the modeling process supported by IIS*Studio. The concept of a business application, as well as the related concepts necessary to specify business application structures, is introduced in Section 4. The main features and functionalities of the *Business Application Designer* tool are presented in Section 5. The concepts concerning visual properties of program forms and functionality of transaction programs are outlined in Section 6. The last section concludes the paper.

2 Related Work

One of the main motives for developing IIS*Studio is in the following. IS design methodologies based on the techniques such as Entity-Relationship (ER) modeling or general purpose family of languages like Unified Modeling Language (UML), and even the relational data model and an appropriate CASE tool, requires advanced knowledge, skills, and high perception power. Failing to find an appropriate number of designers that possess these properties may lead to a risk of designing poor quality ISs [11]. Besides, these methods and techniques are often incomprehensible to end-users. The main idea of our approach was to formalize concepts from a business domain and keep them well-recognizable by the end-users. In this way, for example, the form type concept (explained in Section 3) was "borrowed" from a business domain as a generalization of a business document. The business documents may provide an important input source for database (db) schema design, since the most widely used data are gathered or reported in them. The concept of a form type in our approach mainly corresponds to the concept of a business component that is the main modeling concept *DeKlarit* tool [4] relies on. *DeKlarit,* like the IIS*Studio, can generate relational database schema and SQL commands for various DBMSs. Unlike the *DeKlarit*, IIS*Studio further provides specific concepts and tools for the specification of the transaction programs and business applications.

Batini et al. in [5] and Choobineh et al. in [7] used business forms as input data for the process of database schema design based on generating entity-relationship (ER) diagrams. Unlike them, IIS*Studio generates relational database schemas and executes an efficient transformation of design specifications into error free SQL specifications of relational database (db) schemas for different DBMSs ([1], [2] and [3]). Choobineh and Venkatraman in [8] presented a methodology and tools for derivation of functional dependencies from business form. In our approach, besides the set of functional dependencies F the initial set of constraints, inferred from a form type, withal consists of: a set of non-functional dependencies NF, a set of special functional dependencies F_u, and a set of null value constraints N_c. The importance of form-driven approach is emphasized in [10]. Namely, Kreutzová, Porubän, and Václavík, claim that it is possible to automatically analyze existing user interface and search for domain terms in the set of screen forms.

In contrast to other approaches, by the approach IIS*Studio relies on, a designer simultaneously specify both forms and the structure of a database schema. Namely, by creating form types, a designer specifies screen or report forms, and an initial set of attributes and database constraints, at the same time. Then, created form types are transformed into the database schema specification, and also they are the source for the generation of transaction programs. By means of *Oracle Designer*, for example, a designer may create modules that represent specifications of transaction programs with their screen or report forms and menus, aimed to execute over a database. The concept of a module rather corresponds to the form type concept used in IIS*Studio. But, in order to specify a module in *Oracle Designer*, a necessary part of the database schema must be already completed. Besides, various segments of a business application structure are specified separately, at the level of a module, and there is no any particular tool for application design.

Genero Studio Business Application Modeling (GSBAM) [9] enables modeling of business applications using diagrams describing the

functionality of the applications. A designer may model the components of applications and their interconnection in a high level business diagram that is in a great extent similar to our business application diagram. GSBAM enables prototype generation and the enhancement of the generated code with handwritten custom code. Keeping in mind wide rang of GSBAM features and options, we emphasize the difference in the development lifecycle between GSBAM and IIS*Studio. Namely, forms and reports in GSBAM are specified after the database is designed. The database design process is not incorporated in GSBAM. In our approach, by creating form types, a designer at the same time specifies: (i) a future database schema, (ii) functional properties of future transaction programs, (iii) and a look of the end-user interface.

3 A Form Type Concept for Building a PIM

All the designers' specifications of a system model created by IIS*Studio belong to an IIS*Studio *project* [12]. Each project is organized as a tree structure of *application systems*, where each application system may contain an arbitrary number of form types. A *form type* is the main modeling concept in IIS*Studio. Each form type is an abstraction of business documents, and therefore screens or report forms utilized by the end-users in the communication with the IS. On the contrary to the traditional approaches to the IS design, where database schema design precedes the specification of screen or report forms of transaction programs, in IIS*Studio a designer the first specifies screen or report forms, and indirectly, creates the initial set of attributes and constraints. The set of form types is a platform independent view onto the system from the end-user perspective. IIS*Studio uses the set of form types to generate the relational database schema, and its closure graph [12]. In this way, by creating form types, a designer specifies (i) a future database schema, (ii) functional properties of future transaction programs, (iii) and a look of the end-user interface, at the same time. Furthermore, IIS*Studio provides the generation of SQL/DDL code and the generation of a program code of transaction programs and applications that are executed over a database. In such a way, two new views onto the system are obtained, but this time, both of them are platform specific.

A form type can be designated as *menu* or *program*. If a form type is denoted as *menu*, it will be utilized to generate a menu of screen or report forms, of the future business application. If it is denoted as *program*, it will be utilized to generate a transaction program with its screen or report form. More information about form type concept may be found in [12] and [13].

4 A PIM Concept of a Business Application

In our approach, a *business application* is a named structure of interrelated transaction programs aimed to support business activities at an organization (business) unit. End-user perception of the future IS highly depends on the way how business applications of the IS are structured. We consider as the important that a tool providing the IS design and generation of executable application prototypes, also provides creating business application specifications in the design phase. The major consequence of a lack of these specifications in the model is that the code of transactions programs, generated from such a model, has to be extended and customized with a handwritten code, in order to implement the business application structure. In that case, the resulting code would stay unsynchronized with the system model at the higher abstraction level, and one of the basic model-driven principles would be derogated. Therefore, the design of business applications is an important task in the IS modeling process.

A business application is a concept in the IIS*Studio by means of a designer formally specifies the IS functionality concerned the calls between generated transaction programs, i.e. their forms [15]. The scope of a business application is the application system and each business application belongs to exactly one application system. The specification of a business application in IIS*Studio comprises a structure of the selected form types. Therefore, specifying form types must precede the design of a business application. While a form type is a source for the generation of a sole transaction program code with its screen or report form, a business application specification is a source for the generation of a program code that covers calls between generated transaction programs, i.e. their forms, and a synchronization of their behavior.

The first step in specifying the business application structure is to select necessary form types. One of the selected form types has to be designated as a *start-point* form type. The screen form generated from the start-point form type is the first one accessible to end-users, when they initiate the business application, and by means of they can access the other (subordinated) forms in the application.

After the selection of form types, it is necessary to specify their mutual relationships so as to specify the business application structure. Therefore, a new concept, named *call specification* or *call* for short, is introduced in [15] at the level of a business application. A call specification is a pair of the form types participating in a business application. If (F_1, F_2) is a call specification within a business application, then the screen form generated from the form type F_1 has to support calls of the screen form generated from the form type F_2. In this case, F_1 is named the *calling*

form type, and F_2 is the *called form type*. Besides, each call specification in IIS*Studio has the following properties:
- *Passed values*, which allows specifying the list of passing values from the calling to the called form types;
- *Calling mode*, which allows specifying the rules for data selection and transferring data from the calling to the called form type;
- *Calling method*, which allows specifying a behavior of the calling and the called form type; and
- *UI positioning*, which allows specifying positioning properties of the UI control item for executing the call.

The concepts of a call and a business application structure, defined in this way, enable a designer to formally specify a significant part of the IS functionality concerning relationships between generated transaction programs, i.e. their screen forms. In the following text we present the properties of the call specification in more details.

4.1 Passed Values and Parameter-to-Value Associations

An important role in the business application design plays the specification of data passed from a calling to the called screen form. Very often, it is necessary to pass data to the called form, in order to reach the full form functionality. As an example, a form aimed at browsing courses taken by a student, requires the data about student's ID and it should be passed during the call execution. Therefore, we introduce a concept named *parameter* and associate it to the form type specification. In this way, each form type created in IIS*Studio may contain a set of form type parameters [15]. They are used for storing data passed during call executions. All the necessary form type parameters should be specified prior to the creation of a call specification, because they are used latter on in the call specifications, for specifying *parameter-to-value associations*, or *parameter bindings* for short.

Each parameter-to-value association in a call specifies what would be the passed value from the calling form type to a parameter of a called form type. The possible selections for a passed value, i.e. a parameter-to-value association are:
- a calling form type attribute, if passed value to the parameter should be a value of the selected attribute,
- a constant value, if passed value to the parameter is a constant, and
- a calling form type parameter, if passed value to the parameter should be a value of the selected calling form type parameter.

4.2 Calling Mode

This call property specifies a behavior of the called form concerning data retrieval from the database and passed values from the calling form. Sometimes, the called form may be restricted to retrieve data from the database only in the context of values passed from the calling form. Besides, it is necessary to specify if the data will be retrieved automatically from the database on the activation of the called form. Therefore, this property may have one of the following values:
- *Select on open*,
- *Restricted select*, or
- *Select on open* & *Restricted select*.

If *Select on open* is chosen, the data are automatically retrieved on the activation of the called form and further data retrievals will be unrestricted with respect to the passed values. If *Restricted select* is chosen, the data are not automatically retrieved. Instead, they are retrieved only on a user request. In this case, the data retrieval is always restricted to the context of passed values. If *Select on open & Restricted select* is chosen, that data are automatically retrieved, and the data retrieval is always restricted to the context of passed values.

4.3 Calling Method

This call property specifies a behavior of the calling and the called form type, i.e. a mutual relationship between UI windows of generated forms. The property may have one of the following values:
- *Modal*,
- *Non-Modal*, and
- *Non-Modal* & *Close calling form*.

If *Modal* is selected, the generated screen forms are executed in the dialog mode, i.e. a user cannot access the calling form while the called form is active. If *Non-Modal* is selected, an access to the calling form and the called form are mutually unconstrained, i.e. a user may access both of them independently. If *Non-Modal & Close calling form* is selected, the calling form is closed, and the focus is set to the called form.

4.3 UI Positioning

A UI program control item for executing a call should be generated in a way to provide its convenient and fast usage by the end-users. It is particularly important when the call is frequently initiated. The *UI positioning* property specifies how and where a *UI program control item for executing a call*, or the *call item* for short, will be generated. It may have one of the following values:
- *Show as menu item*,
- *Show as button*, or
- *Show as menu item* & *Show as button*.

If *Show as menu item* is selected, the call item will be generated as a menu item in the calling form menu. If *Show as button* is selected, the call item will be generated as a button in the calling form. If *Show as menu item & Show as button* is selected, both a menu item and a button will be generated in the calling form.

5 Modeling Business Applications in IIS*Studio

An IS may comprise several business applications of an arbitrary complexity. Therefore, designers need a suitable tool that can provide an efficient design of business applications regardless their complexity. *Business Application Designer* is a tool embedded into IIS*Studio that provides the business application design. It supports specifying the structures of business applications in a simple and visually oriented way. It enables including the available form types into a business application, and creating the calls between the selected form types [15] and [16].

In our approach, a business application structure in *Business Application Designer* is a graph that is visually represented in the form of a diagram. Apart from its useful diagram visualization, a graph representation of the business application structure is beneficial due to the utilization of the well known graph algorithms, particularly the algorithms for the graph traversal, and for checking the graph connectivity. In this way, by means of the algorithm for checking the graph connectivity we perform a validity test of the business application structure. If there are any disconnections detected in the graph, i.e. if there are form types that are isolated, we consider the business application structure as invalid, since end-users will not be able to access these forms in the generated business application.

A project tree in IIS*Studio for a project *Faculty organization* is presented in Fig. 1. Each project tree contains the *Application Systems* node enclosing all the application systems created in the project *Faculty organization*. Furthermore, each application system contains the *Business Application* node enclosing all the business applications in the application system. In Fig. 1 the *Faculty organization* application system is presented with its *Business Applications* node that encloses all the business applications created within this application system.

The first step in the creation of a business application is giving its name that must be unique in the scope of the application system, and selecting a start-point form type. The next step is specifying the business application structure by the *Business Application Designer* tool. In Fig. 2 it is presented the main window of the *Business Application Designer* with: (i) a main menu and the toolbar with selected commands, positioned on the top, (ii) a diagram of a simplest business application, containing the only one form type, positioned on the right hand side of the main pane, and (iii) the *Navigator*, positioned on the left hand side of the main pane.

In the step of specifying the business application structure, a designer may include the available form types into the structure, or to exclude the previously selected ones. A designer may select only the form types that belong to the scoping application system of a business application. Each application system may have: (i) its own form types, (ii) the referenced form types owned by the other application systems, as well as (iii) the child application systems with its own form types, as it may be seen in the *Navigator* in Fig. 2. The form types of all those categories may be included into the business application. Whereas a form type may be included in several different business applications, it may not be included into the same business application twice.

After the necessary form types are included into the business application, a designer may specify calls between the form types. The arrow symbolizing the call is always directed from the calling to the called form type. In Fig. 3 it is presented an example of a business application diagram. It may be noticed that the *program* form type *FACULTY* calls the *menu* form type *STUDENT MENU*. Consequently, the application generator will transform the form type *FACULTY* into a screen form, the form type *STUDENT MENU* into a menu, and embed it into the main menu of the generated screen form *FACULTY*. It may also be noticed that the *menu* form type *PERSON MENU* calls the *menu* form type *LIST OF PERSONS*. The application generator will produce two menus: *PERSON MENU* and *LIST OF PERSONS*, and it will create an item in *PERSON MENU* for calling the *LIST OF PERSONS* menu.

After specifying the calling and the called form type of a call, a designer should specify the call properties (explained in Section 4) in order to complete the whole specification. In Fig. 4 it is presented a form that provides specification of parameter-to-value associations. The form presented in Fig. 5 provides specification of calling method, calling mode, and UI positioning.

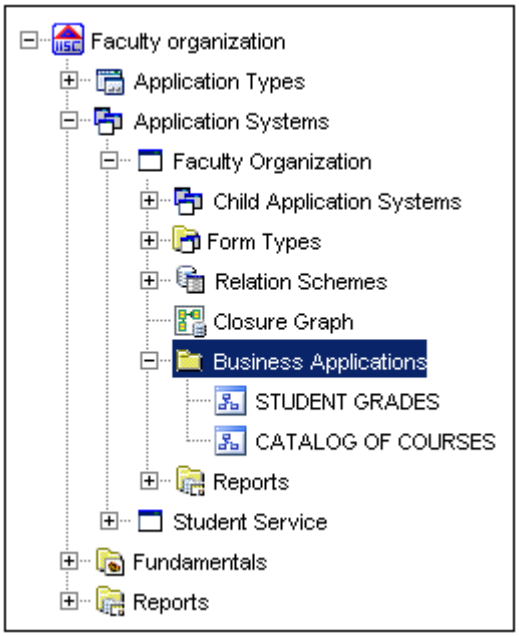

Figure 1. The IIS*Studio project tree for the *Faculty organization* project

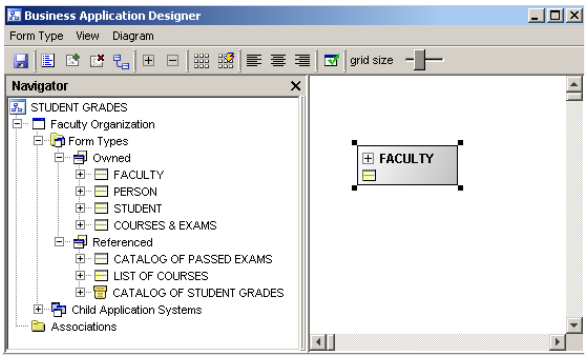

Figure 2. The IIS*Studio diagram of a business application with only one form type

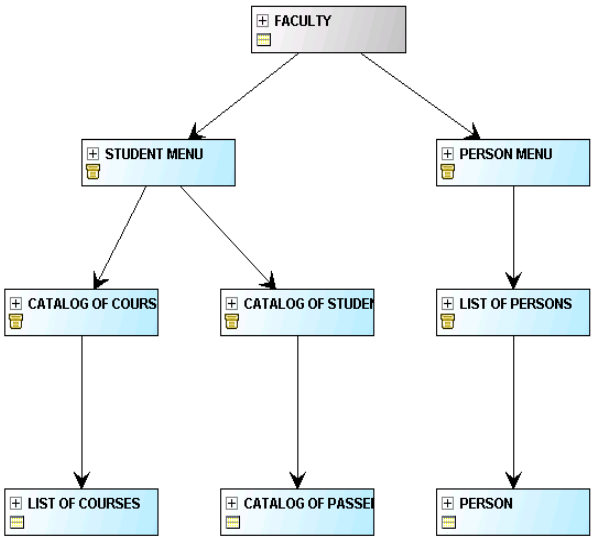

Figure 3. A business application diagram

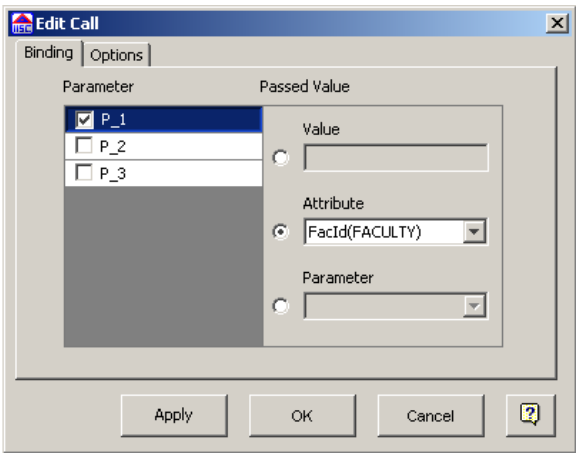

Figure 4. The form for specifying parameter-to-value associations

During the design of a business application structure the errors may occur regarding the graph connectivity. In Fig. 6 is presented an example of an invalid business application structure, since the form type *PERSON* is isolated and cannot be accessed by an application user.

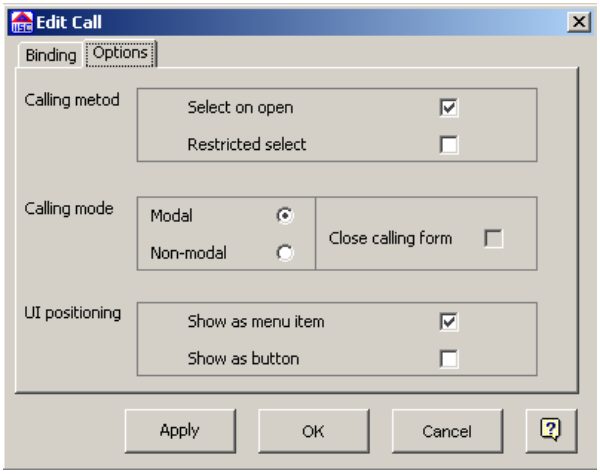

Figure 5. The form for specifying of calling method, calling mode, and UI positioning

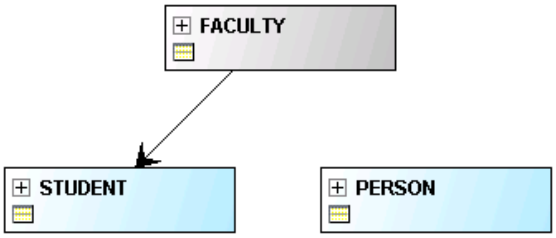

Figure 6. A diagram of an invalid business application structure

6 Related PIM Concepts in IIS*Studio

The main purpose of IIS*Studio is to provide a chain of necessary model-to-model and model-to-code transformations, so as to generate fully functional transaction programs and applications that are executed over a database. Therefore, apart from having a support for modeling PIM specifications of business applications, we consider that it is necessary to have a support for the PIM specifications of:

- visual properties of transaction program forms,
- functionality of transaction programs, and
- common UI models.

In the rest of the section, we briefly outline each of the aforementioned specifications. Due to their complexity, we omit their detailed explanations.

There are several concepts in IIS*Studio, for which it is possible to specify visual properties having an influence on the generated program screen forms. Those are: domain, attribute, component type of a form type and component type attribute [15]. The visual properties of the component type attribute, the attribute and the domain are used to specify the common characteristics of generated screen items that are independent of any programming environment. In this way, a designer may specify a type of each screen

item generated from the attribute of a form type. The allowed item types are: textbox, radio group, check box, combo box, and list item. For each of them, it is possible to specify a number of specific properties. All those properties are organized in a hierarchical way. The most general level in the three-level hierarchy is the level of the domain. The second one is the level of the attribute, whereas the most specific is the level of the component type attribute. For example, if a visual property is specified at the domain level, it will hold in general, for all the attributes associated to that domain. However, the property value may be overridden at the attribute level, for a specific attribute. In that case, such a value will be used for the specific attribute, i.e. for all component types containing that attribute. At the third level, it is possible to override the property value again. In that case, it will hold only for the attribute included into the specific component type. The visual properties of the component type of a form type mainly concern the information about positioning the component type window relatively to the other component type windows, as well as positioning and laying out of the component type content inside the window.

Currently, IIS*Studio provides PIM specifications of transaction program functionality in a great extent. We differentiate between "standard" and "nonstandard" data operations of a transaction program. Standard operations are data retrieve, insert, update and delete. They are specified declaratively for each component type of a form type, as it is presented in [12]. Nonstandard operations are specified in a procedural way, by using the program unit concepts: package, function and event. We have developed and embedded into IIS*Studio a *Function Logic Editor* that provides creating the formal and platform independent specifications of functions. Using the editor, a designer can specify both declarative, procedural and exception parts of a function in a fully structured and visually oriented way. The resulting specifications are stored in the IIS*Studio repository. Based on these specifications IIS*Studio currently provides: (i) coupling of program unit specifications with form types and their component types and attributes; (ii) model-to-model transformations of program unit specifications into a domain specific and platform independent language; and (iii) model-to-code transformations of specifications expressed by the selected domain specific language into the executable program code.

We believe that a lot of visual properties of transaction programs and common logic of a UI may be defined at the level of a project as a whole, or at least at the level of an application system. Therefore, we have developed a model and a tool named IIS*UIModeler for specifying various *Common UI Models*. A *Common UI Model* is a set of platform independent templates that may be applied in the process of generating executable application prototypes provided by IIS*Studio. By these templates, a designer specifies visual and logic properties of a common UI model that each generated application must satisfy. IIS*UIModeler [17] provides a list of predefined templates, as well as creating the new templates in two ways: (i) by the inheritance of existing ones, or (ii) from scratch. In this way, it is possible to specify a number of details about UI logic, layout and visual properties concerning windows, forms, blocks and items for browsing, inserting, updating, and deleting data.

7 Conclusions

We present in the paper a methodological approach to the specification of business applications. The approach requires introducing deliberate concepts and developing a tool by means of a designer can specify the business applications in a unified and visually oriented way. The concepts and the *Business Application Designer* tool that we have developed are embedded into the IIS*Studio tool. In this way, it is provided the design of a database schema, transaction programs, their screen or report forms, and finally the business applications, in an integrated way. By this, we cover the software modeling process in a great extent. Modeling of a system by IIS*Studio is performed at a high level of abstraction, but with a usage of the concepts that are domain specific for IS development. Such software specifications can be used as a sound basis for the generation of DDL database schema specifications and fully executable application prototypes. In contrast to our approach and the IIS*Studio tool, the general purpose modeling tools that are based on the UML language family, provide software design for a wider class of systems, which is an advantage. However, if we consider the usage of such tools in the system modeling, it may be harder to express some specific properties or system characteristics in a particular application domain. As a consequence, such a software model is harder to transform into the fully executable program code in a particular program environment.

A business application specification together with the specifications of selected UI templates is a source for the generation of a program code that covers calls between generated transaction programs, i.e. their forms, and a synchronization of their behavior. A case study illustrating main features of IIS*Studio application prototype generation is given in [17], alongside with the methodological aspects of its usage.

Among all, our current or future research efforts comprise also the following:

- Further improvements of IIS*Studio that will enable a designer to generate complex transaction program functionalities concerning not only "standard" data operations (retrieve, insert, update and delete) expressed by the designed form types,

- Introducing new concepts and tools in IIS*Studio that will provide business process modeling, as well as system architecture modeling, and
- Investigating a possible usage of category theory: i) in order to improve the performance of generated code, on the one hand [18], and ii) as a common language and tool for software engineering task instrumentation on the other hand [19].

8 Acknowledgments

Research presented in this paper was supported by Ministry of Science and Technological Development of Republic of Serbia, Grant III-44010, Title: *Intelligent Systems for Software Product Development and Business Support based on Models*.

References

[1] Aleksić, S.; Luković, I.; Mogin, P.; Govedarica, M. A Generator of SQL Schema Specifications, *Computer Science and Information Systems* (ComSIS), Vol. 4, No. 2, 77-96, 2007.

[2] Aleksić, S.; Luković, I. Generating SQL Specifications of a Database Schema for Different DBMSs, *Info M-Journal of Information Technology and Multimedia Systems*, No. 23, 36-43, 2007.

[3] Aleksić, S.; Ristić, S.; Luković, I. An Approach to Generating Server Implementation of the Inverse Referential Integrity Constraints, In *Proceedings of The 5th International Conference on Information* Technologies, Jordan, CD, 2011.

[4] ARTech. *DeKlarit*™, http://www.deklarit.com/, downloaded: May, 2010.

[5] Batini, C.; Demo, B.; Di Leva, A. A methodology for conceptual design of office data bases, *Information Systems* 9(3/4), 251–263, 1984.

[6] Bézivin, J. In search of a basic principle for model driven engineering. *UPGRADE - The European Journal for the Informatics Professional, 5*(2), 21-24, 2004.

[7] Choobineh, J.; Mannino, M.V.; Tseng, V.P. A form-based approach for database analysis and design, *Communications of the ACM* 35 (2), 108–120, 1992.

[8] Choobineh, J.; Venkatraman, S.S. A methodology and tools for derivation of functional dependencies from business form, *Information Systems* 17 (3), 1992, pp. 269–282.

[9] Genero, Introduction to Genero Studio Business Application Modeling, http://www.4js.com; downloaded: April 27th 2011.

[10] Kreutzová, M.; Porubän, J.; Václavík, P. First Step for GUI Domain Analysis : Formalization, *Journal of Computer Science and Control Systems*. Vol. 4, no. 1, 65-70, 2011.

[11] Kosar, T.; Oliveira, N.; Mernik, M.; Varanda Pereira, M. J., Črepinšek, M., da Cruz, D., Henriques, P. R. Comparing general-purpose and domain-specific languages: An empirical study. *Computer Science and Information Systems*, *7*(2), 247-264, 2010.

[12] Luković, I.; Mogin, P.; Pavićević, J.; Ristić, S. An approach to developing complex database schemas using form types. *Software: Practice and Experience, 37*(15), 1621-1656, 2007.

[13] Luković I.; Ristić S.; Mogin P.; Pavicević J. Database Schema Integration Process – A Methodology and Aspects of Its Applying, *Novi Sad Journal of Mathematics*, Serbia, ISSN: 1450-5444, Vol. 36, No. 1, 115-150, 2006.

[14] Mernik, M.; Heering, J., & Sloane, M. A. When and how to develop domain-specific languages. *ACM Computing Surveys*, *37*(4), 316-344, 2005.

[15] Popović A, *A Specification of Visual Attributes and Structures of Business Applications in the IIS*Case Tool*, M.Sc. (Mr) Thesis, University of Novi Sad, Faculty of Technical Sciences, 2008.

[16] Popović A.; Luković I.; Ristić S. A Specification of the Structures of Business Applications in the IIS*Case Tool, *Info M – Journal of Information Technology and Multimedia Systems*, Belgrade, Serbia, ISSN: 1451-4397, No. 25, 17-24, 2008.

[17] Ristic S.; Aleksic S.; Lukovic I.; Banovic J. Form-Driven Application Generation: A Case Study, In *Proceedings of the XI International Conference on Informatics*, Roznava, Slovakia, 115 – 120, 2011.

[18] Slodičák V. Some useful structures for categorical approach for program behavior, *Journal of Information and Organizational Sciences*, Vol. 35, No. 1, 99-109, 2011.

[19] Szabó, C.; Slodičák, V. Software Engineering Tasks Instrumentation by Category Theory, SAMI 2011, *Proceedings of the 9th IEEE International Symposium on Applied Machine Intelligence and Informatics*, Smolenice, Slovakia, 195-199, 2011.