IPASJ International Journal of Management (IIJM)
Web Site: http://www.ipasj.org/IIJM/IIJM.htm
A Publisher for Research Motivation ..... Email:editoriijm@ipasj.org
Volume 5, Issue 11, November 2017 ISSN 2321-645X

FORM-DRIVEN APPLICATION DEVELOPMENT

Estefan, Gloria

University of Leeds

ABSTRACT

In the paper a form-driven approach to the application development is presented. Our development environment IIS Studio suports presented form-driven approach. It is aimed to provide the information system design and generating executable application prototypes. A form type is central IIS*Studio concept, used to model the structure and constraints of various business documents. On the one hand, a set of specified form types represents a platform independent model (PIM) of a real system. On the other hand, it is a PIM of the future software application. Starting from such a platform independent specification, through the chain of transformations, IIS*Studio generates a set of relational database subschemas in the 3rd normal form and also a global relational database schema by integration of the subschemas. It enables a full implementation of database schemas under different target database management systems. IIS*Studio comprises a tool for the formal specification of business applications, and a generator of the executable application prototypes. The case study presented in the paper illustrates main features of IIS*Studio application generator tool and the methodological aspects of its usage. We consider the chain of transformations from a PIM, through the series of platform specific models with different degree of platform specificity, towards executable program code, as a crucial in our approach.

1. INTRODUCTION

Informally, a form is a way of organizing and presenting data. In the context of organization and its information system one can distinguish between business forms and computerized forms. Business forms (documents) are broadly used in organizations to conduct daily operations and to communicate with their affiliated entities (e.g. staff, superior managers, customers, suppliers, etc.). They may provide an important input source for database (db) schema design, since the most widely used data are gathered or reported in them ([1], [2], [3], [4]). There are tools (CASE or model-driven) aimed at automated application generation using set of forms as input source, such as DeKlarit [5]. On the other hand, users in organizations are experienced in handling and manipulation of databases through screen (computerized) forms which are the most natural interface between user and data. Database systems have been extended to deal with forms and other types of documents used to facilitate the manipulation of data via computerized form (e.g., Oracle SQL Forms, Informix-SQL, Microsoft Access Forms, different reports tools, etc.) [6]. Forms are objects, easy to read and understand, well structured and, consequently, easy to formalize. Therefore, business forms are a source for eliciting user information requirements and also for designing and developing user-oriented information systems. There are various research works about the use of forms (business and/or computerized) in different contexts. In order to integrate services in Office Information system, Tsichritzis in [7] introduces the concepts of form type, form template and form instance. In [8] and [9] Shu et al. proposed using forms to specify system requirements. Batini et al. in [10] and Choobineh et al. in [1] and [2] used business forms as input data for the process of database schema design based on generating entityrelationship (ER) diagrams. Choobineh and Venkatraman in [11] presented a methodology and tools for derivation of functional dependencies from business form. A formbased approach for reverse engineering of relational databases is proposed by Malki, Flory, and Rahmouni [6]. This methodology uses the information extracted from both form structure and instances as a database reverse engineering input using an interaction with a user. This approach inspired later research on extracting personalized ontology from data-intensive web application [12]. Namely, S. M. Bensliman, Malki, Rahmouni and Dj. Bensliman based their approach on the idea that semantics can be extracted by applying a reverse engineering technique on the structures and the instances of HTMLforms which are the most convenient interface to communicate with relational databases on the current data-intensive web application. This semantics is exploited to produce a personalized ontology. In a similar manner, Kreutzová, Porubän, and Václavík in [13], claim that a graphical user interface (GUI) is a partial description of the application domain. They argue that it's possible to automatically analyze existing user interface and search for domain terms in the set of GUI forms. Tailoring some ideas from [14] and [15], Wu et al. in [16] presents a methodology that uses factoring and synthesis to process knowledge involved in forms for designing form-based decision support systems.

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 [5] 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 ([17] and [18]). Our approach has some similarities with the approaches presented in [1], [2], [10], [11] and [19]. But, besides the set of functional dependencies F (like in [11]) 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 Fu, and a set of null value constraints Nc. Their detailed explanation with examples may be found in [3]. While in approaches presented in [1], [2] and [10], just ER diagrams are generated, 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 ([20], [21] and [22]). Although our research does not tackle the same problems as it is in [6] and [12], a common value is a reported necessity of the integration of independently developed database subschemas. Integration of db subschemas in [6] and [12] is done at the conceptual level. In our approach it is done at the implementation instead of the conceptual level [23]. A db schema at the implementation level is expressed by the relational data model. In the paper the case study is presented that illustrates main features of IIS*Studio application prototypes generator tool, as well as the methodological aspects of its usage. The paper is organized as follows. In Section 2 it is presented a real system of Safe House Center (SHC), whose information system is designed by IIS*Studio. Section 3 explains main concepts needed to understand generation of applications in IIS*Studio. Methodological aspects of the usage of IIS*Studio Application Generator are given in Section 4 through an example of Donations subsystem of the SHC information system. The last section concludes the paper.

2. CASE STUDY: THE SAFE HOUSE CENTER

The Safe House Center (SHC) provides support for those children impacted by domestic violence. It offers a safe and nurturing environment where qualified staff assists youth with their basic needs, assessment, advocacy and stabilization. Besides, it organizes foster care and assists and supervises foster families. The SHC is on a great extent based on donations. In the paper we will present the Donation subsystem of the SHC information system. It is simple compared with the whole SHC information system, but suficiently complex to allow an ilustration of main features of IIS*Studio application generator tool, and the methodological aspects of its usage.

A donor may be a natural person, a legal entity or a group of citizens. A contribution to SHC may be made by donating: cash, check, goods, remedies, food, bequests, insurance, stocks, bonds or mutual funds, etc. All donations must be recorded. A donation agreement is made between a donor and SHC. In the agreement all donated items are specified. A donor may require staying anonymous, meaning that all external documents should not contain data about donor. The business forms used in SHC important for Donation subsystem beyond others are: Donor Card (containing basic data about donor: name, type, contact data, etc.) and Donation Agreement (presented in Fig. 1). In the following sections we present the process of formdriven application generating by means of IIS*Studio and using the SHC case study.

3. AN EXAMPLE OF APPLICATION GENERATION BY MEANS OF IIS STUDIO

IIS*Studio relies on the approach that conforms to the principles of model-driven approach. By means of IIS*Studio, a designer specifies only PIM models, because they are free of any implementation details. By the chain of consecutive transformations a set of different semi or fully platform specific models (PSMs) is generated ([3] , [17], [24] and [23]). The chain of transformations in IIS*Studio can be divided into three subsets: (i) transformations aimed to generate a formal and an implementation DB schemas (DB schema generator); (ii) UI (User Interface) prototype generators; and (iii) application generators. The first and the third subset of transformations are mandatory in order to generate application prototype (see Fig. 4). The second subset is optional, and helps designer in the process of selecting the best fitting UI. Therefore, its presentation is omitted here. A case study illustrating the first subset of transformations may be found in [3] and [23]. In the following text a case study illustrating the third subset of transformations is presented. Fig. 4 The chain of model transformations implemented in IIS*Studio The first step of the process of business application specification and generation by means of IIS*Studio is Project Specification (Fig. 4). Fig. 5 presents the Safe House project tree, containing one application system Donation with form types Catalog of Donations, City, Donation Agreement, Donator and Foster Family that are modeling corresponding business forms. The form types are specified through IIS*Studio screen forms, like the one in Fig. 2. After selecting the tab Component type, designer may insert/edit/delete component type, or/and insert/edit/delete the attributes (Fig. 6) of selected component type, or/and specify component type constraints (key, unique or check constraints). During the process of attribute specifications one

would specify the attribute title, behavior (is it modifiable or not), allowed operations, obligingness and default value through tab Definition (see the left-hand side of Fig. 6). This part of specification is important for the database schema generation process. The tab Display is important for the application generator, because it enables specifying the display characteristics of an attribute. The selected display option for the attribute TypeD is ComboBox (Fig. 6), and possible values are a natural person, a legal entity or a group of citizens. The example of Donation Agreement form type illustrates the multiple role of form type. At the same time it is: a model of Donation Agreement business form (Fig. 1); a data model – one can see it as a conceptual database schema (Fig. 2); a model of computerized (screen) form (Donation Agreement screen form in Fig. 8); and a model of transaction program, since the functional properties of future transaction program are specified. The form type structuring rules provide an automatic inference of the set of attributes and relational database constraints.

After the selection of form types, it is necessary to specify their mutual relationships ("calls") so as to specify the business application structure. In the business application diagram in Fig. 3 the arrow between the Donator and Donator Agreement form types indicates a call specification within a business application. In this case, Donator is the calling form type, and Donator Agreement is the called form type. By selecting a form type and Edit option the form types that are to be called from a selected form type are specified (Fig. 8). The consequence of such a specification is that the screen form generated from the calling form type has to support calls of the screen form generated from the called form type. For example, clicking the button Donation Agreement on the screen form Donator causes calling the corresponding screen form (Fig. 9). Besides, each call specification in IIS*Studio has the following properties (Fig. 8): Passed values – the list of passing values from the calling to the called form types; Calling mode – the rules for data selection and transferring data from the calling to the called form type; Calling method – a behavior of the calling and the called form type; and UI positioning – positioning properties of the UI control item for executing the call.

4. CONCLUSIONS

The form-based approaches in the software engineering are present for more than two decades. Some of them are for: database analysis and design; extracting object-oriented db schemas from relational databases; extracting personalised ontology from data-intensive web applications; designing form-based decision support systems; etc. In our approach we are using the notion form-driven, instead of form-based, inspired with the different meaning of the model-based and model-driven approaches. A form type is the central concept of our IIS*Studio tool aimed at automated IS design and application generation. By creating form types, a designer specifies at the same time: (i) a future database schema, (ii) functional properties of future transaction programs, and (iii) a look of the end-user interface. One of the main assumptions of the model-driven approach to software system development is that software systems of large complexity can only be designed and maintained if the level of abstraction is considerably higher than that of programming languages. By means of models, semantics in an application domain can be precisely specified using terms and concepts the end-users are familiar with, such as the business forms. The focus of software development is shifted from the technology domain toward the problem domain. In our future work, in order to justify the correctness of the form-driven adjective we aim to investigate the reverse influence of the changes in the database schema to the screen and business forms. Category theory ([26], [27]) provides a kind of common language and tool, in which a sketch is a specification based on graphs as the formal structure. We plan to investigate a possible usage of category theory: i) in order to improve the performance of generated code, on the one hand ([26]), and ii) as a common language and tool for software engineering task instrumentation on the other hand ([27]).

REFERENCES

[1] CHOOBINEH, J. – MANNINO, M. V. – NUNAMAKER, J. F. – KONSYNSKI, B. R.: An expert database design system based on analysis of forms, IEEE Transactions on Software Engineering 14 (2), 1988, pp. 242–253.
[2] CHOOBINEH, J. – MANNINO, M. V. – TSENG, V.P.: Form-based approach for database analysis and design, Communications of the ACM 35 (2), February 1992, pp. 108–120.
[3] LUKOVIĆ, I. – MOGIN, P. – PAVIĆEVIĆ, J. – RISTIĆ, S.: An Approach to Developing Complex Database Schemas Using Form Types, Software: Practice and Experience, John Wiley & Sons, USA, Vol. 37, No. 15, 2007, pp. 1621–1656.
[4] MOGIN, P. – LUKOVIĆ, I. – KARADZIĆ, Z.: Relational Database Schema Design and Application Generating Using IIS*CASE Tool, Proceedings of International Conference on Technical Informatics, Timisoara, Romania, 1994, Vol. 5, pp. 49–58.
[5] ARTech. DeKlaritTM, Chicago, U.S.A. [Online]. May 2010, Available: http://www.deklarit.com/

[6] MALKI, M. – FLORY, A. – RAHMOUNI, M. K.: Extraction of Object-oriented Schemas from Existing Relational Databases: a Form-driven Approach, INFORMATICA, Vol. 13, No. 1, 2002, pp. 47–72.
[7] TSICHRITZIS, D.: Form management, Communications of the ACM 25 (5), July 1982, pp. 453–478.
[8] SHU, N. C.: Formal: A Form-Oriented, VisualDirected Application Development System, Computer, 1985, pp. 38–49.
[9] SHU, N. C. – LUM, V. Y. – TUNG, F. C. – CHANG, C. L.: Specification of forms processing and business procedures for office automation, IEEE Transactions on Software Engineering -8 (5), 1982, pp. 499–512.
[10] BATINI, C. – DEMO, B. – DI LEVA, A.: A methodology for conceptual design of office data bases, Information Systems 9 (3/4), 1984, pp. 251– 263.