УДК 614.2+574/578+004.38

ВДОСКОНАЛЕННЯ ЯКОСТІ ПРОЦЕСУ РОЗРОБКИ ТА ПІДТРИМКИ ІНФОРМАЦІЙНИХ СИСТЕМ

О.А. Литвинов[1], П.В. Давидов[2], Т.В. Магро[3]

[1] кандидат технічних наук, доцент кафедри електронних обчислювальних машин, Дніпровський національний університет ім. О. Гончара, м. Дніпро, Україна, e-mail: lishu.dnepr.ua@gmail.com

[2] студент групи КІ-14-1, кафедра електронних обчислювальних машин, Дніпровський національний університет ім. О. Гончара, м. Дніпро, Україна, e-mail: davydov_p@ukr.net

[2]студент групи КІ-14-2, кафедра електронних обчислювальних машин, Дніпровський національний університет ім. О. Гончара, м. Дніпро, Україна, e-mail: magro.tanya2602@gmail.com

Анотація. Стаття підсумовує досвід побудови інформаційних систем з використанням фреймового способу подання знань про архітектуру. Представлено опис семантичного ядра системи генерації компонентів, розглянуто особливості процесу підтримки різних архітектур.

Ключові слова: модельно-орієнтована архітектура, фреймовий спосіб подання знань, генерація коду, архітектура програмного забезпечення.

ON IMPROVING THE QUALITY OF THE PROCESS OF INFORMATION SYSTEM DEVELOPMENT AND MAINTENANCE

A.A. Litvinov[1], P.V. Davydov[2], T.V. Magro[3]

[1]Ph. D. in Technical Sciences, Associate Professor at the Department of Electronic Computing Machinery, Oles Honchar Dnipro National University, Dnipro, Ukraine, e-mail: lishu.dnepr.ua@gmail.com

[2]Student, Department of Electronic Computing Machinery, Oles Honchar Dnipro National University, Dnipro, Ukraine, e-mail: davydov_p@ukr.net

[2]Student, Department of Electronic Computing Machinery, Oles Honchar Dnipro National University, Dnipro, Ukraine, e-mail: magro.tanya2602@gmail.com

Abstract. The paper is devoted to summarizing the experience in the field of building information systems using model driven architecture grounded in frame-based knowledge representation paradigm. The work describes the semantic core of the system and the specific features connected to the process of managing different architectures.

Keywords: model driven architecture, frame-based knowledge representation, code generation, software architecture.

Introduction. Today, software developers face the challenge of the rapid creation and effective maintenance of complex information systems. The solution based on the quality of the development process rather than on gathering

team of super-professionals or involving new languages and technologies [1]. The process can be defined as a set of defined, structured activities that will accomplish a specific organizational goal. The improving of the process is based on applying of specific activities directed to identify, standardize and implement best practices, analysis of workflow etc. Thus, the standards become the assets of the organization, which would build a strong foundation the process improvement.

Generally, we can say that the process of information system development is a process of transformation of two sets of requirements (functional requirements and quality attributes) into a set of interacting software components, operating in the operating environment. Thus, the questions connected with the transformation are as follows: how the requirements should be defined to be effectively transformed; what the artifacts could help the developers to build the solution as quickly as it's possible; what the mechanism makes the transformation work.

Goal of work. The main goal of the work is to improve the concept discussed in [2-3] and to solve possible problems it could have, such as incompatibility of systems' units.

Materials and methods. The results of the previous research were published in [2-3]. The provided minimal description is composed of use-case scenarios which can be further transformed into the user acceptance tests based on BDD (behavior driven development) scrips and domain description which can be defined in a form of business simulation using an extended frame-based language described in detail in [3]. The language can describe domain in a full, platform independent manner, considering all necessary information for system construction and maintenance. The expressiveness of the language is based on the descriptive power of facets used by both frame and slot structures. Thus, we can think of a frame as a set of projections, some of which represent more abstract information that can be easily understood by the experts, some form more detailed view needed for generation of more complete and valuable assets. The frame described in such a way as of the complete synthesis-oriented structure able to incorporate all necessary knowledge needed to produce the solution. The model can be used as a foundation for: component generation, allowing to speed up the development process; run-time model interpretation, allowing to exclude the necessity to build and test the classes.

The variant of making an interpreter of the model at run-time seems attractive but it has two serious disadvantages: system becomes closed for changes and inflexible. While it could be effectively used to solve only typical problems but the adaptation of the run-time machine to solve an original one requires efforts and expenses. Another variant is to build a generation-oriented system able

to transform the scripts into a set of artifacts in accordance with the selected architecture (a structured solution meets technical and operational requirements) and technologies. The architecture provides a number of patterns and principles of software organization defining the way of typical tasks solution.

The set of the artifacts includes not only the set of fully or partially generated components (classes and interfaces, database scheme etc.), and unit, integration and user acceptance tests, but also guidelines and recommendations, ordered set of prioritized and estimated tasks, probable risks, sketches of iteration plans and other process-oriented information. It reduces the complexity of the tasks, making process more controllable and robust. On the other hand, developers are not restricted by the provided solution, they can improve and configure the generation system by editing the scripts and even adding new plug-ins.

The questions connected to the problems of compact description of model and the structures able to represent the model were partially discussed in previous works [3]. The goal of the presented work is to summarize the experience in the field of building different applications considering different architectures.

First of all, the description proposed in [3] should be improved by adding new structural element "slot type" responsible to define a set of slots (e.g. properties, methods, namespace).

Concept: SuperConcept facets: entity…
{
properties::
…
methods::
…
update::
…
getBy::
…
}

This improvement makes the description more structured and clear. It also allows to use standard parsers (e.g. Antlr), making the system more open and flexible. The diagram of the semantic core of the system is shown in figure 1.

Another question is connected to different architectures and their management. User should be able not only to add a module responsible for transformation of the concepts into the components of a layer easily but also to define an architecture that is a set of modules and define dependencies occurred among them.

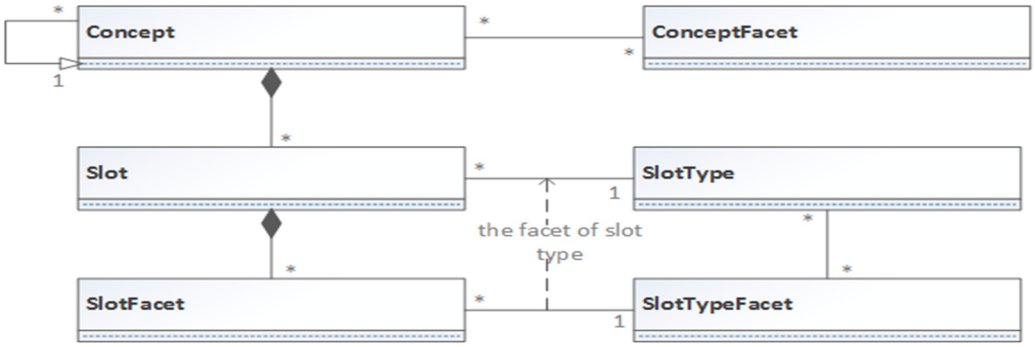

Figure 1 – Semantic core of the system

The solution of the tasks based on the conceptual diagram shown in fig.2. The architecture is regarded as a composition of units, i.e. script interpreters able to generate the artifacts. Each unit is a variation of a unit type – generic entity connected to a facet or facets of the concept and a number of slot types to be interpreted. Such separation dictated by the variations of technological solutions of the same system level problem (e.g. MSSQL and MySql database providers).

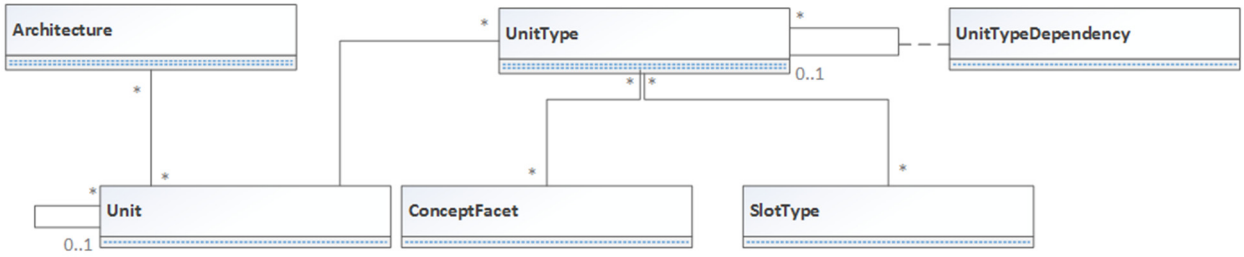

Figure 2 – Units of the architecture

Results. Thus, the proposed work provides improvements to the frame-based model language: adding "slot type" structure makes the language more structured and system more open; provided architecture management sub-system makes the system more flexible. These improvements save time and make the process more defined, predictable, stable and robust.

REFERENCES

1. Mary Beth Chrissis. CMMI® for Development Guidelines for Process Integration and Product Improvement, Addison-Wesley Professional; 3 edition (March 20, 2011). – 688 p.

2. Литвинов О.А., Грузін Д.Л., Гуреєв П.П. Особливості автоматизації розробки функцій в богатошарових інформаційних системах. // Системні технології. Регіональний міжвузівський збірник наукових праць. – Випуск 1(102). – Дніпропетровськ, 2016. – С.- 36-41.

3. Litvinov A.A. On a frame-based language used for software modelling. // System technologies. – N.1(108). – Dnepropetrovsk, 2017. – 55-63 p.