UNIVERSITY OF NOVI SAD
TECHNICAL FACULTY ”MIHAJLO PUPIN”
ZRENJANIN
REPUBLIC OF SERBIA

INTERNATIONAL CONFERENCE ON

INFORMATION TECHNOLOGY AND DEVELOPMENT OF EDUCATION ITRO 2014

PROCEEDINGS

MEĐUNARODNA KONFERENCIJA

INFORMACIONE TEHNOLOGIJE I RAZVOJ OBRAZOVANJA ITRO 2014

ZBORNIK RADOVA

ZRENJANIN, JUNE 2014

Organiser of the Conference:
University of Novi Sad, Technical faculty „Mihajlo Pupin", Zrenjanin, Republic of Serbia

Publisher:
University of Novi Sad, Technical faculty „Mihajlo Pupin", Djure Djakovica bb, Zrenjanin, Republic of Serbia

For publisher:
Milan Pavlovic, Ph. D, Professor, Dean of the Technical faculty „Mihajlo Pupin", Zrenjanin

Technical preparation and design:
Ivan Tasic, Ph. D, Assistant Professor
Dijana Karuovic, Ph. D, Assistant Professor
Marjana Pardanjac, Ph. D, Assistant Professor
Erika Eleven, M.Sc, Assistant
Dusanka Milanov

Lecturer:
Erika Tobolka, Ph. D, Professor

Printed by:
Printing office DIGINET ProStudio, Djure Jaksica street, no. 14, Zrenjanin

Circulation: **60**

ISBN: 978-86-7672-225-9

By the resolution no. 114-451-970/2014-03, Autonomous Province of Vojvodina Provincial Secretariat For Science and Technological Development donated financial means for printing this Conference Proceedings.

The Conference is supported by the Autonomous Province of Vojvodina, the City Administration of Zrenjanin, The National House of Mihajlo Pupin, Idvor and Organizing Committee for the Anniversary of the "Mihajlo Pupin year".

CIP – Каталогизација у публикацији
Библиотека Матице српске, Нови Сад

37.01:004(082)
37.02(082)

INTERNATIONAL Conference on Information Technology and Development of Education (2014 ; Zrenjanin)
Proceedings = Zbornik radova / International Conference on Information Technology and Development of Education, ITRO 2014, Zrenjanin, June 2014 = Međunarodna konferencija Informacione tehnologije i razvoj obrazovanja, ITRO 2014 ; [organiser] University of Novi Sad, Technical Faculty "Mihajlo Pupin", Zrenjanin. - Zrenjanin: Technical Faculty "Mihajlo Pupin", 2014 (Zrenjanin: Diginet ProStudio). – VII, 441 str. : ilustr. ; 30 cm

Tiraž 60. – Bibliografija uz svaki rad .

ISBN 978-86-7672-225-9
1. Technical Faculty „Mihajlo Pupin" (Zrenjanin)
а) Информациона технологија – Образовање – Зборници b)

Образовна технологија - Зборници
COBISS.SR-ID 287020807

CONTENTS

TEACHING MODULAR SOFTWARE ARCHITECTURES

V. Bashovski, S. Koceski
University Goce Delcev – Stip, Faculty of Computer Science, Stip, Macedonia
vasko.bashoski@live.com, saso.koceski@ugd.edu.mk

Abstract - The modern object-oriented software architectures directly target the discovered weaknesses and problems that affect productivity in the development process of software applications and software engineering by using a combination of several known principles and guidelines. They are aimed to increase the productivity and foster the cooperation between developers of software solutions. It is commonly accepted that these problems have roots back in the past, in the universities educational programs for future software developers and the recognition of the new modern object-oriented software architectures can be achieved by their addressing directly and on time in the educational programs. Therefore, this paper presents an efficient methodology of teaching contemporary modular software architectures.

I. INTRODUCTION

Based on the needs for development of large scale systems that implement complex business logic, well designed and adapted user interfaces and data visualization, communication with many data centers, exchanges of heterogeneous data and interaction with different types of end users, raises the usage of modern object-oriented software architectures that are commonly known as architectures by which software applications are meant “built to last” and “built for change”.

The main stream of modern oriented software architectures is using the concept of modularity, accompanied by two main principles, "single responsibility principle" and "separation of concerns", both of them defined by the scholars during 1970’s and 1980’s.

The designs of modern object-oriented software architectures mainly deals with the principles of design easily extensible and easily manageable applications [1], and has as a main objective to give directions for changing the existing and generally accepted ways of software development by using modern object-oriented software architectures that can meet the challenges of development software solutions that rise up in the beginning of the second decade of the twenty-first century [2].

The analysis of current conditions and principles for developing software that are in common use, shows that there are problems that extensively contribute to lowered productivity in the process of software development. The main problem is the lack of standardized and functional methods for unlimited usage of functional components that have already been developed and generally because the functional components are tightly coupled with other functional components within the application. Additionally, documentation of the functionalities of the software applications is difficult due to the lack of independence of the functional components as well as their too vide range. Consequently, due to this the maintenance and upgrading of the software applications in the future becomes inflexible.

The implementation of the modern object-oriented software architectures in the lifecycle of software engineering in the typical software production companies mainly depends on the cost-benefit of their implementation. On the side of the benefits, the greatest ones are the properties of the software product manufactured by using modern object-oriented software architectures, "easy to maintain" and "easy to scale" [3]. These two combined, increase the productivity of the software engineers on a large scale and leave a space for future developments of the companies and easily extend and scale their products. In addition, it must be mentioned that cooperation and interconnectivity with other solutions and companies as well as acquisition with the top market IT companies is simplified to acceptable levels and as result of that, the general managers do not need to worry about it as a business risk. On the other side of the cost-benefit, analysis stays the great effort and cost of re-education of the personnel no matter it is personnel with working experience or nearly graduated students from university.

Current situation on the market show this as a great disadvantage of the implementation for usage of modern object-oriented software architectures, and the problem need to be addressed on time and directly in the educational system by providing a fast backward connection from the industry. Therefore, this paper presents a methodology of teaching contemporary modular software architectures.

II. MODULARITY PRINCIPLE BY SCHOLARS OF 20TH CENTURY

The principle for creation of a software architecture that will target the productivity in the process of the software engineering was one of the focuses of the scholars in the past centuries. The term "separation of concerns" is firstly mentioned by Edsger W. Dijkstra's publication [4] "On the role of scientific thought", where he defines the term as a process for logically separation of the software applications into set of elementary functional parts, taking in consideration their functionalities to overlap as little as possible.

Several years later Chris Reade will reactivate the basic idea of modularity [5] and defines the principle by defining the activities that must be covered by software engineers and set the separation rule of the concerns by these activities. In the following years, the idea of software architectures that implement modularity became more popular. Robert C. Martin [6] introduced the "single responsibility principle", and as stated, it defines the elementary range of the modules with the basic rule, a module can be considered as elementary functional part of a solution only if there exits just one reason or need for future changes of the elementary module itself.

What is common in all the publications that objectives modern object-oriented software architectures in the past century are the architectural goals that must be met and as objectives, they are?

- Allow creation of software applications from modules that can be built, assembled and, optionally, deployed by independent teams.
- Minimize cross-team dependencies and allow teams to specialize in different areas, such as user interface (UI) design, business logic implementation, and infrastructure code development.
- Promote reusability across independent teams.
- Increase the quality of software applications by abstracting common services that are available to all the teams.
- Incrementally integrate new capabilities.

III. MODULARITY IN 21TH CENTURY – COMMONLY ACCEPTED PRINCLIPLE

The modern trends of object-oriented software architectures are in line with the main and widely adopted operating systems architectures such as Microsoft Windows 8 version. The foundation principles on which they are built are parallel processing, portability and by that indirectly energy efficiency of the personal devices. Modularization of the applications by newly introduced object-oriented software architectures is commonly in use on every layer, data processing, user interface adaptations, synchronization with a goal the operating system of the devices to be able to suspend as much passive functionalities of the applications to preserve energy. On the other hand, modularization allows the processing to be done in parallel by more than one processing core because the functionalities of the software application are from scratch designed to communicate in loosely coupled mode.

These trends are enforced by the industry even more as new types of parallel processing system are introduced. Optical processing as a technology of the future resistible to electro-magnetic induction lies on the pillars of future development of parallel processing design and energy consumption.

IV. MODULAR-ARCHITECTURE TEACHING SIMULATOR – SHOW-CASE

In the context of better understanding of the given concept of modularity and eased process of learning object oriented architectures, a simulator shell application for module discovery, downloading, loading, initialization and dependency tracking was developed, as shown on Figure 1.

The foundation idea of the shell application is meant to be an easy to use tool with which modules, as projects with some specific functionality, developed by different isolated groups of students can be mixed into single application. In addition to this foundation idea and as support for the modularity concept the next task that was given to the groups was for them to develop different module with the same functionality that was previously given to another group of students. Evaluation process during the course Software Engineering was made on 50 students, with equal gender and average mark distribution students were separated in two groups. The first one with 24 students, used the multimedia simulations via the simulator shell application for modules as learning tool, and the second group with 26 students took the classic program for the course Software Engineering. The group that used the learning tool was then further divided in 6 groups each counting 4 students and received the tasks as planned. At the end of the course practice labs, the group finished with 6! Equals 720 different versions of one same application, and easily got the idea of loosely coupling between modules and functionalities developed by independent teams, as support for the modern concepts for software engineering. Overall result in comparison with the group that took the classic program for the course was 8.78 average grade for the first group and 7.24 for the second one. This evaluation is showing that nearly 1.5 plus average grade was achieved by the students that were using the modular tool and is a great proof for the thesis given on the usage of modularity principle as a learning tool.

V. The Backward Connection to Education Systems

Educational systems for teaching information technology have arisen mainly from other branches of natural sciences faculties during the 20th century, and primarily from electrical engineering faculties, mechanical engineering, theoretical and applied math etc. All of these fields from with the information technology sciences have separated from are "exact" sciences where more or less focus is on objective reasoning of the problems and solutions. These traditional sciences have roots back in the philosophies of the ancient scholars such as Empedocles, Plato, and Euclid and from that point in time until present days the theories of these sciences are supported by exact experiments.

In comparison to them information technology sciences are going on fast track, meaning that what were the objectives today most probably would not be tomorrow. Neither someone can define a set of experiments by which some theory can be accepted or not, leaving the space for greater creativity and fast adaptations to the newly arisen environment. Therefor the educational system, and by that I mean mainly faculties of information technology sciences cannot easily define what is popular today, or to predict what will be the trend in future or even harder to encapsulate the needs for particular professional profiles for the IT industry.

If modern trends are in position of supporting the modular approach as a modern object-oriented software architectures that does not need to be an exact fact for the years ahead, small changes in the environment can alter upside-down the modern trends so educational system that adapted these study programs with modern object-oriented software architectures from today will fail to match the needs of future.

Providing a backward connecting from the industry is simply not enough because of the mismatch, the average period of student education is four to five year and the average roll down of the technologies is two to three years. For the experienced software engineers the process of accepting newly defined architecture is harder if they managed to specialize certain technology of the past, and that is the main problem of current educational programs that needs to be solved.

VI. Conclusion

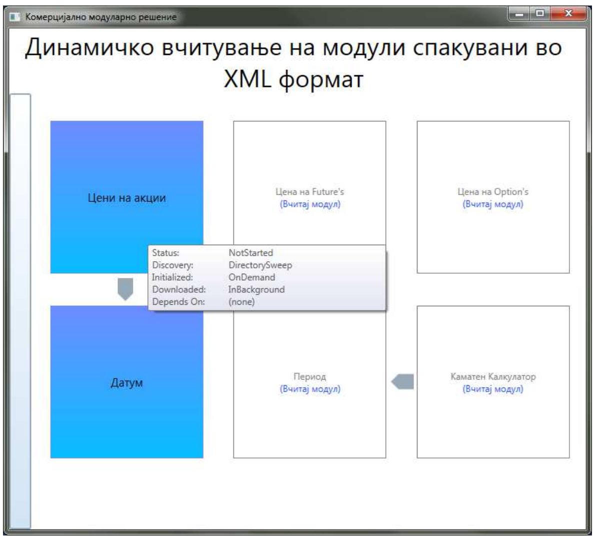

Figure 1. Module Shell Simulator

The usage of the modern object-oriented software architectures and modularity as concept in the information technology industry is inevitable due to the great cost cut advantages. The provider of the basic resource of the information technology industry, the educational systems formal or informal must comply with arisen needs of the industry. That means the educational systems does not need to address it directly in their teaching programs but they need to teach modern object-oriented software architectures by teaching with approach of simplicity. Acceptance of the modern object-oriented software architectures by the industry must be followed by acceptance of modern object-oriented learning techniques in the educational systems.

They must incept the idea of the two principles "single responsibility principle" and "separation of concerns" in the students minds just from the start and try to implement the same two principles in their teaching programs. The modern thinking and teaching of object oriented architectures must take in consideration the objective it needs to meet, fast changing requirements from the industry, maintainability, expandability, easy acceptance and adaptation of new trends as well as to give a way for future developments and subsidation of experimenting for support the theories. As already mentioned information technology science cannot be considered as a classical engineering science but more like a living organism that evolve quickly by the rules of natural selection [4] and therefor classical teaching for it cannot be implemented.

REFERENCES

[1] S. Michal, P. Píša, D. Faggioli, T. Cucinotta, F. Checconi, Z. Hanzálek, G. Lipari. "Modular software architecture for flexible reservation mechanisms on heterogeneous resources." Journal of Systems Architecture 57, no. 4 (2011): 366-382

[2] M., Herwig, J. Verelst, K. Ven. "Towards evolvable software architectures based on systems theoretic stability." Software: Practice and Experience 42, no. 1 (2012): 89-116.

[3] K., Heiko. "Sustainability evaluation of software architectures: a systematic review." In Proceedings of the joint ACM SIGSOFT conference--QoSA and ACM SIGSOFT symposium--ISARCS on Quality of software architectures--QoSA and architecting critical systems--ISARCS, pp. 3-12. ACM, 2011.

[4] E. W. Dijkstra, "On the role of scientific thought," in Selected Writings on Computing: A Personal Perspective, vol. ISBN 0–387–90652–5, New York, NY, USA, Springer-Verlag, ISBN 0–387–90652–5, 1982, p. 60–66.

[5] C. Reade, Elements Of Functional Programming, Boston, MA, USA: Addison - Wesley Longman Publishing Co. ISBN 0201129159, 1989.

[6] G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.

[7] R. C. Martin, Agile Software Development, Principles, Patterns, and Practices, Prentice Hall. ISBN 0135974445, 2002.

[8] C. Darwin, On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life, London: John Murray, 24 November 1859.