The Role of Modelling in Business Software Development: Case Study of Teaching and Industrial Practice in Zrenjanin, Serbia

Ljubica Kazi

University of Novi Sad, Technical faculty "Mihajlo Pupin"
Djure Djakovica bb, 23000 Zrenjanin, Serbia
ljubica.kazi@gmail.com

2019 29th International Conference on Computer Theory and Applications (ICCTA) | 978-1-7281-5276-9/20/$31.00 ©2019 IEEE | DOI: 10.1109/ICCTA48790.2019.9478817

***Abstract* — This paper describes the role of modelling and visualization in artefacts of software development, in the context of diversity in modelling methods, languages and tools, different software development processes and architectures. It particularly addresses issues in reverse engineering, reengineering and modelling with agile software development, which leads to the need for adaptable and adaptive modelling. This paper contributes with the description of the previously presented methodology of integrated modelling approaches in business software development and empirical research with two case studies. First case study shows experiences of modelling business-oriented software in educational process at University of Novi Sad, Technical faculty "Mihajlo Pupin", Zrenjanin, Serbia, while second case study analyses experiences of IT companies in Zrenjanin region, Serbia. Conclusion of the paper is that educational process in teaching modelling gives mental models to students to use them in industrial practice. Agile software development with constant changes in software products leave companies out of modelling or minimize their use. Adaptive and adaptable modelling and visualization is needed in industrial practice, since it brings lower software development costs.**

***Keywords* — Information system, integration, teaching, industry practice, software development artefacts, visualization, development costs.**

I. Introduction

In contemporary construction of products suitable for clients, there are two most important paradigms to be considered: agility and quality. It is important to deliver the product with appropriate quality and to do it fast. Software development process evolved from linear to iterative-incremental and adaptive. Changes in requirements are supported by the agile approach and working software became more important than comprehensive documentation.

It is very important to examine what is the role of design, modelling and visualization in the contemporary process of software development – how are they used in industrial practice and what are their roles in the teaching process at university level. Is there a need for modelling in software development at all? Do we have to continue to teach students modelling and design? Why is it important to teach them modelling? What types of models are used in real-world industrial practice and which types of models are to be taught? Relating industrial practice to teaching practice in modelling and visualization of software development is particularly addressed in this paper.

This paper is organized as follows: second section presents theoretical background with basic terms and categories, while third section describes related work as short literature review related to using modelling in different industrial situations, architectures and development processes, as well as in educational practice. Fourth section presents the first contribution of this paper with description of the methodology that integrates different methods of modelling for the business-related software development, starting with business process modelling, UML modelling and data modelling. Fifth section presents empirical research results in teaching practice of modelling, while sixth section presents empirical research results in industrial practice of modelling in business-related software development. Final section brings conclusions that describe results, limitations and relationships between the two case studies, as well as directions for future research related to adaptive modelling.

II. Theoretical Background

Visualization is often related to presenting numerical data, but it could be considered as a broader term. "Visualization is graphical presenting information (data, processes, relations, concepts), in aim to enable quality understanding of the information content by the user. It is also the process of transforming objects, concepts and data in the form that is visible and understandable to humans." [1] Some of the most popular forms to visualize numerical data are: charts, graphs, plots, maps, images, 3D surfaces, slices. [2]

Special type of visualization is used within design of software systems by creating models. A model could be defined as „graphical, mathematical (symbolic), physical, or verbal representation or simplified version of a concept, phenomenon, relationship, structure, system, or an aspect of the real world. The objectives of a model include (1) to facilitate understanding by eliminating unnecessary components, (2) to aid in decision making by simulating 'what if' scenarios, (3) to explain, control, and predict events on the basis of past observations. Since most objects are very complicated (have numerous parts) and much too complex (parts have dense interconnections) to be comprehended in their entirety, a model contains only those features that are of primary importance to the model maker's purpose.“ [3]

Models are used as tools for creating visual artifacts within software development phases such as specification of client requirements, software design, documenting the implemented solution etc. Activities in software development are often defined within methodologies, i.e. software development life cycles (SDLC). Software development methodology defines activities and the order of their application, and sometimes it also defines tools and techniques that should be applied within the methodology for particular activities [4], as well as expected results of activities, i.e. artifacts. Some of the most popular software development life-cycle models include [5]: traditional waterfall, iterative / incremental, spiral, unified process, rapid application development, agile development, [4] aspect-oriented development, adaptive software development, test-based development etc.

Throughout the history of software development, two most important paradigms are closely related to particular SDLC and modeling methods:

- Structured programming - closely related to waterfall SDLC. Modeling methods include: Structured system analysis and design methodology (hierarchical decomposition of functions, data flow diagrams, data dictionary, ER data modeling).
- Object-oriented programming - closely related to iterative-incremental/spiral/agile SDLC. Modeling methods include Unified Modeling Language and UML diagrams, which became part of international standards [6].

Particular concerns should be made in development of business-related software, which represent the important element of business organization information systems. They are designed to support business processes of an organizational system. Information system could be considered an organizational subsystem [7] and it supports business processes with information useful to an organization members and clients. [8] Information system includes all resources that are engaged in collecting, management, transfer and using information resources in organization [7].

In aim to bring some order into the set of elements, diversity of architectural patterns appeared as different approaches in elements organization. Architecture could be defined as a basic organization of a system and it consists of components, their relationships and connections with the surroundings, including the principles that rule development and evolution of the system. [9] Basic elements of an information system include [10]: information architecture, application architecture and resources architecture [7] (technology architecture, lifeware – employees that will use and maintain the system, orgware – legislation that describes the usage of the system).

According to Software Engineering Body of Knowledge ([11]), the most important aspects of software design include: functional design, data models design, user interface/experience design, as well as software architecture design.

Particularly important is an aspect of software system adjustments, that could be performed manually from humans (adaptable systems) or automatically (adaptive systems) [12]. Therefore, modelling of such systems has certain specifics.

III. Related Work – Short Literature Review

It is common to use standard UML (Unified Modelling Language) diagrams in the contemporary software development. The creation of UML was originally motivated by the need to standardize the various notational systems and approaches to software design. It was developed by Grady Booch, Ivar Jacobson and James Rumbaugh at Rational Software in 1994–1995. In 1997, UML was adopted as a standard by the Object Management Group (OMG), and has been managed by this organization ever since. UML was accepted by the International Standardization Organization (ISO) as a standard and has been periodically revised (with proposing new versions, such as UML 2.0, that includes more diagrams). While trying to use UML models for particular types of software, practitioners and researches use the available adjustment resources within UML constructs, to adapt models towards particular software architectures and technologies, such as distributed mobile software [13], agent-based software [14], context-aware software [15] and others.

Particularly useful is to design software (with the use of models) in such way that the solution will be ready for the anticipated changes in future. Changes in software solutions could be performed by using models in refactoring and reverse engineering [16], as well as reengineering. If models are available, then model-driven reverse engineering [17] could be conducted more easily and with better accuracy.

The most contemporary approach to modelling in software development is related to so-called “agile modelling” [18]. It does not introduce many new models, but it emphasizes that modelling is still used in the agile software development. According to [18], models that are still in use include data flow diagrams, conceptual data models, UML models and others. Within the process of practical adjustment of diverse agile methods to real-world usage, the need for extraction of common characteristics for all particular agile development methods appeared. Recent years brought integration of agile software development methods and approaches into Disciplined Agile Delivery framework [19]. It combines waterfall and iterative-incremental approach, i.e. emphasizes the importance of preliminary business process analysis, requirements specification, initial software design and project planning, while keeping other development activities be performed in iterations. Particularly, it addresses business process modelling as useful before the implementation iterations [20].

Separate research area is related to adaptable and adaptive modelling of software systems [21], which is contemporary in focus. Particularly important aspect is control of such systems, their evolution and practical applicability, such as [22].

Modelling in teaching practice have been analysed from the issues perspective [23], as well from the aspect of their position in computing curricula [24]. Modelling has been also analysed from the perspective of teaching style patterns and their impact on software design results [25]. Teaching patterns create certain mental models and they influence using software design patterns, which is especially useful in software industry. Special attention has been set to using agile practices [26] in teaching software development, particularly in secondary schools.

Industrial practice of using models within software development has not been much explored in published papers [27]. Only certain types of models are examined on how they are used within industrial environment, such as conceptual models [28]. Correspondence between requirements specification and software engineering models has been addressed in recent research, with particular approach to using software engineering models for the reverse engineering into software requirements specifications [29].

IV. The Methodology with Integration of Modelling Methods in Business Software Development

Integration of modelling methods is particularly important in the development of business-oriented software, i.e. business information systems. This section describes, with more detail, previously introduced methodology that integrates particular methods of modelling in business-oriented software development. The brief introduction of business software development methodology with integrated modelling methods is described in [30, 31].

A. The proposed methodology

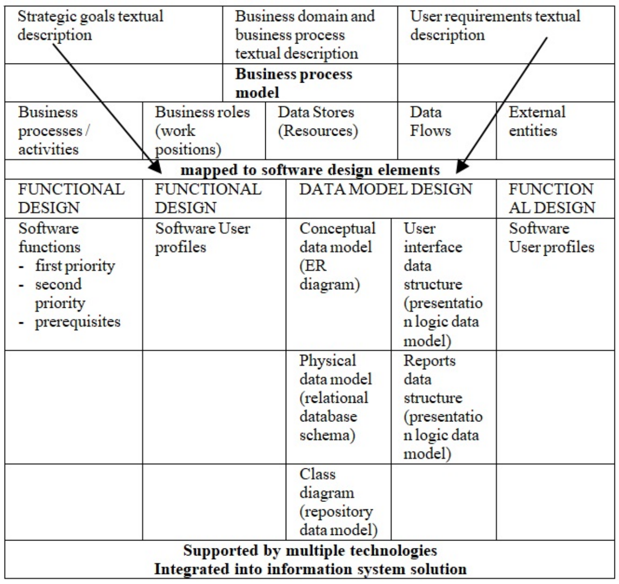
The proposed methodology has been used for many years in teaching practice at University of Novi Sad, Technical faculty "Mihajlo Pupin" Zrenjanin, Serbia. According to the proposed methodology, the modelling process starts with presentation of business process as business process model (BPM), whose elements are mapped into software design in:

- Presentation level (user interface - data elements),
- Functional level (software functions – use cases and user profiles/actors),
- Data level (data models),
- Architecture level (technology solutions that are mapped to implementation of software functions).

Previously introduced methodology has been presented with more detail at Figure 1. According to Figure 1, initial elements that are used in business software design are collected from clients and they include textual description of strategic goals, user requirements and business process. Essential knowledge of business process flow is transformed from textual description into business process model (algorithmic or hierarchical decomposition presentation).

Elements of business process model are directly mapped into certain and aspects of software design:

- Business processes/activities (i.e. primitive processes) are mapped into software functions. Each primitive process is mapped into set of software functions (first priority software functions, second priority software functions and prerequisites software functions). Each software function could be implemented with many different technologies and architectures.
- Work positions from business processes, as well as external entities are mapped into software user profiles.
- Data stores (resources) are mapped into conceptual data model, which is later transformed into physical data model (future relational database) and class diagram, representing future repository classes.



Strategic goals textual description		Business domain and business process textual description		User requirements textual description
		Business process model		
Business processes / activities	Business roles (work positions)	Data Stores (Resources)	Data Flows	External entities
mapped to software design elements				
FUNCTIONAL DESIGN	FUNCTIONAL DESIGN	DATA MODEL DESIGN		FUNCTION AL DESIGN
Software functions - first priority - second priority - prerequisites	Software User profiles	Conceptual data model (ER diagram)	User interface data structure (presentatio n logic data model)	Software User profiles
		Physical data model (relational database schema)	Reports data structure (presentatio n logic data model)	
		Class diagram (repository data model)		
Supported by multiple technologies Integrated into information system solution				

Figure 1. Integrated methodology for modelling in business software development

Integration of set of software functions, data model, user profiles and technology solutions for the software functions support represent the essence of the proposed solution of information system.

B. Discussion

The proposed integrated methodology has certain positive aspects and limitations.

Positive aspects of the proposed approach are related to:

- Direct mapping of business process elements to crucial aspects of software design – data and software functions (and their users).
- The proposed approach introduces delivery planning, i.e. setting priorities for the iterative/incremental development (delivery of software functions of first priority – directly support primitive business processes, second priority – supporting software functions, software function prerequisites – needed to support, i.e. to provide essential data, i.e. coding tables for software functions of first priority).

Limitations of the proposed approach are:

- Practical usefulness of the proposed approach could be questioned in situations where business process is planned to be only partially covered by the developed software solution and when the development team could not access all the details of the business process in the beginning of software development.

- The proposed solution is applicable with direct mapping of knowledge of business domain and process into software design, while strategic goals and user requirements could not be directly mapped into the proposed solution. They could be considered as restrictions or drivers that direct development into certain direction, after the business process mapping into basic software design elements has been done.
- Graphical aspect of user interface/ user experience elements of software design is not covered by the proposed approach, but only structural aspect, i.e. underlying data model for user interface and reports. Basic elements to software architecture planning are provided, but could not be directly mapped from the roots in business process model elements.
- Human role in mapping business process model elements into software design/data model design elements is still largely present. There are certain methods that could help in creating more complete models (and models with appropriate formal, semantic and pragmatic correctness), but integration of these methods is needed as well.

V. Empirical Research in Teaching Practice

Research related to teaching practice of modelling in business software development is focused on pre-exam assessment of knowledge in modelling business-related software. This section describes empirical case, research methodology, sample data, results and discussion related to analysis of teaching experiences and success in the application of modelling in business-related software, i.e. information system.

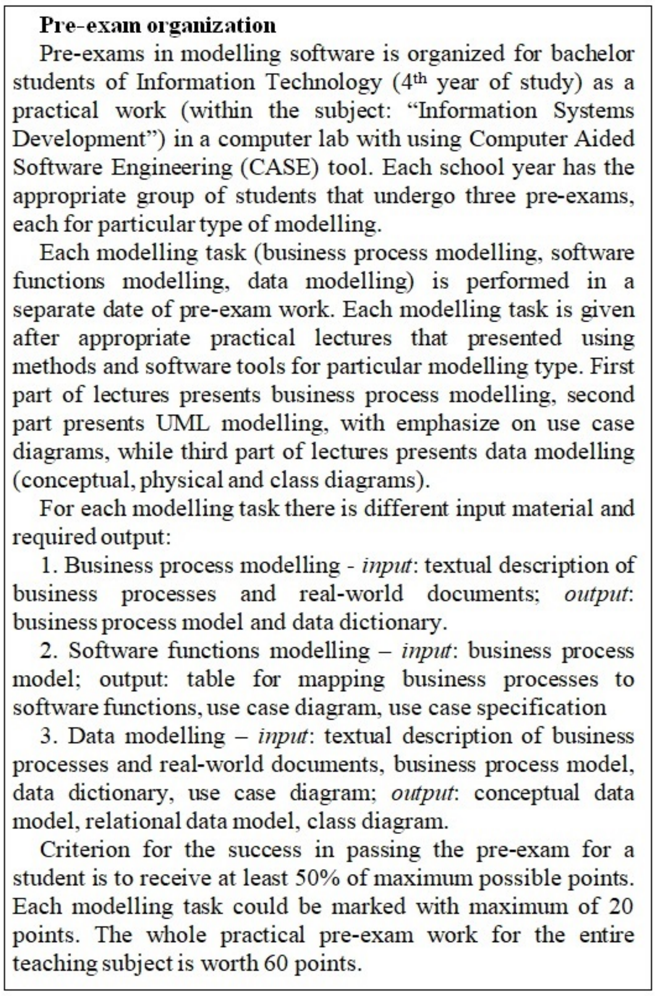
A. *Empirical case*

Empirical case presents the organization of teaching content and knowledge assessment, related to modelling, that has been conducted for many years at University of Novi Sad, Technical faculty "Mihajlo Pupin" Zrenjanin, within the subject System analysis and design (i.e. Information Systems Development). The teaching of modelling and pre-exams organization has been described at Figure 2.

B. *Research Methodology*

Research methodology related to teaching practice is defined by:

- Research questions – what types of models were used in teaching and knowledge assessment, how successful were the students in their knowledge assessment, what issues arise.
- Research methods – statistics on the marks (points) given in assessment of the students' pre-exam practical work in modelling business-related software. This empirical research is based on the particular case of teaching (that includes previously presented integrated methodology, in section IV) and assessment of students' work at University of Novi Sad, Technical faculty "Mihajlo Pupin" Zrenjanin, Serbia.



Pre-exam organization

Pre-exams in modelling software is organized for bachelor students of Information Technology (4th year of study) as a practical work (within the subject: "Information Systems Development") in a computer lab with using Computer Aided Software Engineering (CASE) tool. Each school year has the appropriate group of students that undergo three pre-exams, each for particular type of modelling.

Each modelling task (business process modelling, software functions modelling, data modelling) is performed in a separate date of pre-exam work. Each modelling task is given after appropriate practical lectures that presented using methods and software tools for particular modelling type. First part of lectures presents business process modelling, second part presents UML modelling, with emphasize on use case diagrams, while third part of lectures presents data modelling (conceptual, physical and class diagrams).

For each modelling task there is different input material and required output:

1. Business process modelling - *input*: textual description of business processes and real-world documents; *output*: business process model and data dictionary.
2. Software functions modelling – *input*: business process model; output: table for mapping business processes to software functions, use case diagram, use case specification
3. Data modelling – *input*: textual description of business processes and real-world documents, business process model, data dictionary, use case diagram; *output*: conceptual data model, relational data model, class diagram.

Criterion for the success in passing the pre-exam for a student is to receive at least 50% of maximum possible points. Each modelling task could be marked with maximum of 20 points. The whole practical pre-exam work for the entire teaching subject is worth 60 points.

Figure 2. Teaching and pre-exam organization in business-oriented software design at University of Novi Sad, Technical faculty "Mihajlo Pupin" Zrenjanin, Serbia

- Research population and sample – Population in the research are students from University of Novi Sad, Technical faculty "Mihajlo Pupin" Zrenjanin, Serbia. The research sample consists of students' work in business-oriented software modelling in the seven years period (years 2012-2019). The sample is selected since starting from school year 2012/13 there are three pre-exams in modelling (business process modelling, software functions modelling, data modelling), while in previous school years there were only two – business process modelling and data modelling.

C. *Results*

Results of empirical research in teaching practice are presented with the sample data and assessment of students' modelling work.

Sample in this research is presented by numbers of students that were working on their modelling tasks during pre-exams in computer laboratory work. Numbers of students are categorized by the type of modelling tasks that they were working on. (Table I and Figure 3).

TABLE I. SAMPLE DATA IN THE TEACHING EMPIRICAL RESEARCH

SAMPLE Number	2012/13	2013/14	2014/15	2015/16
Business process	63	64	25	50
Software functions	65	64	25	47
Data model	39	65	25	46
SAMPLE Number	**2016/17**	**2017/18**	**2018/19**	
Business process	49	43	36	
Software functions	44	34	35	
Data model	42	28	32	

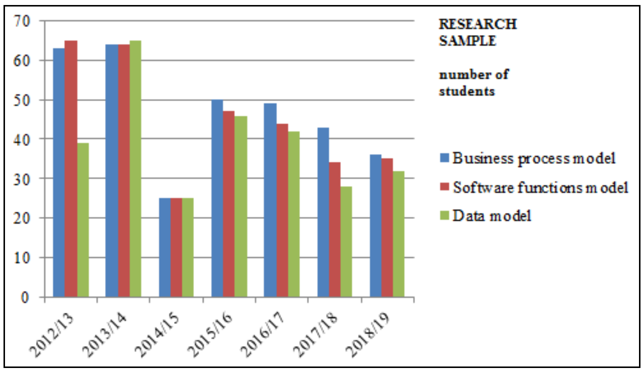


Figure 3. Diagram with number of students in teaching practice empirical research sample, by school years and types of modelling

Results of students' work assessment are presented as a percentage of students that successfully passed the pre-exam for the particular modelling type (Table II and Figure 4).

TABLE II. SUCCESS OF STUDENTS IN THE MODELLING ASSESSMENT

SUCCESS %	2012/13	2013/14	2014/15	2015/16
Business process	67	47	84	94
Software functions	65	70	96	96
Data model	100	66	96	98
SUCCESS %	**2016/17**	**2017/18**	**2018/19**	
Business process	96	74	92	
Software functions	86	85	91	
Data model	81	93	84	

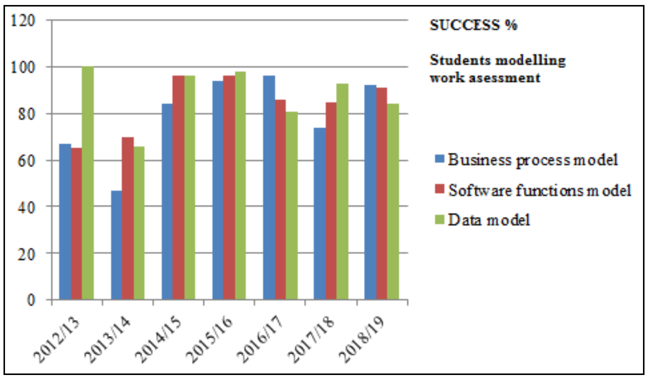


Figure 4. Diagram with students' success percentage by school years and types of modelling

D. Discussion

Analysis of previously presented sample data shows that the average number of students at pre-exams per school year is 44. The sample data show particular trend for the recent four school years at University of Novi Sad, Technical faculty "Mihajlo Pupin" Zrenjanin, Serbia. It is obvious that there is almost equal students behaviour related to pre-exams in software modelling - their presence at pre-exams decrease starting from the first pre-exam. Regularly, the first pre-exam has the largest population, while the last pre-exam always has the smallest number of students present. It could be expected, since the order of pre-exams follow the order of the lectures, starting from business process modelling, then software functions design and modelling, while the lectures and pre-exams in data modelling are scheduled for the end of winter semester. Having in mind winter weather conditions, holidays and travelling issues, these circumstances influence students' appearance at the pre-exams scheduled in that period of time.

Analysis of students' work assessment results (from the whole sample of seven school years, i.e. total number of 921 sample items, i.e. students' works) shows that the average success percentage for the whole sample from seven school years and all modelling types is 84%. Taking into account particular modelling type, business process modelling has average success percentage of 79%, while software functions modelling has success percentage of 84% and data modelling has 88% of success. This average percentage means that students show the best results in data modelling, while their success is the worst for the business process modelling. These results could be expected, since students of the 4th year of bachelor study of IT already had previous knowledge about data modelling from lectures in subject "Databases fundamentals" during their 2nd year of study. Topics in business process modelling are always new to 4th year of bachelor study students, they are not so closely related to students' previous experiences and they are abstract by nature, so it could be expected that students' could have the worst success results in this part.

Comparing particular years' results in all three types of modelling, it could not be concluded that there is some kind of trend in students' behaviour. It could only be stated that most students groups (i.e. groups of students for each school year) show the best results in data modelling than other types of modelling. Two groups (2016/17 and 2018/19) show the best results in business process modelling. Only one group of students show that their best results are in software functions modelling (from school year 2013/14). Obviously, there could not be general attitude since results change for each school year group.

VI. Empirical Research in Industrial Practice

While having first experiences in modelling during the teaching period at university, the same graduates are exposed to the industrial practice. It is important to determine how they use modelling in their professional work. This section presents empirical case, research methodology, results and discussion regarding industrial experiences in modelling.

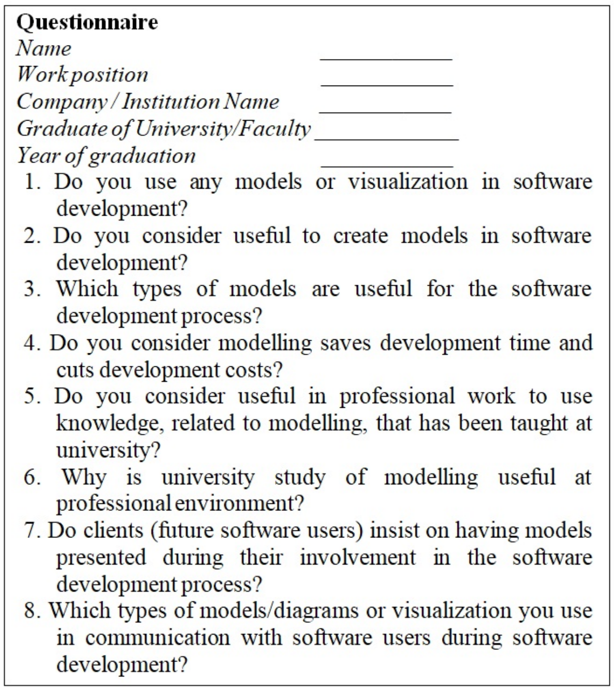
A. *Empirical case*

Research in industrial practice of using models in business-related software development is based on questionnaire given to IT professionals in the region of Zrenjanin, Serbia. The questionnaire is conducted in aim to determine the importance of using models in business-related software development.

B. *Research methodology*

Research methodology is defined by:

- Research questions – what types of models do industrial practitioners use in business software development practice, what are their attitudes regarding the role of modelling and visualization in teaching and their role applied in software industry.
- Research instrument –specially created questionnaire presented at Figure 5.



Questionnaire

Name	____________
Work position	____________
Company / Institution Name	____________
Graduate of University/Faculty	____________
Year of graduation	____________

1. Do you use any models or visualization in software development?
2. Do you consider useful to create models in software development?
3. Which types of models are useful for the software development process?
4. Do you consider modelling saves development time and cuts development costs?
5. Do you consider useful in professional work to use knowledge, related to modelling, that has been taught at university?
6. Why is university study of modelling useful at professional environment?
7. Do clients (future software users) insist on having models presented during their involvement in the software development process?
8. Which types of models/diagrams or visualization you use in communication with software users during software development?

Figure 5. Questionnaire with questions to IT professionals from region Zrenjanin, Serbia

- Research methods – statistics upon questionnaire results given to software development practitioners (from software industry and public institutions). Questionnaire is organized as a set of questions related to the practical role of modelling taught at university and the use of modelling in everyday software development practice.
- Research population – graduates in information technology bachelor studies, from University of Novi Sad, Technical faculty “Mihajlo Pupin” Zrenjanin, Serbia, that are currently employed in software companies and public institutions in the city of Zrenjanin and broader region of Serbia (city of Novi Sad and Belgrade).

C. *Sample data*

Results of empirical research in industrial practice are presented with the sample data and results from the questionnaire with software industry and public institutions’ software development professionals.

Sample data in this research are presented with capital letters or short presentation of names of software companies and institutions, as well as number of IT professionals that are employed in these companies/institutions, which responded to questionnaire (Table III). Sample consists of 11 filled questionnaires from 6 software companies and one public institution (G.U.). The full names of companies are not provided in this paper intentionally.

Table III. Sample data for the questionnaire with IT professionals

SAMPLE Company	Number of IT professionals in questionnaire
C.S.	1
G.U.	2
L.	1
S.	1
V.I.	2
W.A.	1
Y.T.	3
TOTAL:	11

It is particularly important to emphasize work positions from the sample (Table IV).

Table IV. Sample data – work positions of questionnaire respondents

SAMPLE Work position	Number of IT professionals in questionnaire
Software engineer	1
Software developer	2
System administrator	1
Programmer	4
QA developer	1
Software architect	1
Embedded Software Engineer	1
TOTAL:	11

Important aspect of the questionnaire is year of graduation, since it enables grouping of respondents regarding the acquired knowledge. Figure 6 presents number of respondents that graduated before and after year 2000.

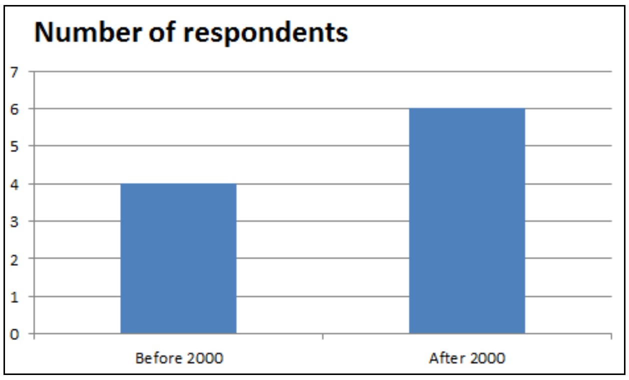


Figure 6. Diagram with number of questionnaire respondents, according to year of graduation group

D. *Sample analysis and discussion*

Considering that the main focus is in research how modelling is used in practice in software development, primarily the focus was on having filled questionnaires from software industry, i.e. companies that produce software, so public institution questionnaires should not be concerned. Still, software production and maintenance exists in public institutions, so the questionnaires from them should be kept as relevant.

To have the sample adequate, it is important to take into account the respondents' work positions. According to the sample data, it is obvious that filled questionnaire from system administrator could not be considered relevant, since this work position is focused on the hardware and operating systems, not software development. Additionally, QA developer's questionnaire could be considered adequate, since this work position closely collaborates with software developers and significantly influences software quality. Finally, only 10 of 11 filled questionnaires are processed for further analysis.

Year of graduation shows relevance to the estimation of the novelty of knowledge acquired from the university, i.e. exposure to certain types of modelling models in university teaching. For this aspect, Year of graduation is analysed and all respondents could be grouped into students that graduated before year 2000 and after year 2000. This way, questionnaire results could be related more closely and represent the group-related characteristics.

Important aspect of the questionnaire results relevance is software development-related work experience of the respondent (years of employment, types of work positions in work history etc.). Questionnaire used in this research lack in the requirement for specification of respondent work experience, which could be influential to the quality of the questionnaire data. This way, the analysis of industrial experiences made in this research could be only considered initial and more precise questionnaires are required in future research.

E. *Questionnaire - results and discussion*

This section presents results of analysing answers from the ten relevant questionnaires.

Questionnaire has two types of questions. First type of questions could be categorized as "close-type" questions that expect YES/NO as an answer. While answering the closed-type questions, some respondents gave descriptive answers as well. Second type of questions could be considered "open-type", with textual description to be entered and many useful information were provided.

In this questionnaire, answers to questions of "closed-type" were mostly YES or NO, while last question was, from some respondents, replied with "Yes and No" answer, since more precise answer depends on the particular circumstances and software project. Results of first type of questions could easily be transformed from the numeric form to graphical presentation (Figure 7).

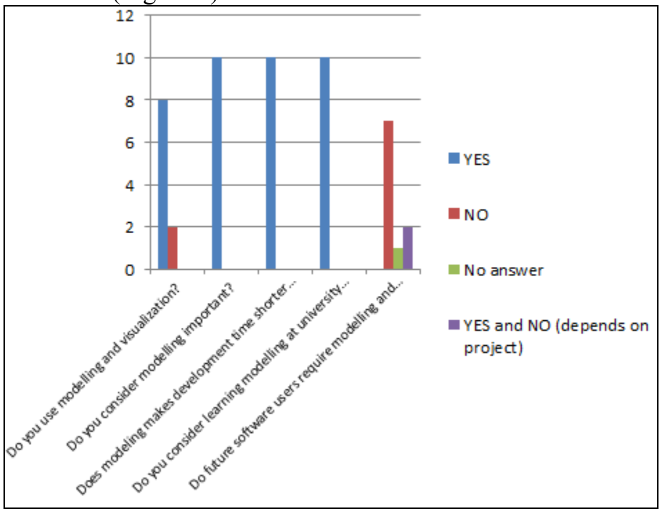


Figure 7. Diagram with number of respondents and answers for the questions with (YES/NO) answers

Figure 7 shows that 8 of 10 respondents use some kind of modelling and visualization in the software development process. All 10 respondents consider modelling important during the software development and they all have opinion that modelling saves software development time and therefore it cuts expenses. All respondents consider modelling as an important topic at university study programmes. Regarding collaboration with future software users during software development, 7 of 10 respondents replied that they did not have experiences in which users explicitly required to have any models or visualization presented regarding elements of software design, while two respondents stated that they were asked, from clients (future software users), to present them models and future software product visualization.

Descriptive part of questionnaire includes additional explanations provided together with answers to "closed-type" questions, as well as replies to "open-type" questions. It has been presented as follows.

Some respondents described that they start software development by creating models and after modelling they start coding the software solution, having models concerned. Others state that they use diagrams occasionally, if there is a need for the better problem presentation, software development planning, issues forecasting and documenting the solution. They state that using previously created models leads the development process to more quality solution and it improves productivity (with or without code generators, as well) and the developer is focused on the problem solution, not the technology, which is often pre-defined and not changeable. Still, some respondents state that having less development time by using modelling depends on the complexity of the software solution itself.

Regarding types of models, respondents state that within development team, data structures are most commonly used in modelling practice. Respondents that graduated after year 2000 mostly use UML diagrams in their software development planning - component diagram, activity diagram, use case diagram, class diagram and sequence diagram. Respondents, which graduated before year 2000, use hierarchical decomposition of business processes (data flow diagram) and ER diagrams. This result was expected, since the UML as a modelling language with set of different model types was world-wide promoted after in 2000-ties, but particularly after it has been promoted to the ISO standard. This fact speaks about the influence of university teaching to practitioners, about the need of their constant methodical and technological knowledge improvements and the role of university teaching staff in presenting latest scientific and professional results as well as standards among students and alumni.

Regarding the role of modelling at university and impact to industrial practice, most respondents concluded that learning modelling is useful and should be considered with even more attention at classes. The role of teaching modelling in business software development helps students to be introduced to problems in software development, object-oriented programming, it brings mental models to students (“teaches them how to think, understand and work”). Modelling is useful to help specify the software solution elements and enables better communication within the development team. It helps to visualize the solution and therefore helps the development by making it less abstract. Modelling teaches students “universal communication language” which is well-accepted as standard in industry environment, but “non-technical” clients also understand this visual language elements. Having models of the developed software solution is particularly useful in situation when new team member is to enter the development process, or new user is to be included in improvement iterations. Teaching modelling also helps students acquire divergent approach to thinking and development, i.e. having the same solution presented from multiple view-points and even considering multiple solutions to the same problem.

Software development practitioners that were included in questionnaire state that, during collaboration with technically educated client representatives, standard UML diagrams are used, while in cooperation with non-technical client representatives, in aim to visualize the solution, mind-maps are used. In most cases, specification of the solution that is used in communication with clients is unstructured, text-based. Client representatives generally do not require models in their collaboration with developers or IT company representatives during development process, but some respondents state that using models in collaboration with clients depends on the complexity and type of software solution to be developed. Even if clients do not require models explicitly, they react positively when exposed to visualization of solution elements. Even better response from clients comes when visualization of the future software product is combined with interaction, i.e. involving clients in their software product design adjustments. In communication with clients, some visualization tools that respondents use are Visio and Pencil.

According to respondents, models bring core of the designed solution. They could be changed during the development and improvement iterations, but they provide the stabile essence of the solution. According to respondents’ experiences, during development process, not whole solution is modelled, but only the most complex, new-to-development team or problematic parts.

VII. Conclusions and Future Work

Aim of this paper is to analyze the role of modelling in business software development, with brief theoretical and literature review as well as with empirical case studies in teaching practice and software development practice (in software industry and public institutions).

First contribution of this paper is in detailed presentation of previously introduced methodology of integrated modelling methods in business software development modelling. It has been shown that the proposed methodology is still human-dependent and provides results in the core software design, i.e. for the functional and data aspects.

Second contribution is related to results in empirical research, which addresses how modelling has been supported by university teaching and how is it used in industrial practice. Empirical research has been organized for the particular case of Zrenjanin city, Serbia, with population of students and software development practitioners that are graduates from the same faculty/university (University of Novi Sad, Technical faculty “Mihajlo Pupin” Zrenjanin, Serbia). This way, relationship between two case studies results could be drawn more easily – from teaching to the industrial practice. Therefore, the research sample has been carefully selected to enable this correlation – industrial practitioners in second empirical study were graduates from the same faculty where the teaching has been performed according to the organization from the first empirical study. The same persons were included in both case studies, first as students and later as software developers.

In this paper, teaching and learning assessment results are presented for the case of modelling that is taught at 4th year of bachelor study of IT, where students have previous knowledge of programming, data structures and databases. During teaching process and knowledge assessment, basic modelling areas include business process modelling and mapping to software design and data structures design. Knowledge assessment do not include all UML diagrams, but gives emphasize to functional part of software design. In this paper, statistics on the assessment results is performed for last seven school years (since 2012/13) and it has been shown that students express the best results in data models design. This result is influenced by the fact of their previous knowledge from databases course.

Research results of industrial practice presented in this paper have many limitations. There were only 10 relevant questionnaires filled from the Zrenjanin's IT professionals (graduates from University of Novi Sad, Technical Faculty "Mihajlo Pupin" Zrenjanin, Serbia), that are employed at software-related work positions. Number of respondents and the employees (companies), as well as geographical area of their location should also be broader, so the conclusions from the empirical research based on questionnaire could have better foundation.

Regardless all the limitation from the second case study, there are many useful conclusions that could be made. Practitioners do use modelling and visualization, but mostly they use it internally within development team and only for complex parts of software solutions. Models that are used in practice are mostly data-centred, with occasional usage of other models, such as UML models, data flow diagrams and ER diagrams. It has been shown that business process modelling and mapping to software design is not adequately used in industrial practice. UML models are used occasionally, mostly for complex parts of software solutions and upon request from technical-educated client representatives. Practitioners have positive experiences with using visualization and models in interaction with clients, during development process.

According to attitudes from software developers included in the questionnaire, it is useful to teach students modelling in software development. Since business software is mostly data-intensive, practitioners emphasize using data models in their work. Students' results in pre-exams assessment show that they have the best results in data modelling. Therefore, it could be concluded that students from University of Novi Sad, Technical faculty "Mihajlo Pupin" Zrenjanin, Serbia come well-prepared, when entering industry, regarding modelling knowledge, particularly for data-intensive software development. Other conclusions could be drawn from the experiences presented in second case study – it is very important for universities to introduce to students the contemporary results in science, technology and standards, as well as to encourage life-long learning of alumni (graduates in professional work), providing them means for the improvement of their knowledge and skills for the benefit of quality of their professional work. It is also important for universities to create strong relationships with industry to enable creative scientific-based solutions (new methods, tools…) that could enhance industrial practice and improve quality of their professional results.

In the contemporary agile world, using models during iterative-incremental process has been minimized, particularly because of the fact that software changes constantly. Modelling and visualization remain important for documenting the solution, automated software creation, teamwork and collaboration with clients.

Models present solid core of the developed solution which could be more easily transformed into other technological platforms. In the world of constant technological changes, models are used as an essence of the particular software solution, which could be changed by refactoring, reverse engineering and reengineering. In the iterative-incremental development process, the need for adaptive modelling arises. Future research results, as well as improvements in development environments could be directed towards improvements of tools for visualization and adaptive modelling in software change monitoring.

REFERENCES

[1] P. Kamran and M. Chignell, *Intelligent Database Tools and Applications: Hyperinformation access, data quality, visualization, automatic discovery*, John Wiley & Sons, Inc., 1993.

[2] Z. Kaidi, *Data visualization*, School of Computing, National University of Singapore, 2000.

[3] Definition of model, http://www.businessdictionary.com/definition/model.html

[4] L. M. Castro Souto, *On the Development Life Cycle of Distributed Functional Applications: A Case Study*, PhD Thesis, Universidade Da Coruna, 2010.

[5] J. Cadle and D. Yeates, *Project Management for Information Systems*, Pearson Prentice Hall, 5th Edition, 2008.

[6] ISO/IEC 19505:2012 Information Technology—Object Management Group Unified Modeling Language (OMG UML).

[7] R. Elmasri and S. Navathe, *Fundamentals of database systems*, Addison Wesley, 2007.

[8] D. Avison and G. Fitzgerald, *Information systems development: Methodologies, Techniques and Tools*, McGraw-Hill, 2003.

[9] ANSI/IEEE Std 1471-2000, *Recomended Practice for architectural description of Sofware-Intensive systems*, 2000.

[10] A. Vasconcelos, P Sousra and Tribolet J: *Information system Architecture Metrics: an Enterprise Engineering Evaluation Approach*, The Electronic Journal of Information Systems Evaluation, Vol. 10, Issue 1, pp 91-122, ISSN 1566-6379, 2007.

[11] *A Guide to the Software Engineering Body of Knowledge* SWEBOK v.3, IEEE Computer Society, 2014.

[12] M. Salehie, L. Tahvildari: *Self-Adaptive Software: Landscape and Research Challenges,* ACM Transactions on Autonomous and Adaptive Systems, March 2009.

[13] C. Simons, *CMP: A UML context modeling profile for mobile distributed systems*, 40th Annual Hawaii International Conference on System Sciences (HICSS'07). IEEE, 2007.

[14] B.Bauer, *UML class diagrams revisited in the context of agent-based systems*, In International Workshop on Agent-Oriented Software Engineering (pp. 101-118). Springer, Berlin, Heidelberg, May 2001.

[15] D. Ayed and Y. Berbers, *UML profile for the design of a platform-independent context-aware applications*, Proceedings of the 1st workshop on Model Driven Development for Middleware (MODDM'06). ACM, 2006.

[16] S. Tichelaar, *Modeling object-oriented software for reverse engineering and refactoring*, Doctoral dissertation, der Philosophisch-naturwissenschaftlichen Fakultät der Universität Bern, 2001.

[17] Rugaber, S., & Stirewalt, K. *Model-driven reverse engineering*. IEEE software, 21(4), 45-53, 2004.

[18] Agile modeling vebsite, http://www.agilemodeling.com/

[19] S/ W. Ambler, M. Lines, *Disciplined Agile Delivery, The Foundation for Scaling Agile*, CrossTalk, November/December 2013.

[20] Lj. Kazi, M. Ivkovic, B. Radulovic, M. Bhatt, N. Chotaliya: *The Role of Business Process Modeling in Information System Development with Disciplined Agile Delivery Approach*, ICIST 2015 5th International Conference on Information Society and Technology, Proceedings, pp. 455- 493.

[21] M. Brown, and C. Harris. *Neurofuzzy adaptive modelling and control*. Diss. University of Southampton, 1993.

[22] A. Heidari, A. K. Khandani, and D. McAvoy. *Adaptive modelling and long-range prediction of mobile fading channels*. IET communications 4.1 (2010): 39-50.

[23] R.F. Paige, F. Richard, F.AC. Polack, D. S. Kolovos, L. M. Rose, N. D.atragkas, and J. R.Williams, *Bad Modelling Teaching Practices*, In EduSymp@ MoDELS, pp. 1-12, 2014.

[24] J. Börstler, L. Kuzniarz, C. Alphonce, W. B. Sanders, and M. Smialek. *Teaching software modeling in computing curritcula*. ACM, 2012.

[25] I. Warren, *Teaching patterns and software design.* In Proceedings of the 7th Australasian conference on Computing education-Volume 42 (pp. 39-49). Australian Computer Society, Inc., 2005.

[26] P. Kastl, P., U. Kiesmüller and R. Romeike, *Starting out with projects: Experiences with agile software development in high schools*. In Proceedings of the 11th Workshop in Primary and Secondary Computing Education (pp. 60-65). ACM, 2016.

[27] S. Chandra, P. Godefroid, , and C. Palm, *Software model checking in practice: an industrial case study*. In Proceedings of the 24th International Conference on Software Engineering (pp. 431-441). ACM, 2002.

[28] H. Störrle, *How are conceptual models used in industrial software development?: A descriptive survey*. In Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering (pp. 160-169). ACM, 2017.

[29] J. Nicolás, and A. Toval. *On the generation of requirements specifications from software engineering models: A systematic literature review*. Information and Software Technology 51, no. 9 (2009): 1291-1307.

[30] Lj. Kazi, B, Radulovic, D. Radosav, M. Hbatt, N. Grmusa and N, Stiklica, *Business Process Model and Elements Of Software Design: The Mapping Approach*, International conference on Applied Internet and Information Technologies ICAIIT2012, Zrenjanin, ISBN 978-86-7672-166-5, pp.17-20, 2012.

[31] Lj. Kazi, B. Radulovic, I. Berkovic and Z. Kazi, *Integration of Conceptual Data Modeling Methods: Higher Education Experiences*, IEEE International convention on Information and Communication Technology, Electronics and Microelectronics MIPRO, Opatia, Croatia, 2014.