Novel ICT Trends that can Empower Business Again

Dražen VUKOTIĆ
Superius d.o.o., Pula
drazen.vukotic@superius.hr

Nikola TANKOVIĆ
Superius d.o.o., Pula
nikola.tankovic@superius.hr

Mario ŽAGAR
University of Zagreb Faculty of Electrical Engineering and Computing, Zagreb
mario.zagar@fer.hr

Abstract:
High availability of computational infrastructure enabled every company to use business software applications, ERP (Enterprise Resources Planning) systems in particular. ERP systems became inevitable part of business. With many positive characteristics that ERP systems provided, they also brought some negative effects. Given the way they evolved, ERP systems are highly standardized and inflexible as they provide customization only in a way that is built in design time. That is why they are becoming a lagging factor to fast and innovative business process changes which enable competitive advantage. ERP system represents a realistic image of a business system, and vice versa. This raises a possibility to change and adapt business system and its processes by adapting ERP system. In an evolved business environment that is always changing and growing, adaptations to ERP systems must follow in speed. This paper covers new technical possibilities to enable faster and efficient changes to the business systems by its end-users, without the need to involve IT departments or companies. ADBE (Application Design by Example) possibilities are specially covered as a way to create and modify business software applications

Keywords:
PbE, Programming by Example, ADBE, Application Design by Example, Data Collection and Data Estimation Methodology; JEL Code: C890

Introduction

In early 1980's the era of personal computers began which brought big changes in business management. Utilizing IT in business processes became possible for small and medium businesses, not only for large - scale companies which could afford large mainframes. Networking of small personal computers enabled collaborative work and faster information exchange. Working on-line meant unprecedented number of ways of doing business in that period like MRP and JIT managing of production and stocks. In a wave of new technical possibilities came () great new applications covering all main business processes available for variable size companies in all markets, since 1990 known as ERP (Enterprise Resource Planning).

ERP software market had great potential so IT companies started to build wide use ERP applications for unknown buyers. In a struggle to gain market share, they coded the best common practice to satisfy the needs of different buyers across several business areas. IT companies further advanced their solutions by incorporating business knowledge gained from their customers, so applications where adapted to better satisfy customer needs. After some time, a reversed process began: companies taking the role of customers started to change their business processes to accompany their software solutions. On one side, this enabled standardization of business processes and possibility of B2B[1] cooperation with positive effects on company efficiency. On other hand, companies which introduced ERP system faced quite big difficulties when they wanted to alter certain business process outside the frame that ERP system provided.

Characteristics of business applications development and implementation

Development of business applications began in parallel with computerization of business processes specific for certain business area on one side, and computerization of common processes used by majority of companies on the other. By combining these two paths, ERP systems for specific businesses evolved. In the beginning, applications were developed by own IT departments or ordered from outer IT companies. With time, ERP systems as product or end solution, became available which could more or less be configured to satisfy specific needs. It is a common case to have also a combination of these different solutions depending on size and abilities of businesses own IT departments. Custom development means having a tailored application, and buying an ERP product means having best practices for certain industry implemented already. Figure 1 displays the difference between custom tailored classically developed application and end-product application across five different aspects: (1) process coverage, (2) speed of development, (3) speed of implementation, (4) economic factor, and (5) ease of modification. Level of compliance for each category is an approximation based on 20 years of author's experience.

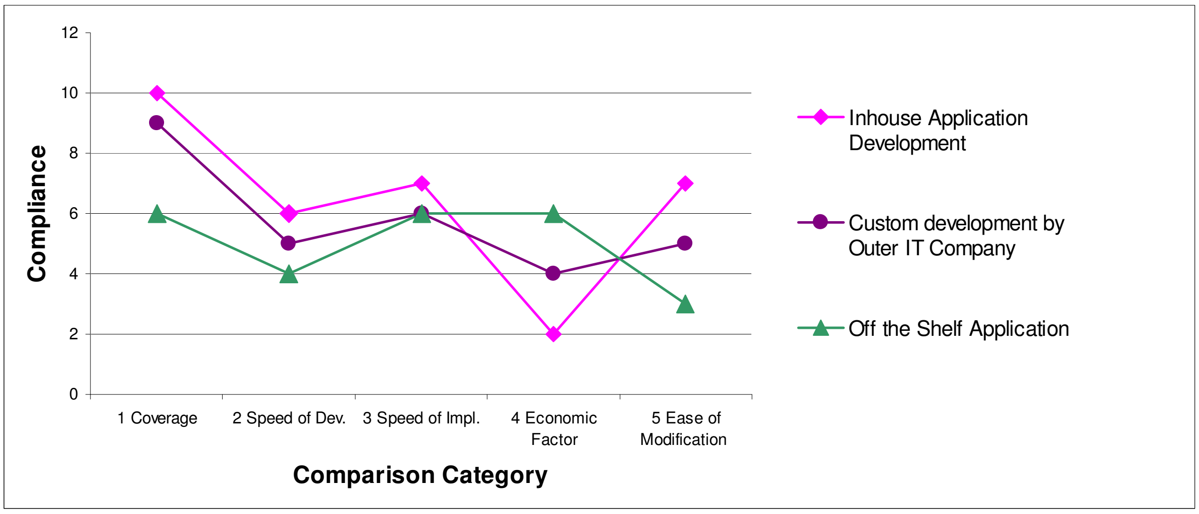

Figure 1: Comparison of custom design and end solution

It can be observed that all three types of development have similar characteristics, except in level of customizability and price, which vary significantly.

[1] **B2B - Business-to-business**, commerce transactions between businesses

The role of ERP in business

Main positive characteristics of ERP system are: computerization of all important aspects of business, great increase in efficiency of many processes, having knowledge about best business practices and enabling better B2B cooperation by introducing standardization of business processes. During development, ERP systems adapted quite well to specific business areas and according to [1] and [2], they represent a realistic image of business system covering most business functions. The other way around also stands: by changing the functionality of ERP it is possible to change business processes, especially in those parts where processes are taking place only by information exchange (B2B, B2C[2], C2B[3] and G2C or C2G[4]).

Classical approach to ERP application support

Classical ERP systems are complex, they often possess thousands of relations and tens of thousands lines of code. They are also developed in long cycles, often in even a decade, involving many people, some of which have also changed working positions. This is way ERP modules are poorly integrated, with documentation missing, being incomplete or too extensive. Business processes are commonly described in programming code and hard to modify. To solve such problems, two approaches are used: more efficient development, and parameterization. In first approach, in-house IT departments are working to perfect earlier developed modules, following new technologies and educating themselves. They are also developing new modules and exploring existing solutions. In contrast, outside IT companies are working closely with their customers and are implementing their requirements promptly. They are also following new technologies, educating own IT experts and suggesting new technological changes. Such IT companies incorporate vast customer experience in their products and optimize them accordingly. They are guided with majority of their customers, law changes, and technological trends. They often use parameterization in their products to adapt to specific customer needs. Such parameterization enables managing certain functionalities, switching them on or off, or changing their behavior.

In first approach, in-house IT department is stuffed with providing technical support of earlier developed modules and "locked" with using technologies used so far. Every new development represents a new risk for them threatening the stability of an existing system. Outside IT companies that develop custom tailored solutions are trying to unify their solutions for different users to lessen the burden of support, but that approach is making further changes quite difficult to achieve. By switching to new technologies they must also support older functionalities for existing customers. IT companies that develop for wider markets often implement larger amount of functionalities then is commonly needed for their customers with solutions being too restrictive [3]. Parameterization of their systems is so complex that it can only be achieved by experienced consultant. Such solutions are commonly too complex for average users with rules that are forced upon their business system. Business is shaped according to these complex solutions that are often not optimal for the tasks at hand.

[2] **B2C - business-to-consumer**, electronic commerce, e.g. online retailing

[3] **C2B - Consumer-to-business** is a business model in which consumers create value, and firms consume this value

[4] **G2C - Government-to-Citizen** is the communication link between a government and private individuals and vice versa (C2G)

Additions and changes of systems are complex, long-lasting and expensive, with most commonly implemented late or not consistently in all segments. That is why innovative and specific business processes are hard to enclose into existing ERP solutions which in final lower concurrent advantage of their users. That is why we suggest using new technological and programming options which will enable their users manually changing and adapting their applications without the need for specialized knowledge of application development, databases and servers with goal of efficient and quick change of their business processes.

New technological trends

In the past 10 years of IT industry has been developed many IT achievements that push the limits of applicability in IT. Main characteristics of these technologies are that they are mature, inexpensive and available and that there are strong interdependencies of business and consumer solutions. For this reason, there is a decrease of users refusing the IT solutions, which is one of the prerequisites of the proposed concept. From new IT trends we emphasize (1) ubiquitous computing, (2) wireless communication, (3) cloud computing, (4) dynamic DBMS, (5) application developing without coding and (6) interoperability. Each of these six trends is necessary to achieve an efficient modeling of business processes through changes in business information systems. The first three technologies are important to ensure that every employee (or other business entity), regardless of location and type of work could be equipped with appropriate computer device and business applications, that can exchange information on-line with the central office and other employees (or other business entities). The following two technologies are important to enable simple and effective changes in business applications, and consequently the business models that will propagate to all users of business applications. The latest technology allows an increasing number of business processes between different business entities to be conducted virtually, and thus further strengthen the ability to change business processes through changes in business information systems. In the following paragraphs we briefly describe these trends and characteristics that are important to achieve this goal.

Ubiquitous computing

During the development of IT industry, the ratio of the number of users to the number of computers were changed as follows: in the beginning one computer served more than one user, then in the PC era one computer served one user, while today one user has many different computers at their disposal. Availability of computer equipment went from a few mainframe computers available exclusively in specialized computer centers, over desktop computers available at home or in the office, laptop computers available mostly during sedentary activities to handheld and smartphone computers available in any situation, even when the user is mobile. In the beginning their use was exclusively dedicated to the business or to the science. Later on, computers started to be used for private purposes and for entertainment, while today various purposes (private, business and fun) are firmly interwoven. Computers used to be focused exclusively on a particular task, while today they are multifunctional devices focused on the user. An important feature of modern day computers is the fact that they are equipped with various kinds of sensors (touch screen, accelerometer, camera, GPS, photometer…) which substantially expand the possibilities of application.

Wireless communication

The capabilities of wireless communication today are characterized by increasing coverage of the population, the globe, micro locations which is heading toward absolute coverage. Transfer speeds are increasing all the time. The level of reliability and service availability is very high, while prices have a downward trend due to relatively competitive market.

Cloud computing

Cloud computing has become a mature technology and a common part of IT infrastructure. It has gained the trust of both private and business users. The concept of cloud computing popularizes a model of publishing and accessing data as opposed to the old model of gathering and distributing data. As a result, new usage models arise, based on the collaboration of different types of users. Usage models such as SaaS[5], PaaS[6] and IaaS[7] are available. What they all have in common is the capability of upsizing and downsizing depending on the current needs of the user. New business models are also developed, where cloud computing represents a platform for creating markets of data and flow of information.

Dynamic DBMS

Due to their characteristics such as reliability in carrying out transactions, ACID properties, maintaining referential integrity, SQL[8] data access etc., relational databases have become almost unparalleled in creating data oriented business applications. An additional complication for the developers emerges when trying to achieve the M2-M1-M0 approach to metamodelling data, which allows for a dynamic modification of the data model after the business application has already entered production phase. There are entire programming environments [4] that allow that approach but they are impractical for a non-technical user. The problem is that the metamodel of the relational database is based on fundamental data units consisting of entities and relations. Entities are realized through a tabular structure, while relations are realized through referential integrity. This structure is implicitly built into the database engine which is accessed using SQL DDL[9] standard commands. The data model is created in the design phase of the application so it is difficult to change it during the production phase by, for instance, moving a column from one table to another table or break it away into a separate entity. On the other hand, non-relational databases like Key-Value or Graph databases have a more atomized structure than relational databases. The fundamental data unit in a Key-Value database is a Key-Value pair that corresponds to the column in the relational table. In the graph database, the fundamental data unit is a node, which corresponds to a cell in the graph database.

[5] **SaaS - Software as a service** is a software delivery model in which software and associated data are centrally hosted on the cloud.

[6] **PaaS - Platform as a service** is a category of cloud computing services that provide a computing platform (operating system, database, Language is a programming language designed for managing data in relational database management systems (RDBMS).web server etc.) as a service.

[7] **IaaS - Infrastructure as a service** provides hardware (servers, storage, network etc.) as a service.

[8] **SQL - Structured Query Language** is a programming language designed for managing data in relational database management systems (RDBMS).

[9] **DDL – Data Definition Language** - subset of Structured Query Language (SQL) for creating tables and constraints

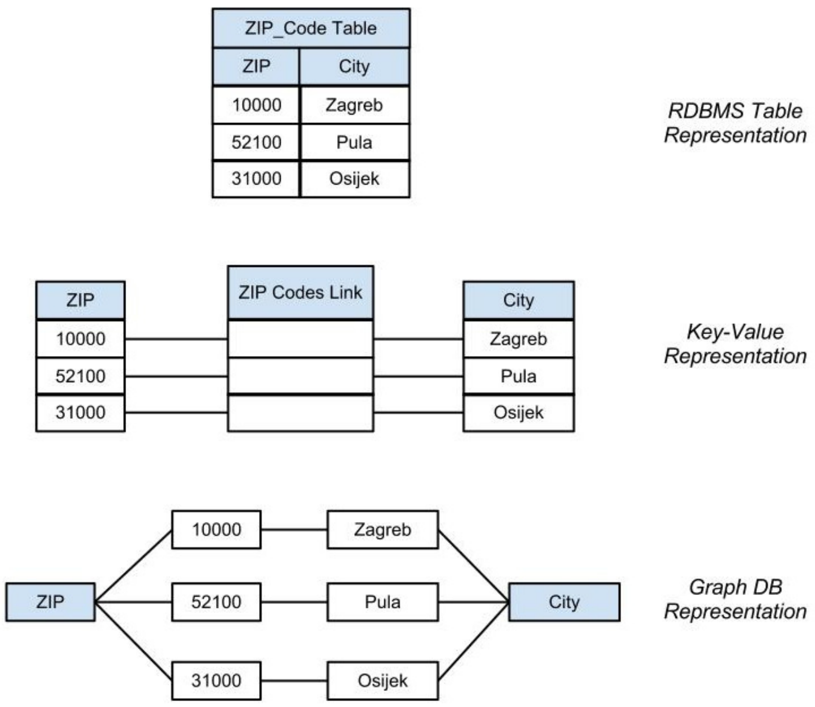

Figure 2: Key-Value and graph structures are much more flexible and suitable for a dynamic modification

Relational structure groups data by type and meaning (table and a column in a table), Key-Value structure groups them by meaning, while graph structure views all data equally, where their grouping can depend on the content of an individual node and its relations to other nodes. That's why Key-Value and Graph structures are much more flexible and suitable for a dynamic modification of metamodel and data models in the same way that a modification in production data is possible as shown on Figure 2.

Codeless application development

Programming through coding is the usual approach in computer program developing. There are more than 8000 registered programming languages. However, programming through coding has some drawbacks that prevent a wider circle of users from developing computer applications. The procedure of programming through coding is too abstract, demands a lot of specialized knowledge and experience, which is why there is a chronic lack of programmers on the market. There is insufficient overlap of knowledge between the buyer and the supplier of the programming product, which often leads to developing a programming product that is not in accordance with the user's wishes and needs. Programming is an iterative procedure involving many participants which further slows down and complicates the whole process, while the product requires comprehensive and thorough testing. Business knowledge built into the application is hidden in the programming code, which poses an additional problem during later modifications of the application. As a result of all the problems listed above, a process of developing applications without coding (which eliminates one or more drawbacks arising from programming through coding) is being increasingly explored and advanced. However, developing applications with no coding has still not reached a phase in which it would be simple and intuitive enough to be usable by non-programmer users.

Interoperability

Using computer applications in business is so common that many business processes have become completely virtual. For example, a large part of business processes in taking orders, sales, payment, human resources etc. no longer have any material traces except for records in computer applications. Standardization of formats and protocols makes it possible to implement these virtual processes not only within a company, but also between various companies. In order to achieve this, formal and industrial standards have been developed, which greatly expand the realm of possibility for B2B, B2C, C2B and B2G cooperation. These possibilities create new opportunities and new business models for companies, while posing a challenge for application developers in finding a way to, on the one hand, ensure standardization of formats to enable data exchange and, on the other hand, to ensure sufficient flexibility for later modifications and expansion. Apart for technical issues, this segment raises new legal, ethical and moral issues regarding the allowed degree and purpose of sharing of available data.

Codeless developing tools

In the previous chapter we talked about already existing technologies that make the concept of changing business processes through modification of business applications that support or enable these processes possible. The technology that is currently not sufficiently developed is the technology for codeless developing. This technology is key factor for achieving speed and efficiency in the process of developing or modifying business applications, especially because it would enable end users to make those modifications. There are tools on the market that make codeless developing of applications possible. According to research [11], these tools can be divided into five basic categories: (1) visual programming and parameterization tools, (2) modeling tools, (3) ontology tools (4) composing tools and (5) PbE tools (Programming by example). Next we will briefly describe the elementary characteristics of each of the listed categories.

Visual programming and parameterization tools

Basic characteristic of this group of tools is using visual techniques for creating the program logic and interface. This group can be divided into three subcategories: (1) tools for graphical editing of the user interface, (2) tools for defining data and processes through special diagrams and (3) visual programming tools. Even though tools of this category make the creation of complex applications without coding possible, their drawback is the fact that users have to have knowledge of programming principles and database structure.

Modeling tools

Tools from this group describe the business process through special types of diagrams, with UML[10], BPMN[11] or OPM[12] used most frequently. Based on these models, an application is

[10] **UML - Unified Modeling Language** is an object modeling and specification language used in software engineering

generated through a generative or an interpretative approach. Generative approach consists of creating a programming code in a higher-level programming language and compiling it into the executive code of the application. Interpretative approach is based on the existence of a run-time engine that generates and delivers requested program elements. This method enables automatic generation or modification of program specifications on demand, but has a somewhat poorer performance than generative approach. The drawback of this approach is an additional level of abstraction introduced into the modeling process by introducing graphical symbols and syntax that describe the observed business process.

Ontology tools

Ontology in IT terminology represents a model of knowledge that describes the observed area (domain) as a group of concepts and relations between them. Two main groups of ontology can be defined: domain-specific ontology and foundation (or upper) ontology. Domain-specific ontology describes concepts from various specific domains, whereas foundation ontology describes common concepts. The focus of the tools from this group is to describe a business process. This is achieved by way of an application user interface adapted for this task. User enters his knowledge of the business process into the application by describing objects participating in the business process and the relations between them. The same interface gives him the ability to input and edit instances of objects created in the previously described way. The drawback of these tools stems from the fact that external add-ins programmed in the old-fashioned way are required for dealing with more complex processes.

Composing tools

Studies [2] show that every business IS consists of a finite group of business functions amounting to 50.000 different business functions in all types of business. Differences between business types are reflected in less than 20% of business functions. Therefore, IS representation of every common business function has to implement 80% of standardized functionalities and has to be able to support up to 20% of specificities. If program modules supporting every single one of the 50.000 different business functions were available on the market, the user could assemble a complex information system using only the modules he needs for his work. Authors [5] start from the assumption that every module must have its representation through a user interface, regardless of whether we are dealing with a manual or an automatic process, and suggest a procedure of simple creation of complex applications using the method of composing the application from modules that support basic business functions. The drawback lies in the fact that supporting 20% of specificities is much more challenging than supporting 80% standard functionalities and the 20% specificities is what makes the difference between a satisfied and an unsatisfied user. Therefore, it is necessary to provide a procedure that would allow for a simple modification of existing or creation of new modules designed for specific business functions.

PbE Tools

[11] **BPMN - Business Process Model and Notation** is a graphical representation for specifying business processes in abusiness process model.

[12] **OPM - Object Process Methodology** is an approach to designing information systems by depicting them using object models and process models.

Even though here are different applications of the PbE principle, the main attributes of a tool that enables PbE are: (1) using the same user interface both for working on the application and for programming the application and (2) programming consists of teaching the program system by demonstrating by example what is required. Programming the user interface is done by pointing out the interface element the user wants to put onto the application user interface via a Drag&Drop or a Point&Click method. Advanced PbE tools independently make conclusions of the nature of the data entered and any rules to be applied to this data, based on a small amount of data entered in the learning process. For example, if a new field is added into the application, the type of the field can be explicitly set (text, number, date), or the system will independently suggest that this is a date field after a date value has been entered into it. Afterwards, the system will require that only date values be entered into this field. Programming of a process is carried out by giving examples as to how a process should be done. A similar case is with spreadsheets by entering mathematical, logical or other operation whose result is shown in a particular cell. Another method that allows for iteration is based on collecting user's actions and storing them for executing at a user request, but with data which are relevant at the moment of request. The most well known example of this technique is macro-programming, frequently used in spreadsheets. Drawbacks of PbE tools stem from the fact that they are focused on creating a user interface and memorizing user's actions, while too little attention is given to modeling and storing data. In certain cases, when there isn't enough information, the PbE system must ask the user to solve the dilemma by entering the appropriate parameters, which is sometimes non-intuitive or too complex.

Application Design by Example

Classical approach of creating a computer program consists of several common phases such as defining business processes, database modeling, implementing user interface and program logic, creating basic application modules, and connecting them into one functional unit. In the previous section, we describe the tools that enable development of applications without coding, but these tools focus only on some of these phases. Thus, the tools for visual programming and parameterization are focused only on making the user interface and program logic, modeling tools are focused only on defining the business processes and databases, tools for defining domain ontology are focused only on the description of business processes, and tools for composing solve only the problem of connecting different modules in the functional unit. With such a partial approach, these tools can not fully achieve the basic goal - that the average user can create new or modify an existing application.

The paper [6] highlights two key factors that such a tool must meet. First, do not introduce new levels of abstraction. Most users will easily describe their business processes, but will have difficulty in abstract modeling. For example, the process of taking customer orders, users usually describe in a way similar to this: "When I take an order, I record the name of the customer, delivery date and place of delivery, and then I write the items, quantities and discounts." The average user will seldom describe the process in a structured and generalized manner suitable for the computer program implementation as outlined in the following example: "Order is the process of creating a document that has a header and one or more items. The header contains information about the customer, date and place of delivery. The items contain information about the ID and name of the item, quantity, and discounts. Product and quantity are mandatory, and the discount is not mandatory."

Second, it is extremely important to use only the familiar user interface. Most users are very familiar with the business applications they use in their daily work but have difficulty in getting used to new programs. Therefore, special tools designed to create applications without programming make a large barrier to their acceptance. In addition, the result of programming can be seen only after the "compile and deploy" procedure. The concepts that are important for modeling databases such as foreign keys, referential integrity, relationships, etc. are IT terms that average users are not interested. Research [7] shows that users easily express themselves using the familiar user interface elements. They want to say: "At this point I want to have information about due debt in the same form as total debt is presented." It is therefore important to find a way that will enable to easily add, delete or modify user interface elements, and all other activities related to the technical work needed to achieve this task, the system should process automatically in the background.

For all these reasons, we propose a new approach to application development - ADBE (Application Design by Example), which takes the best of each method described above and combines them into a single, comprehensive new method.

The technological structure of ADBE implementation

Technologically speaking, ADBE implementation is a client-server application based on HTML5 and JavaScript technology on the client side and based on the Java technology on the server side. Neo4J graph database is used for storage purposes on the server side. Server side is powered by a server in the cloud and application is used by the SaaS principle. It is a multiplatform solution, optimized to work with touch screen and can be used on a smartphone, tablet or PC computers. The solution is unique because it allows you to create or modify applications directly on the smartphone, which is also one of the key prerequisites for enabling a rapid change in business processes through changes in business application.

The logical structure of ADBE implementation

Logically speaking, ADBE solution is interpretative application model based on three-layer metamodel-model-data (M2-M1-M0) architecture [8]. M2 model implements OPM [9] structure that contains object description of all modeling elements. M1 contains model of all parts of the application (data structure, user interface, program logic) which are modeled using M2 elements. The system can support several applications at the same time. M0 contains information created by using of these applications.

As previously mentioned, graph database stores elementary information in the form of nodes and links. This type of storage makes it easy to inherit objects between M2-M1-M0 layer models. The model is therefore autoreflective because all components of the model are described by components that are also part of the model. This feature is extremely important to enable the PbE process. Figure 3 shows the concept how the application can be created or changed. Modeler is a modeling software extension that uses information from the fixed M2 metamodel and allows reading and changing elements of the M1 application model, and thus the behavior of the application.

Client application reads the M1 model and uses it to generate the User Application user interface and the starting point process. The application then retrieves data from the M0 layer needed to conduct next process, also written in the M1 model. If the Data model of the M1 layer is changed by Modeler Extension, this change is automatically reflected in the M0 layer. Due to the atomicity property of the graph database, M1 data model can be arbitrary changed over the data already stored in the M0 layer.

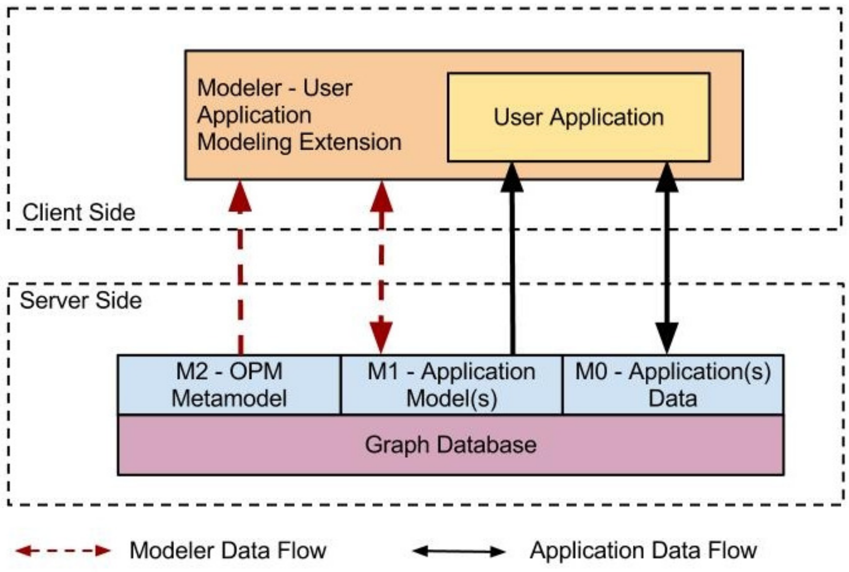

Figure 3: The logical structure of ADBE implementation

Example of application creating by ADBE

Below we will show how to create an application using ADBE method. The application allows you to create short custom message (Twiit). For ease of understanding, let us first consider an example of serialization of M1 application model and corresponding instances created in M0 model. M2 layer metamodel contains OPM structure, and Figure 4 shows M2 model of objects and relationships that have source and destination. Using Modeler we created two objects: message – Twiit object and its sender – User object.

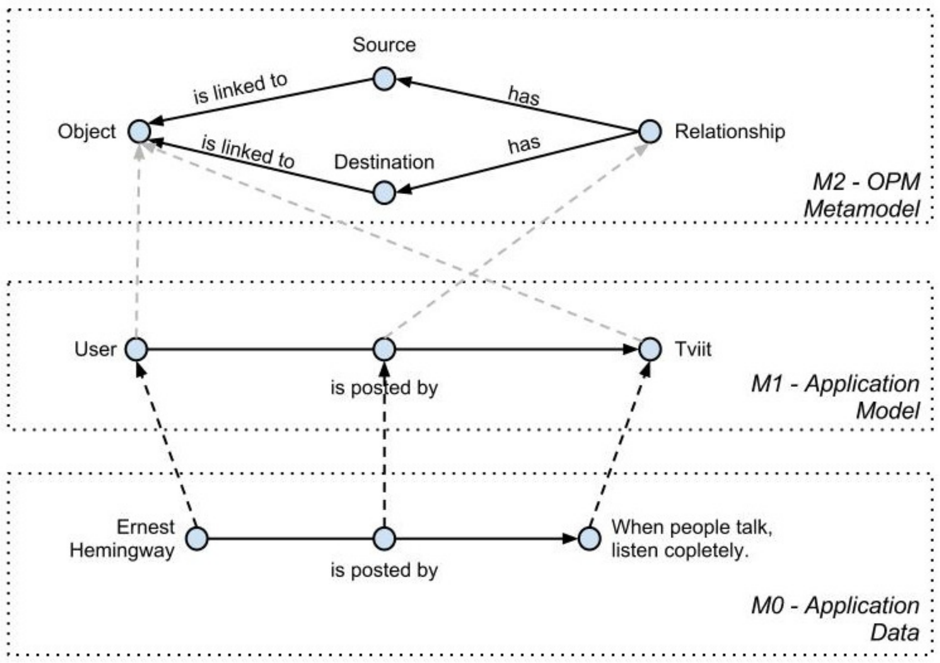

Figure 4: M2-M1-M0 layers of Twiit example application

Using this application model, we launched the application that created the Twiit "When people talk, listen completely". The Twiit is sent by the User with the name "Ernest Hemingway".

Figure 5 shows the complete process of generating an application using ADBE method and visual editing of user interface using the Point & Change method. At the same time the user interface is created, automatically is created the ontology of the business process and model of objects and processes too. Table 1 gives step by step description of the ADBE process.

Table 1: Description of ADBE example

Nr.	Procedure description
1.	The process begins by logging into the system. The system recognizes users 'Ernest Hemingway', but also determined that for him there is no application. Therefore, automatically opens the M1 node to create a new application and start the application in design mode.
2.	The system has showed an application form that has several standard elements of user interface that expect the entry of its content or removal from the application model (M1). These elements are (1) the application name (<Enter application name>, (2) exit button (<logout>) and a company logo <Insert logo or company name>. With Tap & Hold method user shows to the system what element he wants to change.
3.	After entering the required information, the user can, without leaving the design mode, to test the application by clicking on the button "Logout". However, the user decides that he wants to add a new element to the application by pressing <Insert element>.
4.	Based on the context of the previous step, the system decides which elements of the interface can offer for selection. The user selects the option "Document".
5.	The system displays the basic elements of the interface and automatically enters data into them from the current context: User name and time. Initial elements of the interface are described in the M1 layer. These are: (1) <back button>, (2) <document Title>, (3) <home button>, (4) <Current user>, (5) <Current date and time>, (6) <Cancel > and (7) <Apply>. The user changes the labels for <back button>, <document Title> and <home button>.
6.	Interface elements from the M1 document model are given new label values that are written to the M0 model: <back button> the value "Go back", <document Title> the value "Twiit" and <home button> value of the "Home". Thereafter, the user indicates that he wants to add a new element to the interface by pressing <Insert New element>.
7.	Based on the context of the previous step, the system concludes what elements of the interface can offer at this time. The user chooses the option "Text Field" and by selecting its properties explicitly instructs the system that this is mandatory multirecord text field.
8.	The system adds a new node in the M1 model and automatically displays the modified document. The user indicates that he wants to add another element of the interface by pressing the <Insert new element>.
9.	The user repeats the process of selecting a new element of the interface, this time it's a dropdown list. He entered the name of the label: "Message Priority". In the background, system automatically creates a new data type and M1 structure which describes it.
10.	Dropdown list is added to the interface. By pressing the "Home" button system returns to the Home screen of the application.
11.	The application is ready for use. The user selects the option "Message Priority" and enters an element called "Normal", then select "Twiit".
12.	Based on the context, system decides that it should display the M1 node that lists all the documents of this type, with the possibility of adding a new document. The user changes the <new screen Title> in “Twiit search” then presses the button "Add twiit".
13.	The system opens the document "Twiit" and automatically fills in the fields "User" and "Time" with the values of context. The user enters messages text and "Priority" and presses the button "Twiit it!" which publishes the message.
14.	The system returns a list of all twiits. At the top of the list shows the most recently added twiit. The user selects the option "go back"
15.	Once the user selects the Modeler option <Apply>, the application returns to the user mode and is ready to share and use.

In the example it should be noted that to navigate through the application we use only the elements of the application itself.

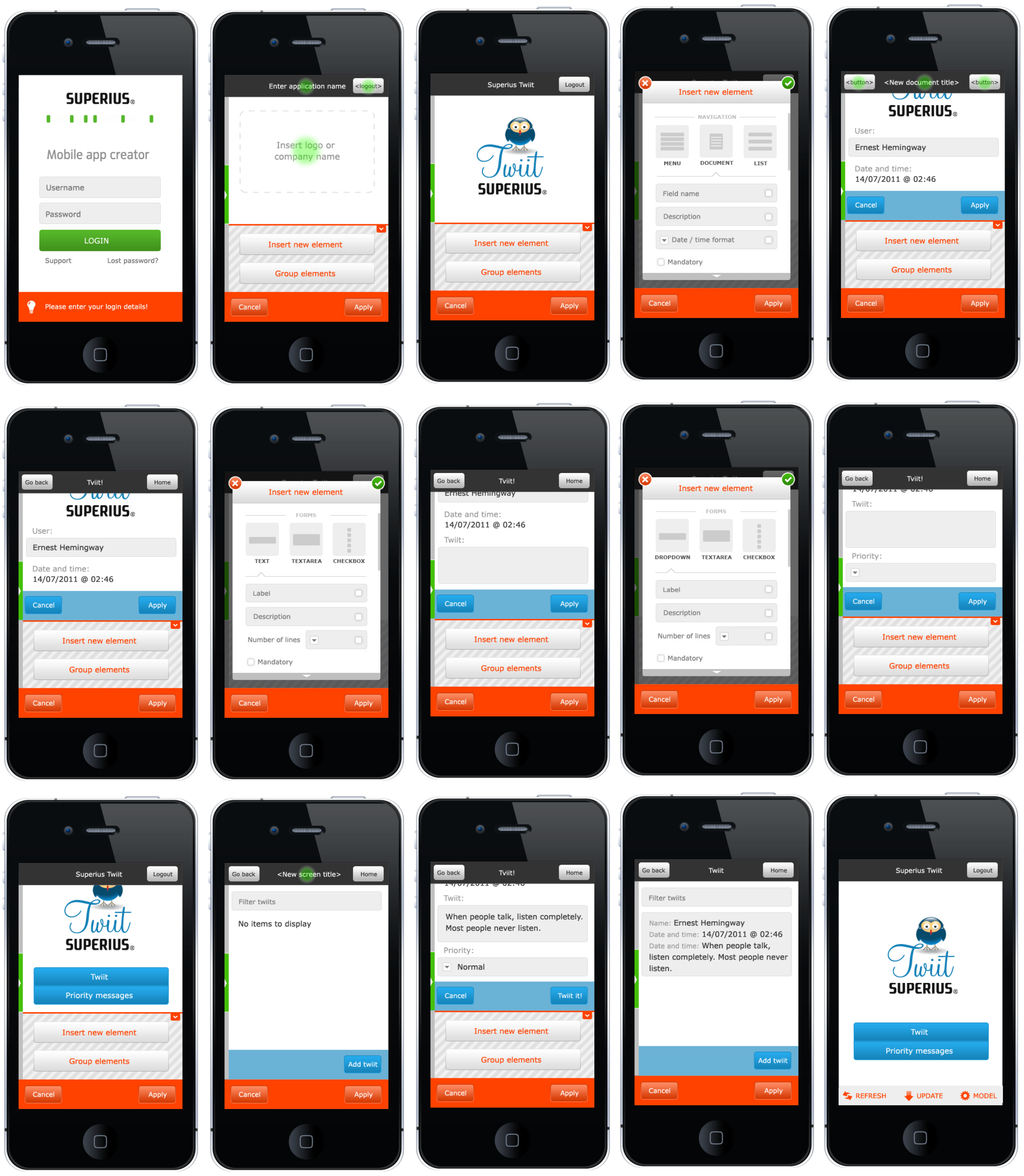

Figure 5: The complete process of generating an application using ADBE method

Composition of elementary ADBE solutions

Atomicity of the graph database provides another important property of ADBE solutions - data sharing. While the M2 model is common to the whole system, in the M1 layer can be stored a lot of elementary applications. These applications can be grouped into more complex application in K1 layer. K1 layer applications then can be grouped together much more complex applications in layer K2, etc. to the arbitrary level of complexity marked „m“. Figure 6 shows how to create a new application by using composing method. It should be noted that because atomicity of the graph database, data on the M0 layer may also be composed into the new group, which method would be equivalent to creating a view in the relational model, but with the additional possibility of changing data.

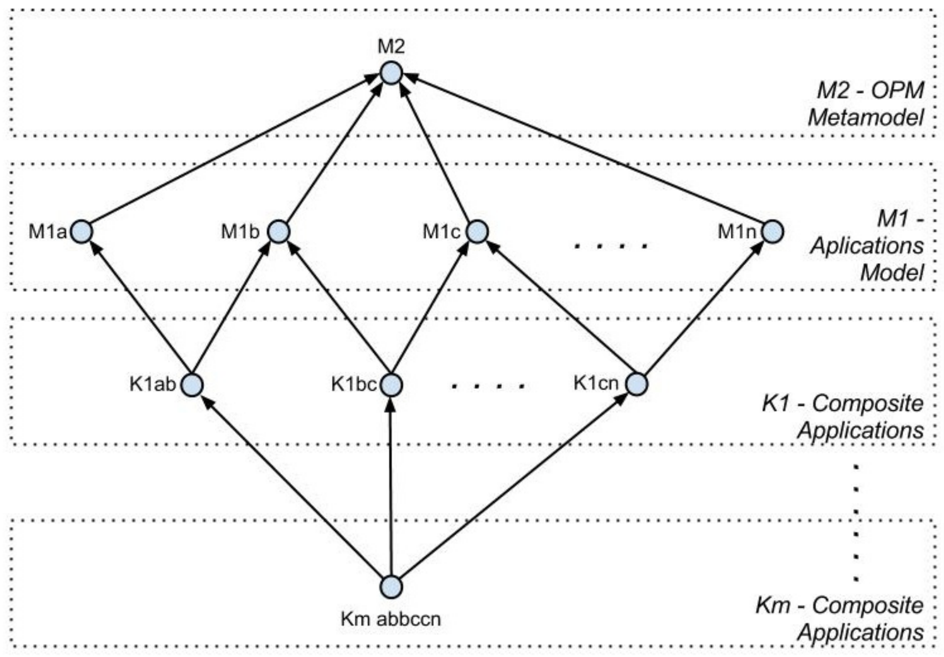

Figure 6: Creating complex application by using composing method.

Future work

Although the complete infrastructure described in this paper have been implemented and used in real-world applications, described ADBE approach is not fully developed in the part of application modeling. Further research will try to find the most intuitive and efficient solutions for the PBE approach. It involves the measurement of application creation efficiency of the classical approach in comparison with ADBE approach. Figure 7 displays the prediction of comparison diagram between classical application development approach and ADBE approach across five different aspects: (1) process coverage, (2) speed of development, (3) speed of implementation, (4) economic factor, and (5) ease of modification.

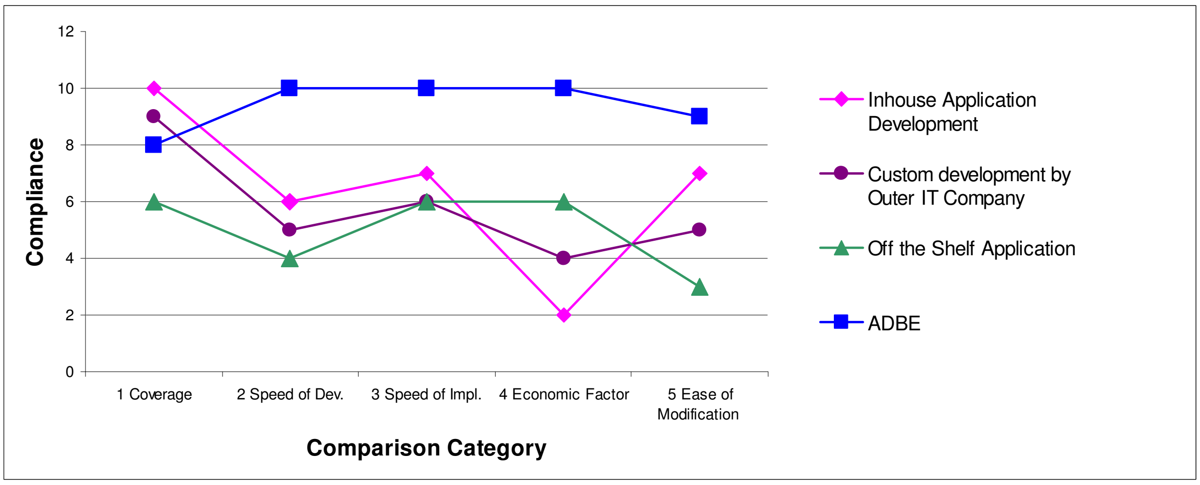

Figure 7: Comparison of classic design and ADBE

The expectation is that ADBE method will have a significant advantage over conventional methods of development in all categories except for category (1) process coverage.

In addition, further research will focus on the implementation of BI tool that will be able to enable the analysis of data stored in the graph database, assuming the frequent changes in data structures. This tool should be able to use the structure of the data model of the M1 layer and based on this information to offer the user options for analysis, also using PBE approach. Once the user selects the desired option, the tool should allow fast searching of data stored in the M0 layer.

Conclusion

IT has all the prerequisites to become a powerful driving force to improve business again. To achieve the goal of modification of business processes through the modification of business computer applications, it is necessary to provide a solution in which the business user with average knowledge and skills in using IT could independently and efficiently change their business applications, and thus the business process. For this become real, it is necessary to simplify the programming process as much as possible. The user should not be even aware that it is about programming. He just needs to show with an example what he needs done and a computer program automatically does the rest in all aspects of an applications. ADBE is a suggested procedure that combines the best of all methodologies developed so far for codeless application development. It is described in the concept of the Superius G2 software system that implements all of these principles.

Acknowledgements

This work is supported in part by the Croatian Ministry of Science, Education and Sport, under the research project “Software Engineering in Ubiquitous Computing”.

References

[1] Jakupović, Alen; Pavlić, Mile, Measuring the Complexity of Business Organization and Business Software Using Analytic Hierarchy Process (AHP). // Computer Technology and Application. 2 (2011) , 9; 736-747

[2] Jakupović, Alen; Pavlić, Mile. Procjena veličine poslovnih djelatnosti podržanih ERP rješenjima // CASE22 / Polonijo, Mislav (ur.). Rijeka : CASE d.o.o, 2010. 51-57

[3] Fried, Jason, Rework / Jason Fried and David Hansson.--1st ed. eISBN: 978-0-307-46376-0

[4] Ramljak Darije,Dinamički prilagodljivi poslovni procesi kroz BPM i SOA koncepte// CASE22 / Polonijo, Mislav (ur.). Rijeka : CASE d.o.o, 2010. 5-10

[5] Srbljić, Siniša; Škvorc, Dejan; Skrobo, Daniel, Widget-Oriented Consumer Programming. // Automatika : časopis za automatiku, mjerenje, elektroniku, računarstvo i komunikacije. 50 (2009) , 3-4; 252-264

[6] Halbert, Daniel, 1984. Programming by example. Ph. D. diss. University of California, Berkeley

[7] Lieberman, Henry, Your wish is my command: programming by example, p.cm. ISBN 1-55860-688-2

[8] The UML Metamodel, [Online] http://umlbase.com/learn/fundamentals/the-uml-metamodel/

[9] D. Dori, Object-process methodology: a holistics systems paradigm. Springer, 2002, no. s. 1.

[10] N. Tankovic, D. Vukotic, and M. Zagar, “Executable graph model for building data-centric applications,” in Information Technology Interfaces (ITI), Proceedings of the ITI 2011 33rd International Conference on, june 2011, pp. 577 –582.

[11] Vukotić, Dražen, Tanković, Nikola. Alati za razvoj aplikacija bez kodiranja // CASE23 / Polonijo, Mislav (ur.). Rijeka : CASE d.o.o, 2011. 15-22