Information Systems Development Based on Visual Domain Specific Language BiLingva

Jana Ceriņa-Bērziņa, Jānis Bičevskis, Ģirts Karnītis

Information Systems Development Based on Visual Domain Specific Language BiLingva

Jana Ceriņa - Bērziņa, Jānis Bičevskis, and Ģirts Karnītis

Datorikas Institūts DIVI, A.Kalniņa str. 2-7, Rīga, Latvia
{Jana.Cerina, Janis.Bicevskis, Girts.Karnitis}@di.lv

Abstract. This paper is devoted to the system modelling and information transfer to information system. The BiLingva (Bilingual Language) is a domain specific modelling language used to model event oriented information systems. Unlike many other modelling languages BiLingva allows unconstrained use of state diagram concepts and activities diagram concepts in the same diagram. This approach is most appropriate in this specific domain.
In the second part of the paper authors analyze a new approach for transferring information system model in BiLingva language to the information system. It is proposed to make user friendly and model consistent configurable software that takes over information from BiLingva model.
In the end this paper provides a few examples of running systems developed with this technology thus confirming usefulness of this approach.
Keywords: Software engineering, Modelling, Domain specific languages

1 Introduction

A number of years the efforts of software developers have been devoted to creation of IS development technology that would enable communication on system requirements in a language comprehensible to both, users of IS (client) and system developers. Such a language could prevent misunderstandings in communication reduce the number of cases when user failed to explain the requirements and developer failed to understand them. Despite over a decade experience including development of CASE tools [1], MDA (Model-Driven Architecture) [2] in 1990ies and later development of MDSD (Model-Driven Software Development) [3], the issue of graphical modelling languages has not been solved yet.

Concept of the CASE tools was based on assumption that information system development is a continuous process that begins with conceptual modelling of system, is followed by detailing and translating upper level concepts into lower level concepts, and ends with detailed specification, which serves as the basis for the development of application with appropriate tools. The proposed approach implied - software development without programming assuming that all activities are related to specification of requirements and not actual program coding. This concept has been embodied in practically used tools such Oracle Designer [4]. However, the real environment raises more serious requirements to the system quality than CASE tools can provide, e.g. user interface, usability, maintainability, performance etc. Therefore, program code generated by CASE tools must be modified without including these modifications into specification

and, thus, creating a problem of discrepancy between the code and specification. Additional difficulties are brought about the fact that CASE tools can generate application only with features built into that CASE tool. Due to rapid development of technologies and supply of new interface objects and concepts applications developed with CASE tools are lagging behind.

Above mentioned problems that can hardly be overcome in CASE concept are better solved in MDA. In order to make generated applications more flexible, MDA proposes system development in 2 steps. In the first step platform independent model (PIM) is created using a universal modelling language, for example UML (Unified Modelling Language) [5]. In the second step the environment for running a program (PDM) is defined. Application is generated from PIM using PDM that allows generate more flexible and user friendlier application. The MDA approach is being developed very rapidly nowadays [6,7], however, the main problems have already been identified [8]:

- Modelling languages used in information system modelling (including UML) are universal and difficult to understand for non-IT specialists (users). Although users accept specifications proposed by IT specialists they rarely understand them and hardly evaluate possible consequences.
- Automatic code generation from PIM specifications cannot create high quality software with good usability and reliability.
- Modification of automatically generated software is complicated and difficult due to efforts to maintain consistence between code and specification. If programmer changes code these changes are not included into specification. Repeating code generation can eliminate individually made changes.

Therefore, in many cases MDA ends with specification in UML and software is more or less manually developed from that specification.

The authors propose a new approach to solve the above mentioned problems.

1. We know, based on experience that each problem domain has specific concepts and activities. Business processes and objects are being modelled, leaving technical implementation details to the IS development stage (similarly as developing PIM in MDA architecture). Few the most popular modelling languages and tools, however, provide insufficient support to creation of a complete model. For example, the frequently applied state diagram uses a general term “state” without possibility to define attributes and their values to determine the current state. Also, frequently one diagram includes concepts from different types of diagrams, even from different modelling languages. For example, in the event oriented information systems it is useful to include concepts from UML activities diagrams and UML state diagrams as well as add timer concept from SDL. Therefore, it is useful to create own Domain Specific Modelling Language (DSML) to serve a specific task.
2. For the event driven IS modelling we propose Domain specific modelling language, called BiLingva, that combines business process concept from BPML, state concept from state transition diagrams and time control concept from SDL. Applied in practice the BiLingva proved that it is easily comprehensible to developers as well as to users without significant IT background.

3. In order to use DSML several supporting tools are needed for visual diagrams drawing, step by step detailing, model consistency check, navigation from diagram to diagram etc. Usually, development of such drawing tool is expensive and time consuming. Though, the newest achievements in modelling tool design [10,11] enable development of a tool for a specific DSML within an acceptable amount of resources. For example, the development of BiLingva supporting tool required one person-month.
4. In order to achieve one of the main modelling goals – to develop information system that runs according to business model – we propose the following solution: instead of application generation automatically we propose use of "the narrowed" information transfer from DSML model to the application. Practically, it means that an application is developed in order to solve a specific task. This application is reading from database many parameters that are necessary to ensure application's operation. The parameters are configured by developers or in case of using DSML these parameters are transferred from DSML model to the database. Thereby, the PDM model is not directly defined, but it is indirectly included into particular application. By using domain specific, highly configurable and user friendly solutions, it is possible to develop a high quality programs that operate according to DSML model. In the case of BiLingva the data transfer from model to the IS database was carried out automatically.

Approach described in this paper has been tested in a number of medium sized projects. The results of testing experience are given in the last chapter. The most surprising was the very positive user's attitude toward the models developed with BiLingva. These models were not only a part of specification but also became a part of user's manual. Users switched from reading thick user's manual to studying graphical models that describe system more precisely and are easy comprehensible. Full compliance between graphical specification and real system also was achieved. The authors do not assert having the same success in development of any IS since it largely depends on how suited specific IS domain is for proposed solution. Implemented in event oriented information systems (such as CRM, document management systems, project evaluation systems etc.) proposed solution proved to be very successful.

2 Introduction to BiLingva Modelling Ideology

This chapter deals with a short concept of BiLingva modelling language. The conventional IS specification languages include two approaches for modelling of information processing:

- **State diagrams**; the main objects and their states are identified. Information processing is designed as objects move from one state to another. Object moves form one state to another are caused by activities that determine the state of the object. Thereby vertexes in state diagrams are some abstract object states and they are connected with edges that represent activities.
- **Process diagrams (DPD with many variations, Grade BM, activity diagrams etc.)**; the activities of the information processing and their sequence are identified.

Information system is designed as input messages are processed into output messages.

In BiLingva diagrams it is possible to use both state diagrams and process diagrams simultaneously. There are 2 types of vertexes in BiLingva models – object states and activities. Vertexes are connected with edges representing either activities that change object state or activities that trigger execution of another activity. BiLingva permits use of condition symbol in that is not allowed in state diagrams and DPD, but is allowed in business process modelling languages. Step-by-step detailing is one of the cornerstones of BiLingva. The designer may detail activities and states as deep as it is necessary.

The following example based on project evaluation system demonstrates the main idea of BiLingva. The example is built on information system *Project-System* base that is foreseen to provide support to the project evaluation process (to evaluate projects submitted for innovation or science tenders etc.)

The project submitted to tender is the main object processed by system. Project compliance with formal requirements including mandatory information (legal status and address of the submitter etc.) is verified at the very beginning of the workflow. The next procedural step evaluation of project content implemented by experts. The outcome of the evaluation processes includes either approval or rejection of a project. Main tasks of the system are to record all activities related to project evaluation and control the sequence and execution term of activities. The system must provide reporting on approved and rejected projects, as well as on projects in different phases of the evaluation process. Our proposal is to use the concept of state and the concept of activity equally in one diagram. The idea is shown in Fig. 1. – the model combines both concepts, the states and the activity.

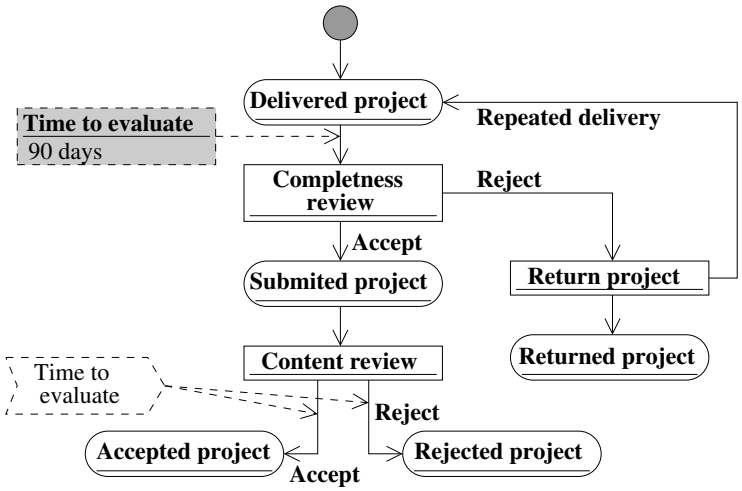


Fig. 1. State transition and activity (BiLingva) diagram.

This model provides additional advantages:

- It is possible to detail both states and activities.
- In the case of forking it is enough to add the result of condition at the edges that go out from activities.

The detailed BiLingva graphical modelling language description is given in the next chapter.

3 BiLingva Description

The main concepts of BiLingva language are described in this chapter.

3.1 Object

The concept of object in this paper describes an object processed by information system. A project or data submitted for evaluation, a personal identification number in the Population Register, a document, etc. is considered an object. As practiced in other modelling languages and, similarly, in BiLingva language we considered that only one object is being processed, although a real information system processes many objects simultaneously.

3.2 Object Attribute

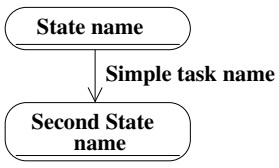
The object attribute is a variable that describes a property of an object and has assigned value in a property's domain. For instance, a project's identification number – comprised of a tender number and project serial number – is an attribute that is formed according to specific rules and has a specific semantic meaning. Another example, a personal identification number – a unique number assigned to a person contains person's birth date. The most typical attributes are:

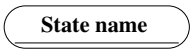
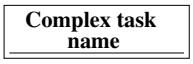
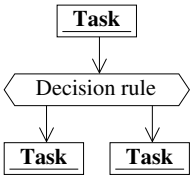
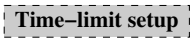
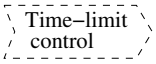
- object identifier – a unique value for each object,
- object name (can be used instead of identifier provided the names are unique),
- object status,
- performer – position or role,
- comment – free text,
- others.

3.3 BiLingva Language Base Elements

The base graphical objects used to draw BiLingva diagrams are given in this chapter.

Elementary task a task with no further detailing required during modelling. The task attributes are identifier, name, task performer (role), comment, etc.



Object state	a set of attribute values, more precisely – equivalence class of attributes' values of a marked object	
Complex task	a task available for further optional detailing during modeling. The attributes of a complex task are identifier, name, task performer, comment etc.	
Decision		
Time limit setup	The time limit attributes are time limit identifier, time limit length in months or days, time limit warning before time limit end in months or days, comment.	
Time limit control	Time limit control attributes are time limit identifier, comment.	
Process start		
Process end		

3.4 Diagram

The diagram is a graphical representation of the above defined symbols considering the following rules:

1. The diagram should have exactly one process start.
2. The diagram should have at least one process end.
3. The vertexes can connect:
   - State S1 with state S2, meaning elementary task causes the object move from S1 to S2. Task name adjacent to the vertex is mandatory.
   - State S1 with complex task D1 meaning the object moves from state S1 to start execute complex task D1. Task name adjacent to the vertex is optional.

- Complex task D1 with state S1 meaning a move from execution of complex task D1 to the state S1. Task name adjacent to the vertex is optional.
- Complex task D1 with complex task D2 meaning a move from execution of the complex task D1 to execution of the complex task D2. Task name adjacent to the vertex is optional.
- State S1 with same state S1 meaning execution of a simple task that does not change the object's state. Task name adjacent to the vertex is mandatory. A number of vertices can be applied.
- Many vertices can enter each state and complex task.
- Many vertices can exit each state and complex task.

4. The time limit setup and time limit control are connected before or after the state or complex task
5. The construction of mutually exclusive tasks has exactly one Enter and one Exit and is constructed in the same way as in UML activity diagrams.

The complex task can be considered as function that can be used repeatedly.

3.5 Model and Step-by-Step Detailing

The result of a process modelling implies the model containing several diagrams. One of the diagrams, called the main diagram, consist only of states and tasks that are not a result of object detailing. Each of the main diagram's states or complex tasks can be detailed with exactly one diagram that contains new states and complex tasks. Thus continuing the author can detail the model as far as necessary.

The detailing rules are the following:

- Each state and complex task can be detailed by exactly one diagram.
- Each diagram details exactly one state or complex task.

Navigation exists between the diagrams according to detailing.

3.6 Diagram Editor

A specific visual tool build platform is used in order to develop a tool for editing of domain specific visual diagram [10,11]. The developed tool has the following capabilities:

- Navigation between diagrams according to step-by-step detailing principles,
- Automated object placement,
- User – defined object placement,
- Object visualisation with attribute filtering. Attributes can be shown all or only part of them.
- Attributes' style configuration (font, colour, style),
- User defined views can be used,
- User actions' sequence support, including UNDO,
- Access from outside applications to tool repository in read/write mode.

The project evaluation process diagram developed with this tool is shown in Fig. 2. This diagram has 4 (four) new possibilities added:

- Each task has a performer (e.g. *Registering of Project – Registration unit*)
- Each state has a performer that owns the project in a specific state (e.g., *Submitted Project – Registration unit*)
- Time limit for accomplishment of a task and time limit control can be set.
- Projects while in a specific state can have elementary tasks that do not change the object's state (e.g., a project in the state *Submitted project* has assigned a performer of the next task *Set Completeness reviewer*).

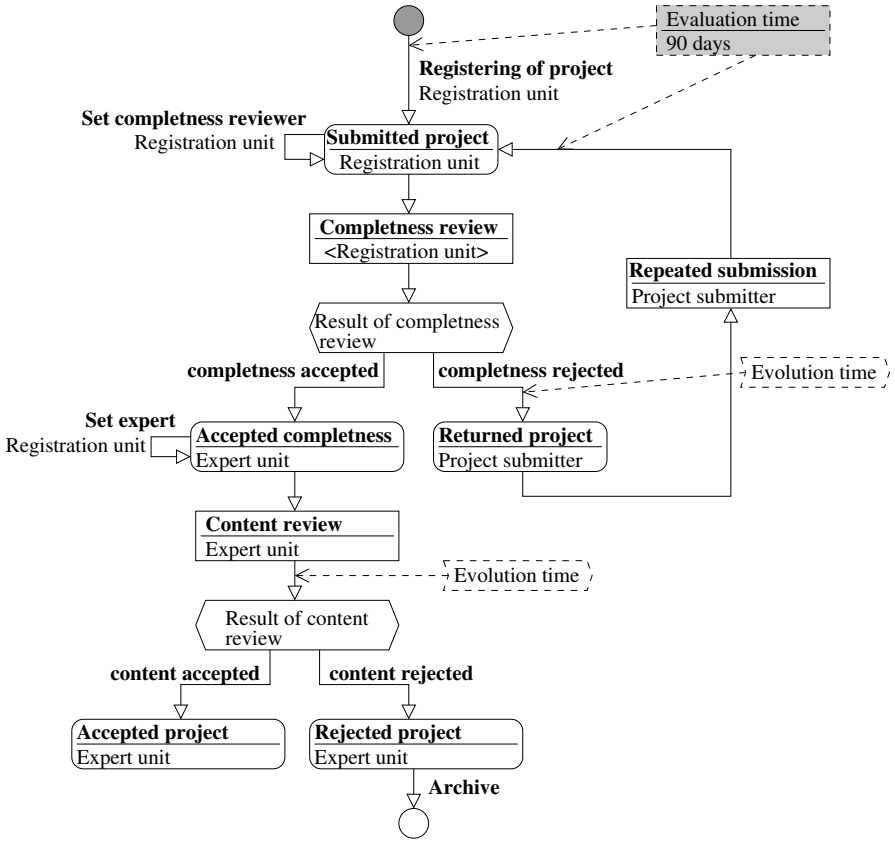


Fig. 2. BiLingva diagram.

4 Supporting Tools and Transition to Application

The business process modelling cannot be an aim in itself. The purpose of a model is to develop and operate the application. According to the concept the proposed development of information system consists of several interrelated activities such as:

- **Development of the BiLingva graphical editor**. Built in 2 months using tool building platform the BiLingva graphical editor provides considerable spectrum of options to draw models. Moreover, models developed with this tool also have high aesthetical qualities. Thus, the main problem of the existence of supporting graphical tool in graphical DSML usage is solved successfully.
- **Development of a business model**. The business processes of a specific information system are modelled with the graphical editor. The result of such modelling contains both, model information and information used for graphical editor operation and is stored in repository.
- **Development of a specific application**. To the contrary of MDA this approach does not make an automated creation of "a fit-to-model" application an ultimate goal. According to this approach the developed application is corresponding to a specific domain and is able to interpret constructions from model in BiLingva. Information of those constructions is stored in a specific database not tightly coupled with BiLingva editor repository.

In order to use the information captured in model also in application it is necessary to transfer model's information from graphical tool's library to the application's database. It can be done automatically or manually. However, there is one problem to be solved – object identification. It is necessary to assign a unique internal number to each object. Such number is stored as a specific attribute of an object. This number is allocated by application – in fact it is primary key to the table where object's information is stored. Such technology imposes some restrictions to model modifications – numbers of an existing objects should be traceable in order to estimate how modification, deletion or addition of objects induce changes is (identifiers) numbers. Changes in object's identification number in database should be reflected in model as well.

The information transfer from the model to the application database can be implemented manually or automatically. In case of manual transfer, the user fills information on model objects into database and assigns object identifiers in model afterwards. This is a very long and exhausting process, besides it leaves room for mistakes. A safer and more effective way is to transfer information automatically using specific tool. A tool that can transfer model information from graphical editor repository to the application's database has been developed. This tool uses graphical editor's API to access repository, reads model data and fills application database. This tool has the following specific functions:

- Read model data from repository;
- Find model objects and their identification numbers in application's database and make necessary changes;
- Delete from application's database object's that cannot be found in repository;

- Add new model objects to application's database;
- Fill in the attributes of a specific object in repository with new object's identification.

  This model can be used to operate the application in a number of ways:
- Show objects distributed by states;
- Allow only activities/transfers defined in model to be applied to an object in a specific state.

The application is operating according to the model developed in BiLingva modelling language, thus achieving the goal. Now the qualities of the application, namely usability, security, performance, etc. depend only on application itself and its developers and are not subject to a hypothetical tool's functional ability to generate application of a high quality.

5 Practical Applicability

The given technology can be applied for a process modelling and application configuration in event oriented information systems in order to create practically applicable applications. The new BiLingva language, its technology and application have been applied to develop supporting information systems for project evaluation:

- Latvian Investment and Development Agency;
- Ministry of Education and Science of the Republic of Latvia;
- State Education Development Agency;
- State Regional Development Agency.

The system's primary goal is to provide support to registration, evaluation and monitoring processes of the project management. All of the above mentioned institutions share several common characteristics - institutional processes are stipulated by instructions that describe workflow activities, functionally responsible employees and authorities for each business process. As a rule, these institutions have sophisticated organizational structures with clearly established and separated responsibilities. Therefore, the time management (control) becomes one of the most significant features of the event oriented information system.

In hindsight, the development of a new technology – creation of new DSML language - was not an aim in itself. Moreover, one of the crucial requirements for information systems is to provide functionality to follow the state of an information object, e.g. the senior employee of the structural division should be provided with an option to monitor project evaluation progress: how many projects are in evaluation process or have been completed and how many projects have been rejected.

Initially, attempts were made to model the processes according to stipulations by applying state diagrams. However, merely applying state diagrams soon revealed that activities and their sequence significantly contribute to the changes in object's state and should be considered as important as object's state. This is the place where elements of activities' diagram become apparent - beginning and ending of a business process as well as sequence of activities involved are important. Yet, another issue on assigning

responsible employees involved in workflow throughout the organisational structures remained unsolved since the state diagrams do not provide such modelling support. Similarly, it was necessary to introduce an element 'decision' in order to split the process based on the results of a specific activity. E.g. if project evaluation result is positive, then project has to be approved; negative project evaluation result means, project has to be rejected. As a result state diagram model was combined with activities' diagram model thus creating a new bilingual language BiLingva.

Time control is another important requirement for business process modelling. Certain workflow activities have time limits or are subject to deadline set either by senior employee or stipulations of the procedure. Therefore, the concept of "term" was introduced into diagram model. According to this concept a certain activity (project registration) automatically activates time control for the term for project evaluation is 20 work days. In this example the time control function can be deactivated by another activity – project evaluation completed.

The BiLingva language developed gradually. Initially, models were created empirically for case specific notation. Information was manually transferred from models to application database. Application by interpreting information in database was performing according to modelled business processes. To the great surprise of the system developers, business process diagram models were remarkably useful in training and teaching the basics of information system to users (non-IT specialists). Business processes reflected in diagram models appeared to be a very successful information material to complement user's manual.

Now, the use of this new technology in the development of information system has reached the level of applicability that information captured in business models is practically embodied in configurations of real applications. The current scientific work focuses on development of automated information transferring from diagram model into system's database, including reversibility of this functionality - automated transferring of system's configuration changes into diagram model.

6 Conclusions

The informal nature of IS specification is the main problem in the further use of specification. When applying conventional methods in specification it is not possible to reach the level of precision that would eliminate all contradictions and collisions during programming. This paper presents a partial solution to the above mentioned problem and proposes the use of modelling language BiLingva and new technology to transfer information from model to application. The experience of using the BiLingva can be summarized in the following conclusions:

- The initial steps of system development include design of a business process model in BiLingva and approval form a customer.
- The organisational business procedures (system's business model) are captured in the information system in a form of graphical diagrams.
- The agreed system's business model is transferred to the information system data base thus ensuring software compliance with system's business model.

- The organisation's decision makers (without any specific IT education) are able to read and understand the system models and make decisions. The operation of an information system depends directly on these models. The workflows captured in the model specify the amount and responsibilities for information input.
- The users do not have to learn the complete workflow "by heart" since every user can access only those tasks that are assigned to a specific user
- "Learning by doing" – a new user can learn business processes by using the system therefore the user training becomes redundant.

In the future, research efforts should be focused on limits and usage boundaries of the presented technology as well as on development of new DSML for other types of systems aiming at expanding BiLingva's abilities to describe models that involve several parallel workflows.

References

1. Bergin T.J.: Computer-aided Software Engineering: Issues and Trends for the 1990s and Beyond. Idea Group Inc (IGI) (1993)
2. Object Management Group (OMG), http//www.omg.org
3. Volter M., Stahl T.: Model-Driven Software Development. John Wiley & Sons, Ltd. (2006)
4. Oracle Products, http://www.oracle.com/technology/products/designer/index.html
5. Booch G., Jackobson I., Rumbaugh I.: The Unified Modeling Language. Reference Manual (1999)
6. OMG Committed Companies and Their Products, http://www.omg.org/mda/committed-products.htm
7. OMG Model Driven Architecture, http://www.omg.org/mda/products_success.htm
8. MSDN, http://blogs.msdn.com/alan_cameron_wills/archive/2004/11/11/255831.aspx
9. Wikipedia Domain Specific Language, http://en.wikipedia.org/wiki/Domain_Specific_Language
10. Kalnins A., Barzdins J., Celms E.: Model Transformation Language MOLA. In: Model-Driven Architecture, European MDA Workshop, Revised Selected Papers, LNCS, vol. 3599, pp. 62–76. Springer, Heidelberg (2005)
11. Kalnins A., Barzdins J., Celms E.: Efficiency Problems in MOLA Implementation. 19th International Conference OOPSLA'2004, Vancouver, Canada, October (2004)