Larisa Globa
National Technical University of Ukraine „Kiev Polytechnic Institute", Institute for Telecommunication Systems
e-mail: lgloba@its.kpi.ua
lgloba@hotmail.com

Modified model driven software development

Abstract.

The approach to development and implementation of the complex methodology and integrated software environment for rising the quality and efficiency of Internet based information systems (IBIS) developing process; development of new high-end software products based on optimal software and functionality structure; increasing of IBIS performance by computing threads paralleling; efficiency increasing and costs decreasing for software reengineering; minimization of human factor influence in the process of IBIS development and reengineering at the expense of data threads modeling, project database designing and data storing on all development stages.

Keywords: Internet-based information systems, information processing, enterprise business processes, computing business processes.

Introduction

In up-to-date internet-based business-world and communications, there is a growing need for different organizations and enterprises to work together and communicate in global environment in order to achieve the desired business goals. Realizing this purpose, distributed Internet-based information systems (IBIS) are characterized by complex and hierarchical processes of information processing. Their complexity increases time and recourses, required for their realization, making the process of their realization complex and information processing inefficient [1].

This paper is focused on designing of the modified model driven software development (MMDSD) methodology which includes some additional approaches. The first one is software system modeling on the first development stage – requirements analysis, which includes three levels of activities presentation in graph forms. These three levels are described by enterprise business processes (EBP) model, computing processes (CP) model, interfaces and functionalities level (services and components) models. The second one is the abstract notions (model components) division into separate entities – interfaces and functionalities (services), with using models stored in database to describe them. The third one is the EBP, CP analysis and optimization including the methods of parallelization. The

fourth one is storing project information in database and knowledgebase, including all IBIS models that support generating correct and complete project documentation.

Problems of IBIS development

Internet based information system can be described as multimodule application software; a large networked system which stores, processes, and transmits sensitive data and information; complex integrated applications that perform a number of functions simultaneously; migrating to the Web and include numerous applications.

There are some problems, appearing when using state-of-the-art development methodologies and tools (extreme programming, waterfall model, model driven software development (MDSD)) for IBIS software development. Waterfall model, extreme programming have a lot of errors and buzz when IBIS realization (coding); over-costs of IBIS development; increase of IBIS development time, as a result there are some projects failures because of IBIS unconformity to fast changeable requirements and requirements are not formalized and clear.

MDSD doesn't have above mentioned disadvantages, but there are still some. The EBP and CP are not optimized, resource-intensive EBP and CP reengineering, no tools for EBP and CP analysis, error recovery and paralleling, development process isn't integral, i.e. the separate development stages are poorly connected, that brings to high time and money costs when high-end IBIS development.

For overcoming all this disadvantages the Modified MDSD (MMDSD) methodology is suggested like a basis for IBIS development and implementation. It includes new models, algorithms and software toolkit.

Approaches for IBIS development

MMDSD basic concepts are used some additional approaches too. First one is modified software basic components model. It includes informational resources, computational resources and models to characterize the unified environment for IBIS functioning in global environment. The second approach is three-level hierarchical models of information processing. This model is developed for three levels complex analysis and optimization the enterprise business processes (EBP-level), computing processes (CP-level) and tasks performance (Task-level) in distributed information environment. EBP-level considers enterprise business model in general and application area of business processes realization in particular. On EBP-level EBP are developed and represented like the models of the weighted graph and data graph. The EBP models are specified to separate operations. The

designed integrated EBP models include data flows and EBP detailed specification. The important task to solve on this level is business process analysis and parallelization using set of the enterprise criteria. The main criteria for EBP parallelization is their efficiency. All this models form problem domain models, which are stored in project knowledgebase.

CP-level considers the IBIS computing processes. On CP-level enterprise business process model is mapped on computing processes model taking into account information exchange. It means that problem domain model is mapped to solution domain model (from EBP model to IBIS model) and business system data model transfer to IBIS data model without data loss.

Task level provides separate tasks of computing processes implementation. On Task-level separate functional components models are used. They are user interface model, model of functionality (web-services) and model of connections between them. The structural mathematical user interface model is:

$$F = (N, E_1(x_{11},...x_{1N}),...., E_N(x_{N1},..., x_{NM}), C_N(i_{NN}, o_{NN}), P(pl, ob, func), L(con, des, pos)),$$

where N – number of the points where user or system can produce an events; E_1, .., E_N – names of the points where user or system can produce an events; $x_{11},....x_{1N}$ – input data for the function E_1, and $x_{iN} \in X$ – data region of acceptability; $C_N(i_{NN}, o_{NN})$ - a list of interfaces (relations input-output) optional; $P(pl, ob, func)$ - a places where processing code is situated (it means: *pl* - platform name (COM object, WEB services, CORBA object, DLL and etc.), *ob* - object name, *func* – function name; $L(con, des, pos)$ - a face (*con* - state (active or not), *des* – presentation (colored or not and so on), *pos* – a position on diagram).

The structural mathematical model of functionality is:

$$f = (E_{form}, E_{func}, E_p, \ x_1,..., x_N, y_1,..., y_N, \ C_N(i_{NN}, o_{NN}), P(pl, ob, func), L(con, des, pos)),$$

where E_{form} – the activity points where user or system can produce an events, in this point the events activate functions connected with the user interface; E_{func} – the activity point which belong to the function and transfers data to the external object (user interface or function); E_p - the activity point which belong to the external object (user interface or function) where activity stated after running function stop; $x_1,....x_n$ – input function data and $x_{iN} \in X$ – data region of acceptability; $y_1,....y_m$ – output function data and $y_i \in Y$ - data region of acceptability; $C_N(i_{NN}, o_{NN})$ - a list of interfaces (relations input-output) optional; $P(pl, ob, func)$ a places where processing code is situated; $L(con, des, pos)$ - a face (*con* - state (active or not), *des* – presentation (colored or not and so on), *pos* – a position on diagram).

All the functions can be represent in following format: *string Func(string text),* use *text* like input and output data. Functions call is working used next construction:

(Access point)<input data>→string →handler call→ string→ (Access point)<output data>

There are three types of functions: dummy functions between user interfaces, functions which are called one of internal functions, functions which call external modules. For different types of functions the activity point *Ep* can get different values to characterize next computing situations: uniquely determine the user interface F_j to be called like a function executing result; return the activity to user interface F_i ; to specify the external module K_l which is activated as a result of the function execution.

Dummy functions are the functions which have the same model like usual functions but additional have a special mark and have no code. They have only interface for integrating into the system and test data. In this way we have two type of the model of functionality. They are the model of dummy functionality and model of runtime functionality. For integration user interface model and model of functionality it need to use model of connections between them. Model of connections between user interface model and model of functionality represents the events in system in executive mode.

On the task level IBIS model is mapped to IBIS prototype model like a part of it. IBIS prototype model includes IBIS user interface model, IBIS model of functionality (web-services) and model of connections between them. It is used for system early (before coding functions) testing that gives the possibility to analyze system operation in test mode. The dummy functions are used with test data for early testing processing instead of real functionality.

All these models implementation provide realization of system early testing without coding based on abstract platform-independence complex IBIS model. For this realization the additional notations are used and toolkit for prototyping is developed. The toolkit includes tools for IBIS user interfaces design and represent. Suggested interfaces model consist of following elements: interface view, interface model and interface behavior description. There are tools for IBIS functionality design and representation (precedent controller, precedent, service levels). Services model consist of following elements: functionality view, function model and functionality description (algorithms and formulas). For early prototyping FFD Designer [3] is used. This is prototyping toolkit based on graphical notation for fast design, executing tool, testing tool. The mathematical apparatus of finite-state machine and Petri networks are used for early testing. So, each level provides special tasks for IBIS processing analysis and performance.

EBP and CP design and optimization

Enterprise business processes level is to provide the conformity between IBIS functionality and enterprises business tasks; formalizing the requirements analysis when information processing is designing; raising the probability of their proper implementation.

According to the suggested approach, the following models are developed on the EBP level: the business-model, which is the model of EBP system, represented as weighted graph; the model of EBP and operations, which is the result of the business-model decomposition and specification. The example of simple business-model, represented by weighted graph, is represented in Fig.1. Nodes of the weighted graph represent the tasks. The execution process of every task presents the result data, which transfer to another task using EBP graph. The branches of weighted graph represent information threads. Every branch has weight, represented as w_{ij}, where i,j – are numbers of nodes, connected by corresponding branch. It is suggested to take minimized time of EBP execution using developed algorithms [1].

Time and required resources for EBP with their executing operations are stored in database. At the next step every node is considered as a separate EBP graph, which can consist of several operations and data thread (Fig.2). Time and required resources for EBP and operations execution are analyzed and optimized using developed algorithms [1]. The results are used for further business model analysis, using graph analysis methods [2]. Mathematical basis for EBP model analysis is developed. This includes EBP simulation; EBP parallelization; EBP optimization used business-processes performance efficiency criteria like EBP execution time *(T)*, dedicated resources *(X)*, probability of EBP execution *(P)* and business-processes performance requirements. The main business-processes performance requirements are execution time minimization, recourses optimal distribution, high probability of EBP execution. For all of this the following algorithms are used: the algorithm for optimization of business-process concurrent operations, algorithm for optimization of business-process sequential stages, integrated algorithm for optimization of business-process sequential stages, consisting of concurrent operations. Dynamic programming principles will be applied.

Information threads can be also represented by weighted graph $G=(M,V)$, where nodes (M_i, M_j) represent structural components of information threads, which are connected with branches in case when there is information transfer between them. Such informational graph is represented on Fig.3. It is possible to integrate informational (Fig.3) graph and business-model graph (Fig.1) to get business-model graph, representing designed workflow (Fig.4).

Next step of information processing is the CP level. First task is the conversion of EBP model to CP model. The steps using for this task realization are considerate below.

1. To show the processes that could be automated, could not be automated or could be automated partly on the EBP graph and to represent information threads on it. It is possible to use special graph colouring. Mapping realized in two steps. The first step is marking the processes that could be automated on the integrated EBP graph used white colour, could not be automated used black colour and could be automated partly used two colours at the same time. The second step is representing information threads used special arrows (fig. 5). Selecting subgraph from integrated EBP graph is shown in represented graph containing automated and partly automated processes. After this analysis of result graph integrity is stated.

2. To make colouring graph analysis. To represent graph containing automated processes and reform partly automated nodes the information threads are reorganized without data loss. These data are required for each automated tasks performing. The additional system data are added for specification the information threads. At the end of reorganizations the input and output information for EBP and CP shouldn't be changed.

3. Result graph is optimized and then performed as UI-WS graph (UI-User Interface, WS – Web Services). System early testing is produced using this graph [3]. One of the important tasks on this step is development of algorithms of IBIS model information threads analysis and paralleling with guarantying their integrity.

The last step of information processing is the tasks performance level. The CP are considered as set of separate tasks. Each task is analyzed to be parallelizable. The main analysis criteria are paralleling appropriateness and availability of necessary computing resources. If tasks paralleling is appropriate, the technologies of parallel programming are chose and possibility of high-optimized multithreaded mathematical operations libraries attachment are considered.

The main steps of IBIS development

IBIS development steps according to MMDSD conceptually shown on table 1.

Table 2. The main steps of IBIS development according MMDSD

Step	The used suggested tools	Results
Step 1 Specifications definition, EBP definition analyzing and optimization	Special data base designing (FFD designer)	IBIS project data base
Step 2 Specifications definition, EBP mapping on computing processes	Special data base designing (FFD designer)	IBIS project data base
Step 3	Special data base designing (FFD	IBIS project data base

Specifications definition, CP definition analyzing and optimization	designer)	
Step 4 Specifications definition, task designing and optimization	Special data base designing (FFD designer)	IBIS project data base
Step 5 Early testing	Special CASE-tools for testing	Proper pre-prototype IBIS interfaces IBIS stub-functions
Step 6 Optimization	Special CASE-tools for optimization based on parallel executing process	Optimized pre-prototype IBIS interfaces IBIS stub-functions
Step 7 Coding	Other well known CASE–tools	IBIS wireframe code IBIS complete code
Step 8 Testing	Other well known tools for testing	IBIS correct code
Step 9 Deployment and maintenance	Special CASE-tools and data base for re-engineering	IBIS project data base Functioning IBIS

All the development steps are executed iteratively. IBIS binding to system architecture levels using up-to-date software technologies and Internet protocols when its deployment. IBIS implementation is realized based on integrated environment of IBIS development, which includes toolkit for modeling, analysis, testing (estimation of critical paths), IBIS prototyping and data storing on every IBIS development steps.

Conclusions

The suggested approaches to IBIS development have following advantages to have time and financial benefit:

- Software system modeling on the first development stage – requirements analysis, which includes three levels of activities presentation: enterprise business processes (EBP) model, computing processes (CP) model, interfaces and functionalities level (services and components) models.
- Abstract notions (model components) division into separate entities – interfaces and functionalities (services) with using models stored in database to describe them.
- EBP and CP analysis and optimization using methods of parallelization.
- Storing project information in database and data knowledgebase, including all IBIS models that support generating correct and complete project code and documentation.

References

[1] Larysa Globa, Tatiana Kot, Alexander Schill, "Business-processes optimization while information systems design", ACS'08 Proceedings, Poland
[2] R. Basaker, T. Saaty, "Finite graphs and networks", 1974, p. 368.
[3] Larysa Globa, "CASE Tools for Distributed IT-System Accounting Multithreading", Polish J. of Environ. Stud, Vol. 16, No. 5B (2007), 135-140

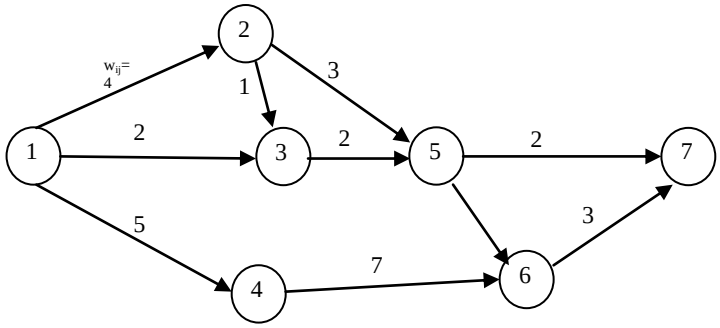

Fig.1 Example network diagram (business-model graph)

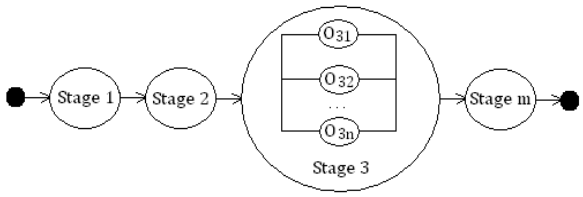

Fig.2 Model of application area business-process

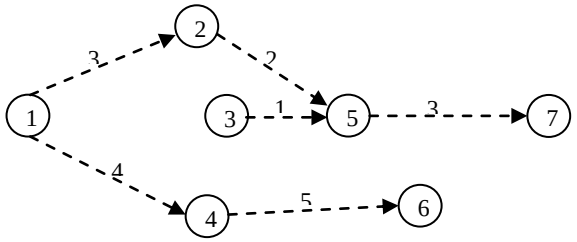

Fig.3 Informational process graph

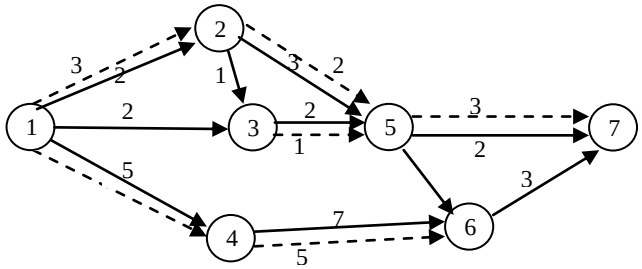

Fig.4 Integrated business-model graph, representing designed workflow

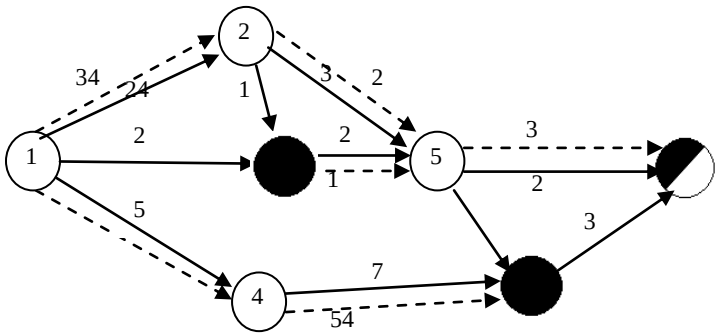

Fig.5 Integrated business-model graph after graph analysis