Towards Automatic Generation of Project-Based Solutions

Krikor Maroukian, Kevin Lano, Mohammad Yamin

Towards Automatic Generation of Project-based Solutions

Krikor Maroukian[1], Kevin Lano[1], Mohammad Yamin[2]
[1] School of Natural & Mathematical Sciences, Department of Informatics, King's College London, Strand, London, WC2R 2LS, UK
[2] Department of Information Technology, King Abdulaziz University, Jeddah, 21589,
Saudi Arabia

krikor.maroukian@kcl.ac.uk, kevin.lano@kcl.ac.uk, myamin@kau.edu.sa

Abstract. Modern business models and processes usually demand an integrated utilisation of business frameworks and methodologies, such as PRINCE2® and PMBOK®, to produce meaningful business documentation and solutions. Often, the use of such frameworks is a prerequisite to engage with public or private sector large-scale projects. However, models contained in such frameworks usually lack formal semantics which may lead to inconsistencies between modeling solutions. The maintainability and reusability of such models tends to require manual intervention which is susceptible to human error. Software engineers used to experience similar issues and partially solved these by introducing a model-driven approach called Model Driven Architecture. In an attempt to adapt to industry needs, over the past five years Domain Specific Modeling has experienced increased popularity. The authors propose a transfer of concepts and logic from MDA and DSM to a project-based model-driven approach; facilitating the automated production of supportive documents for business decision making.

1 Introduction

Rapidly changing business environments require frequent re-calculation of business strategies. Such changes are frequent and unpredictable. Human responses to these changes can be prone to human error and not within the required timeframe. The current industrial landscape predisposes business solutions (business decisions and supporting documentation) with a number of defects in terms of lack of understanding and implementation of frameworks, methodologies and best practices. As a consequence, informal models or even non-modelled business solutions offer limited value to the business.

Such informalities, may lead to a number of limitations such as: the requirement for model specific training, difficulty in capturing changing business requirements and the use of inconsistent models which are often out dated. Effectively, changes that will not be included in all corresponding models will create inconsistencies since the models will no more reflect the actual business requirements. Subsequently, models

will be discarded. “Often, the modellers themselves have disappeared, and any knowledge that wasn’t captured in the specialised models is inaccessible, forgotten, or written off” [1]. In addition, informal models are limited to use by individuals, small teams or within single organisations due to the lack of information clarity and understanding of non-standard models among different teams or members. This leads to the inability to address key business environment factors.

In the last decade, software engineering solved most of the system modelling problems with the introduction of the Model Driven Architecture framework (MDA). MDA divides models into four abstraction layers; Computational Independent Model (CIM), Platform Independent Model (PIM), Platform Specific Model (PSM) and generated code. The key idea behind MDA is the production of formal models through consistent model transformation. Models of the PIM layer are transformed to models of the PSM layer and then to code. At each layer, the user can add details or tune the models as needed. Any model can participate in an MDA transformation as long as it has a corresponding meta-model. A meta-model is a model that explains the semantics of its corresponding model. In other words, a meta-model is data about data.

Our research extends the applicability of MDA and uses it to solve business modelling problems in the project management domain. In effect, a *domain* can be characterised as a business discipline, customer, company, contact, location. Domain Engineering such as Product Line Engineering, is the entire process of reusing domain knowledge in the production of new software systems. An essential idea in systematic software reuse is the application domain, a software area that contains systems sharing commonalities [18].

The reason the author’s attempt to shed light on project-based modeling and automation regards noteworthy references from the MDA community on benefits realised in software development project management. As a result, this paper proposes a project-based approach to inherit the MDA concepts of Modelling and Meta-modelling, separate modelling layers and model transformations. This could potentially inherit a number of MDA established benefits realised over the past decade as follows ([2],[3],[4], [5], [6], [7], [8], [9]):

- Increase productivity and reliability through automated generation of business related documentation;
- Reduce time-to-market solution;
- Richer model semantics;
- Models with higher formality.

The proposed framework will be capable of supporting corporate decision making through business solutions provided that their corresponding meta-models are present. This project-based approach can be utilised as a solution generation tool to offer artefacts given the appropriate meta-model or pool of metamodels, model transformations and/or reusable project-based artefacts (meta-model or transformation).

Further to the introduction, this paper is organised as follows; in section (2) the definition of a model is illustrated and current modelling issues in software development are presented. Section (3) discusses in brief the various aspects of Model

Driven Architecture. Section (4) presents a thorough account of the proposed project-based approach. Research conclusions and future work are discussed in section (5).

2 Project-based Modelling Issues in Software Development

A model is a representation of a concept from the real world. An interpretation of a model gives a model meaning [10]. Models are widely used and are essential in other disciplines. For instance, prior to the construction of a bridge civil engineers produce a design that will be utilised as a blueprint for the construction of the bridge. It is not possible to start the construction of a bridge without any designs. However, it is not uncommon to start a business project without any planning and sometimes without concise and well-defined requirements or specifications or even clear business goals and objectives.

The paradox is that, software engineering can benefit more from models than other disciplines [11]. The current problem with models [12] is that most of the models are described in an abstract layer which is not very useful, indicating, what needs to be done at a given moment in time.

Nevertheless, business requirements change so rapidly that it is possible that the requirements might change while the project is still under development. Most of the cases the business solution will not reflect the design due to the high abstraction level of the design, time and/or cost constraints as well as incorrect or incomplete design. Provided there is a change request it is very likely that it will only change in the business solution and there is a possibility that nobody will update the design. These reasons will create an inconsistency between solution and design that will lead to the infrequent use or ultimately the disposal of the design.

Therefore, there is an explicit industry need to address platform complexity and the inability of third generation languages to alleviate this complexity and express domain concepts effectively. There exist MDA tools which can be employed to address these issues such as Eclipse Modeling Framework (EMF), Graphical Editing Framework (GEF), Graphical Modeling Framework (GMF) to which IBM contributes, Microsoft MS/DSL Tools, and Model Integrated Computing (MIC) utilised by Generic Modeling Environment (GME) developed by Vanderbilt University.

3 Model Driven Architecture

Prior to proposing a solution it is worth investigating how software engineering solved modelling issues presented in section (2). Model Driven Architecture (MDA) is a software development approach launched by OMG, [13]. The key idea behind the MDA framework is the separation of the development into three layers and the automatic transformation of models between the four layers by software tools. The business processes and requirements of the CIM layer are mapped to PIMs. These are then transformed to PSMs which are then transformed to code.

Human experts can execute manual tuning to each model and these changes will be carried over to the next model. MDA software tools allow changes made at a

higher layer of the MDA to be reflected at the lower layers of the framework."MDA is potentially advantageous because it shifts complexity away from developers and into the tool chain and, hence, the PIM-to-PSM transformation" [14]. MDA uses the Unified Modelling Language (UML), OMG's main modelling standards which are ISO standards and ITU-T recommendations. There are several model transformation languages; UML-RSDS [20], Epsilon [21], QVT-R, ATL, Kermeta, GrGen.NET. A comparison on the characteristics of some of these languages can be found in [22].

4 A Project-based Approach

A project management oriented approach attempts to address Model Business Engineering (MBE) issues with an aim to assist project managers and other project stakeholders generate day-to-day business documents and/or perform decision making activities in an increasingly automated manner.

Research in the area of transformations of UML Activity Diagrams to BPMN 2.0 has indeed been stated in e-Government systems [15], [16]. There is also evidence for the BPMN 2.0 to UML Activity Diagram transformation [17]. However, such empirical research has not been considered in the industrial domains of project management for frameworks such as PRINCE2® and PMBOK® as well as service management such as ITIL®.

The project-based approach can be characterised as *'a structured approach to automated generation of modelled artefacts in the context of business disciplines, that can form the basis of decisions, business documents and/or business activities.'*

This approach can reach its end result i.e. business solution generation, through two abstraction layers; Project Specific Layer (PSL) and Business Solution Layer (BSL). The end result e.g. documentation, can lead to management decisions and/or a set of actions. A mechanism to support reuse of best practices when creating families of business solutions would be appropriate to consider at this stage.

To visualize a software model transformation consider Java or C++ code as the implementation solution i.e. PSM, see section 3. The model describing the code functions and variables is one abstraction layer higher than the implementation layer i.e. PIM, see section 3. In a similar way, project management documents or decision making artifacts can be characterised as part of the implementation layer or BSL. The model describing the BSL is the PSL.

The project-specific layer ensures a modelled business and leads to business solution. The business solution layer would effectively depict the real data relating to information fed in the previous layer.

In certain instances, capturing project stakeholder related information can be of substantial value in formulating an accurate business solution such as business policies that do not allow employees to work beyond the eight-hour shift since these set their rules in the environment in which the project is executed. Hence, capturing environment information can be vital for the success of projects.

Information pertinent to a specific project framework should be utilised in the project specific layer. Any pertinent information to e.g. PRINCE2® or PMBOK®

should be utilised. Finally, PRINCE2® or PMBOK® produced documentation, roles, processes and functions signify a modelled business solution.

The project-based approach can help architects commit changes at the project-specifc layer which can then propagate to the business solution layer instead of having a monolithic transformation. The next sections describe thoroughly the project-specific layer.

4.1 Project Specific Layer

The project-specific layer can be defined as *'the depiction of project-based elements e.g. tasks, activities, resources, that can facilitate real world business solutions.'*

In this layer, it is recommended to select models from well established frameworks or industry standards with worldwide recognition. The accuracy of the result will heavily depend on the selected framework.

Taking into consideration the information available such as a meta-model that clearly states that the more certified PMs in structured PM frameworks the more successful that PM framework could prove to be in an organisation, it is clear that a structured PM framework would be selected for use within the enterprise. The business solution would relate to real data such as strategic corporate decision of whether to use a structured or agile PM framework. The business solution can be anything from a simple decision to complex models supported by vast documentation. In the scenario considered the business solution can either be a 'Yes' or a 'No'. In order to reach this stage, the data from PSM has to be extracted.

4.2 Business Solution

The business solution layer can be defined as *'the resulting business document(s), charts and descriptive information for a specfic project.'*

The business solution layer, contains the produced business documents such as business plans, progress reports, status reports, risk analysis documents, time tables, schedules and more artifacts that can be used for both day to day operation or strategic level information. Before this static documents are generated their corresponding meta-models are required.

There can be actions defined as activities to be performed by a human or software agent. Such actions can include, sending emails, perform transactions, make payments and more. To support the generation of such dynamic artifacts their corresponding meta-models should also include triggers with pre and post conditions. These can be defined in OCL or any other constraint language and must also be supported by the software tool producing the project-based model.

The project-based layers have been presented and thoroughly discussed. However, a closer examination of the approach with an example can form a concrete definition.

5 BPMN to UML-RSDS Transformation

Project change is inevitable whether it comes from within the project or from external factors influencing project scope. For these reasons, whenever change occurs, an agreed logical change management process has to be initiated that allows the project to identify, assess and control any potential and approved changes to the original baselines that where originally agreed for the project.

In addition, the use of a standardised change management approach can serve as a control agent to any Request-for-Change (RfC). Once the project scope and other key associated documents have been approved, these become the project "baselines" and can only be changed after approval by the appropriate authority; normally the Change Advisory Board. Change control and hence change management is not there to prevent changes, but to ensure that every change is agreed by the relevant authority before implementation. This section presents the transformation of a BPMN project-specific model from PRINCE2®, see Fig 1, to its UML-RSDS derivative model.

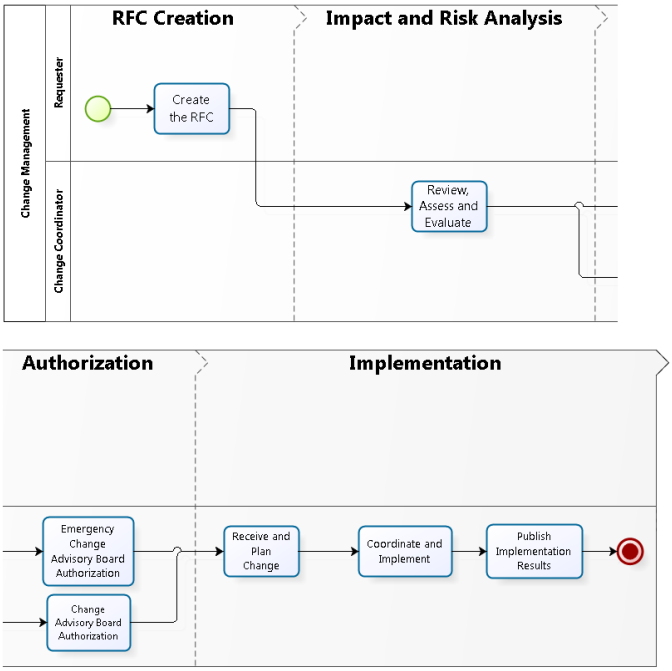

Fig. 1. Change Management Procedure of a Project

The next step is to produce the corresponding model using the UML-RSDS transformation engine based on a set of rules that apply for BPMN models. The transformations supported by UML-RSDS regard BPMN 2.0 elements set i.e. flow

objects, connecting objects, artifacts. Swim lanes are not included in the below example nor can they be currently supported.

5.1 Rules

The mapping is described by one rule, and the execution semantics by several update-in-place rules defining how a process instance may evolve, and how its tokens may move around the process. The rules are described in textual representations derived from UML-RSDS models.

Process Instantiation This is formalised by the following use case *initialise* postcondition on Process:

```
sn : flowElements & sn : StartEvent &
sn.eventDefinitions->forAll( ed I ed : TimerEventDefinition ) =>
ProcessInstance->exists( pi I pi.state = RUNNING & self :
pi.process &
  Token->exists( t I t : pi.tokens & sn : t.element ) )
```

"If the process has a StartEvent sn which has only TimerEventDefinition, create a process instance pi for the process, with one token at sn".

Normal termination These are postcondition use cases of Process Instantiation:

```
state@pre = RUNNING &
process.flowElements->exists( e I e : EndEvent ) & tokens@pre->forAll(
t I t.element <: EndEvent ) => state = FINISHED &
  tokens@pre->isDeleted()

state@pre = RUNNING &
process.flowElements->forAll( e I e /: EndEvent ) &
tokens@pre.element->forAll( n I n : FlowNode & n.outgoing->size() =
0 ) =>
  state = FINISHED &
  tokens@pre->isDeleted()
```

Either (i) the process has an EndEvent, and all its tokens occupy EndEvent nodes, or (ii) the process has no EndEvent, and all its tokens occupy nodes with no outgoing flow. In either case the process is set to FINISHED and all its tokens deleted.

Starting a process instance A process instance can start if it has a token t on a start event with at least one outgoing flow:

```
  state = RUNNING & t : tokens &
  fe : t.element@pre & fe : StartEvent &
  fe.outgoing->size() > 0 =>
  fe.outgoing->exists( sf I t.element = Set{ sf } )
```

The token on the start event is then moved to one of the outgoing flows of the start event.

Ending a process If a process instance has a token on a SequenceFlow with target node an EndEvent, then the token can be moved to the EndEvent:

```
state = RUNNING & t : tokens &
fe : t.element@pre &
fe : SequenceFlow &
fe.targetRef : EndEvent =>
t.element = Set{ fe.targetRef }
```

Entering a task

The same step applies if the target is a Task:

```
state = RUNNING & t : tokens &
fe : t.element@pre &
fe : SequenceFlow &
fe.targetRef : Task =>
   t.element = Set{ fe.targetRef }
```

Leaving Tasks

A process instance which has a token t on a Task fe can leave fe if fe has at least one outgoing flow:

```
state = RUNNING & t : tokens@pre &
fe : t.element@pre &
fe : Task & fe.outgoing->size() > 0 =>
   t->isDeleted() &
   fe.outgoing->forAll( sf I
        Token->exists( t1 I sf : t1.element & t1 : tokens ) )
```

t is deleted, and new tokens are created for the process instance on each outgoing flow.

Entering parallel gateway

Here we assume that there is at most one token for a given process instance on each flow element.

The process instance can enter parallel gateway pg if it has a token on every incoming flow of pg, and there is at least one such flow:

```
state = RUNNING &
pg : Parallel1Gateway &
v = tokens->select( t I pg.incoming->exists( sf I sf : t.element )
) &
v.size > 0 &
v.size = pg.incoming->size() =>
   Token->exists( t1 I pg : t1.element & t1 : tokens ) &
   v->isDeleted()
```

A single token ti for the process instance on pg is then created, and the set v of the instance tokens on the incoming flows of pg is deleted.

In this case the constraint requires fixed-point iteration, as it writes the same data (Token: :element) that it reads. The let variable v is used to store the pre-value of the expression it is assigned.

Leaving parallel gateway
This is formalised by the following postcondition use case on Process Instantiation*:*

```
state = RUNNING & t : tokens@pre &
fe : t.element@pre &
fe : Parallel Gateway &
fe.outgoing->size() > 0 =>
   t->isDeleted() &
   fe.outgoing->forAll( sf I
        Token->exists( t1 I sf : t1.element & t1 : tokens ) )
```

"If the process instance is running, and has a token t in a parallel gateway fe, with an outgoing flow, then delete t, and create a token for the process instance in each outgoing flow of fe."

5.2 Modeled Solution

The corresponding modelled solution of Fig. 1 is described with seven (7) tasks, two parallel gateways and a start and end node as follows below.

```
p1 : Process
p1.name = "BPMN2UMLRSDS"
pg1 : Parallel Gateway
pg1.name = "pg1"
pg1 : pl.flowElements
pg2 : Parallel Gateway
pg2.name = "pg2"
pg2 : p2.flowElements
se : StartEvent
se.name = "start event"
se : pl.flowElements
ee : EndEvent
ee.name = "end event"
ee : pl.flowElements
tl : Task
t1.name = "Create the RFC"
tl : pl.flowElements
t2 : Task
t2.name = "Review, Assess
and Evaluate"
t2 : pl.flowElements
t3 : Task
t3.name = "Change
Advisory Board
Authorization"
t3 : pl.flowElements
t4 : Task
t4.name = "Emergency Change
Advisory Board
Authorization"
t4 : pl.flowElements
t5 : Task
t5.name = "Receive and Plan
Change"
t5 : pl.flowElements
t6 : Task
t6.name = "Coordinate and
Implement"
t6 : pl.flowElements
t7 : Task
t7.name = "Publish
Implementation Results"
t7 : pl.flowElements
sf1 : SequenceFlow
sf1.name = "startTotask1"
sf1 : pl.flowElements
sfl.sourceRef = se
sfl.targetRef = t1
sf2 : SequenceFlow
sf2.name = "task1Totask2"
sf2 : pl.flowElements
sf2.sourceRef = t1
sf2.targetRef = t2
sf3 : SequenceFlow
sf3.name = "task2Topg1"
```

```
sf3 : pl.flowElements
sf3.sourceRef = t2
sf3.targetRef = pg1
sf4 : SequenceFlow
sf4.name = "pg1Totask3"
sf4 : pl.flowElements
sf4.sourceRef = pg1
sf4.targetRef = t3
sf5 : SequenceFlow
sf5.name = "pg1Totask4"
sf5 : pl.flowElements
sf5.sourceRef = pg1
sf5.targetRef = t4
sf6 : SequenceFlow
sf6 .name = "task3Topg2"
sf6 : pl.flowElements
sf6.sourceRef = t3
sf6.targetRef = pg2
sf7 : SequenceFlow
sf7 .name = "task4Topg2"
sf7 : pl.flowElements
sf7.sourceRef = t4
sf7.targetRef = pg2
sf8 : SequenceFlow
sf8 .name = "pg2Totask5"
sf8 : pl.flowElements
sf8.sourceRef = pg2
sf8.targetRef = t5
sf9 : SequenceFlow
sf9 .name = "task5Totask6"
sf9 : pl.flowElements
sf9.sourceRef = t5
sf9.targetRef = t6
sf10 : SequenceFlow
sf10.name = " task6Totask7"
sf10 : pl.flowElements
sf10.sourceRef = t6
sf10.targetRef = t7
sf11 : SequenceFlow
sf11.name = "task7Toend"
sf11 : pl.flowElements
sf11.sourceRef = t7
sf11.targetRef = ee
```

The following shows a trace of the execution of the transformation on this model:

```
Model loaded
Entering startTotask1
Left startTotask1
Entered Create the RFC
Left task Create the RFC
Entered flow task1Totask2
Left task1Totask2
Entered Review, Assess and
Evaluate
Left task Review, Assess and
Evaluate
Entered flow task2Topg1
Entered parallel pg1
Left pg1
Entering pg1Totask3
Left pg1
Entering pg1Totask4
Left pg1Totask3
Entered Change Advisory
Board Authorization
Left pg1Totask4
Entered Emergency Change
Advisory Board Authorization
Left task Change Advisory
Board Authorization
Entered flow task3Topg2
Left task Emergency Change
Advisory Board Authorization
Entered flow task4Topg2
Entered parallel pg2
Left pg2
Entering pg2Totask5
Left pg2Totask5
Entered Receive and Plan
Change
Left task Receive and Plan
Change
Entering task5Totask6
Left task5Totask6
Entered Coordinate and
Implement
Left task Coordinate and
Implement
Entering task6Totask7
Left task6Totask7
Entered Publish
Implementation Results
Left task Publish
Implementation Results
Entered flow task7Toend
```

```
Leaving task7Toend                    Finished process instance
```

The aforementioned textual workflow indicates consistency with explanations provide in section 5.1 whereby each step is thoroughly described. The benefits such an approach can realise in project management regard formalised models which respect project management processes modelled in BPMN. The textual representation of these models can potentially lead to advantages described in section 1.

6 Conclusions and Future Work

This paper attempts to highlight the potential research areas that can extend MDA aspects on DSM to business-specific model and more specifically project management. There are suggestions [19], that corporate decision analysis and decision making leading to changes, can be linked to business needs and improved decision making techniques by adopting approaches in model driven environments for software development.

Developing an integrated methodology on the marriage of MDA and business models is a multi-faceted issue. The paper proposes the utilisation of a renewed project-based approach that will form part of a structured treatment to business models and contribute to increased clarity and formality.

The proposal includes two layers; that can signify business oriented solutions and result to modelled decisions and management guidance documentation.

Extended research in the area of project-based automation could include transformations that support the full BPMN 2.0 elements set i.e. flow objects, connecting objects, swim lanes, artifacts. Future work should also focus on other practices outside MDD such as business analysis and service management. The MDA and BPM communities have taken steps towards attaining a more business-oriented approach to identified parts of projects which can be standardised. However, it is the authors' belief that building on the already available knowledge of both research communities, there are valuable lessons to learn and apply to other standardised business frameworks such as PRINCE2® for project management, BABOK® for business analysis and even ITIL® for service management.

References

1. Denno, P., Steves, M. P., Libes, D., Barkmeyer, E. J.,: Model-Driven Integration Using Existing Models. IEEE software, IEEE computer society. (2003)
2. Mohagheghi, P., Dehlen, V.: Where Is the Proof? A Review of Experiences from Applying MDE in Industry. In: Schieferdecker, I., Hartman, A. (eds.): ECMFA, Lecture Notes in Computer Science, Vol. 5095, Springer-Verlag, Heidelberg, Germany (2008) 432-443
3. Guttman, M., and J. Parodi.: Real-Life MDA: Solving Business Problems with Model Driven Architecture. Morgan Raufmann (2007)

4. Presso, M. J., Belaunde D.: Applying MDA to Voice Applications: An Experience in Building an MDA Tool Chain. In: Hartman, A., Kreische D. (eds.): ECMDA-FA. Lecture Notes in Computer Science, Vol. 3748, Springer-Verlag, Heidelberg, Germany (2005) 1-8
5. Staron, M.: Adopting Model Driven Software Development in Industry – A Case Study at Two Companies. In: Nierstrasz, O., Whittle J., Harel, D., Reggio, G. (eds.): MODELS. Lecture Notes in Computer Science, Vol. 4199, Springer-Verlag, Heidelberg, Germany (2006) 57-72
6. Bloomfield, T.: MDA, Meta-Modelling and Model Transformation: Introducing New Technology into the Defence Industry. In: Hartman, A., Kreische D. (eds.): ECMDA-FA. Lecture Notes in Computer Science, Vol. 3748, Springer-Verlag, Heidelberg, Germany (2005) 9-18
7. Burgstaller, R., Wuchner E., Fiege L., Becker M., Fritz T.: Using Domain Driven Development for Monitoring Distributed Systems. In: Hartman, A., Kreische D. (eds.): ECMDA-FA. Lecture Notes in Computer Science, Vol. 3748, Springer-Verlag, Heidelberg, Germany (2005) 19-24
8. Baker, P., Loh, S., Weil, F.: Model-Driven Engineering in a Large Industrial Context – Motorola Case Study. In: Briand, L.C., Williams, C. (eds.): MoDELS. Lecture Notes in Computer Science, Vol. 3713, Springer, Heidelberg, Germany (2005) 476-491
9. M1 Global Solutions.: Model Driven Software Development and Offshore Outsourcing. (2004)
10. Seidewitz, E.: What models mean. IEEE Software, Vol.20, Issue 5 (2003) 26
11. Bran, S.: The pragmatics of model-driven development. IEEE Software, (2003) 20:19
12. Anneke, G., Kleppe, J. W., Bast, W.: MDA Explained: The Model Driven Architecture: Practice and Promise. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (2003)
13. OMG, Meta Object Facility (MOF) Specification, 2003. Object Management Group, http://doc.omg.org/formal/02-04-03
14. Weis, T., Ulbrich, A., Geihs, K.: Model Metamorphosis. IEEE software, IEEE computer society (2003)
15. Kalnins, A., Vitolins, V.: Use of UML and Model Transformations for Workflow Process Definitions, Databases and Information Systems. In: Vasilecas, O., Eder, J., Caplinskas, A. (eds.): BalticDB&IS'2006. Vilnius, Technika (2006) 3-15
16. Heitkoetter, H.: Transforming PICTURE to BPMN 2.0 as Part of the Model-Driven Development of Electronic Government Systems. In Proceedings of HICSS (2011) 1-10
17. Macek, O., Richta, K.: The BPM to UML activity diagram transformation using XSLT. In Proceedings of DATESO (2009) 119-129
18. Roebuck, K.: Model-driven Architecture (MDA): High-impact Strategies - What You Need to Know: Definitions, Adoptions, Impact, Benefits, Maturity, Vendors, USA (2011) 88-90
19. Apostolopoulos, H., Maroukian, K.: Model Driven Architecture Transformation for Business Models: Decision Analysis Based on a Collateral Analysis Model. 13th International Conference on Informatics and Semiotics in Organisations, Leeuwarden, The Netherlands (2011) 33-39
20. Lano, K.: Advanced Systems Design with Java UML and MDA. Elsevier, Oxford, UK (2005)
21. Kolovos, D., Paige, R., Rose, L., Polack, F.: The Epsilon Book, Structure (2010) 178
22. Lano, K., Poernomo, I., Kolahdouz-Rahimi, S.: Comparative Evaluation of Model Transformation Specification Approaches. International Journal of Software and Informatics, Vol. 6, Issue 2, (2012) 233-269