VILNIUS UNIVERSITY

MARTAS AMBRAZIŪNAS

ENTERPRISE MODEL BASED MDA INFORMATION SYSTEMS ENGINEERING METHOD

Summary of doctoral dissertation

Physical Sciences, Informatics (09 P)

Vilnius, 2014

This doctoral dissertation was written at Vilnius University in 2010-2014.

Scientific supervisor
Prof. Dr. Audrius Lopata (Vilnius University, Physical Sciences, Informatics – 09 P).

The defence of the dissertation will take place at a public meeting of the Counsil of Informatics Sciences of Vilnius University in the auditorium 203 at the Vilnius University Institute of Mathematics and Informatics at 12 a.m. on 29 September, 2014.

Address: Akademijos st. 4, LT- 08663, Vilnius, Lithuania.

The summary of the doctoral dissertation was distributed on 29 August, 2014.

A copy of the doctoral dissertation is available at the Library of Vilnius University.

VILNIAUS UNIVERSITETAS

MARTAS AMBRAZIŪNAS

VEIKLOS ŽINIŲ BAZE IŠPLĖSTOS MODELIAIS GRINDŽIAMOS ARCHITEKTŪROS TAIKYMO INFORMACIJOS SISTEMŲ INŽINERIJOJE METODAS

Daktaro disertacijos santrauka

Fiziniai mokslai, informatika (09 P)

Vilnius, 2014

Disertacija rengta 2010 – 2014 metais Vilniaus universitete.

Disertacija bus ginama viešame Informatikos mokslo krypties tarybos posėdyje 2014 m. rugsėjo 29 d. 12 val. Vilniaus universiteto Matematikos ir informatikos instituto 203 auditorijoje.

Adresas: Akademijos g. 4, LT- 08663, Vilnius, Lietuva.

Disertacijos santrauka išsiuntinėta 2014 m. rugpjūčio mėn. 29 d.

Disertaciją galima peržiūrėti Vilniaus universiteto bibliotekoje ir VU interneto svetainėje adresu: www.vu.lt/lt/naujienos/ivykiu-kalendorius

SUMMARY OF DOCTORAL DISSERTATION

INTRODUCTION

The relevance of the work

The majority of IT project failures (about 68% [9]) are caused by inconsistent user requirements and insufficient problem domain analysis. Although new methods of information systems engineering (ISE) are being researched and developed, they are empirical in nature: the project models repository of CASE system is composed on the basis of enterprise problem domain. The problem domain knowledge acquisition process relies heavily on the system analyst and user; therefore it is not clear whether the knowledge of the problem domain is comprehensive. The expert plays a pivotal role in the problem domain knowledge acquisition process, and few formalized methods of knowledge acquisition control are taken into consideration. Currently, despite the existing tools and CASE systems, user requirement analysis largely depends on the expertise of system analyst and the user. The knowledge stored in repository of CASE tool is not always verified through formalized criteria, thus it is necessary to use advanced user requirements specification and validation techniques that ensure iterative knowledge acquisition process during which the missing or incorrect data elements are obtained and fixed. Another challenge that should be handled is knowledge transferring (transformation) process from one software development stage to another. This process heavily depends on skill of the systems analysts and other specialist. Wrong decisions or interpretation may lead to occurrence of logical gaps, thus causing various issues for the whole project. To reduce the empirical factors impact for the mentioned process OMG (Object Management Group) provided Model Driven Architecture (MDA [28]) approach to software engineering where MDA focuses on functional requirements and system architecture not on technical details only. Model Driven Architecture allows long-term flexibility of implementation, integration, maintenance, testing and simulation of software. It means that user requirements specification and enterprise modelling stages of software development life cycle have not been covered enough by MDA yet. There is lack of formalized problem domain knowledge management and user requirements acquisition techniques for composition and verification of MDA models. In order to solve this problem enhancement of MDA approach by the best practices [10], [11], [16] of Knowledge Base IS engineering (including Enterprise Knowledge repository) can be used. Knowledge based software development methods can ensure that the collected user requirements are validated against Enterprise meta-model and rules defined by this meta-model. This validation is reducing risk of project failures caused by inconsistent user requirements and insufficient problem domain knowledge verification.

Overview of scholarly investigation of the problem

User requirements acquisition techniques heavily depend on the selected software development approach. Two main branches of software development approaches are depicted in literature: Plan Driven [36] and Agile [5]. Plan driven software development is formal with strictly defined stages and required outputs at the end of each stage. *Heavy* and formal requirements acquisition processes are used.) This means that a comprehensive documentation is created that includes requirements specified by text and various visual models etc. [35]. Different approach for requirements specification process are being used in agile methods. The main focus of Agile methods is usage of iterative requirements acquisition process and more investing into employees' expertise rather than into developing precise requirements acquisition procedures [22]. This approach allows adapting to rapidly changing user requirements or problem domain environment, but it lacks capability to forecast resources and time needed for the project to complete. Model driven software engineering methods [2] are heavily relying on system models creation and usage in software development process. These methods require specific knowledge of modelling languages and models transformation. Requirements are specified by using visual models created using domain specific or general purpose modelling languages. The main element of Knowledge Based software development methods is Enterprise model which is used to store the collected knowledge (requirements including) about the problem domain and developed system. There are many different enterprise modelling standards including: CEN EN 12204, CEN EN 40003 (CIMOSA), UEML. Enterprise Meta-Model [13], developed by a joint scientific group of Vilnius University and Kaunas University of Technology, was used in the research. Enterprise Meta-Model's internal structure is based on Control Theory [14] and best practices of the above mentioned enterprise modelling standards. Several authors

[24], [40] have recently proposed EMM based UML models generation methods in order to decrease empirical nature of Information Systems Engineering process. These methods are able to cover some aspects of MDA but not fully support the whole process. Changes are necessary to be made in order to fully integrate Enterprise model into MDA process. This is the main objective of the current research.

The **object** of research is the enterprise model's integration into MDA based process.

The **aim** of the research is to create the MDA and Knowledge Based software development method that will allow validation of the acquired problem domain knowledge against formal criteria.

For the aim of the research to be achieved the following tasks have been set:

- to analyse software development processes and define impact of empirical factors to requirements acquisition and business modelling stages;
- to determine the advantages and disadvantages of MDA based methods during requirements acquisition and business modelling stages;
- to create a new Knowledge Based MDA software development method:
    - to define necessary changes to Enterprise model in order to integrate it into MDA process;
    - to create mappings between MDA models and Enterprise model;
    - to specify Knowledge Based MDA method‘s process.
- to create CASE tool's prototype in order to experimentally check the efficiency of the proposed method;
- to experimentally examine the efficiency of the proposed method:
    - to create application for mobile devices by using the proposed method and CASE tool prototype;
    - to document process of mobile application development and prepare/draw conclusions and make suggestions;
- to summarize the results of the performed theoretical and empirical research and formulate the conclusions.

The assumptions of the dissertation

- Enhancement of MDA process with Enterprise Model will improve the quality of software documentation and code by these aspects: the requirements will be specified using visual models created using de facto standard modelling languages (SysML, UML), the created models will be validated against formal criteria using Enterprise Meta-Model, automatic programing code generation from validated models will ensure the consistency and will reduce logical gaps as well as will ensure code standardization in objects naming and other aspects.
- Knowledge Base MDA method will reduce the time consumption when developing application with the same functionality for multiple platforms.

The novelty of the research

Knowledge Based MDA method combines main principles of Knowledge based and MDA based ISE methods. This provides opportunity to create advanced software development approaches especially for applications that should run on multiplatform systems. During the research the existing Enterprise Meta-Model [13] was extended with new elements and relations, thus improving its capabilities to store problem domain knowledge as well as use it for acquainted knowledge validation. In order to perform transformations among MDA CIM, PIM and EM, new models transformation algorithms and mappings were created.

Practical application of the work

Application of the Knowledge Based MDA method provides the possibility to use additional models validation techniques that are defined by Enterprise Meta-Model. This allows system analyst and architect to work with SysML and UML models and have Enterprise Model based validation of specified requirements. Knowledge Based MDA tool's prototype was created during the research. The prototype is capable of transforming Use Case and Activity UML models to Enterprise model as well as validating models consistency. The prototype was developed with intention to expand it to fully working application that can be integrated as plug-in into software modelling tools in order to improve the created models quality.

The methodology of the research

The following methods were applied in the present research: systemic and comparative analysis of scholarly literature, development of prototype, visual modelling, and case study analysis. For empirical research the following tools were used: Microsoft Visual Studio and Eclipse software development environments, Microsoft database management software SQLServer, MagicDraw UML and MS Visio modelling tools.

The structure of the work

The first part embraces the analysis of scholarly literature where the focus is laid on the requirements acquisition and enterprise modelling stages of the software development processes. *The second part* presents the Knowledge Based MDA method. In this part a detailed description and composition of the provided method is defined including: description of models used, models mappings, method process, and model transformations. *The third part* presents CASE tool's prototype (architecture and functions) that was created to support the suggested method. *The fourth part* presents description of the performed case study, during which real-life application was created for mobile devices, using newly developed method and CASE tool prototype.

REQUIREMENTS SPECIFICATION AND MANAGEMENT PROCESSES

Requirement specification and management stage is one of the most important stages of software development life cycle [9], [23], [17], but in most case this stage is empirical in nature [3], [22] thus various issues may occur (e.g. due to incomplete or inconsistent requirements project plan can be created with high risk of over budget or overtime, or with incorrectly allocated resources). Various software development paradigms are dealing with such challenges by implementing different approaches. *Plan Driven* methods user requirements acquisition process has strictly defined the procedures and heavy usage of documentation (e.g. using Volere [37]). The process is not versatile and there are difficulties in handling requirements that are often changed, so the stability of the problem domain and requirements is one of the key success factors [42]. *Plan Driven* methods can provide more efficient time and resource control by usage of various metrics as well as comprehensive documentation creates better conditions to introduce new employees to projects, thus reducing time needed to familiarize with the product.

The biggest disadvantage of *Plan Driven* methods is necessity of additional resources and time for managing the complicated process. This reduces the possibility to react in fast changing environment. Agile methods such as scrum [21], FDD [12] and others [4] [43] suggest different approach to requirements acquisition and management. The main attention is shifted from tools and procedures [1] to people skill and involvement. Requirements instability is more manageable compared to *Plan Driven* methods [22], in fact the lack of this versatility on Plan Driven methods enforced the evolution of Agile based ISE development approaches. Despite the aforementioned advantages *Agile* methods are heavily dependent on empirical factors (e.g. specialist skill and expertise), the problems may arise when trying to introduce new employees to the product development as the documentation is scarce or in some cases not existing at all. Model Driven IS engineering emphasises the models usage during the whole software development process. During the requirements acquisition stage business models are created. For models creation two types of modelling languages can be used: Domain Specific modelling languages [6] and General Purpose modelling language [15]. Depending on the selected modelling language models can be used as a special kind of documentation or in case of fUML[8] [31] they can be used as code i.e. models can be built and executed. This is a fairly new approach which suggests an idea how to reduce the logical gaps between pure requirements specified by text and some visual models and application code. Knowledge based [26] ISE uses the Enterprise Model and Meta-Model to store knowledge from problem domain. The knowledge includes user requirements for developed system. The requirements acquisition can be performed using ordinary technique i.e. specifying them by text forms and converting to Enterprise Model‘s elements. This transformation can be performed by using mappings and it requires special expertise of the system analyst. There are numerous enterprise models standards like IDEF [25], OMT [38], CIMOSA [16], GERAM [20], dodaf[7], UEML[41] that can be used to store and operate problem domain knowledge. Enterprise model, created by the joint scientific group of Vilnius University and other science institutions, was used in this research. Defined Enterprise Meta-Model is provided in the picture below:

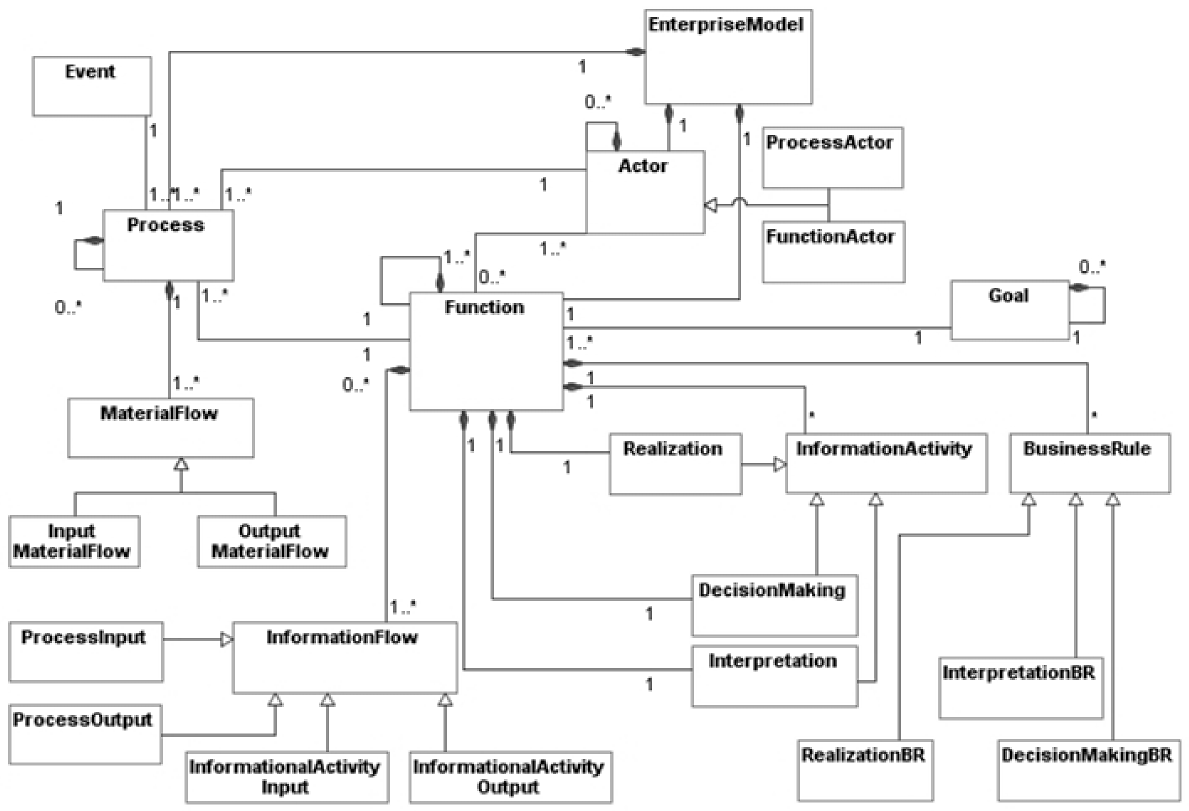


Fig. 1. Main Elements of Enterprise Meta-Model [24]

The Enterprise Meta-Model is based on control theory and its main idea is: each process on the system should have the function that is managing this process. This rule can be expressed by Elementary Manageable Cycle.

Generalization of the first part

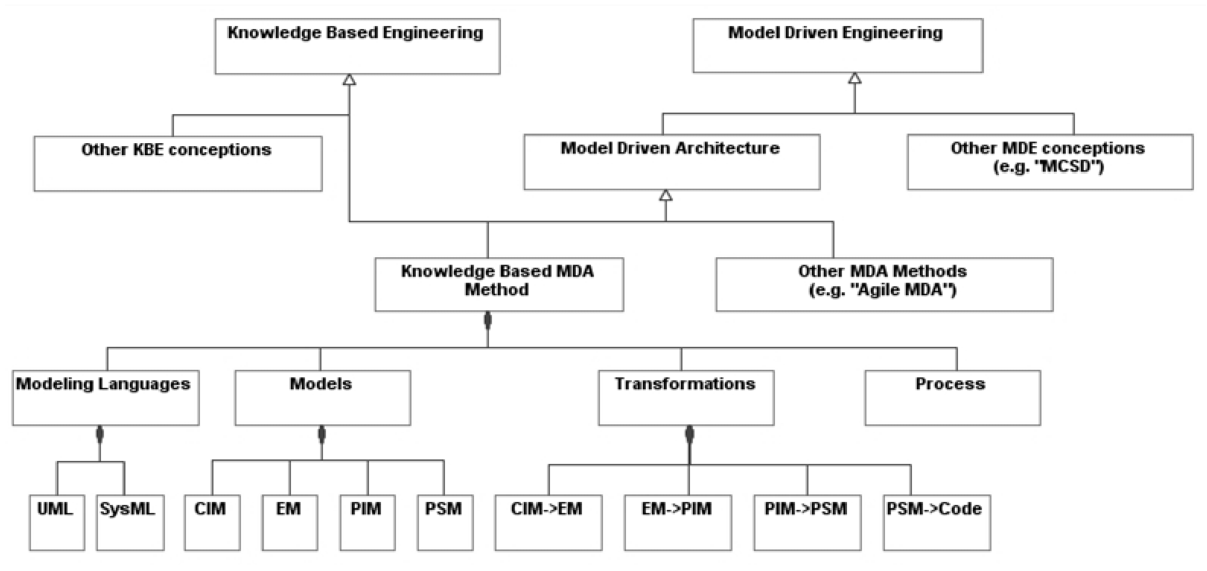
After the examination of various sources the conclusion was made that classical software development approaches deal with requirements specification and validation by implementing one of the two strategies: strictly defining formal procedures and their output as well as document templates which must be followed by employees; and another approach emphasises iterative requirements specification and validation processes that are managed by skilful specialists. These are two main paradigms and the solutions implemented by particular organizations may have attributes of both, but one will still be dominating. Both approaches are empirical in nature i.e. they depend on expertise and skill of specialists (although Plan Driven methods is less dependent). To reduce the dependency of empirical factors on software development process, Knowledge based ISE principles can be used. The core idea of Knowledge-Based ISE is as follows: problem domain knowledge is validated against formal criteria and stored in

Enterprise Model (which is part of Knowledge-Based subsystem consisting of Enterprise Model, Enterprise Meta-Model and transformation algorithms). The integration of Knowledge Based subsystem into classical ISE methods is not a trivial task as it requires special skills and expertise of specialist as well as additional CASE tools. These requirements may not always be met by traditional Agile or Plan driven methods or may require too many changes to particular method to be relevant. Thus natural integration can be done with model based software development approaches as these methods are using models and models transformations as the core concept. MDE methods for user requirements acquisition are using various models created with different notations (e.g. WF, SysML, UML). Thus models cannot be validated against formal criteria in a universal way. Models consistency validations can be implemented, but Enterprise Model provides opportunity to test the collected requirements independently from initial notation. Knowledge Based MDA method which combines the basic principles of Model Driven and Knowledge Base software development approaches can address the issue of empirical nature of requirements by introducing requirements validation against formal criteria, automatic generation of architectural models and programming code.

KNOWLEDGE BASED MDA METHOD

Knowledge-Based MDA (KB-MDA) software development method combines the basic principles of Model Driven ISE (MDE) and Knowledge Based ISE (KBE). MDE defines the main ideas of domain models usage in software development process and KBE defines Enterprise Model's and Meta-model's usage in software development process. By combining these two approaches we are creating new method that is inheriting main components from both parent concepts. Knowledge-Based MDA extends traditional MDA with additional model (Enterprise Model) and two transformations (CIM to EM and EM to PIM). Thus KB MDA defines four types of models: CIM, EM, PIM, and PSM. CIM specifies system requirements of a particular problem domain (it can also be named as Business Model). Usually this model is created by a system analyst and user, therefore CIM's nature is empirical. The following modelling languages are selected for creation of Knowledge-Based MDA visual models: SysML [32] and UML [33]. SysML is used to capture user requirements by creating four types of diagrams

(Use Case, Activity, Block Definition and Requirements). UML diagrams (Class, Sequence, and Use Case) are used for PIM and PSM models. The place of KB- MDA method (and components) in ISE hierarchy is presented in Fig. 2.

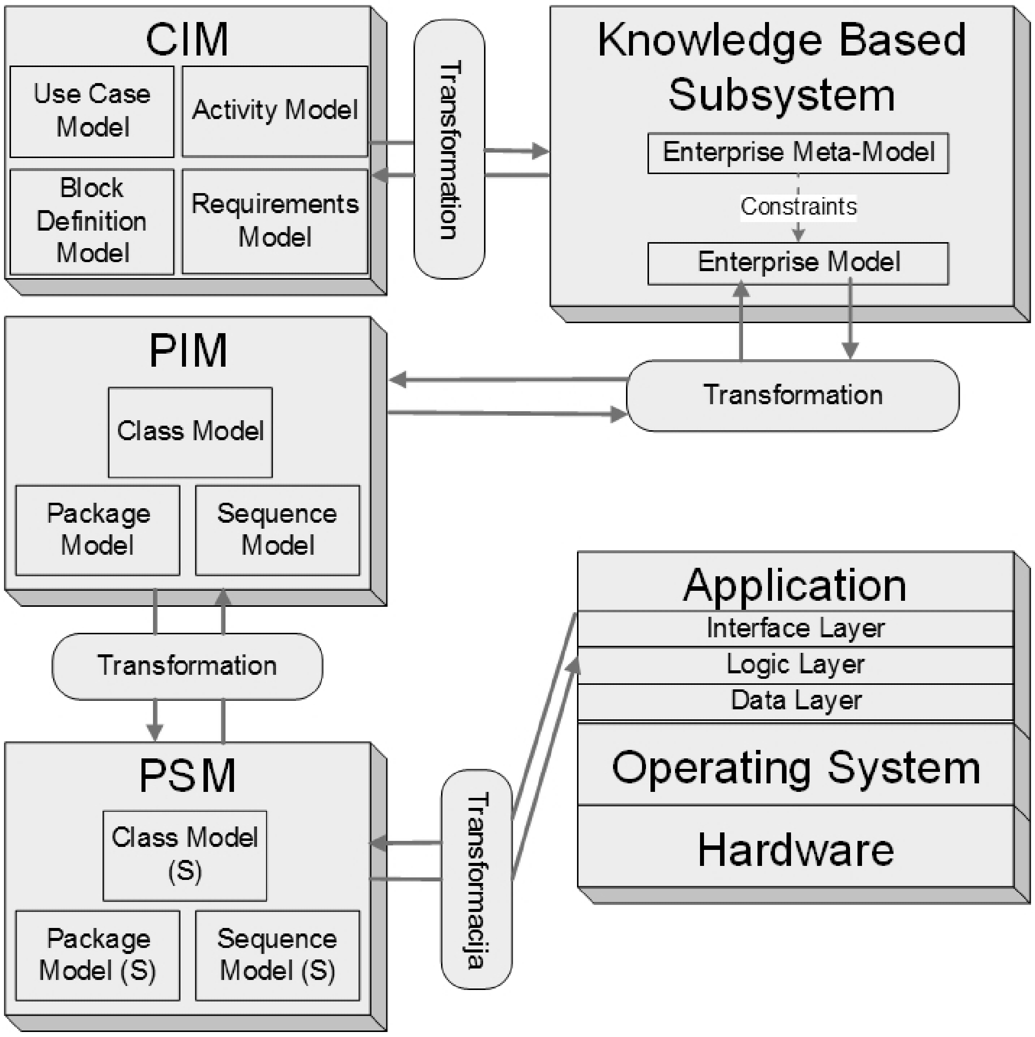


Knowledge Based Engineering
Model Driven Engineering
Other KBE conceptions
Model Driven Architecture
Other MDE conceptions (e.g. "MCSD")
Knowledge Based MDA Method
Other MDA Methods (e.g. "Agile MDA")
Modeling Languages
Models
Transformations
Process
UML
SysML
CIM
EM
PIM
PSM
CIM->EM
EM->PIM
PIM->PSM
PSM->Code

Source: created by the author of the dissertation.

Fig. 2. The place of Knowledge Based-MDA method in ISE hierarchy

PIM provides system's architecture and functionality without platform specific information and technical details that are used in design stage. PSM is constructed on the basis of PIM enhancing it with platform specific details e. g. information about object types, design patterns and other implementation and deployment information. PSM is the main data source for the programing code generation process. Enterprise Model is a part of the element called Knowledge-Based subsystem. This subsystem also contains EMM and transformation algorithms that handle CIM to EM and EM to PIM transformations. Physically EM is implemented as database different from CIM, PIM, and PSM that are visual models. Knowledge-Based MDA process involves five actors: User, System Analyst, Knowledge-Based subsystem, MDA model transformation (generation) tool, and Developer. User is participating in the first stage of software development (CIM creation) by providing requirements and requirements related data. Based on the information provided System Analyst creates SysML models that represent particular requirements area i.e. Use Case model and Activity model for functional requirements, Block Definition model for Systems structure, and Requirements model for non-functional requirements. After all internal CIM models are created, syntactic and semantic models validation is performed. The elements of Knowledge-Based MDA method are presented in the picture below:



Source: created by the author of the dissertation.

Fig. 3. Conceptual Knowledge-Based MDA method's architecture

After validation is successfully completed, Knowledge-Based subsystem creates Enterprise Model based on the existing SysML models. The next stage is Enterprise Model's validation against Enterprise Meta-Model. This process produces validation report. Validation report is analysed by System Analyst and a particular decision is made whether to proceed with the next steps or repeat the initial steps. If the decision is made to proceed, System Analyst selects a particular area for which programing code will be generated. Based on this decision the PIM and PSM models are created. From the PIM layer generation the process follows standard MDA process with one exception: any changes that are made to internal PIM or PSM models must be synchronized with Enterprise Model.

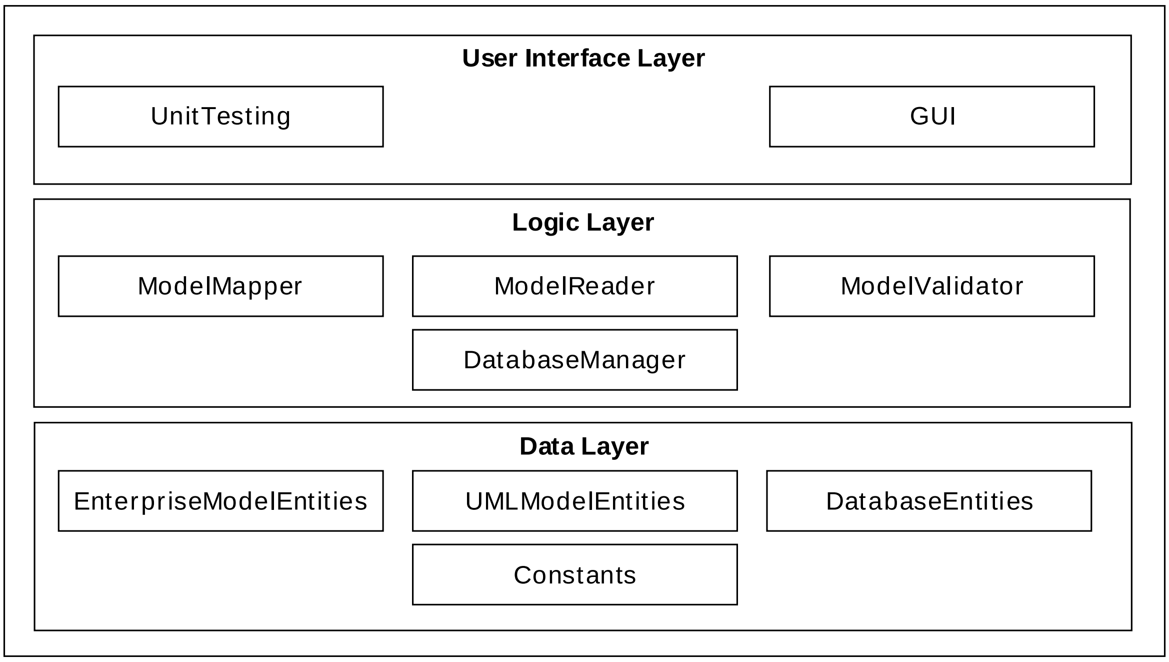
Generalization of the second part

Knowledge Based MDA method is created by enhancing MDA approach with the main elements from Knowledge base ISE – Enterprise Model and Enterprise Meta-Model. Traditional set of three MDA models (CIM, PIM, PSM) is complemented with Enterprise Model, which is created by following the rules defined in Enterprise-Meta model. In order to transfer data from CIM models to Enterprise Model and from Enterprise Model to PIM model, transformation algorithms are created. In conclusion, knowledge based MDA method consists of the following elements:

- Modelling languages: SysML and UML;
- Models: CIM, EM, PIM, PSM, Code;
- Transformations and mappings: CIM => EM, EM=>PIM, PIM=>PSM, PSM=>Code;
- Knowledge Based MDA method process.

KNOWLEDGE BASED MDA TOOL's PROTOTYPE

In order to test the efficiency of the created ISE method Knowledge Based MDA tool's prototype was created. Prototype is capable of parsing and processing common to SysML and UML Use Case and Activity diagrams, validating consistency of these models as well as transforming models' elements to Enterprise Model's elements, and validating Enterprise Model according to function-process rule. The Use Case and Activity diagrams must be created using "Magic Draw" modelling tool [30] and saved in XMI [34] format. Prototype consists of ten separate modules connected by dependencies. All modules can be assigned to one of the three categories: UI Layer, Logic Layer, and Data Layer. The architecture is created in such way that most elements can be reused as separate libraries. Architecture of the prototype is depicted in the picture below:



Source: created by the author of the dissertation.

Fig. 4. Knowledge-Based MDA tool prototype's components

UI Layer:

- *GUImodule* contains user interface elements such as forms and dialogs. This module is responsible for handling system's interaction with user. User interface is organized using dialogs. All dialogs can be divided into two main categories: input dialogs and output dialogs.
- *UnitTesting* module contains classes and routines that are used for testing purposes only. Unit testing is used to automate testing procedures and test application functionalities such as model parsing, validation, and mapping, entities saving and loading to DB. The main purpose of unit testing is to maintain application consistency and integrity during development.

Logic Layer:

- *ModelParsers* module contains objects for UML models parsing from XMI format and creating *UMLModelsEntities* in prototype. This module contains objects and routines used for XMI file parsing. Each UML model type (Use Case, Activity, and Class) should have its own parser. All parsers implement IParser interface and are created using Factory Design Pattern. Parser Factory is used to return particular Parser depending on model type. Each Parser can have various methods for parsing different models as models internal structure is different. The parsers are using

Response/Request data input output patterns. Each parser responses should contain not only information about parsed UML model's objects but as well as status about parsing process and error messages if some exceptions were encountered. ProjectParser is used to parse the whole XMI file. All newly introduced parsers should implement IParser interface and be included into ParserFactory and ProjectParser routines.

- *ModelValidator* module is responsible for various UML models validations (e.g. consistency). This module contains objects and routines used for UML models validation. Currently model consistency validation for Use Case and Activity models is implemented in the prototype. Consistency validation as input uses two models Main model and Mirror model. Main model is the model from which all elements must be in Mirror model. For example, consistency validation checks if elements from one model e.g. (Use Case Actor) are represented in another model (e.g. Activity Swimline). Model consistency validation can be used to resolve syntactic (e.g. misspelled element name) or semantics (missing element) error. All newly introduced validators should implement IValidator interface and be included into ValidatorFactory routines.
- *ModelMapper* module contains objects that are used for *UMLModelEntities* transformation to enterprise model (EM) Entities. This module provides UML models conversion to EM functionalities. Model Mapper takes as input UMLModelObjectInfo entities and creates EnterpriseModel ObjectInfo entities. A set of predefined and user selected options can be used for conversion. Model Mapper is created as static class as its functions are not model specific. Model Mapper has information about all UMLModelObjectInfo objects and EnterpriseModelObjectInfo objects and matches these objects using the defined parameters.
- *Database Manager* module handles *EnterpriseModelEntities* transformation to DatabaseEntities and saving/loading to enterprise model database. DatabaseManager module provides database interaction functionalities. Each EM entity has its own data manager (e.g. ActorManager, FunctionManager etc.). Parts of the routines are common like save and load object by id, but most of them are dependent on object type. CommonManager is used for actions that are common for all objects or as a

helper for custom actions.

Data Layer:

- *UML Model Entities* are created by parsing XMI file. These entities are used for model validation as well as data input for EM Entities generation. *EM Entities* are used for EM analysis and validation. These are the building blocks of EM. In this chapter modules that contain data layer objects are described. UMLModelEntities module contains objects that are used for storing and manipulating UML model data. In fact, all parsed data are stored in these entities. Basically, the names of these entities are the same as UML model element type e.g. UMLUseCase, UMLActor, UMLAction etc. UMLModelObjectInfo is the base class from which all other classes are derived. This class contains basic UML model object's information such as Id, Name, UMLObjectType and source. EnterpriseModelEntities module contains entities that are created from UMLModelEntities after mapping process was performed. These entities have specific the EM information that was appended during mapping process. Using EM Entities Enterprise Model validation and various analyses can be performed. There is one base element EnterpriseModelObjectInfo from which all other elements are derived. This element contains basic enterprise model element's information such as Name, Type and Source. *Constants* module contains static information for the application such as Constants, Enums, Messages, etc.
- *Database Entities* contains entities used by ORM for DB related actions. DatabaseEntities contains objects that are created by ORM Mapper (in particular case, Net Entity Framework). Each Database table has its own representing object. These objects provide functionality to manipulate database data in application. DatabaseManager classes have routines to query database tables, save, update or delete records as well as perform other actions. The described entities provide architecture backbone for prototype to be able to read Use Case and Activity models information from XMI file, validate these models, map to enterprise model elements as well as perform Database related actions. Constants module is created as main source of static information (such as model type's definitions, object types definitions, messages, error codes etc.). Data are saved as Constant or Enum variables. In most cases this module is reused by Logic Layer's objects.

Generalization of the third part

Knowledge Based MDA method tool's prototype was created by using three layers architecture (data layer, business logic layer, user interface layer). Each layer consists of several packages divided by performed functions. Data layer was implemented using *MS SQL Server Express* database management system. Business logic layer and user interface layer were implemented using *MS .Net Framework*. Prototype is capable of reading and processing XMI files created by MagicDraw modelling tool, validating consistency of Activity and Use Case models, transforming these models' elements to Enterprise Model's elements and validating Enterprise Model according to function-process rule. Prototype was designed having in mind expansion possibilities as in long term perspective prototype should be developed in order to create plug-in for the selected modelling tool.

EMPIRICAL STUDY: POST TRACKING APPLICATION

According to statistical reports on 2014 year leading Operating systems of mobile devices are *„Android"*, *„iOS"*, *„Windows Phone"*. All three operating systems contain ~95% of mobile devices market [29], thus for organisation to reach full market potential applications should be created at least for dominating platforms (e.g. *„Android" and „iOS"*). Applications for these platforms are created by using different programing languages (for *„Android"* platform applications are created using *„Java"* programming Language, for *„Windows Phone"* - *.Net* Framework and C# language, for *„iOS"* is used *Objective-C* programing language). So the main goal when creating application for different mobile devices is to maintain the same functionality (*feel and look*), but to achieve this by using different programing languages. In ideal case the application should be created by using high level of abstraction, platform independent programing language. This requirement can be met by MDA based software development methods. In these methods requirements are specified by creating Computation Independent Model and in the next stage CIM model is transformed to technical Platform Independent Model, which contains basic structure and behaviour describing elements but without implementation attributes. When developing software for multiplatform systems the requirements quality becomes even more important success factor than in single platform systems as any issues will be spread through all platforms and fixes will needed to be made on all platforms too. This may mean big time and resource losses and overall risk of project failure. Requirement quality can be ensured by performing additional requirements validation, e.g. by using principles of Knowledge Base ISE. So, both criteria (usage of high level of abstraction, platform independent programing language and additional requirements validation) are met by Knowledge Based MDA software development method. The application was created by following Knowledge Based MDA method steps. Initially CIM model consisting of four SysML (Use Case, Activity, Block Definition and Requirements) models was created to capture user requirements. By using KB MDA tool's prototype Activity and Use Case models were cross-validated for missing or inconsistent elements. After the validation was completed CIM elements were converted to Enterprise Model's elements, thus allowing to perform internal Enterprise Model validation base on function-process rule. After this validation was completed, the elements from Enterprise model were transformed (manually) to PIM internal models (Class, Sequence and Package models were created). PIM's class model was annotated with platform specific elements (Java language attributes) in order to generate programming code, packages were used as namespace defining element and sequence diagrams defined the calls of class methods.

Generalization of the fourth part

The creation of SysML models to capture user requirements is time and effort consuming process. The CIM model‘s level of abstraction should be such that automatic models validation can be performed. This means that at least two CIM models (CIM0 and CIM1) should be created there. CIM0 is high abstraction and contains elements that are fully understandable to customer, CIM1 is created based on CIM0 but with more details which allows performing models validation and transformation. Using the created prototype models consistency validation was performed on Use Case and Activity models. This allowed to identify semantic (e.g. missing block in Block Definition model) and syntactical (e.g. different element names) errors and inconsistencies. Validation against Enterprise Meta Model allowed to separate data validation and data transformation processes, thus allowed to identify the missing data validation function. Due to time consuming CIM models creation process there is no time saving (compared to Plan Driven or Agile) methods when developing application for a single platform.

Despite that, judging based on personal work experience time economy would be achieved when creating application for two or more platforms. The reasons for this is as follows: detailed and consistent requirements specification process, unified requirements storing format (Enterprise model), detailed SysML/UML models that are created with reduced logical gaps, automated code generation from these models and shorter testing stage.

CONCLUSIONS AND SUGGESTIONS

After performing the research, the following conclusions can be made:

1) The analysis of various sources led to the conclusion that user requirements acquisition stages of classical software development approaches are empirical in nature and are heavily dependent on skill and expertise of systems analyst and involvement of user. Thus the decision was made to create ISE method that would have reduced impact of empirical factors on software development process.
2) In order to reduce the impact of empirical factors on software development process, the decision was made to use Knowledge Base IS engineering approach. The main advantage of this approach is the possibility to validate specified user requirements against formal criteria, in that way reducing the possible issues and ensuring more efficient software development process compared to classical software development methods.
3) Despite the existing advantages during requirements validation stage the integration of Knowledge Based components into traditional ISE methods is complicated. The main element of Knowledge Based ISE – the Enterprise Model has no direct representation in traditional software development methods, thus requires special skills and expertise to be used.
4) To solve this problem Model Driven software development methods can be used. MDE methods are heavily using models and models transformations during software development process. This allows integrating Enterprise Model with less effort as well analysts are familiar with basic model elements.
5) Knowledge Based MDA method is developed by using basic MDA approach and enhancing it with the main elements of Knowledge based ISE (Enterprise model, Enterprise Meta-Model, model transformation algorithms). To achieve this goal the following tasks were completed:
    1. Selected Enterprise model was complemented with new elements that allowed integration with internal MDA models (CIM, PIM);
    2. Mappings CIM=>EM and EM=>PIM were created;
    3. Model transformation algorithms CIM=>EM and EM=>PIM were created;

4. Knowledge Based MDA tool‘s prototype with basic functionalities required by new method (models transformation and consistency validation) was created;
5. By using the developed prototype empirical research was performed in order to validate the efficiency of the new method. The results of the research showed that the created method can be applied for development of applications for multiplatform systems and for applications with low tolerance of errors. The conclusions were made after performing the case study – creating real life post parcels tracking application for mobile devices („Android“platform).

6) During the empirical research it was established that by using Knowledge Based MDA method software requirements quality is improved and comprehensive documentation is created due to Enterprise Model based validation, occurrence of logical gaps between software development stakeholders is reduced, time consumption needed for creation of application for multiplatform systems is reduced due to automated code generation and shorter testing stage.

DAKTARO DISERTACIJOS SANTRAUKA

Darbo aktualumas

Šiuolaikinius IS inžinerijos metodus (pvz. *RUP[19], MDA[28], SCRUM[39]*) nuolat vysto ir tobulina IT srities bendruomenės (pvz. *Agile Alliance[1], OMG*), komercinės (pvz. *IBM[18], Microsoft[27]*) bei mokslo organizacijos, tačiau iš esmės šie metodai yra grindžiami empiriniais procesais t.y. dalykinės srities žinių surinkimo veikla priklauso nuo sistemų analitiko bei kliento (užsakovo) kompetencijos ir patirties. Praktinio darbo patirtis leidžia teigti, jog empiriškai išgautų žinių kokybė gali būti nepakankama sėkmingam projekto įgyvendinimui, nes netikslus ir/arba nepilnas vartotojo reikalavimų surinkimas neigiamai įtakoja visus programinės įrangos kūrimo etapus, o tai lemia didesnes laiko, žmogiškųjų, finansinių ir kitų išteklių sąnaudas bei didina projekto įgyvendinimo riziką.

Darbo objektas – Veiklos modelio integracija į MDA architektūra grindžiamą programinės įrangos kūrimo procesą.

Darbo tikslas – sukurti IS inžinerijos metodą, kuris įgalins empiriniais būdais surinktas žinias patikrinti formalių kriterijų atžvilgiu.

Tikslui pasiekti iškelti tokie ***uždaviniai***:

- Atlikti klasikinių programinės įrangos kūrimo metodų analizę, vartotojo reikalavimų surinkimo ir veiklos modeliavimo kontekste, siekiant įvertinti empirinių veiksnių įtaką šiems programinės įrangos kūrimo etapams.
- Išanalizuoti žiniomis grindžiamų bei MDA metodų privalumus ir trūkumus, vartotojo reikalavimų specifikavimo ir veiklos modeliavimo etapuose.
- Pasiūlyti žiniomis grindžiamą MDA metodą, iškeltam tikslui pasiekti, pateikiant realizacijos specifikaciją:
    - specifikuoti metode taikomo veiklos modelio pakeitimus, būtinus integracijai į MDA procesą;
    - sukurti veiklos modelio ir MDA modelių sąsajų žemėlapius (angl. *„mappings“*);
    - detaliai specifikuoti žiniomis grindžiamo MDA metodo vykdymo procesą.

- Siekiant patikrinti metodo funkcionalumą, sukurti dalykinės programos prototipą.
- Eksperimentiškai patikrinti siūlomo metodo efektyvumą:
    - taikant metodo funkcionalumą realizuojančios dalykinės programos prototipą bei žiniomis grindžiamo MDA metodo principus, sukurti mobiliems įrenginiams skirtą programėlę;
    - aprašyti siūlomu metodu grindžiamą programėlės kūrimo procesą, pateikti eksperimentinio tyrimo išvadas.
- Apibendrinti teoriškai ir eksperimentiškai gautus rezultatus. Pateikti rekomendacijas dėl siūlomo metodo panaudojimo galimybių ir tolimesnių metodo vystymo krypčių.

Ginamieji disertacijos teiginiai

- MDA architektūros išplėtimas veiklos modeliu padės užtikrinti dokumentacijos ir programinio kodo kokybę šiais aspektais: tiksliau ir pilniau bus specifikuojama dalykinė sritis, veiklos modelio pagrindu sugeneruota dokumentacija teigiamai (standartizuotas programinio kodo generavimas ir susiejimas su vartotojo reikalavimais) įtakos analitiko, projektuotojo ir programuotojo darbo rezultatus.
- MDA architektūros išplėtimas veiklos modeliu sumažins laiko sąnaudas, kuriant identiško funkcionalumo dalykines programas skirtingoms operacinių sistemų platformoms.

Darbo mokslinis naujumas

Darbo metu naujais struktūriniais elementais papildytas VU mokslininkų (bendradarbiaujant su kitomis mokslo institucijomis) sukurtas veiklos metamodelis [24]. Šie pakeitimai atlikti, siekiant veiklos metamodelį integruoti į MDA procesą bei panaudoti dalykinės srities žinių tikrinimui formalių kriterijų atžvilgiu [14] ir projektinių modelių generavimui. Sukurti transformavimo algoritmai, įgalinantys perkelti MDA CIM žinias į veiklos modelį ir veiklos modelyje saugomas dalykinės srities žinias į MDA PIM. Sukurta žiniomis grindžiamos IS inžinerijos ir MDA architektūros apjungimo koncepcija, suteikia galimybę vystyti progresyvius IS inžinerijos metodus, skirtus dalykinių programų ir programėlių, veikiančių daugiaplatformėse sistemose, kūrimui.

Praktinė nauda

Realizuoti vidinių CIM modelių (SysML *Use Case* ir *Activity* modelių*)* transformavimo į veiklos modelį algoritmai. Tai leidžia analitikui ir projektuotojui tikrinti šių modelių elementų tarpusavio sąsajas ir juose saugomų žinių atitikimą veiklos metamodelio taisyklėms. Tokiu būdu analitikas ir projektuotojas gali dirbti su standartiniais SysML ir UML modeliais bei lygiagrečiai naudotis veiklos modelio teikiamais privalumais (centralizuota žinių saugykla, žinių tikrinimas formalių kriterijų atžvilgiu, galimybė atvaizduoti žinias skirtingomis notacijomis).

Darbo metu sukurtas žiniomis grindžiamo MDA metodo dalykinės programos prototipas, kuris realizuoja dalį (*Use Case* ir *Activity* modelių transformavimas į veiklos modelį, tarpusavio suderinamumo tarp šių modelių tikrinimas, modeliuose apibrėžtų elementų, remiantis EVC kriterijais, tikrinimas) teoriškai apibrėžtų žiniomis grindžiamo MDA metodo funkcijų. Prototipas gali būti naudojamas kaip pagrindas kurti įskiepį konkrečiam modeliavimo įrankiui bei suteikti galimybę sistemų analitikams atlikti papildomą modelių tikrinimą veiklos metamodelio atžvilgiu.

Tyrimo metodika

Darbe taikomi teoriniai ir eksperimentiniai tyrimo metodai. Dalykinės srities turinio analizės tyrimas apima sisteminę mokslinės literatūros analizę, nagrinėjančią programinės įrangos kūrimo metodų veiklos modeliavimo bei vartotojo reikalavimų procesus. Taikomi lyginamosios analizės bei apibendrinimo metodai. Teoriniams ir eksperimentiniams tyrimams atlikti buvo naudojamos CASE priemonės (*MagicDraw 17.0, MS Visio 2013*) ir kita programinė įranga (*MS SQLServer*, integruotos programavimo aplinkos *MS Visual Studio* bei *Eclipse).*

Darbo struktūra

Pirmasis skyrius skirtas programinės įrangos kūrimo metodikų apžvalgai vartotojo reikalavimų ir veiklos modeliavimo aspektu. Apžvelgiami tradicinių programinės įrangos kūrimo metodų reikalavimų surinkimo etapai, reikalavimų valdymo bei veiklos modeliavimo žiniomis grindžiamose IS inžinerijos metoduose bei modeliais grindžiamuose IS kūrimo metoduose problematika. Antrajame skyriuje aprašomas siūlomas programinės įrangos kūrimo metodas, jo sudėtis, veiklos procedūros, modelių sąsajų žemėlapiai, modelių transformavimo algoritmai. Trečioji dalis skirta aprašyti siūlomo metodo dalykinės programos prototipą, jo struktūrą ir funkcijas. Ketvirtojoje dalyje aprašomas, siūlomu metodu grindžiamas, mobiliesiems įrenginiams skirtos programėlės kūrimo procesas. Šioje dalyje taip pat pateikiamos eksperimentinio tyrimo išvados.

IŠVADOS IR PASIŪLYMAI

1) Atlikus klasikinių PĮ kūrimo procesų veiklos modeliavimo ir vartotojo reikalavimų specifikavimo etapų analizę, nustatyta, kad analitiko ir projektuotojo patirtis turi didelę įtaką šiuose etapuose vykstantiems procesams, todėl nuspręsta, kad tikslinga sukurti metodą, kuris sumažintų empirinių veiksnių įtaką.
2) Empirinių veiksnių įtakos mažinimui tikslinga taikyti perspektyvią IS inžinerijos šaką – žiniomis grindžiamą IS inžineriją, kurios vienas iš esminių taikymo privalumų yra surinktų dalykinės srities žinių tikrinimas formalių kriterijų atžvilgiu, o tai įtakoja efektyvesnį (lyginant su klasikiniais IS kūrimo metodais) IS inžinerijos procesą (detalesnį vartotojo reikalavimų specifikavimą, mažesnį loginių trukių skaičių tarp programinės įrangos kūrimo dalyvių).
3) Problemai spręsti pasirinkta modeliais grindžiamos IS kūrimo krypties OMG sukurta MDA koncepcija, UML ir SysML modeliavimo kalba ir XMI modelių pernešimo formatas, todėl kad šiomis priemonėmis veiklos modelį galima funkcionaliai integruoti su vidiniais MDA modeliais (CIM ir PIM), išsaugant galimybę automatizuoti IS kūrimo procesą.
4) Žiniomis grindžiamas MDA metodas sukurtas klasikinę MDA architektūrą papildant nauju elementu - veiklos modeliu. Šis veiklos modelis papildytas naujais elementais, kurie įgalina jo integraciją su vidiniais MDA modeliais (CIM, PIM). Sukurti sąsajų žemėlapiai tarp CIM=>VM ir VM=>PIM, suprojektuoti CIM=>VM ir VM=>PIM modelių transformavimo algoritmai, sukurtas modelių transformavimo ir tikrinimo prototipas (panaudos atvejų ir veiklų modeliams);
5) Siekiant nustatyti metodo efektyvumą buvo atliktas empirinis tyrimas, panaudojant sukurtą dalykinės programos prototipą. Tyrimo rezultatai parodė, kad šį metodą tikslinga naudoti daugiaplatformių sistemų kūrimo atvejais bei taikyti sistemoms, kurioms svarbūs kokybės reikalavimai bei išplečiamumo galimybės. Tai grindžiama

atliktu eksperimentu, kurio metu buvo sukurta pašto siuntų stebėjimo programėlė „Android“ platformai.

6) Tyrimo metu nustatyta, kad taikyti žiniomis grindžiamą MDA metodą IS inžinerijoje tikslinga nes: detaliau dokumentuojami vartotojo reikalavimai (jie yra tikrinami formalių kriterijų atžvilgiu), sumažinama loginių trukių atsiradimo galimybė (tarp programinės įrangos kūrimo dalyvių), daugiaplatforminiuose sprendimuose sumažinamos projekto įgyvendinimo laiko sąnaudas (dėka automatinio kodo generavimo iš patikrintų modelių).

Literatūra

1. **Agile Alliance** [interaktyvus]. 2014 [žiūrėta 2014 m. gegužės 06 d.]. Prieiga per internetą: <http://www.agilealliance.org/>
2. **Cabot, J.:** *Clarifying concepts: MBE vs MDE vs MDD vs MDA* [interaktyvus]. 2014 [žiūrėta 2014 m. gegužės 07 d.]. Prieiga per internetą: <http://modeling-languages.com/clarifying-concepts-mbe-vs-mde-vs-mdd-vs-mda/>
3. **Chanda, J.; Kanjilal, A.; Sengupta, S.; Bhattacharya, S.** *Traceability of requirements and consistency verification of UML use case, activity and Class diagram: A Formal approach.* In Methods and Models in Computer Science, ICM2CS 2009, 2009.
4. **CMS Office of Information Service**. *Selecting a development approach* [interaktyvus]. 2009 [žiūrėta 2014 m. gegužės 05 d.]. Prieiga per internetą: <http://www.cms.gov/Research-Statistics-Data-and-Systems/CMS-Information-Technology/XLC/Downloads/SelectingDevelopmentApproach.pdf>.
5. **Cohen, D.; Lindvall, M.; Costa, P.:** *An introduction to agile methods*, Advances in Computers, New York, Elsevier Science, 2004, pp. 1-66. Prieiga per internetą: <http://robertfeldt.net/courses/agile/cohen_2004_intro_to_agile_methods.pdf>
6. **Dalgarno, M.; Fowler, M.:** *UML vs. Domain-Specific Languages* [interaktyvus]. 2008 [žiūrėta 2014 m. gegužės 06 d.]. Prieiga per internetą: <http://www.methodsandtools.com/archive/archive.php?id=71/>
7. **Department of Defence.** *DoD Architecture Framework Version 2.0 Volume 2: Architectural Data and Models Architect's Guide* [interaktyvus]. 2009 [žiūrėta 2014 m. gegužės 05 d.]. Prieiga per internetą: <http://dodcio.defense.gov/Portals/0/Documents/DODAF/DoDAF%20V2%20-%20Volume%202.pdf>
8. **Eichelberger, H.; Eldogan, Y.; Schmid, K.** *A Comprehensive Analysis of UML Tools, their Capabilities and thei Compliance* [interaktyvus]. 2011 [žiūrėta 2014 m. gegužės 06 d.]. Prieiga per internetą: <http://www.uni-hildesheim.de/media/fb4/informatik/AG_SSE/PDFs/UML-Tools-2.0.pdf>.
9. **Ellis K.** *The Impact of Business Requirements on the Success of Technology Projects*. Benchmark, IAG Consulting (2008)
10. **European Committee for Standardization.** *ENV 12 204: Advanced Manufacturing Technology – Systems Architecture – Constructs for Enterprise Modelling,* CEN TC 310/WG1, 1996, p.42.
11. **European Committee for Standardization.** *ENV 40 003: Computer Integrated Manufacturing – Systems Architecture – Framework for Enterprise Modelling,* CEN/CENELEC, 1990.

12. **Goyal, S.:** *Agile Techniques for Project Management and Software engineering* [interaktyvus]. 2007 [žiūrėta 2014 m. gegužės 07 d.]. Prieiga per internetą: <http://csis.pace.edu/~marchese/CS616/Agile/FDD/fdd.pdf>

13. **Gudas, S.; Lopata, A.** *Žiniomis grindžiama informacijos sistemų inžinerija.* Informacijos mokslai: mokslo darbai, Vilnius: Vilniaus universiteto leidykla, 2004, Vol. 30, p 90-98. ISSN 1392-0561

14. **Gupta, M.M.; Sinha N. K.** *Intelligent Control Systems: Theory and Applications.* The Institute of Electrical and Electronic Engineers Inc., New York. 1996.

15. **Haan, J. D.** *Combining general purpose languages and domain specific languages for Model Driven Engineering* [interaktyvus]. 2008 [žiūrėta 2014 m. gegužės 05 d.]. Prieiga per internetą: <http://www.theenterprisearchitect.eu/blog/2008/04/15/combining-general-purpose-languages-and-domain-specific-languages-for-model-driven-engineering/>

16. **Heuluy, B.; Vernadat, F.B.** *The CIMOSA Enterprise Ontology.* Proceedings of the IFAC Workshop–MIM'97, Vienna, 1997. p. 37–41.

17. **Hull, E.; Jackson K.; Dick J.:** *Requirements Engineering, Second Edition*. Springer, ISBN 1-85233-879-2, 2005

18. **IBM** [interaktyvus]. 2014 [žiūrėta 2014 m. gegužės 05 d.]. Prieiga per internetą: <http://www.ibm.com/us/en/>

19. **IBM.** *RUP: Best practices for design, implementation and effective project management*[interaktyvus]. 2014 [žiūrėta 2014 m. gegužės 06 d.]. Prieiga per internetą: <http://www-01.ibm.com/software/rational/rup/>

20. **IFIP–IFAC Task Force on Architectures for Enterprise Integration.** *GERAM:Generalised Enterprise Reference Architecture and Methodology* [interaktyvus]. 1999 [žiūrėta 2014 m. gegužės 05 d.]. Prieiga per internetą < http://www.cit.gu.edu.au/~bernus/taskforce/geram/versions/geram1–6–3/v1.6.3.html>

21. **Jeffries, R.** *Extreme Programming* [interaktyvus]. 2014 [žiūrėta 2014 m. gegužės 05 d.]. Prieiga per internetą: < http://xprogramming.com/book/whatisxp/>**.**

22. **Juhani, W.; Abrahamssom, P.; Salo, O.; j. Ronkainen.:** Agile software development methods [interaktyvus], ISBN 951-38-6010-8. 20024 [žiūrėta 2014 m. gegužės 05 d.]. Prieiga per internetą: < http://www.vtt.fi/inf/pdf/publications/2002/P478.pdf>

23. **Leszek M.,** Requirements Analysis and Systems Design (3rd Edition), June 22, 2007 | ISBN-10: 0321440366 | ISBN-13: 978-0321440365 | Edition: 3rd

24. **Lopata A., Gudas S.** Veiklos modeliu grindžiamas kompiuterizuotas funkcinių vartotojo reikalavimų specifikavimo metodas, daktaro laipsnio disertacija, VU KHF, 2005.

25. **Mayer, R.** *IDEF Family of Methods for Concurrent Engineering and Business Re–engineering Applications* [interaktyvus]. 1992 [žiūrėta 2014 m. gegužės 05 d.]. Prieiga per internetą: < http://www.idef.com/Downloads/pdf/IDEFFAMI.pdf>.

26. **Maskeliūnas, S.** Žinių technologijų (ir saityno technologijų) terminų žodynėlis (2012-03-01 d. versija). [interaktyvus]. 2012 [žiūrėta 2014 m. gegužės 05 d.]. Prieiga per internetą: <http://eta.ktl.mii.lt/~mask/LIKS-IS/Z%27iniu_technologiju_z%27odyne%27lis.pdf>

27. **Microsoft.** [interaktyvus]. 2014 [žiūrėta 2014 m. gegužės 05 d.]. Prieiga per internetą: < http://www.microsoft.com>.

28. **Miller, J.; Mukerji, J.** MDA Guide Version 1.0.1 [Interaktyvus]. [žiūrėta 2009 02 08]. Puslapio nuoroda: www.omg.org/docs/omg/03-06-01.pdf.

29. **Nielsen.:** *Top U.S. Smarthphone operating systems by market share* [interaktyvus]. 2014 [žiūrėta 2014 m. gegužės 05 d.]. Prieiga per internetą: <http://www.nielseneurope.pl/us/en/newswire/2014/smartphone-milestone-half-of-americans-ages-55-own-smartphones.html>

30. **NoMagic.** *MagicDraw* [interaktyvus]. 2014 [žiūrėta 2014 m. gegužės 05 d.]. Prieiga per internetą: <http://www.nomagic.com/products/magicdraw.html>

31. **Object Management Group.** *Semantics of a Foundational Subset for Executable UML Models (fUML)* [interaktyvus]. 2014 [žiūrėta 2014 m. gegužės 05 d.]. Prieiga per internetą: <http://www.omg.org/spec/FUML/1.1/PDF>.

32. **Object Management Group.** *Systems Modeling Language* [interaktyvus]. 2014 [žiūrėta 2014 m. gegužės 05 d.]. Prieiga per internetą: <http://www.omg.org/spec/SysML/1.3/PDF>.

33. **Object Management Group.** *Unified Modeling Language (UML)* [interaktyvus]. 2014 [žiūrėta 2014 m. gegužės 05 d.]. Prieiga per internetą: <http://www.omg.org/spec/UML/2.4.1/>.

34. **Object Management Group.** *XML Metadata Interchange* [interaktyvus]. 2014 [žiūrėta 2014 m. gegužės 05 d.]. Prieiga per internetą: <http://www.omg.org/spec/XMI/2.4.2/PDF>.

35. **Passing, J.**: *Requirements Engineering in the Rational Unified Process.*Hasso Plattner Insitute for Software Systems Engineering, 2006 [žiūrėta 2014 m. gegužės 14 d.]. Prieiga per internetą: <http://int3.de/res/RUP/RUP_Paper_JohannesPassing.pdf>

36. **Petersen, K.; Wohlin, C.** *The effect of moving from a plan-driven to an incremental software development approach with agile practices.* Empirical Software Engineering, Vol. 15, No. 6, 2010 p.p. 654-693

37. **Robertson, S.; Robertson, J.:** *Volere Requirements Techniques: an Overviev* [interaktyvus]. 2008 [žiūrėta 2014 m. liepos 20 d.]. Prieiga per internetą: http://www.softed.com/Resources/Docs/VolereOverview1.pdf

38. **Rumbaugh, J.** *OMT Insights: Perspective on Modeling from the Journal of Object–Oriented Programming.* Signature sounds recording, 1997. p. 412. ISBN: 0138469652.

39. **Schwaber, K.; Sutherland, J.** *The Scrum Guide* [interaktyvus]. 2014 [žiūrėta 2014 m. gegužės 05 d.]. Prieiga per internetą: <https://www.scrum.org>.

40. **Skersys T.** *Business Knowledge-Based Generation of the System Class Model*. Informatica, Vol. 37, No. 2, Vilnius, 2008, p. 145 - 153, 2008

41. **Vernadat, F.:** *UEML: towards a unified enterprise modelling language.* International Journal of Production Research, Vol. 40, no. 17, p. 4309–4321, 2002

42. **Watson, A.:** *Visual Modelling: past, present and future* [interaktyvus]. 2014 [žiūrėta 2014 m. gegužės 05 d.]. Prieiga per internetą: <www.uml.org/Visual_Modeling.pdf>

43. **Williams, L.:** *A Survey of Agile Development Methodologies* [interaktyvus]. 2007 [žiūrėta 2014 m. gegužės 07 d.]. Prieiga per internetą: http://agile.csc.ncsu.edu/SEMaterials/AgileMethods.pdf

INFORMACIJA APIE DISERTACIJOS AUTORIŲ

Vardas, Pavardė: Martas Ambraziūnas

Gimimo data: 1984 05 02

Išsilavinimas:

Aukštoji mokykla	Baigimo metai	Įgyta kvalifikacija arba specialybė
Vytauto didžiojo universitetas	2006	Informatikos bakalauro kvalifikacinis laipsnis
Vilniaus universiteto Kauno humanitarinis fakultetas	2008	Vadybos ir verslo administravimo magistro kvalifikacinis laipsnis (verslo informacijos sistemų programa)

Mokslinė veikla:

Metai	Aprašymas
2013 m.	Mokslinis tyrimas „Inovatyvių programinės įrangos kūrimo metodų taikymo išmaniesiems įrenginiams tyrimas„ (VP2-1.3-ŪM-05-K-02-057) pagal MITA „InočekiaiLT“ priemonę (VP2-1.3-ŪM-05)

Atliktų tyrimų paskelbtos mokslinės publikacijos:

Mokslinės informacijos instituto (ISI) pagrindinio sąrašo leidiniuose

1. Lopata A., Ambraziūnas M., Veitaitė I., Masteika S., Butleris R. „SysML and UML models usage in Knowledge Based MDA process“, Electronics and Electrical Engineering, ISSN 1392-1215, 2013 (Spaudoje);
2. Lopata A., Ambraziūnas M., Gudas S., Butleris R., Butkienė R. „Enterprise Knowledge-Based Generation of Class Model“, Electronics And Electrical Engineering, Vol. 19, No 2 (2013), pp. 79-82, ISSN 1392-1215, 2013;
3. Lopata A., Ambraziūnas M., Gudas, S. „Knowledge Based MDA Requirements Specification and Validation Technique“, Transformations in Business & Economics, Vol. 11, No 1 (25), pp. 248-261, ISSN 1648 - 4460, 2012;
4. Lopata A., Ambraziūnas M., Gudas S., Butleris R. „The Main Principles of Knowledge-Based Information Systems Engineering“, Electronics And Electrical Engineering, Vol. 120, No 4 (2012), pp. 99-102, ISSN 1392-1215, 2012;
5. Lopata, A., Ambraziunas, M., Gudas, S. “Knowledge-Based Approach to Business and IT Alignment Modelling”, Transformations in Business & Economics, Vol. 10, No 2 (23), pp. 60-73, ISSN 1648 - 4460, 2011.

Kituose Mokslinės informacijos instituto (ISI) duomenų bazėse referuojamuose leidiniuose [Proceedings ir kt.]

1. Veitaitė I., Ambraziūnas M., Lopata A. „Enterprise Model and ISO Standards Based Informations System’s Development Process“ // BIS 2014 International Workshops,

Larnaca Cyprus May 22-23, 2014, Series: Lecture Notes in Business Information Processing. (Spaudoje);

2. Lopata A., Ambraziūnas M. „Knowledge-Based MDA Approach“ // BIS 2011 International Workshops, Poznan, Poland, June 15-17, 2011, Series: Lecture Notes in Business Information Processing, Vol. 97, Abramowicz, Witold; Maciaszek, Leszek; Węcel, Krzysztof (Eds.), 2011, XV, 307p., Softcover ISBN: 978-3-642-25369-0;
3. Lopata A. Ambraziūnas M. „MDA Compatible Knowledge - Based IS Engineering Approach“ // International Conference, AICI 2010, Sanya, China, October 23-24, 2010, Proceedings, Part II, Series: Lecture Notes in Computer Science, Vol. 6320, Subseries: Lecture Notes in Artificial Intelligence, Wang, Fu Lee; Deng, Hepu; Lei, Jingsheng (Eds.) 1st Edition., 2010, XXII, 386 p., Softcover ISBN: 978-3-642-16526-9;
4. Lopata A., Ambraziunas M. „MDA Compatible Knowledge Based IS Development Process“ // BIS 2010 International Workshop, Berlin, Germany, May 3-5, 2010, Series: Lecture Notes in Business Information Processing, Vol. 57, Abramowicz, Witold; Tolksdorf, Robert; Wecel, Krzysztof (Eds.) 1st Edition., 2010, XVI, 324 p., Softcover ISBN: 978-3-642-15401-0.

Kituose recenzuojamuose mokslo leidiniuose

1. Lopata A. Ambraziunas M. „Knowledge Subsystem’s Integration into MDA Based Forward and Reverse IS Engineering“ // Proceedings of 16th International Conference on Information and Software Technologies “Information Technologies 2010”. , Kaunas, Lithuania April 21-23, 2010 / Kaunas University of Technology, Kaunas: Technologija. 2010, p. 205-210, ISSN 2029-0020;
2. Lopata A. Ambraziūnas M. „Veiklos žinių posisteme grindžiamas MDA metodas“ // Vytauto Didžiojo Universitetas: 15-osios tarpuniversitetinės magistrantų ir doktorantų konferencijos „Informacinė visuomenė ir universitetinės studijos“ (IVUS 2010) medžiaga 2010.05.13 Kaunas. Kaunas: Vytauto Didžiojo Universitetas 2011. p. 5-9, ISSN 2029-249X.