https://doi.org/10.15388/vu.thesis.448
https://orcid.org/0000-0001-9046-0788

VILNIUS UNIVERSITY

Ilona Veitaitė

Enterprise Knowledge-Based UML Dynamic Models Generation Method

DOCTORAL DISSERTATION

Technological Sciences,
Informatics Engineering [T 007]

VILNIUS 2023

https://doi.org/10.15388/vu.thesis.448
https://orcid.org/0000-0001-9046-0788

VILNIAUS UNIVERSITETAS

Ilona Veitaitė

Organizacijos žiniomis pagrįstas UML dinaminių modelių generavimo metodas

DAKTARO DISERTACIJA

Technologijos mokslai,
Informatikos inžinerija [T 007]

VILNIUS 2023

Disertacija rengta 2011 – 2015 metais studijuojant doktorantūroje Vilniaus universitete.
Disertacija ginama eksternu.

Disertacija ginama viešame Gynimo tarybos posėdyje 2023 m. kovo mėn. 23 d. 14:30 val. Vilniaus universiteto Duomenų mokslo ir skaitmeninių technologijų instituto 203 auditorijoje. Adresas: Akademijos g. 4, LT-04812 Vilnius, Lietuva.

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

ACKNOWLEDGEMENT

The thesis author would like to express her gratitude to her supervisor, Prof. Dr. Audrius Lopata, for the valuable scientific consultations and insights provided during the PhD studies, for the advice, suggestions and constant support, to the colleagues of the Kaunas Faculty, to the PhD students for their cooperation and help, to her family for inspiration, support and understanding.

ACRONYMS

CASE – Computer-Aided Software Engineering
CISE – Computerised IS Engineering
CIM – Computation Independent Model
EM – Enterprise Model
EMM – Enterprise Meta-Model
GUI – Graphic User Interface
IEC – International Electrotechnical Commission
IS – Information System
ISEDS – IS Engineering Development Stages
ISO – International Organisation for Standardization
IT – Information Technology
KB – Knowledge-Based
KBISE – Knowledge-Based IS Engineering
MDA – Model Driven Architecture
MOF – Meta-Object Facility
OMG – Object Management Group
PIM – Platform Independent Model
PSM – Platform Specific Model
SAM – Strategic Alignment model
SIM – Society for Information Management
TISE – Traditional IS Engineering
UML – Unified Modeling Language

GLOSSARY

Computer-Aided Software Engineering (CASE) – the implementation of computer-facilitated tools and methods in software development.

Computation Independent Model (CIM) – model to describe user requirements for a system and business processes. This model does not contain information about the technology platform on which the system will be implemented.

Enterprise Knowledge – knowledge subsystem consisting of the Enterprise Meta-Model and Enterprise Model.

Enterprise Meta-Model (EMM) – a formal structure ensuring a more qualified project development process and knowledge base data collection. This model was created by the collaborative scientific group of Vilnius University and other science institutions.

Enterprise Model (EM) – the primary source of the necessary knowledge of the particular problem domain for IS engineering and IS re-engineering processes based on the Enterprise meta-model (EMM).

Information system (IS) – Enterprise software, also known as enterprise application software (EAS), is computer software which is used to satisfy the demands of an organisation rather than individual users. Enterprise software can be categorized by type of business function, and each type of enterprise application can be considered a "system" that supports an organization's particular business management process or function.

Graphic User Interface (GUI) – a form of a user interface allowing users to interact with electronic devices through graphical icons and an audio indicator such as primary notation, instead of text-based UIs, typed command labels or text navigation.

Model Driven Architecture (MDA) – an approach to software design, development and implementation spearheaded by the OMG. MDA provides guidelines for structuring software specifications that are expressed as models.

Meta-Object Facility (MOF) – an Object Management Group (OMG) standard for model-driven engineering.

Object Management Group (OMG) – an international, open membership, not-for-profit technology standards consortium.

Platform Independent Model (PIM) – a model describing the architecture and behaviour of a system without specifications for a specific technology platform. By annotating the model with platform-specific information, a platform-dependent model is created.

Platform Specific Model (PSM) – a model describing the architecture and behaviour of a system with specifications for a specific technology platform (e.g. programming language, operating system).

Strategic Alignment Model (SAM) – a conceptual model that has been used to understand strategic alignment from the perspective of four components, i.e. Business Strategy, IT Strategy, Organisational Infrastructure, and IT Infrastructure, and their interdependencies.

Unified Modeling Language (UML) – is a general-purpose, developmental modelling language in the field of software engineering intended to provide a standard way to visualize the design of a system.

UML Dynamic models – also known as behavioural models: Use Case, Information Flow, Activity, State Machine, Interaction: Sequence, Communication, Timing, Interaction Overview. These models emphasize the behaviour of objects in a system, including their methods, interactions, collaborations, and state histories.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

The alignment of business and IT has been a significant management concern for over two decades because it remains an essential goal. Superior strategic alignment between business and IT strategy should lead to superior performance compared to lower stages of strategic alignment. There is much discussion about the term "business-IT alignment" [12]. Some would argue that alignment is a weak goal that can be achieved only by a small number of organisations, where everyone prefers the help desk and the IT budget is enough to fund any proposed project. Moreover, some would agree that business and IT alignment ensure the proper function of the entire organisation [10][11]. Professionals suggest that organisations need to achieve strategic business and IT alignment to be competitive. Strategic business and IT alignment affect business performance and IT effectiveness [11][12].

In today's organisations, issues of shared understanding between business and IT exist. Information technology strategy planning is a multistage process, and information system development depends on implementing each phase of IS development life cycle. A combination of the leading business and IS development goals does not have a direct relation. To ensure business and IT alignment, it is essential to create communication between these two sides. Implementing management processes provides clarity in decision-making, at the same time, explains what IT is responsible for. Management takes decisions away from IT and gives them to those who have a concern for the success of initiatives and projects [11][12][34]. It means that the business stakeholders present the case to an appropriate working group. That kind of communication often leads to stronger relationships and partnerships and also can help identify common needs between disparate user groups. This knowledge improves communication, and people become more enlightened about the challenges IT faces when trying to satisfy user requirements [12][34].

In order to help stimulate communication and understanding, many organisations are establishing business-facing roles that have significant responsibility for establishing and maintaining alignment between IT and the business fields. The balance of technology and business vision and the ability to explain business and technology challenges with equal clarity should be ensured. It is pretty often that forward-thinking and understanding how to adopt new technologies to impact business is clear just for one side, and talking about this technology does not always serve well [12][34][36]. It is vital to move away from using technical terms or talking about the challenges to explain correctly what the business needs and how the working team will be able to meet these needs only if some conditions change or how the end-stage will better support the organisation and its business goals, not just how the changes will make the IT team more effective.

Motivation

A knowledge-based IS engineering offers system modelling and decision-making methods and tools, which help to develop a more accurate and detailed subject domain corresponding to the project [18][31[33]. IS project developer and/or programmer is allowed to use not only the knowledge of the project, which is stored in the traditional CASE tool repository, but also the knowledge-based repository, where subject domain knowledge tested according to formal criteria is stored. There are many standards and business modelling methodologies which can make this process easier [21][23][30][31][47].

The Unified Modeling Language (UML) is a general-purpose, developmental modelling language in the field of software engineering. It is a universal IS modelling language applied to many methodologies and used in the most popular modelling tools, such as Enterprise Architect, System Architect, MagicDraw, and others [5][13][15]. The method of UML model generation from the Enterprise Model presented by the author implements a knowledge-based design phase in the IS development cycle [26][27][28].

A knowledge-based subsystem as a CASE tool component containing Enterprise Meta-Model (EMM) and Enterprise Model (EM) inside can help solve this issue. The Enterprise Meta-Model and Enterprise Model (EM) are a formal structure that ensures a more qualified project development process and knowledge base data collection [26][27]. Enterprise Model and Enterprise Meta-Model make the generation of UML project models possible, which makes the design process more efficient and qualitative and ensures fewer errors in the final phase of IS development [26][50][56][57].

Object and Scope of Research

The research object of this work is the process of generating UML dynamic models from the Enterprise Model (EM) as a part of knowledge-based IS engineering following MDA approach with the integration of ISO standards and MOF.

The scope of the research encompasses the following fields:

- Knowledge-based information system engineering.
- ISO standards that provide possible solutions and help achieve benefits for almost all sectors of activity. These standards are documented agreements containing technical specifications or other precise criteria to be used consistently as rules, guidelines, or definitions of characteristics, to ensure that materials, products, processes, and services are fit for their purpose [38][44].
- MOF is an Object Management Group (OMG) standard for model-driven engineering. Its purpose is to provide a type system for entities and a set of interfaces through which those types can be created and manipulated. MOF only provides a means to define a language or data's structure or abstract syntax [74].
- MDA – Model Driven Architecture is an approach to software design, development and implementation spearheaded by the OMG. MDA provides guidelines for structuring software specifications that are expressed as models.

Problem Statement

Traditional IS engineering phases – from the specification of user requirements to project development – are empirically based, and traditional CASE tools implement an empirical UML-based IS project development process. It means that IS project creation depends mainly on analyst knowledge and experience [26][27][57][60].

One of the many computerised IS engineering methods' disadvantages is the fact that IS design models are generated only partially since, in the design phase, the designer forms design models based on one's personal or analyst's experience rather than applying the principles of using knowledge, which is stored in Enterprise Model, for design models generation. The solution to these problems is the integration of the Enterprise Model, as the core subject knowledge repository, into the knowledge-based IS engineering process [49]. The knowledge-based subsystem implemented as a CASE tool component, including Enterprise Meta-Model and Enterprise Model, can manage these issues.

Research Goal and Objectives

The goal of this research is to extend capabilities for generating system design phase UML dynamic models from the Enterprise Model via the usage of transformation algorithms. Research tasks are the following:

1. To analyse the different perspectives of UML dynamic models in IS engineering.
2. To create transformation algorithms UML dynamic models from Enterprise Model in a knowledge-based IS engineering.
3. Experimentally check the transformation algorithms with real-world subject domain examples.
4. Evaluate the advantages of a knowledge-based method of UML dynamic models generation.

Research Methodology

The current situation and the relevance of the research have been evaluated by analyzing the scientific literature related to IS engineering, IS development life cycle phases, Enterprise Modelling, UML, ISO standards, MOF architecture, model-driven development, and other areas. The research was carried out using the Design Science research methodology [35]. The method of UML dynamic model generation from Enterprise Model by using transformation algorithms was created by the author to ensure a more effective and qualified generation process, and a lower number of mistakes and errors in the final IS development life cycle phase. All the examples were included to evaluate the presented method's applicability and prove its suitability.

Defended Propositions

Propositions, defended by the research:

1. An Enterprise Model (EM) that corresponds to an Enterprise Meta-Model (EMM) is a sufficient source of subject domain knowledge to generate the dynamic models defined in the UML specification.
2. The created transformation algorithms ensure the generation of UML dynamic models from the enterprise model (EM) in different subject domains.

Major Contributions and Novelty

The significant contribution of this work is the Enterprise knowledge-based method for UML dynamic models generation from Enterprise Model. The knowledge-based generation method combines the main principles of knowledge-based techniques, ISO standards, MOF and MDA. This method gives the possibility to create advanced software development approaches. Within the research, the existing Enterprise Meta-Model [23][32][57] was analysed and used for the UML dynamic model generation of different subject domains. In order to implement UML dynamic model generation from Enterprise Model, new UML model transformation algorithms were created.

Practical Significance

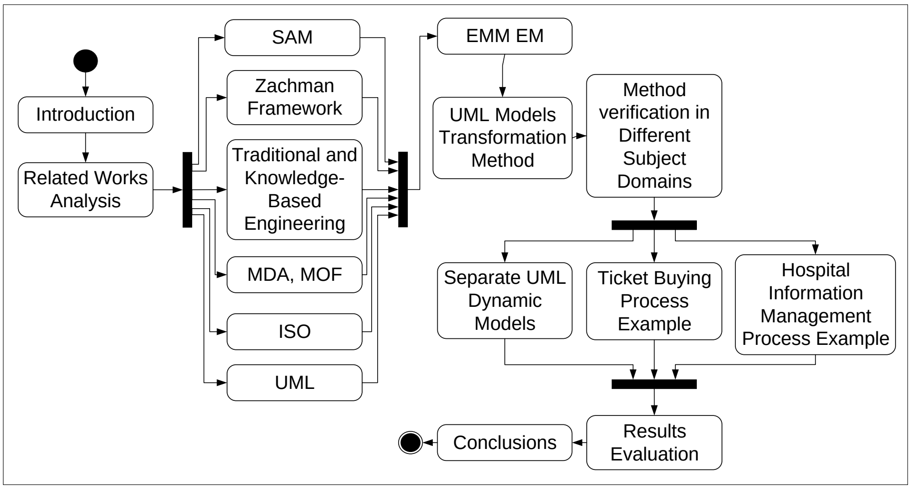
The primary practical significance of the research is the step towards making the modelling process better understandable and usable for all participants of IS development life cycle. Application of the proposed knowledge-based generation method provides the opportunity to use supplementary models validation methods that Enterprise Meta-Model defines.

Scientific Approval

The research results were presented at 14 international and 3 Lithuanian conferences, 6 international seminars as poster presentations, and 1 PhD symposium. Section "List of conferences and scientific events" contains a detailed list of scientific events and conferences. 4 articles were published in periodical scientific publications in journals referred in Clarivate Analytics Web of Science database publications with a citation index. 9 articles were published in conference proceedings referenced in the Clarivate Analytics Web of Science database. 9 articles were published in peer-reviewed conference proceedings. 2 articles as a book's chapters were published in Springer book series, classified as other peer-reviewed periodicals, continuous or one-time scientific publications. Section "List of publications" contains a detailed list of publications.

Thesis Structure

The thesis structure and main parts are provided in the research schema (Figure 1). The introduction presents the entire content of the thesis, then as follows: related works analysis, models transformation method, examples, results and conclusions.


SAM
Zachman Framework
Introduction
Related Works Analysis
Traditional and Knowledge-Based Engineering
MDA, MOF
ISO
UML
EMM EM
UML Models Transformation Method
Method verification in Different Subject Domains
Separate UML Dynamic Models
Ticket Buying Process Example
Hospital Information Management Process Example
Results Evaluation
Conclusions

Source: created by the author according to [35]

Figure 1. Research Schema

The first chapter analyses scholarly literature with a primary focus on business and IT alignment challenges, ISO standards and MOF architecture usage, traditional and knowledge-based IS engineering, Enterprise Modelling solutions, phases of the IS development life cycle processes and model-driven architecture solutions. The second chapter presents the knowledge-based UML model generation method. This chapter presents a detailed description and composition of the provided method by describing UML dynamic models compositions, transformation rules, and their transformation algorithms from Enterprise Model. The third chapter introduces separate and two combined UML dynamic model generation examples from different subject domains by using transformation algorithms proposed by the author. This chapter verifies and proves that the Enterprise Model is suitable for UML dynamic model generation and can be used through created transformation algorithms.

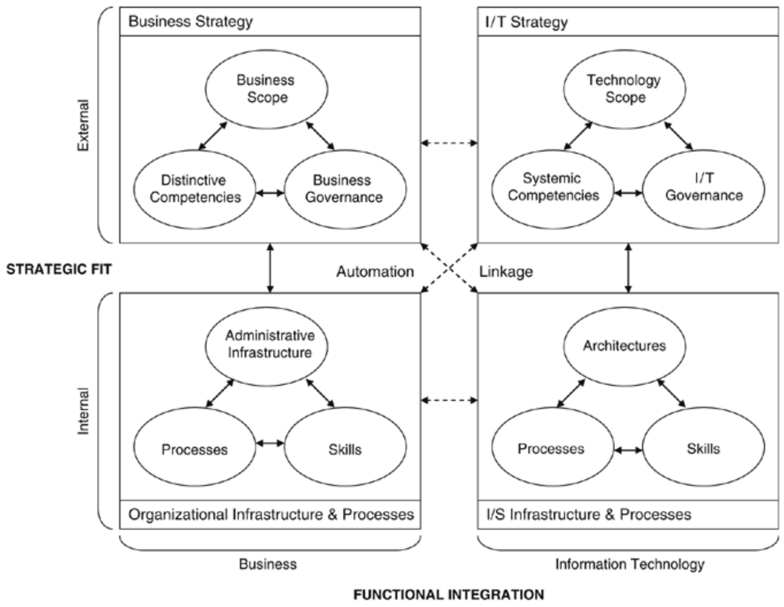
1. ANALYSIS OF RELATED WORKS

The first section consists of an analysis of scholarly literature with a particular focus on business and IT alignment challenges, ISO standards, MOF and MDA usage, traditional and knowledge-based IS engineering, Enterprise Modelling solutions, and phases of the IS development life cycle processes.

1.1. Strategic Alignment Model Relation to Enterprise Model

One of the organisational solutions in enterprises is the application of business and IT alignment. The most widespread and accepted conceptual model of alignment was proposed by J.C. Henderson and N. Venkatraman [11][12][34]. This theoretical construct is known as the strategic alignment model (SAM). The concept of strategic alignment is based on two dimensions (Figure 2): first – the strategic fit between external focus, directed towards the business environment, and internal focus, directed towards infrastructure and processes and second – functional integration between business and IT [11][34][76]. Strategic fit refers to the concordance between internal and external domains. Functional integration refers to the incorporation of the IT strategy into the business strategy, particularly the integration of the internal IT strategies into the internal organisational procedures and strategies. Altogether, the model defines four domains that must be harmonized to achieve alignment [78][87][93]. The alignment models derived from it help us understand alignment from the view of the involved components, such as Business Strategy, IT Strategy, Organisational Infrastructure, and IT Infrastructure, and their interdependencies [12][34]. However, analyzing the alignment among Business and IT requires a more detailed interpretation and definition than presented in this model [77][78].

All later alignment models and consulting practices start from or refer to Henderson and Venkatraman’s SAM [34]. Several assessment frameworks and processes are developed from such foundations to indicate an expected value of alignment.



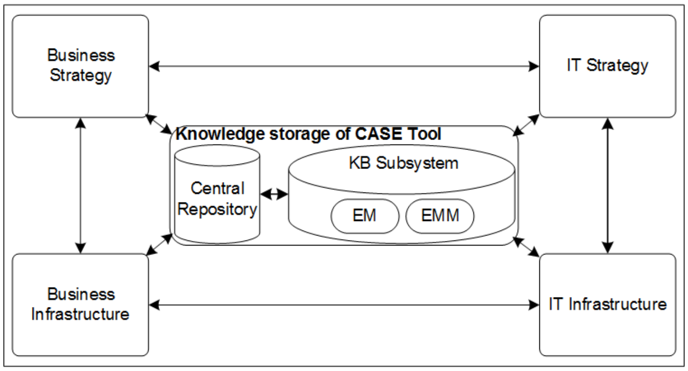
Source: [34]

Figure 2. Strategic Alignment Model

In the field of strategic business and IT alignment, competing approaches (typically labelled models or frameworks) are found to describe the nature of the alignment phenomenon [1-8]. In this research, Luftman's strategic business and IT alignment assessment are adopted [67][68] as the operationalized theory that correctly describes the complex phenomena of alignment since it is empirically well-founded [34][67]. It is based on a combination of twelve relationships between SAM components and research results from previous studies on alignment inhibitors and enablers and has been used in 60 global companies [68][93]. Some might, of course, argue against considering his strategic business and IT alignment as a genuinely representative approach, but that discussion is not the subject of this research. Luftman's strategic business and IT alignment assessment approach, from now on cited as Luftman's alignment assessment approach, has been the subject of benchmark studies jointly sponsored by the Society for Information Management (SIM) and The Conference Board and has been applied in large and small companies at all levels. The alignment allows one to measure how well the technical and business organisations work together. It examines six dimensions, rating each on a scale of 1 (lowest) to 5 (highest) [12][34][67].

Henderson and Venkatraman described the interrelationship between business and IT strategies. Strategic Alignment Model (SAM) was created in order to describe these relations. The model consists of four domains: Business Strategy Domain, Business Infrastructure Domain, IT Strategy Domain, IT Infrastructure Domain and relationships between them. The model is based on two parts: strategic integration and functional integration [34][67]. As mentioned previously, strategic integration recognizes that the IT strategy should be expressed in terms of an external domain (how the organisation is positioned in the IT environment) and an internal domain (how the IT infrastructure should be configured and managed). There are two types of functional integration: strategic and operational. The first one is the relationship between business strategy and IT strategy. The second one covers the internal domain and deals with the relationship between business infrastructure and IT infrastructure [11][12]. Functional integration considers how choices made in the IT domain impact those made in the business domain and vice versa. The strategic alignment model describes four basic alignment perspectives: Strategy Execution, Technology Transformation, Competitive Potential, and Service Level [34][93].

The proposed business and IT strategy alignment model (SAM) is conceptual. It does not provide a practical framework to implement this kind of alignment, despite that there are alignment mechanisms developed and used in organisations to achieve the business and IT synthesis, but these mechanisms are mainly oriented to business, not to IT [34][76][78].



Source: created by the author [63A]

Figure 3. Business and IT Alignment Model and Knowledge Storage of CASE Tool

The knowledge-based CASE Tool subsystem in business and IT management can be used as the primary source for particular alignment [64A][100A]. These frameworks specify all relevant structures within the organisation, including business, applications technology, data and their relationships to perform business [22][23][24]. For business and IT alignment processes, specific data is necessary and used in a knowledge-based subsystem (Figure 3). The business strategy domain provides business goals to the knowledge base. Business goals are described by the business managers and are received from the business environment, and IT goals are described by the IT managers. The business infrastructure domain delivers business rules, constraints, processes, functions and other related data; this can be defined using diverse methods, tools and techniques: natural language, formal templates, decision tables, and applications. IT infrastructure domain delivers information about IT infrastructure, which describes current software and hardware and can also be defined using various methods [70][88]. All this information is stored in the knowledge base and can be used through the Enterprise Model, which is validated according to the Enterprise Meta-Model [63A][64A][65A].

1.2. Strategic Zachman Framework's Approach

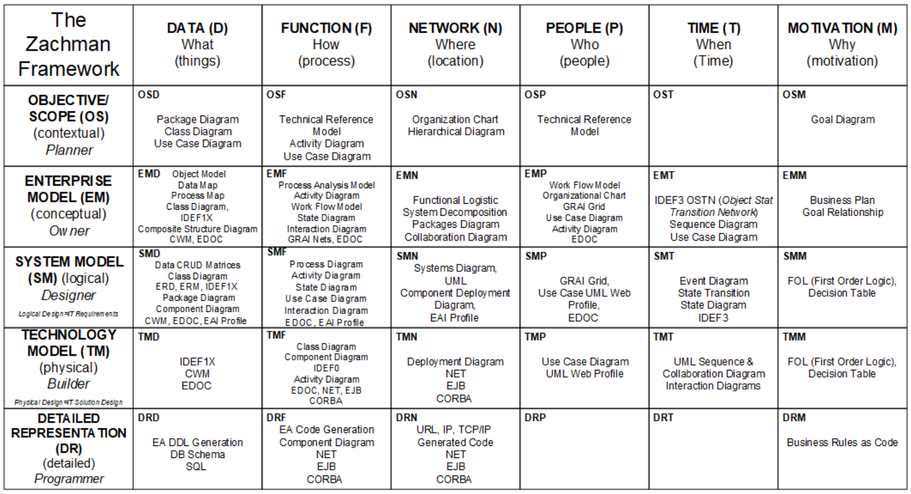
Another possible organisational solution in the enterprise is the adaptation and application of a specific framework. The Framework for Enterprise Architecture is a two-dimensional classification scheme for descriptive representations of an enterprise [1][5][7]. It was derived through observation of descriptive representations (design artefacts) of various physical objects like aeroplanes, buildings, ships, computers, and others in which it was empirically observed that the design artefacts (the descriptive representations, the product descriptions, the engineering documentation) of complex products could be classified by the audience for which the artefact was constructed (the Perspective) as well as classified by the content or subject focus of the artefact (the Abstraction) [1][5][7][19][112].

Different perspectives are represented in the process of engineering and manufacturing complex products. The descriptive representations of the product that are prepared over this process are designed to express concepts/constraints relevant to the various perspectives [7][46]. That is, not only do the design artefacts depict the necessary engineering information, but they depict it in such a manner that it is intelligible to the perspective (audience) for which they were created [53][54].

This set of perspectives appears to be universal and is easily observed in the architecture of buildings, independent of geography, culture, language, politics, or technology. Thousands of years of precedence establish that presently, in every case, there is the Bubble Charts or sketches (Scope), Architect’s Drawings (Owner’s View), Architect’s Plans (Designer’s View), Contractor’s Plans (Builder’s View) and Sub-Contractor’s Plans (Out-of Context View) and finally, the building (end product) itself [46][82][112].

Model Driven Generation (MDG) Technology for Zachman Framework was proposed, which further extends the Enterprise Architect diagram set to support the framework. Figure 3 shows the diagram types appropriate to each cell of the Zachman Framework [53][54][80][112].

The Zachman Framework pages of the Enterprise Architect UML Toolbox provide elements and relationships for all the Zachman Framework diagrams that the MDG Technology supports. The Zachman Framework Toolbox pages can be accessed using more tools in the Zachman Framework menu option. They can be docked on either side of the diagram or freely float on top of the diagram to expose more surface for editing [7][19][112].



The Zachman Framework	**DATA (D)** What (things)	**FUNCTION (F)** How (process)	**NETWORK (N)** Where (location)	**PEOPLE (P)** Who (people)	**TIME (T)** When (Time)	**MOTIVATION (M)** Why (motivation)
OBJECTIVE/ SCOPE (OS) (contextual) *Planner*	**OSD** Package Diagram Class Diagram Use Case Diagram	**OSF** Technical Reference Model Activity Diagram Use Case Diagram	**OSN** Organization Chart Hierarchical Diagram	**OSP** Technical Reference Model	**OST**	**OSM** Goal Diagram
ENTERPRISE MODEL (EM) (conceptual) *Owner*	**EMD** Object Model Data Map Process Map Class Diagram, IDEF1X Composite Structure Diagram CWM, EDOC	**EMF** Process Analysis Model Activity Diagram Work Flow Model State Diagram Interaction Diagram GRAI Nets, EDOC	**EMN** Functional Logistic System Decomposition Packages Diagram Collaboration Diagram	**EMP** Work Flow Model Organizational Chart GRAI Grid Use Case Diagram Activity Diagram EDOC	**EMT** IDEF3 OSTN (*Object Stat Transition Network*) Sequence Diagram Use Case Diagram	**EMM** Business Plan Goal Relationship
SYSTEM MODEL (SM) (logical) *Designer* *Logical Design=IT Requirements*	**SMD** Data CRUD Matrices Class Diagram ERD, ERM, IDEF1X Package Diagram Component Diagram CWM, EDOC, EAI Profile	**SMF** Process Diagram Activity Diagram State Diagram Use Case Diagram Interaction Diagram EDOC, EAI Profile	**SMN** Systems Diagram, UML Component Deployment Diagram, EAI Profile	**SMP** GRAI Grid, Use Case UML Web Profile, EDOC	**SMT** Event Diagram State Transition State Diagram IDEF3	**SMM** FOL (First Order Logic), Decision Table
TECHNOLOGY MODEL (TM) (physical) *Builder* *Physical Design=IT Solution Design*	**TMD** IDEF1X CWM EDOC	**TMF** Class Diagram Component Diagram IDEF0 Activity Diagram EDOC, NET, EJB CORBA	**TMN** Deployment Diagram NET EJB CORBA	**TMP** Use Case Diagram UML Web Profile	**TMT** UML Sequence & Collaboration Diagram Interaction Diagrams	**TMM** FOL (First Order Logic), Decision Table
DETAILED REPRESENTATION (DR) (detailed) *Programmer*	**DRD** EA DDL Generation DB Schema SQL	**DRF** EA Code Generation Component Diagram NET EJB CORBA	**DRN** URL, IP, TCP/IP Generated Code NET EJB CORBA	**DRP**	**DRT**	**DRM** Business Rules as Code

Source: created by the author according to [7][19][54][80][112].

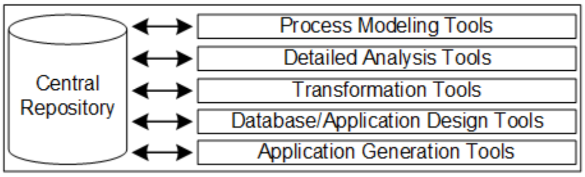
Figure 4. Zachman Framework Cells with Possible Diagrams’ Types

Figure 4 shows that the diagram or model of different notations could be used for each Zachman Framework cell. A pretty high number of cells can be covered with UML diagrams. That is one more confirmation of how important the role of UML models in IS engineering is essential [53][82][112].

1.3. Traditional IS Engineering Definition

Information systems engineering – is an information system development process, the execution of which is implemented through the IS development life cycle phases (planning, requirements analysis, modelling, design, development, testing and integration, implementation, maintenance.

Each phase of the IS development life cycle was implemented independently from the others because there was no computerised knowledge repository that included all phases of the life cycle. Analysts and designers used conventional or object-oriented IS engineering methods [23][25]. IS projects were implemented either in a non-computerised way or by using specially created computerised tools intended for the specific user requirements specifications to solve the problem [23][29]. Later, computerised IS development methods were implemented. IS development was improved by computerised IS development tools – CASE systems which included a part of the IS development life cycle phases – design, documentation and coding. Analysts and designers used the CASE system's IS development tools for conceptual and detailed design phases, and in the practical realization, phase programmer used the tools designed to generate code. In this case, a computerised system was based on traditional IS engineering and (or) object-oriented IS engineering methods [28][59][61].



Source: created by the author [60][62A][105A]

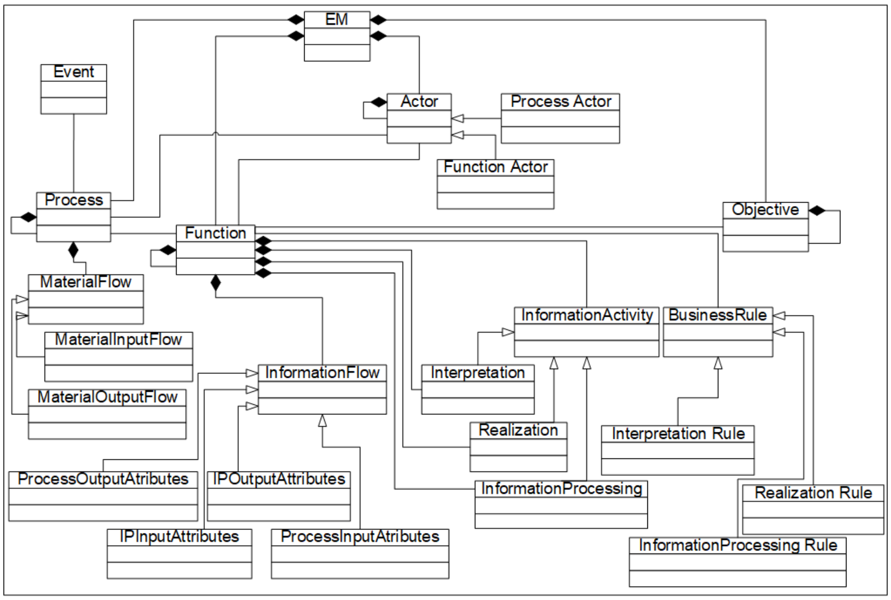
Figure 5. Traditional IS Engineering CASE Tool Components

With the development of the traditional IS engineering process, the CASE system covered the entire IS development life cycle – from Enterprise Modeling to code generation. These life cycle phases are implemented with CASE tool components (Figure 5) [16]. The Enterprise Modeling process is integrated into IS engineering instruments.

1.4. Enterprise Model Concept

The Enterprise Meta-Model (EMM) is a formally defined Enterprise Model (EM) structure, which consists of a formalized Enterprise Model in line with the general principles of control theory [23][26][27]. Enterprise Model is the primary source of the necessary knowledge of the particular subject domain for IS engineering and IS re-engineering processes (Figure 6) [28][58][70].

Enterprise Meta-Model manages Enterprise Model composition. Enterprise Model stores knowledge that is necessary for IS development process only and will be used during all phases of IS development life cycle [26][27][57]. EM and EMM are the main components of the knowledge-based subsystem of the CASE Tool. Collecting the necessary knowledge to the knowledge-based subsystem of the particular CASE Tool would be appropriate. This subsystem is the main source of knowledge necessary for design phase models (including UML diagrams) and the source code generation process.



Source: [23][26][23]

Figure 6 Enterprise Meta-Model Class Model

Figure 7 presents the Enterprise Meta-Model class model proposed two decades ago [23][6]. Also, Enterprise Model can be described as Malcev algebra-based algebra system (Figure 6) [57][64A]:

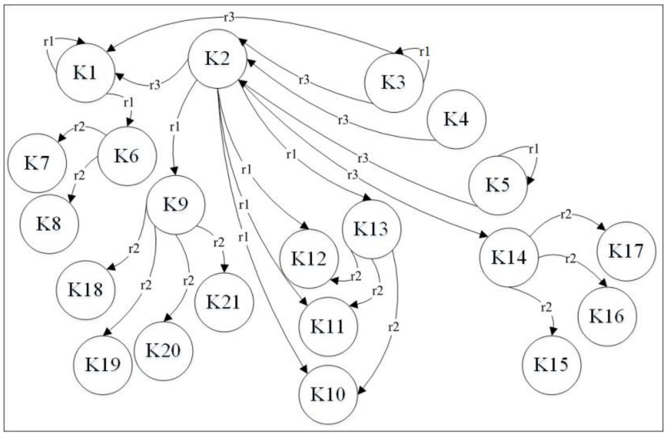
$$M1=<K, R> \quad (1)$$

where M1 – Enterprise Model as algebra system; K – elements set of M1 system; K={K1, K2,…, K21}, where K1,....K21 EM meta-classes; R – set of relationships between elements, where R={r1, r2, r3}.

For each set of K element Kn composition is defined as: Kn=<{an1, an2,…,ank}, {mn1, mn2,…,mnl}>, where {an1, an2,…,ank} – attributes of Kn element, {mn1, mn2,…,mnl}– methods of Kn element.

Enterprise Model M1 composition is as follows:

$$M1=<\{K1, K2,...,K21\}, \{r1, r2, r3\}> \qquad (2)$$



Source: [57]

Figure 7. An Enterprise Meta-Model Graphical Schema Based on Malcev Algebra

where: K1 – meta-class Process, K2 – meta-class Function, K3 – meta-class Actor, K4 – meta-class Event, K5 – meta-class Goal, K6 – meta-class Material Flows, K7 – meta-class Input Material Flow, K8 – meta-class Output Material Flow, K9 – meta-class Information Flow, K10 – meta-class Interpretation, K11 – meta-class Data Processing and Solution Making, K12 – meta-class Realization, K13 – meta-class Information Activity, K14 – meta-class Business Rules, K15 – meta-class Interpretation Business Rules, K16 – meta-class Data Processing and Solution Making Business Rules, K17 – meta-class Realization Business Rules, K18 – meta-class Process Output, K19 – meta-class Information processing Input Attributes, K20 – meta-class Information processing Output Attributes, K21 – meta-class Process Input, r1 – Aggregation, r2 – Generalization, r3 – Association [57].

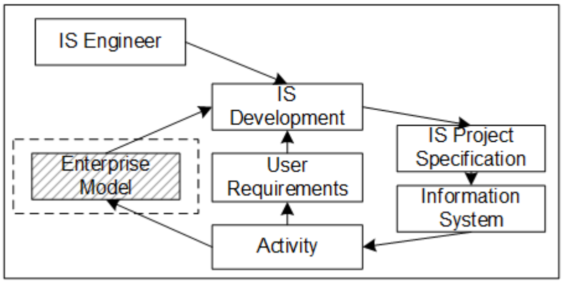
1.5. Comparison of Traditional and Knowledge-Based IS Engineering

Information systems (IS) are becoming more complicated, and textual information with elementary schemes is insufficient to describe business processes. Specific models are applied for computerised requirements specification and modelling of IS. Enterprise Modelling has become an integral part of IS development process. During this process, user requirements, business domain knowledge, software architecture, and other essential components are modelled [17][20][25][26]. Recently, the organisation's business modelling has become an important phase of modelling design processes [14][22][23][56][57].

IS engineering phases, from modelling to code generation, are traditionally implemented empirically. A computerised model of the subject domain is formed by the experience of the analyst, while traditional CASE system design models are formed based on the information collected on the business domain, which is being computerised [28][30][48]. The leading role is given to the analyst, and enhanced knowledge of the subject domain is not fully or insufficiently controlled by formalized criteria [56][57].

Currently, computerised IS engineering, in order to avoid the empiric, is developing based on new knowledge-based methods [28][30]. Computerised information system in knowledge-based information system engineering is developed by using the stored enterprise knowledge base of the subject domain, i.e., the Enterprise Model, the composition of which is defined by formal criteria. Computerised IS software development based on that model is known as knowledge-based IS engineering [48][57].

A knowledge-based subsystem as a CASE tool component with Enterprise Meta-Model (EMM) and Enterprise Model (EM) inside it can solve this issue. The Enterprise Meta-Model is a formal structure which ensures a more qualified project development process and knowledge base data collection [29][30][33][57] Usage of the Enterprise Model and Enterprise Meta-Model makes the UML project models generation process more effective and qualified.



Source: created by the author according to [23][27]

Figure 8. Principal Schema of Traditional IS Engineering and Knowledge-Based IS Engineering Differences

Traditional IS engineering and knowledge-based IS engineering have qualitative differences (Figure 8). In the case of traditional IS engineering – there is an empirical IS engineering, where the individual user does not include whole enterprise processes [22][33]. A knowledge-based IS engineering covers the whole enterprise specification (Enterprise Model) because it specifies the essential organisation's characteristics. Formalized Enterprise Model is based on a theoretical organisation's management principles [32][33][45].

The information system is developed empirically in traditional computerised IS engineering [23][24][29]. In knowledge-based computerised IS engineering, an IS is developed by using an enterprise knowledge repository, where necessary and sufficient computerised knowledge of the subject domain is stored [26][33][58][101A].

The table presented below shows the differences in the types of IS engineering methods, sources of knowledge, i.e. participants who are involved in IS development process and necessary software [30][32][33][64A].

CASE system is applicable not only in IS engineering but also in Enterprise Modelling and re-engineering. CASE systems are complemented with Enterprise Modelling tools and CASE system repository – Enterprise Model subsystem. A system analyst is the main user of the Enterprise Modelling CASE tool component. The computerised IS development process is based on the design from the model sequence execution, where every other phase of the design model is created interactively in the presence of the analyst, designer and programmer [30][32][33].

At that period, CASE systems automatically generated only the logical database diagrams, user interface fragments and code, which programmers basically had to adjust. Thus, the traditional computerised IS engineering, IS development life cycle starting from the initial and ending with its last phase, takes place empirically, and many design models in CASE tools were generated partially (Table 1). The only analyst could fully implement them based on empirical experience [30][32][33][105A]. Explanations of abbreviations in the table: ISEDS – IS Engineering Development Stages; TISE – Traditional IS Engineering (non–computerised); CASE – Computer-aided software engineering; CISE with EM – Computerised IS Engineering (with Enterprise Modelling); KBISE – Knowledge-based IS engineering.

Table 1. Differences of IS Engineering Development Stages

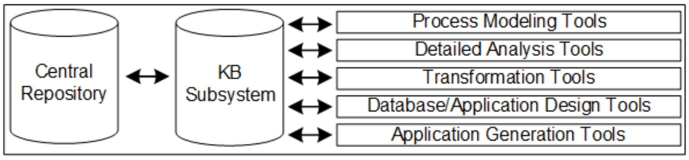
ISEDS	Methods	Sources of knowledge (participants)/ People	Software
TISE	IS development life cycle phases based design using individual methods	User Analyst Designer Programmer	Database design tools Development Tools
CASE	Computer-aided software engineering method includes parts from IS development life cycle: Design method Realization method	User Analyst Designer Programmer	User requirements specification tools Database design tools User interface design tools Use Case tools Development tools
CISE with EM	Computerised IS Engineering (CASE) method includes all IS development life cycle: Enterprise Modelling Design method IS realization method	User Analyst Designer Programmer	Enterprise Modelling tools User requirements specification tools Database design tools User interface design tools Use Case tools Development tools
KBISE	Knowledge-based CASE method includes: Enterprise knowledge modelling Enterprise Modelling Design method Realization method	User, Analyst, Designer, Programmer *CASE knowledge-based subsystem*	Enterprise Modelling tools Knowledge-based subsystem User requirements specification tools Database design tools User interface design tools Use Case tools Development tools

Source: created by the author according [30][32][33][64A]

The computerised IS engineering-specific methods are developed based on common requirements, which systematise the selected methodology. There are some necessary components for knowledge-based IS engineering methodology (Figure 9) [30][32][33]:

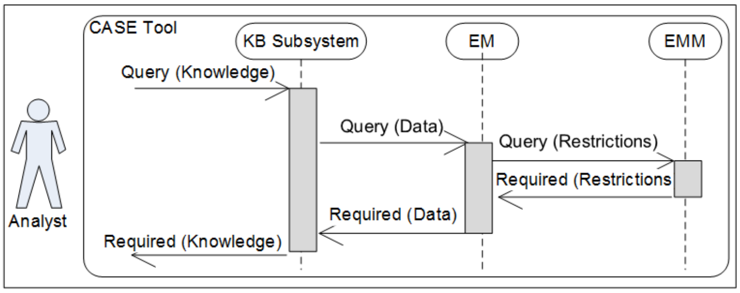
- The core of Enterprise Modelling's theoretical basis is the theoretical Enterprise Model. Its purpose is to identify the necessary and sufficient business components for IS engineering, which implements the organisation's business management. Theoretical Enterprise Model is a formalised enterprise management model that identifies business components and their interactions, enterprise management and their interactions.

- The theoretical enterprise knowledge model is the Enterprise Meta-Model. Based on the IS development life cycle phases, the theoretical model components and their interactions are described as Enterprise Meta-Model. The Enterprise Meta-Model is a structural model which specifies the necessary and sufficient components of IS engineering enterprise management features and interactions.
- The theoretical basis of the knowledge-based IS engineering process is IS engineering process model (methodology) that justifies the knowledge-based IS development process by using an Enterprise Model as an additional knowledge source besides the analyst and the user.
- Computerised IS engineering systems development is based on knowledge-based IS engineering tools: an Enterprise Modelling approach, design models, and use case methods. The right tools are needed for the implementation of the knowledge-based IS development life cycle phase, such as practical knowledge-based modelling methods or model sets. Practical knowledge-based modelling methods are intended for the development of functionality of traditional CASE systems by creating a knowledge-based – CASE intelligent system.
- A computerised knowledge-based IS engineering project management basis is a CASE system knowledge-based subsystem. CASE system’s knowledge-based subsystem's core component is the knowledge base, which essential elements are Enterprise Meta-Model specification and Enterprise Model for the particular subject domain. A knowledge-based subsystem is one more active participant in IS engineering process besides the analyst, whose purpose is to verify the results of IS development life cycle phases.



Source: created by the author [23][64A]

Figure 9. Knowledge-Based CASE Tool Components



Source: created by the author [23][65A]

Figure 10. Knowledge-Based Subsystem Connection to the Enterprise Model and Enterprise Meta-Model Inside a CASE Tool

Knowledge-based CASE systems contain essential components which organize knowledge: knowledge-based subsystem's knowledge base, which essential elements are Enterprise Meta-Model specification and Enterprise Model for certain subject domains. In the principal figure scheme of interaction between the CASE tool's knowledge-based subsystem, Enterprise Model and Enterprise Meta-Model and how this system is related to the analyst is presented (Figure 10) [23][65A][71][79].

There are fewer logical breaks in a knowledge-based, computerised IS engineering design than in traditional computerised IS engineering. The logical break of design is when a consistent automated information system design process is terminated to allow the analyst to enter the missing information of IS engineering process [23][26]. Logical breaks exist not only in theoretical IS engineering models but also are observable in many CASE tools, such as System Architect, MagicDraw, Enterprise Architect and others. The majority of CASE tool's project models are generated only partially, and their complete implementation is possible just by using the systems analyst experience [51][57][73].

In a knowledge-based computerised IS engineering, all project models can be generated interactively using generation algorithms if the necessary knowledge is collected into the knowledge repository. At least the participation of an analyst and designer is required to ensure a missing knowledge entry [26][51][57]. Knowledge is tested for completeness and verified to ensure automatically generated design models and software code quality in knowledge-gathering into the knowledge repository stage [73][85].

The differences in the IS engineering development stages are engineering methods, sources of knowledge, i.e. participants who are involved in IS development process and the necessary software are mentioned in Table 1 and more detailed described below [23][26][100A]:

- Traditional IS Engineering (non-computerised) – Methods: IS development life cycle phases based design using individual methods; Sources of knowledge (participants)/People: User, Analyst, Designer, Programmer; Software: Database design tools, Development Tools.
- Computer-aided software engineering – Methods: Computer-aided software engineering method includes parts from IS development life cycle: Design method, Realization method; Sources of knowledge (participants)/People: User, Analyst, Designer, Programmer Software: User specification tools, Database design tools, User interface design tools, Use Case tools, Development tools
- Computerised IS Engineering (with Enterprise Modelling) – Methods: Computerised IS Engineering (CASE) method includes all IS development life cycle: Enterprise Modelling, Design method, IS realization method; Sources of knowledge (participants)/People: User, Analyst, Designer, Programmer Software: Enterprise Modelling tools, User requirements specification tools, Database design tools, User interface design tools, Use Case tools, Development tools
- Knowledge-based IS engineering – Methods: Knowledge-based CASE method includes: Enterprise knowledge modelling, Enterprise Modelling, Design method, Realization method; Sources of knowledge (participants)/People: 1. User, Analyst, Designer, Programmer 2. CASE knowledge-based subsystem Software: Enterprise Modelling tools, Knowledge-based subsystem, User requirements specification tools, Database design tools, User interface design tools, Use Case tools, Development tools.

1.6. Relation Between MOF and UML

Nowadays, information system description with textual information and basic schemes is not enough because of their complexity of them [15][45][62A]. Modern information systems are engaged in a variety of data, information, and knowledge-based problems. Earlier, most information systems were data-oriented only, and their initial purpose was to store, retrieve and control data [77][89][96A][98A].

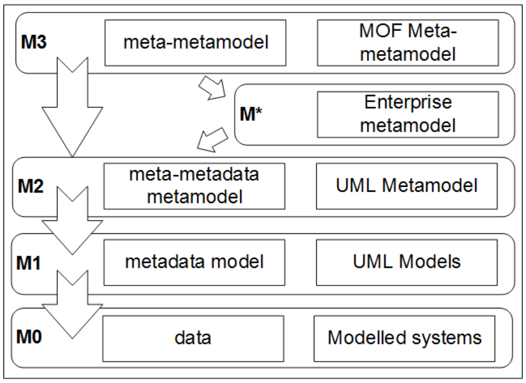
Information system engineering extends through the whole life cycle of systems. Information system engineering is based on the traditional knowledge and empirical experience of the analyst combined with supplementary abilities achieved from previous practice [45][62A][77][89]. The information systems engineering process is a logical sequence of actions and solutions that converts operational demands into a description of system performance configuration [90][95A][96A].

The Meta-Object Facility (MOF) is an Object Management Group (OMG) standard for model-driven engineering. Its purpose is to provide a type system for entities and a set of interfaces through which those types can be created and manipulated. MOF only provides a means to define the structure or abstract syntax of a language or data [74].

Enterprise Modelling has become an inextricable part of the information system development process. MOF architecture supplemented with Enterprise Meta-Model assures the appropriate information system development process [62A][77][89].

Computerised IS engineering is developing based on new knowledge-based methods in order to avoid empirical influence. A knowledge-based IS engineering offers system modelling and decision-making methods and tools, which help to develop a more precise and detailed subject domain corresponding to the project [62A][77][89].

The Meta-Object Facility (MOF) standard is designed as a four-layered architecture. It provides a meta-meta model at the top layer, called the M3 layer. This M3-model is the language used by MOF to build meta-models, called M2-models. The most prominent example of a Layer 2 MOF model is the UML meta-model, which describes the UML itself. These M2-models describe elements of the M1-layer, and thus M1-models, for example, models written in UML. The last layer is the M0-layer or data layer. It is used to describe real-world objects of the particular subject domain. MOF is a closed meta-modelling architecture; it defines an M3-model, which conforms to itself [74].



Source: created by the author according to [74][90][96A][98A]

Figure 11. MOF Architecture with Additional EM Layer

As described above, M3 is a meta-meta model, the base for a meta-modelling architecture, which defines the language to describe meta-models. Moreover, M2 meta-model is an instance of a meta-meta model, which defines the language to describe models. According to the thesis author, among M3 and M2 layers, one more layer is needed to ensure more accurate usage of MOF architecture [74][96A][98A]. This additional layer consists of Enterprise Meta-Model (EMM) (Figure 11). Enterprise Meta-Model, as already mentioned, is a formal structure which ensures a more qualified information system development process and knowledge-based data collection. Enterprise Model and Enterprise Meta-Model make the UML design model generation process more efficient and eligible [62A][96A][98A].

1.7. ISO Standards in IS Engineering

ISO standards make a positive contribution to the world. They provide solutions and achieve benefits for almost all sectors of activity [38][39]. ISO Standards are documented agreements containing technical specifications or other precise criteria to be used consistently as rules, guidelines, or definitions of characteristics, to ensure that materials, products, processes and services are fit for their purpose [41][42][44].

Practically, information systems engineering extends during the entire life cycle of systems, involving requirement definitions, functional designs, development, testing, and evaluation. As it is mentioned, information systems engineering is based on the traditional knowledge and personal experience of the analyst combined with additional abilities gained from previous practice [24][29][95A].

ISO (the International Organisation for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. The main task of the joint technical committee is to prepare International Standards for IT services and software and systems engineering [39][41][42][44].

International standards in IT service management and software and system engineering are also excellent references to what is considered good practice by the international community of professionals that work in these areas [15][25]. Modelling languages are quite strictly specified in ISO standards. Several standards and business modelling methodologies are used in the information systems development process [24][29], and UML is one of the most common software specification languages. UML modelling language has become popular for modelling software-intensive systems [52][72][91]. There is part of ISO standards where UML usage or methods based on UML are described.

According to creators, the Enterprise Meta-Model is a formally defined Enterprise Model structure, which consists of a formalized Enterprise Model in line with the general principles of control theory [24][25][58]. Enterprise Meta-Model manages Enterprise Model composition and stores knowledge that is necessary for IS development process only and can be used during all phases of IS development life cycle. The necessary knowledge is collected to the knowledge-based subsystem, where the main components are Enterprise Model and Enterprise Meta-Model. Formalized Enterprise Model provides a knowledge base, which ensures quality and verified knowledge in specific IS development-related situations [33][63A][98A].

ISO standards-based information system development life cycle (19505, 19501) ensures formality, where appropriate sets of criteria and restrictions for the professional practice of software engineering are established [40][43]. Regarding these criteria, professional decisions can be made in the information system development process [39][41][44].

In the past, UML was often criticized as being too large to be implemented as a whole and too complex to be realized in detail [52][91]. UML models based on Enterprise Model implement knowledge-based information systems development cycle because Enterprise Model stores knowledge necessary for IS development process. The usage of the Enterprise Model is appropriate to collect the necessary knowledge for the UML model generation process [63A][95A].

1.8. ISO Standards in the Requirements Specification Phase

The ability to design, implement and manage information systems has highly improved in the last twenty years. A core body of knowledge in software and systems engineering now exists. Its necessity remains because of the demand to create and deliver more complex applications, systems and services in a shorter period [55].

Developing more complex information systems under a short time-frame and delivering the required IT services in the most cost-effective range will remain an aspiration. International standards play an essential part in this process [38].

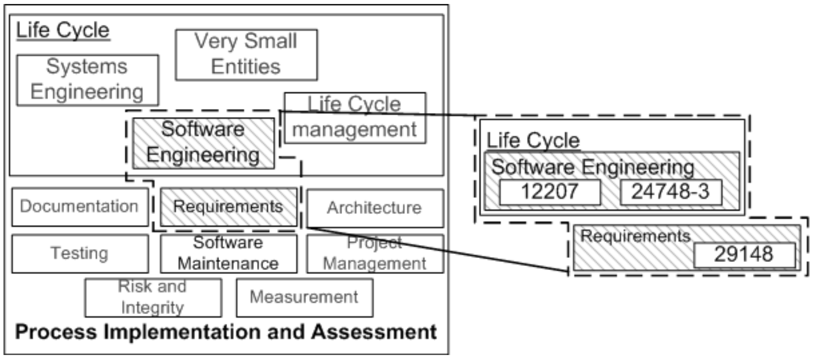
As it was mentioned, international standards in software and system engineering are an excellent indication of what is considered good practice by the international community of professionals that work in these areas [38][39][41][42][44].

Subcommittee 7 (SC7) is responsible for IT services and software and systems engineering standardization: software and systems engineering processes, software system products, enterprise architecture, software engineering environment, software engineering body of knowledge, management of IT assets [42][44].

SC7 standards are constantly updated by developing and improving on standards. One of the main scopes is integrating IT and business system definitions as one of the main SAM goals and providing the software and system engineering tools to implement enterprise information systems [42][44].

SC7 standards collection consists of several blocks, where one of the most relevant is process implementation and assessment. It consists of standards for the life cycle in major, assessment and certification, IT service management and all phases of the information system development process. Life cycle standards are divided into the following groups: systems engineering, software engineering, life cycle management and small entities. The requirements specification phase is one of the essential phases of the information systems life cycle phases. There are standards designated for this phase (Figure 11) [42][44].

ISO/IEC 12207 was published on 1 August 1995 and was the first International Standard to provide a comprehensive set of life cycle processes, activities and tasks for software that is part of a more extensive system and for standalone software products and services [42][44][95A].



Source: created by the author [95A]

Figure 12. ISO Standards in the Requirements Phase of the Information Systems Development Process

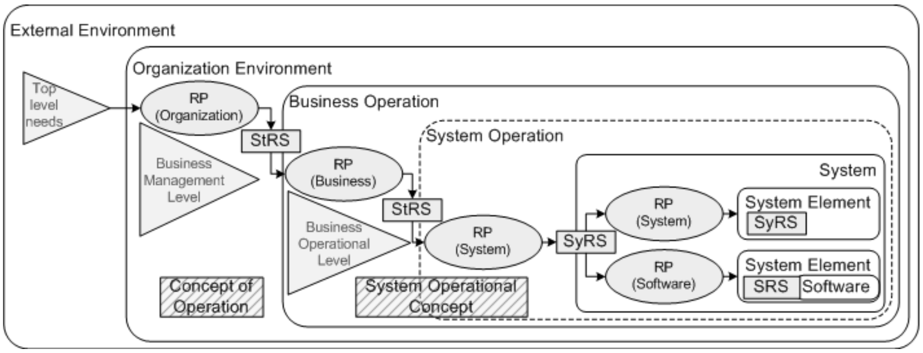
The current version ISO/IEC 12207:2008 establishes a common framework for software life cycle processes, with well-defined terminology that can be referenced by the software industry (Figure 12). The limitation of this International Standard is that it does not detail the life cycle processes in terms of methods or procedures required to meet the requirements and outcomes of a process [38][42][44][88]. ISO/IEC TR 24748-3:2011 is a guide for applying ISO/IEC 12207:2008. It addresses system, life cycle, process, organisational, project, and adaptation concepts [42][44][95A]. ISO/IEC/IEEE 29148:2011 provides additional guidance in the application of requirements engineering and management processes for requirements-related activities in ISO/IEC 12207. The content of ISO/IEC/IEEE 29148:2011 can be added to the existing set of requirements-related life cycle processes defined by ISO/IEC 12207 or can be used independently [39][42]. It is also possible to rely on ISO/IEC 19501 and 19505 and use them independently [40][43].

It is widely known amongst researchers and industry practitioners that software projects are significantly vulnerable when the requirements related activities are barely accomplished [39][95A]. According to ISO 29148 requirements quality characteristics, qualities of requirements specification are completeness, consistency, affordability, boundedness, and characteristics of individual requirements are a necessity, implementation freeness, non-ambiguity/uniquity, completeness, singularity, feasibility, traceability and verifiability [42][95A].

The software requirements knowledge area is related closely to software design, software testing, software maintenance, software configuration management, software engineering management, software engineering process, software engineering models and methods, and software quality knowledge area [38][44].

1.9. ISO-Based Requirements Storing to Enterprise Model

According to ISO 29148 requirements, processes and their specifications depend on the coverage of the system for which the requirements are defined. The stakeholder requirements specification (StRS), the system requirements specification (SyRS) and the software requirements specification (SRS) are intended to represent different sets of required information items. The specifications correspond to the requirements in Figure 13 as follows: StRS – stakeholder requirement (business management level and operational business level); SyRS – system requirements; and SRS – software requirements. These information items can be applied to multiple specifications (instances) iteratively or recursively [42][87][95A]; they can be applied to various templates, tools or applications and can be used in decision tables. Enterprise model element Business rules can be stored for the requirements, especially all constraints.



Source: created by the author [95A]

Figure 13. The Sequence of Requirements Processes and Specifications

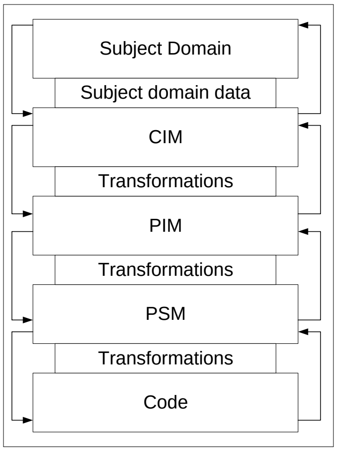
The concept of operation and the system operational concept are helpful in eliciting requirements from various stakeholders in an organisation and as a practical means to communicate and share the organisation’s intentions. At the organisation level, the concept of operation addresses the leadership's intended way of operating the organisation. The system operational concept addresses the specific system-of-interest from the user's viewpoint. Information items represented in the StRS, SyRS, SRS, the concept of operation and the system operational concept documents are interdependent [42][95A].

1.10. Model Driven Architecture and Enterprise Modeling

The main idea of Model Driven Architecture (MDA) is to separate the models that describe the functionality of a system from the models that describe how that functionality should be implemented (the technical aspects of implementation), i.e. to separate the “what to do” from the “how to do” [2][4][6]. MDA distinguishes three qualitatively different types of models (layers) that are used in IS development [4][8]:

- CIM – computation Independent Model. It is a model of the functional and non-functional requirements of a system. The procedures for building this model, its structure, and change management are equivalent to the requirements gathering and management phases of classical software development methods. The MDA does not contain a precise specification for the structure of this model as well as for the compilation procedures, which is why the CIM compilation has been interpreted differently by different authors. A CIM can be composed of a set of models that separately describe a system’s static and dynamic information, with UML as one example.
- PIM – a platform-independent abstract model that contains enough information to be transformed into a platform-specific model (PSM). A platform-independent model describes the structure and functions of a system at the architectural level but not how they are implemented.
- PSM – a platform-specific model that contains information about what the system should do (functional information) and how it should do it (technical implementation information). This model is created by transforming the PIM model, i.e. by using additional annotations on object types, platform-specific object relationships and elements.

The OMG specifications are used to create a single CIM model for software development, which must specify all system requirements and processes, and to create a single PIM model based on this model using transformations (Figure 14). The number of PSM models is unlimited (but not less than one) and depends on the specific project’s needs. In software development based on MDA principles, the process starts with the specification of user requirements and the creation of a CIM model [86][111]. This phase is empirical, i.e. the systems analyst interprets the information provided by the user about the system requirements and formalises them using the chosen modelling language. To avoid empirical impact for further process Enterprise model as storage of subject domain knowledge can be used [86].



Source: created by the author according to [86]

Figure 14. MDA Layers and the Process

The model is transformed into an architectural model (PIM). This transformation can be performed automatically, provided that the CIM model has been described using formal modelling languages, UML, in this research case and that CIM to PIM interface maps exist. PIM to PSM transformations are performed automatically by replacing the base PIM types with platform-specific types and relationships. Software code generation from the PSM model is also done automatically using code generation tools.

1.11. First Part Conclusions

The first part consists of an analysis of the business and IT alignment model and its challenges, of how ISO standards can be used in IS development process, its advantages and disadvantages of them, of differences between traditional and knowledge-based IS engineering, of definitions of MOF and MDA frameworks. This part presents how the Enterprise model can be used as one of the IS development process elements.

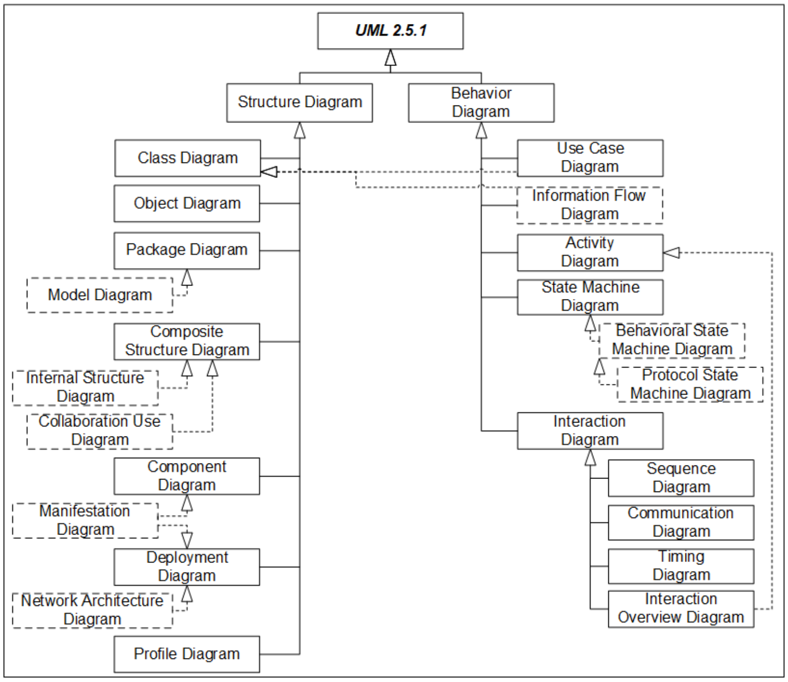
2. EM TO UML MODELS TRANSFORMATION METHOD

The second section presents the knowledge-based generation method. This chapter introduces a detailed description and composition of the provided method by describing UML dynamic model compositions, transformation rules and presenting transformation algorithms from the Enterprise Model [5][24][81].

The Unified Modeling Language (UML) is a common language for software architects, business analysts and developers used to specify, describe, design and document the existing or new business processes and structure of IS under development. UML 1.4.2 specification, which became the international standard ISO 19051, explains software development importance: delivers guidance as to the order of a team's activities, refers to what artefacts should be developed, specifies the tasks of the team as a whole and individual developer and proposes criteria for measuring and monitoring a project's activities and products [9][75][91].

The current version of UML is UML 2.5.1, released in December 2017. 2.5.1 tools will have to support a complete UML specification (Figure 13). Information flows, models, and templates will no longer be auxiliary UML constructs. At the same time, use cases, deployments, and information flow become UML supplementary concepts [75][91].

UML specification defines two major kinds of UML diagrams: structure or static diagrams and behaviour or dynamic diagrams (Figure 15). Structure diagrams show the static structure of the system and its parts on different implementation and abstraction levels and how they are related to each other [75][77][81]. The elements in structure diagrams represent the meaningful concepts of a system. The elements may include abstract, real-world and implementation concepts. Behaviour diagrams show the dynamic behaviour of the objects in a system, which can be described as a series of changes to the system over time. [84][91][110].



Source: [75][91]

Figure 15. UML Diagrams (Version 2.5.1)

Computer-aided design tools facilitate the designer’s work but still do not automate enough because serious inclusion of the analyst is required. Great attention is paid to modelling, and the designer has to spend a lot of time on that, especially when the final result is the whole IS instead of design models. The primary purpose is to formalise the model creation process as much as possible. A variety of methodologies has been designed, but still, there are missed opportunities that would improve the IS design methods that are controlled by CASE tools.

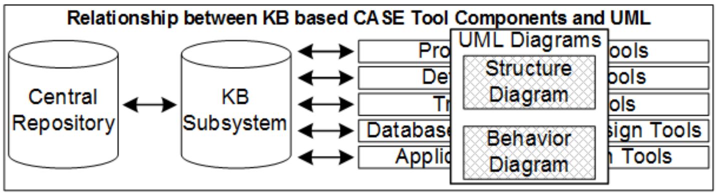
UML modelling language intended to describe the design decisions. It is a modelling language that defines a graphical notation for the various aspects and perspectives of software architecture modelling [77][92].

Information systems design methods specify the sequence of systems engineering actions, i.e. how, in what order and what UML diagrams to use in the design process and how to implement the process [75][77][81]. Many of them are based on several types of diagrams describing various aspects of the system’s properties. The meaning of each of them can be defined individually, but more important is the fact that each diagram is the projection of the same system [84][91][110].

This kind of system description is quite confusing because most of the information in the diagrams overlaps and describes the same things, just in different ways. An inexperienced specialist can misuse UML diagrams, and the description of the system will possibly be inconsistent, incomplete and controversial. Currently, UML CASE tools cannot help the designer much [24][100A].

Formalisation brings new software and information systems design and development opportunities. When the maximum coherence of UML models is reached, models are linked to each other, clearly articulated by the rules, expressed stronger and more completely. It greatly facilitates the task of automated software development.

Since December of 1997, when the OMG announced UML as a standard. This increasingly popular language makes a significant impact on IS design. During the first year of using this language, it was used for modelling information systems, but it is suitable for modelling business processes and gained ground among business analysts. UML is capable of defining the appropriate business structural and behavioural rule aspects. The use of UML language can provide stability of documentation and make communication easier between the designers and all other types of stakeholders [75][84][91].

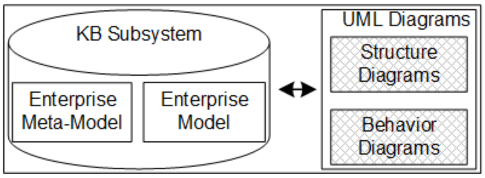


Source: created by the author [64A][98A]

Figure 16. Knowledge-Based IS Engineering CASE Tool Components and UML

Interaction between UML diagrams and Enterprise Model can be realized through the transformation algorithms. Business elements (participants, processes, functions, constraints) – subject domain knowledge stored in the Enterprise Model can be generated as UML diagrams elements using transformation algorithms. The integration of knowledge-based subsystem and Enterprise Model usage makes it possible to generate knowledge proven by formal criteria into the UML models (Figure 16). Usage of the Enterprise Model and knowledge-based IS development process automatization saves the working hours of designers and other stakeholders. Knowledge-based subsystems and UML interaction inside the CASE tool are shown in Figure 17.

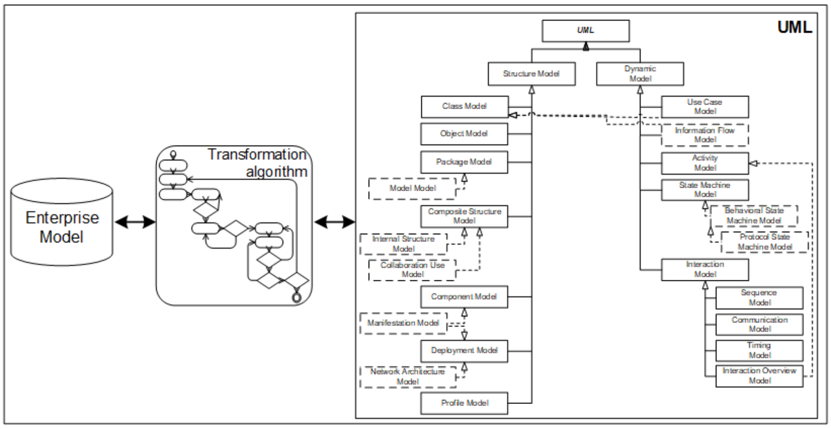
Enterprise Model as an organisation's knowledge repository allows to generate UML diagrams after using the transformation algorithms (Figure 18). Such a repository can be used not only for knowledge of the organisation's accumulation but also as a tool that minimizes IS reengineering scope of work if changes occur in an organisation.



Source: created by the author [64A]

Figure 17. Knowledge-Based Subsystem and UML Diagrams Relationship

Enterprise Models have been concluded in accordance with the notations (such as Data flow diagrams, workflow models and so on). However, their composition has not been checked by the characteristics of the specific subject domain, but this knowledge may be used as a background for UML model generation [64A][98A].



Source: created by the author [64A]

Figure 18. UML model generation by transformation algorithms

UML dynamic models can be generated from the Enterprise Model using transformation algorithms (Figure 18). Firstly, a specific UML model must be identified for the generation process; after this identification, the initial – main element of this UML model must be selected from the Enterprise Model. Secondly, all related elements must be selected according to the initial element, and all these related components must be linked according to related constraints necessary for the UML model type identified at the beginning of the process [64A][98A].

2.1. UML Models Top-Level Transformation Algorithm

Information systems design methods specify the arrangement of systems engineering actions and which UML models' element can be generated from the EM element (Table 2). Many of them are based on diverse types of models describing different aspects of the system properties. The sense of each model can be defined individually, but more important is that each model is the projection of the same system and will be helpful in the whole IS development process [100A].

Table 2. EM Process, Function, Actor and Business Rules elements roles variations as different UML dynamic models elements

EM	**UML Model element**	**UML Model**	**Description**
Process/Function	Use Case	Use Case model	A use case is a type of behavioural classifier that defines a unit of functionality achieved by actors or subjects to which the use case applies in combination with one or more actors.
	Activity	Activity model	Describes a parameterised behaviour as a correlative flow of actions.
	Behavioural State Machine	State Machine model	Specifies individual behaviour of a part of a designed system through limited state transitions
	Protocol State Machine	Protocol State Machine model	Expresses a usage protocol or the life cycle of some classifier.
	Message	Sequence model	Describes one specific kind of communication between the lifelines of an interaction.
	Frame	Communication model	Describes a unit of behaviour that concentrates on the appreciable exchange of information between connectable elements.
	Frame	Interaction Overview model	Describes a unit of behaviour that concentrates on the appreciable exchange of information between connectable elements.
Actor	Actor	Use Case Model	Represents a role played by some person or system external to the modelled system.
	Subject	Use Case Model	Represents the behaviour of the participant.
	Partition	Activity Model	Describes actor or actor group actions that have some common characteristics.
	Lifeline	Sequence Model	Represents the lifeline of the actor.
	Lifeline	Communication Model	Represents part of the sequence model lifeline.

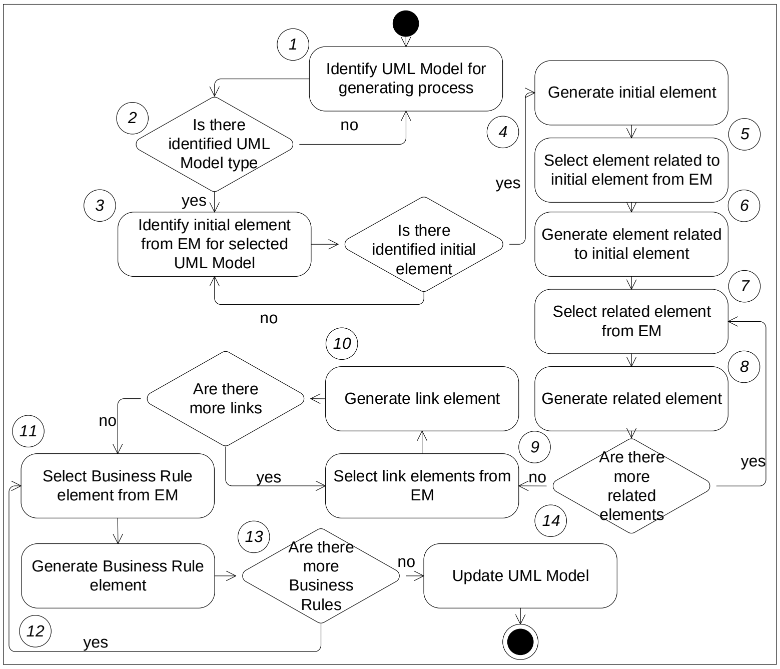
	Lifeline	Timing Model	Represents an individual participant in the interaction
Business Rules	Extend	Use Case Model	Extend is a directed relationship that specifies how and when the behaviour defined in usually supplementary (optional) extending use case can be inserted into the behaviour defined in the extended use case.
	Include	Use Case Model	Use case include is a directed relationship between two use cases which is used to show that the included use case’s behaviour is inserted into the included use case's behaviour.
	Association	Use Case Model	Each use case represents a unit of proper functionality that subjects provide to actors. An association between an actor and a use case indicates that the actor and the use case somehow interact or communicate with each other.
	Control Nodes	Activity Model	Used to coordinate the flows between other nodes. It includes: initial, flow final, activity final, decision, merge, fork, join.
	Pseudostate	State Machine model	An abstract node that encompasses different types of transient vertices in the state machine graph
	Protocol Transition	Protocol State Machine model	Used for the protocol state machines, which specifies a legal transition for an operation
	Execution Specification	Sequence model	Represents a period in the participant’s lifetime.
	Combined Fragment	Sequence model	Defines a combination (expression) of interaction fragments. An interaction operator and corresponding interaction operands define a combined fragment.
	Interaction Use	Sequence model	Allows to use (or call) another interaction.
	State Invariant	Sequence model	Represents a runtime constraint on the participants of the interaction.
	Destruction Occurrence	Sequence model	Represents the destruction of the instance described by the lifeline.
	Duration constraint	Timing model	Refers to a duration interval. The duration interval is the duration used to determine whether the constraint is satisfied.
	Time Constraint	Timing model	Refers to a time interval. The time interval is the time expression used to determine whether the constraint is satisfied.
	Destruction Occurrence	Timing model	Represents the destruction of the instance described by the lifeline.
	Duration Constraint	Interaction Overview model	Refers to a duration interval. The duration interval is the duration used to determine whether the constraint is satisfied.
	Time Constraint	Interaction Overview model	Refers to a time interval. The time interval is the time expression used to determine whether the constraint is satisfied.

	Interaction Use	Interaction Overview model	Allows to use (or call) another interaction.
	Control Nodes	Interaction Overview model	Used to coordinate the flows between other nodes. It includes: initial, flow final, activity final, decision, merge, fork, join.

Source: created by the author according to [75][91][98A]

Identifying a particular UML model and selecting the initial model element is quite significant because the further generating process depends on it. Many UML model elements repeat in different UML models, but these elements define various aspects of the system [64A][75]. In the example, Enterprise Model elements, such as Process, Function, Actor and Business Rules, can be generated into different UML dynamic model elements depending on which UML model is chosen for a generation [91][98A].

In IS engineering, all design models are implemented based on empirical expert experience. Experts, who participate in the IS development process, do not gain enough knowledge, and process implementation in requirements analysis and specification phases can take too much time. Enterprise Meta-Model contains essential elements of business modelling methodologies and techniques, that ensure a suitable UML model generation process [64A][98A].


1
Identify UML Model for generating process
2
Is there identified UML Model type
no
yes
3
Identify initial element from EM for selected UML Model
Is there identified initial element
no
4
yes
Generate initial element
5
Select element related to initial element from EM
6
Generate element related to initial element
7
Select related element from EM
8
Generate related element
Are there more related elements
yes
9
no
14
10
Generate link element
Are there more links
no
yes
Select link elements from EM
11
Select Business Rule element from EM
Generate Business Rule element
13
Are there more Business Rules
no
yes
12
Update UML Model

Source: created by the author [101A][103A]

Figure 19. The Top-Level Transformation Algorithm of UML Model Generation from the EM Process

The transformation algorithm (Figure 19) is a top-level algorithm for Enterprise Meta-Model-based UML models generating process. The main steps for generating process are identifying and selecting the UML model for generating process, identifying starting elements for the selected UML model and selecting all related elements, mapping Enterprise Model elements to UML model elements and generating the selected UML model [101A][103A].

The transformation algorithm of UML model generation from the Enterprise Model process is depicted by the following steps [101A][103A][104A]:

- Step 1: Particular UML model for generation from the Enterprise Model process is identified and selected.
- Step 2: If the particular UML model for generation from the Enterprise Model process is selected, then the algorithm process is continued, else the particular UML model for generation from the Enterprise Model process must be selected.
- Step 3: The first element from Enterprise Model is selected for the UML model, identified previously, generation process.
- Step 4: If the selected Enterprise Model element is an initial UML model element, then the initial element is generated, else the other Enterprise Model element must be selected (the selected element must be an initial element).
- Step 5: The element related to the initial element is selected from the Enterprise Model.
- Step 6: The element related to the initial element is generated as a UML model element.
- Step 7: The element related to the previous element is selected from the Enterprise Model.
- Step 8: The element related to the previous element is generated as a UML model element.
- Step 9: If there are more related elements, they are selected from the Enterprise Model and generated as UML model elements one by one, or the link element is selected from Enterprise Model.
- Step 10: The link element is generated as a UML model element.
- Step 11: If there are more links, then they are selected from the Enterprise Model and generated as UML model elements one by one, else the Business Rule element is selected from the Enterprise Model.

- Step 12: The Business Rule element is generated as a UML model element.
- Step 13: If there are more Business Rules, then they are selected from Enterprise Model and generated as UML model elements one by one, else the generated UML model is updated with all elements, links and constraints.
- Step 14: The generation process is finished.

2.2. UML Use Case Diagram Transformation Algorithm

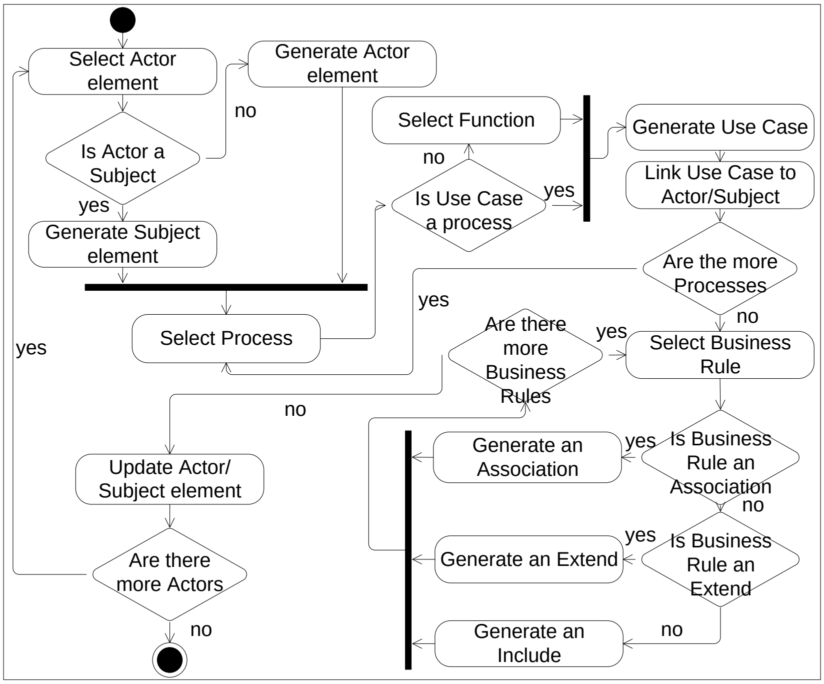
UML Use Case diagrams are usually referred to as dynamic models used to describe a series of actions that some system or systems should or can implement in contribution to one or more external users of the system. Each use case should grant some observable and valuable results to the actors or other participants of the system. UML Use Case model elements [101A][102A][103A][104A].

Table 3. UML Use Case Model Elements

EM element	**UML Use Case model element**	**Description**
Actor	Actor	An actor is a behavioural classifier that defines a role played by an external entity.
	Subject	A subject is a classifier representing a business, software system, physical system or device under analysis, design, or consideration, having some behaviour to which a set of use cases applies.
Function, Process	Use Case	A use case is a type of behavioural classifier that describes a unit of functionality performed by actors or subjects to which the use case applies in collaboration with one or more actors.
Business Rule	Extend	Extend is a directed relationship that specifies how and when the behaviour defined in usually supplementary (optional) extending use case can be inserted into the behaviour defined in the extended use case.
	Include	Use case include is a directed relationship between two use cases which is used to show that the included use case’s behaviour is inserted into the included use case's behaviour.
	Association	Each use case represents a unit of proper functionality that subjects provide to actors. An association between an actor and a use case indicates that the actor and the use case somehow interact or communicate with each other.

Source: created by the author according [101A][102A][103A][104A]

Input (elements from Enterprise Model) and output (elements generated to UML Use Case model) elements are presented in Table 3 [101A][102A][103A][104A].



Source: created by the author in previous publications [101A][102A][103A][104A]

Figure 20. The Transformation Algorithm of UML Use Case Model Generation from EM Process

Figure 20 presents the transformation algorithm of UML Use Case model generation from the EM process [101A][102A]. UML Use Case model generation from Enterprise Model initial element is actor or subject, after the generation of this element, follows the selection of Enterprise Model element: process or function and Use Case element are generated. After the generation of these two elements, they have to be linked to each other with some type of relationship: association, extension or inclusion, defined by the Enterprise Model element Business Rule. After all these elements are generated, there is an update of the actor or subject element and check if more actor elements are left in the Enterprise Model [103A][104A].

2.3. UML Activity Diagram Transformation Algorithm

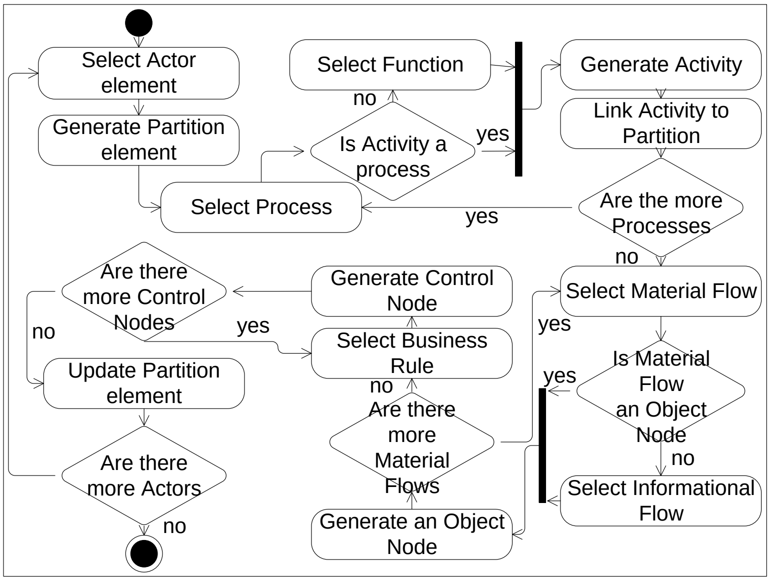
The Activity diagram is a UML dynamic model that shows a flow of control or object flow with emphasis on the sequence and conditions of the flow. The actions which are coordinated by activity models can be initiated because other actions finish executing because objects and data become available or because some events external to the flow occur [94A][101A][102A][103A][104A].

Table 4. UML Activity Model Elements

EM element	**UML Activity model element**	**Description**
Actor	Partition	Describes actor or actor group actions that have some common characteristics.
Function, Process	Activity	Represents a parameterized behaviour as the coordinated flow of actions.
Material Flow, Informational Flow	Object Nodes	Used to define object flows in an activity.
Business Rules	Control Nodes	Used to coordinate the flows between other nodes. It includes: initial, flow final, activity final, decision, merge, fork, join.

Source: created by the author [91][101A][102A][103A][104A]

Input (elements from Enterprise Model) and output (elements generated to UML Activity model) elements are presented in Table 4 [94A][101A][102A][103A][104A].



Source: created by the author[101A][102A][103A][104A]

Figure 21. UML Activity Model Transformation Algorithm

In UML Activity model generation from Enterprise Model (Figure 21) initial element is a partition; after the generation of this element, follows a selection of Enterprise Model element: process or function and activity element is generated. Afterwards, the generation of these two types of elements must be linked to each other. Object nodes are generated from Enterprise Model material or informational flow elements in activity models. Furthermore, all these generated elements are connected through control nodes based on Enterprise Model business rule elements. Later all these elements are generated, there is an update of the partition element, and check there are more actor elements left in Enterprise Model [101A][102A][103A][104A].

2.4. UML State Machine Diagram Transformation Algorithm

UML State machine diagram is used for modelling discrete behaviour through finite state transitions. In addition to expressing the behaviour of a part of the system, state machines can also be used to express the usage protocol of a part of a system. These two types of state machines are referred to as behavioural state machines and protocol state machines [91][101A][102A][103A][104A].

Table 5. UML State Machine Model Elements

EM element	**UML State Machine model element**	**Description**
Process, Function	Behavioural State Machine	Used to specify discrete behaviour of a part of a designed system through finite state transitions
Informatio n Flow	Simply State	Defined as a state that has no substates.
	Composite State	Defined as a state that has substates.
Business Rule	Pseudostate	An abstract node that encompasses different types of transient vertices in the state machine graph

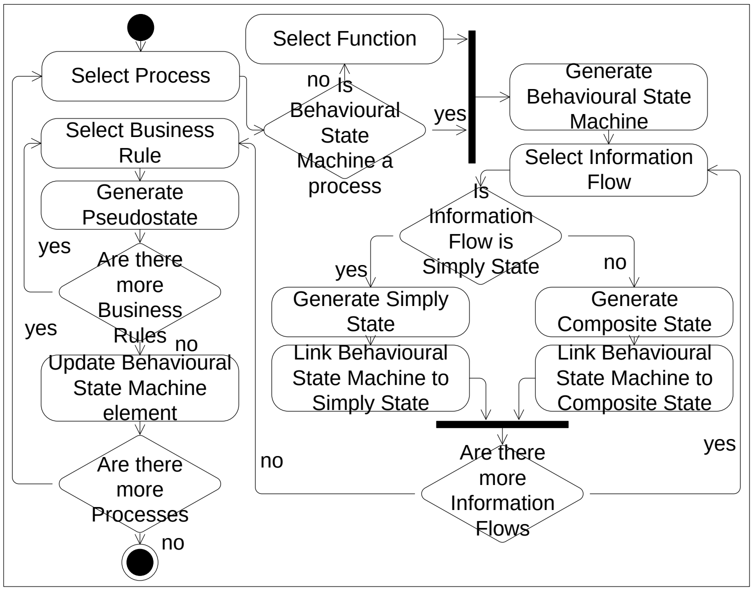
Source: created by the author [91][101A][102A][103A][104A]

Table 6. UML Protocol State Machine Model Elements

EM element	**UML State Machine model element**	**Description**
Process, Function	Protocol State Machine	Used to express a usage protocol or a life cycle of some classifier.
Informatio n Flow	Protocol State	Present an external view of the class that is exposed to its clients.
Business Rule	Protocol Transition	Used for the protocol state machines, which specifies a legal transition for an operation

Source: created by the author [91][101A][102A][103A][104A]

Input (elements from Enterprise Model) and output (elements generated by the UML State Machine model and UML Protocol State Machine model) elements are presented in Tables 5 and 6 [101A][102A][103A][104A].

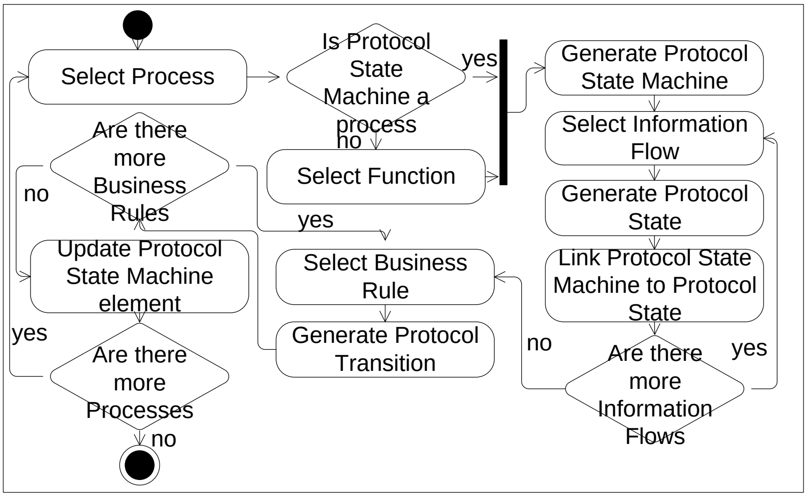

Select Process
Select Function
Is Behavioural State Machine a process
no
yes
Generate Behavioural State Machine
Select Information Flow
Select Business Rule
Generate Pseudostate
yes
Are there more Business Rules
yes
no
Update Behavioural State Machine element
Are there more Processes
no
Is Information Flow is Simply State
yes
no
Generate Simply State
Generate Composite State
Link Behavioural State Machine to Simply State
Link Behavioural State Machine to Composite State
Are there more Information Flows
no
yes

Source: created by the author [101A][102A][103A][104A]

Figure 22. UML State Machine Model Transformation Algorithm

In UML Behavioural State Machine model generation from Enterprise Model (Figure 22), the initial element is a process or function, which means that the behavioural state machine element is generated from these Enterprise Model elements. Subsequently, this element generation second related element is simply state or composite state, which is generated from information flow. Furthermore, the first two elements are linked to each other and with the pseudostate element generated from the business rule. After that, there is an update of the initials element and check if there are more process elements left in Enterprise Model [101A][102A][103A][104A].

Moreover, the second type of State machine is the UML Protocol State Machine model, which defines usage protocol or the life cycle of some classifier [14].


Select Process
Is Protocol State Machine a process
yes
no
Generate Protocol State Machine
Select Information Flow
Generate Protocol State
Link Protocol State Machine to Protocol State
Are there more Information Flows
yes
no
Select Function
Are there more Business Rules
no
yes
Select Business Rule
Generate Protocol Transition
Update Protocol State Machine element
Are there more Processes
yes
no

Source: created by the author [101A][102A][103A][104A].

Figure 23. UML Protocol State Machine Model Transformation Algorithm

In UML Protocol State Machine model generation from Enterprise Model (Figure 23), initials element is also process or function, from these Enterprise Model elements protocol state machine element is generated. After this element generation, the information flow is selected, and the Protocol state element is generated. These two elements are linked to each other and also with the protocol transition element, which is generated from the business rule. Afterwards that there is an update of the initial element and check if there are more process elements left in Enterprise Model.

2.5. UML Sequence model Diagram Transformation Algorithm

UML Sequence diagram is the most common kind of interaction model which focuses on the message interchange between objects (lifelines). The Sequence model shows how the objects interact with others in a particular scenario of a use case [91][101A][102A][103A][104A].

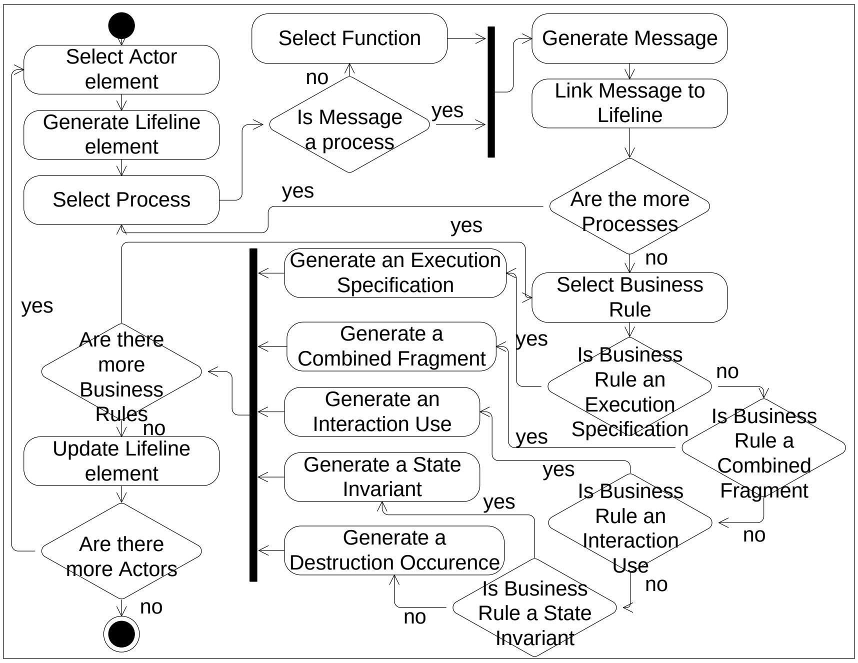
Table 7. UML Sequence Model Elements

EM element	**UML Sequence Model element**	**Description**
Actor	Lifeline	Represents an individual participant in the interaction. While parts and structural features may have a multiplicity greater than 1, lifelines represent only one interacting entity.
Process, Function	Message	Defines one specific kind of communication between the lifelines of an interaction.

Business Rules	Execution Specification	Represents a period in the participant's lifetime.
	Combined Fragment	Defines a combination (expression) of interaction fragments. An interaction operator and corresponding interaction operands define a combined fragment.
	Interaction Use	Allows to use (or call) another interaction.
	State Invariant	Represents a runtime constraint on the participants of the interaction.
	Destruction Occurrence	Represents the destruction of the instance described by the lifeline.

Source: created by the author according [91][101A][102A][103A][104A]

Input (elements from Enterprise Model) and output (elements generated to UML Sequence model) elements are presented in Table 7 [102A][103A][104A].


Select Actor element
Generate Lifeline element
Select Process
Select Function
no
Is Message a process
yes
Generate Message
Link Message to Lifeline
yes
Are the more Processes
yes
no
Generate an Execution Specification
Select Business Rule
yes
Generate a Combined Fragment
yes
Are there more Business Rules
Is Business Rule an Execution Specification
no
Generate an Interaction Use
no
yes
Is Business Rule a Combined Fragment
Update Lifeline element
Generate a State Invariant
yes
yes
Is Business Rule an Interaction Use
no
Generate a Destruction Occurence
Are there more Actors
no
Is Business Rule a State Invariant
no
no

Source: created by the author [101A][102A][103A][104A]

Figure 24. UML Sequence Model Transformation Algorithm

In UML Sequence model generation from Enterprise Model (Figure 24) initial element is the lifeline; after the generation of this element, follows the selection of Enterprise Model element: process or function and message element are generated. Afterwards, the generation of these two types of elements must be linked to each other. In sequence models, execution specification, combined fragment, interaction use, state invariant and destruction occurrence are generated from Enterprise Model business rule elements. Later all these elements are generated, and there is an update of the lifeline element and check whether there are more actor elements left in Enterprise Model.

2.6. UML Communication Diagram Transformation Algorithm

The UML Communication diagram (called collaboration diagram in UML 1.x) is a type of UML interaction model which shows interactions between objects and/or parts (represented as lifelines) using sequenced messages in a free-form arrangement [91][101A][102A][103A][104A]

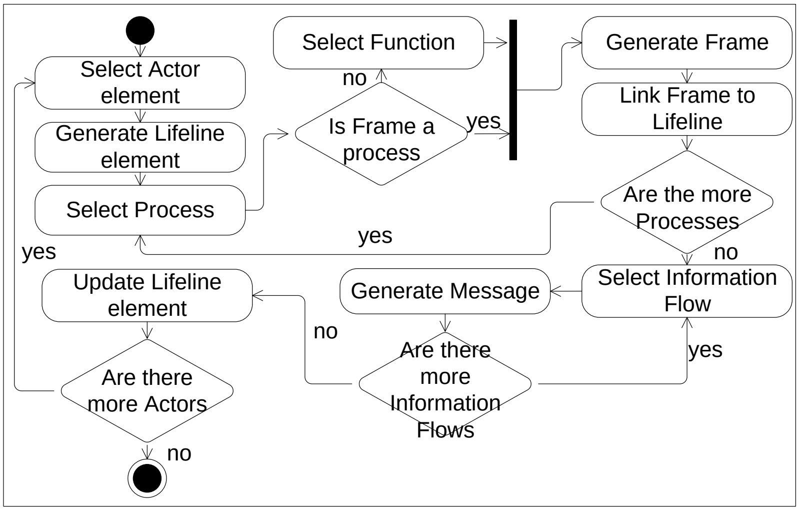
Table 8. UML Communication Model Elements

EM element	**UML Communication Model element**	**Description**
Process, Function	Frame	Represents a unit of behaviour that focuses on the observable exchange of information between connectable elements.
Actor	Lifeline	Represents an individual participant in the interaction.
Informatio n Flow	Message	Indicates the direction of the communication.

Source: created by the author [101A][102A][103A][104A]

Input (elements from Enterprise Model) and output (elements generated to UML Sequence model) elements are presented in Table 8 [101A][102A][103A][104A].

The UML Communication Model is another type of interaction models group, and it focuses on the interaction between participants called lifelines where the architecture of the internal structure and how this corresponds with the objects called message passing is central [91][101A][102A][103A][104A].



Source: created by the author [101A][102A][103A][104A]

Figure 25. UML Communication Model Transformation Algorithm

The initial element is the lifeline in the UML Communication model generation from Enterprise Model (Figure 25). After the generation of this element, the selection of Enterprise Model element follows: process or function and frame element is generated. Subsequently, the generation of these two types of elements must be linked to each other. In communication models, message elements are generated from the information flow element. Later all these elements are generated, there is an update of the lifeline element, and check if there are more actor elements left in Enterprise Model [101A][102A][103A][104A].

2.7. UML Timing Diagram Transformation Algorithm

The UML Timing diagram is an interaction model that shows interactions when the primary scope of the model is to reason about time. The timing model focuses on terms changing within and among lifelines along a linear time axis. Timing models define the behaviour of both individual classifiers and interactions of classifiers, focusing attention on the time of events causing changes in the modelled terms of the lifelines [91][97A][101A][102A]103A][104A].

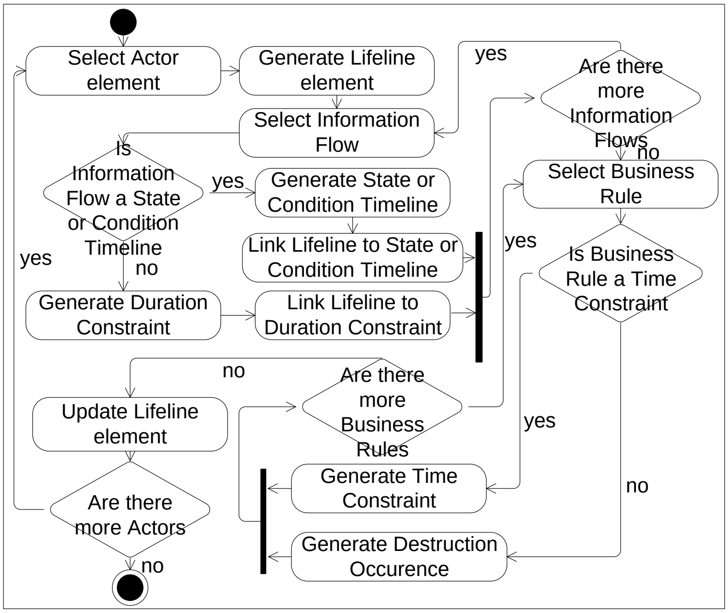
Table 9. UML Timing Model Elements

EM element	**UML Timing Model element**	**Description**
Actor	Lifeline	Represents an individual participant in the interaction. While parts and structural features may have a multiplicity greater than 1, lifelines represent only one interacting entity.
Information Flow	State or Condition Timeline	Shows states of the participating classifier or attribute, or some testable conditions
Business Rules	Duration constraint	Refers to a duration interval. The duration interval is the duration used to determine whether the constraint is satisfied.
	Time Constraint	Refers to a time interval. The time interval is the time expression used to determine whether the constraint is satisfied.
	Destruction Occurrence	Represents the destruction of the instance described by the lifeline.

Source: created by the author [91][101A][102A][103A][104A].

Input (elements from the Enterprise Model) and output (elements generated to UML Timing model) elements are presented in Table 9 [91][101A][102A][103A][104A].

The UML Timing model is also one of the interaction model groups and defines interactions when the model's primary purpose is to reason about time. Timing models concentrate on conditions changing inside and between lifelines ahead of a linear time axis [91][101A][102A][103A][104A][105A].


Select Actor element
Generate Lifeline element
Select Information Flow
yes
Are there more Information Flows
no
Is Information Flow a State or Condition Timeline
yes
Generate State or Condition Timeline
Link Lifeline to State or Condition Timeline
no
Select Business Rule
yes
Is Business Rule a Time Constraint
Generate Duration Constraint
Link Lifeline to Duration Constraint
no
Update Lifeline element
Are there more Business Rules
yes
Generate Time Constraint
no
Are there more Actors
Generate Destruction Occurence
no

Source: created by the author [91][101A][102A][103A][105A].

Figure 26. UML Timing Model Transformation Algorithm

The initial element is a lifeline in the UML Timing model generation from Enterprise Model (Figure 26). After the generation of this element, the selection of Enterprise Model element information flow follows, and a timeline or duration constraint element is generated. Subsequently, the generation of these two types of elements must be linked to each other. In timing models, time constraint and destruction occurrence elements are generated from the business rule element. Later all these elements are generated, there is an update of a lifeline element, and check whether there are more actor elements left in Enterprise Model [91][101A][102A][103A][106A].

2.8. UML Interaction Overview Diagram Transformation Algorithm

The UML Interaction Overview diagram identifies interactions through a variant of activity models in a way that sustains an overview of the control flow. The interaction Overview model focuses on the overview of the flow of control where the nodes are interactions or interaction uses, and the lifelines and the messages do not fulfil this overview level. UML Interaction Overview model coordinates elements from activity and interaction models [91][101A][102A][103A][104A]:

- from the activity model: initial node, flow final node, activity final node, decision node, merge node, fork node, join node;
- from the interaction models: interaction, interaction use, duration constraint, time constraint.

Table 10. UML Interaction Overview Model Element

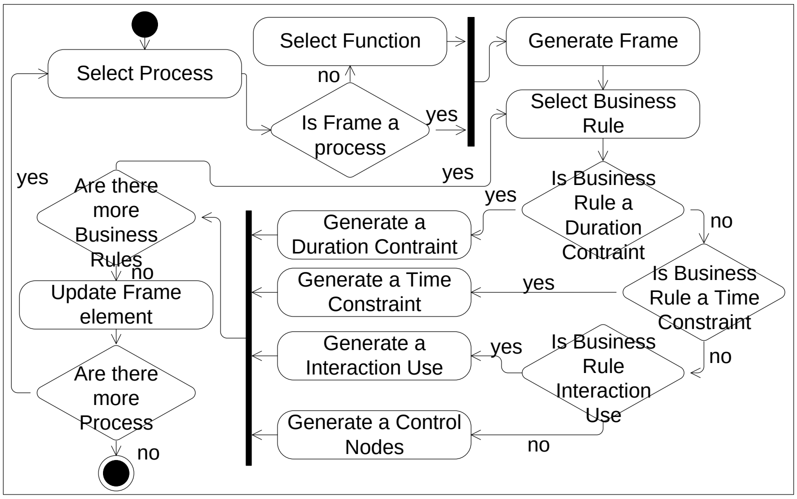
EM element	**UML Interaction Overview model element**	**Description**
Process, Function	Frame	Represents a unit of behaviour that focuses on the observable exchange of information between connectable elements.
Business Rules	Duration constraint	Refers to a duration interval. The duration interval is the duration used to determine whether the constraint is satisfied.
	Time Constraint	Refers to a time interval. The time interval is the time expression used to determine whether the constraint is satisfied.
	Interaction Use	Allows to use (or call) another interaction.

	Control Nodes	Used to coordinate the flows between other nodes. It includes: initial, flow final, activity final, decision, merge, fork, join.

Source: created by the author [91][101A][102A][103A][104A]

Input (elements from the Enterprise Model) and output (elements generated to UML Timing model) elements are presented in Table 10 [91][101A][102A][103A][104A].

The UML Interaction Overview model is the last of the interaction models group, which defines interactions through a variant of models in a way that stimulates an overview of the control flow. Interaction overview models focus on the overview of the flow of control where the nodes are interactions or interaction uses. The participants, like lifelines and objects like messages, do not appear at this overview level [91][101A][102A][103A][104A].


Select Function
Generate Frame
Select Process
no
Is Frame a process
yes
Select Business Rule
yes
yes
Are there more Business Rules
Is Business Rule a Duration Contraint
yes
no
Generate a Duration Contraint
no
Update Frame element
Generate a Time Constraint
yes
Is Business Rule a Time Constraint
Generate a Interaction Use
yes
Is Business Rule Interaction Use
no
Are there more Process
Generate a Control Nodes
no
no

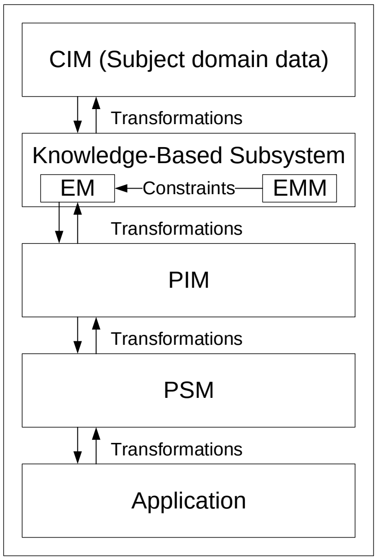
Source: created by the author [101A][102A][103A][104A]

Figure 27. UML Interaction Overview Model Transformation Algorithm

In the UML Interaction model generation from Enterprise Model (Figure 27), the initial element is a process or function, which means that the frame element is generated from these Enterprise Model elements. All other elements: duration constraint, time constraint, interaction use and control nodes, are related to the initial frame element and depend on the Enterprise Model business rule element. Later all these elements are generated, there is an update of a frame element and check if there are more process or function elements left in Enterprise Model [101A][102A][103A][106A].

2.9. MDA Approach Extended with the Knowledge-Based Subsystem

As mentioned above, MDA is an IS development concept based on the use of models to capture the functional and non-functional requirements of information systems, as well as for IS design and software code generation. Within the scope of the study, the focus is on the development of IS design models [4][6]. The MDA only defines general principles for using models but does not provide detailed specifications [69]. Thus, most MDA-based IS engineering approaches need to improve the construction of the CIM layer, and the verification of the system models concerning the subject domain and a knowledge-based subsystem would resolve this issue.



Source: created by the author according to [86]

Figure 28. MDA Approach Extended with the Knowledge-Based Subsystem

It can be done by adding to the MDA concept the principles of knowledge-based IS engineering, represented by an Enterprise model (based on an Enterprise meta-model) and transformation algorithms into UML dynamic models (Figure 28). Model transformation algorithms are used to perform transformations between MDA models (CIM, PIM) and the Enterprise model. The Enterprise model is built between the CIM and PIM models and is responsible for verifying the user requirements and business processes of the information system under development. The main advantage of the knowledge-based MDA approach is that the requirements gathered in the

MDA CIM model can be checked in terms of the rules of the Enterprise meta-model; thus, the other MDA models (PIM and PSM) are developed with less influence of empirical factors. The knowledge-based MDA approach incorporates the principles of traditional MDA concept models in IS engineering. It extends this methodology by introducing a new element, the Enterprise knowledge-based model, used to validate the empirically collected subject domain data.

2.10. Second Part Conclusions

The second part defines the transformation method from Enterprise to UML models. Seven UML dynamic model transformation algorithms are presented with EM and UML element mapping descriptions. After the UML dynamic model transformation algorithms are presented, the extended knowledge-based subsystem MDA approach is defined. It helps to understand how a knowledge-based subsystem can improve the generation process and how it may be useful in avoiding negative empirical factors in IS development process.

3. METHOD REPRESENTATION WITH THE EXAMPLES

The third section presents two different and combined UML dynamic model generation examples from different subject domains using transformation algorithms. This section verifies and proves that the Enterprise Model is suitable for UML dynamic model generation and can be used through created transformation algorithms.

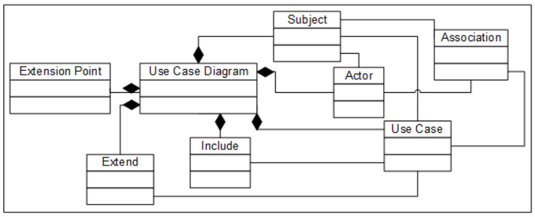
As it was mentioned earlier, information system design methods specify the sequence of systems engineering actions and which UML models to use in the design process; to make the process of implementation easier [44][58][59][61][63A]. According to the previous section, it can be declared that interaction between Enterprise Model and UML models is realized through the transformation algorithms (Figure 17) [64A][99A][100A].

Enterprise Model as an organisation's knowledge repository allows generating UML models after using the transformation algorithms. Enterprise Meta-Model contains essential elements of business modelling methodologies and techniques, ensuring suitable UML model generation [63A][64A].

The ISO 12207 purpose of the Software requirements analysis process is to establish the requirements of the software elements of the system. Analysis process tasks of software requirements are solved using knowledge storing into Enterprise Model and generating UML models from its match with the outcomes and quality characteristics of the ISO 29148 [37][38][84].

3.1. UML Use Case Model Generation

UML Use Case model, at its simplest, is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved (Figure 27). The UML Use Case model can identify the different types of users of a system and the different use cases and will often be accompanied by other types of models as well. Commonly, it is used to define possible requirements.



Source: created by the author [102A]

Figure 29. UML Use Case Metamodel

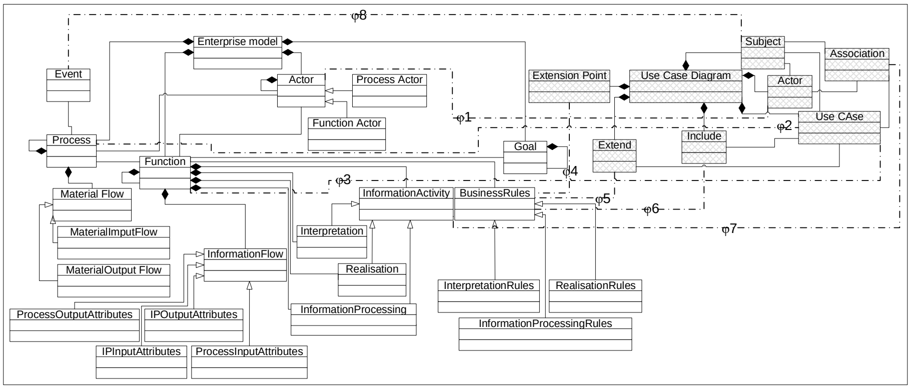
Mainly, the UML Uses Case model is created by a system analyst, which analyses the domain field and functional and non-functional user requirements for the system.

The transformation algorithm presented in Figure 18 is an algorithm for Enterprise Meta-Model based UML Use Case model generation process. The main steps for generating process are: identifying and selecting UML use case model for generating process, identifying starting element, selecting all related elements, mapping Enterprise Model elements to UML model elements and generating selected UML use case model elements (Table 11, Figure 30 (Figure 6 and Figure 29 elements)).

Table 11. Generated UML Use Case Elements

EM elements	Formal description	Generated UML Use Case elements
Actor	φ1	Actor
Process, Function	φ2 φ3	Use Case
Business Rules	φ4 φ5 φ6	Extension Point, Exclude, Include
Information Activity	φ7	Association
Event	φ8	Subject

Source: created by the author [102A]



Source: created by the author [102A]

Figure 30. UML Use Case Elements Generated from EM

The quality of requirements phase specifications is significant for the success of the information systems development process, and mistakes in this phase can cause huge problems and cost a lot of time and expenses to fix. International standards provide qualified guidance for the information system development process. However, international standards do not detail specific methodologies or methods that should be used.

There are many standards and business modelling methodologies, and UML is one of the most common software specification standards. Enterprise

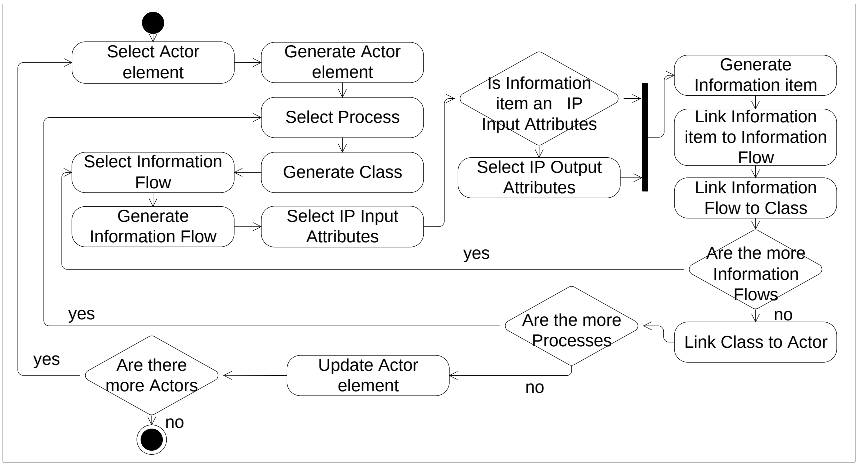
Model includes business management information process essential properties. Enterprise Meta-Model specification and Enterprise Models with particular business data ensure quality and verified knowledge in specific situations. Each element of the UML model can be generated from the Enterprise Model using a knowledge-based Enterprise Model and transformation algorithms. The method of the UML model generation process from the Enterprise Model can implement the whole knowledge-based IS development cycle design phase. This is partially proved by the example with the UML use case model's generated elements.

According to the combined usage of ISO Standards and UML models that can be generated from a knowledge-based Enterprise Model, the information systems development process becomes consistent, more efficient.

3.2. UML Information Flow Model Generation

The UML Information Flow Model belongs to the dynamic UML models part and shows the exchange of information among system entities at some high levels of abstraction, and it is directly related to UML Class and Use Case models. This model describes information flows and provides information to Class and Use Case models [99A][101A][102A][104A].

Information flows can be useful in describing the circulation of information through a system. These flows represent aspects of models not yet wholly specified or with fewer details.


Select Actor element
Generate Actor element
Select Process
Generate Class
Select Information Flow
Generate Information Flow
Select IP Input Attributes
Is Information item an IP Input Attributes
Select IP Output Attributes
Generate Information item
Link Information item to Information Flow
Link Information Flow to Class
Are the more Information Flows
yes
no
Link Class to Actor
Are the more Processes
yes
no
Update Actor element
Are there more Actors
yes
no

Source: created by the author [99A]

Figure 31. Transformation Algorithm of UML Information Flow Model Generation from EM Process

The transformation algorithm of UML information Flow model generation from the Enterprise Model process is presented in Figure 31 and is illustrated by the following steps [99A]:

- Step 1: According to the top-level transformation algorithm of UML model generation from the EM process, the UML Information Flow model is identified for the generation process. So the initial element for the UML Information Flow model is the Actor element.
- Step 2: UML Information Flow model Actor element is generated from the Enterprise Model.
- Step 3: The process element from Enterprise Model related to the initial actor element is selected.
- Step 4: UML Information Flow model Class element is generated from the Enterprise Model.
- Step 5: The information Flow element as a link to other elements from the Enterprise Model related to the process element is selected.
- Step 6: UML Information Flow model Information Flow element as a link of other elements is generated from the Enterprise Model.
- Step 7: Information processing Input Attributes element as a definition of link element from Enterprise Model related to the process element is selected.
- Step 8: If the UML Information Flow model Information item element is a definition of a link to the next element, it is generated from the Enterprise Model.
- Step 9: Else, the Information processing Output Attributes element as a definition of a link to the previous element from the Enterprise Model is selected.
- Step 10: UML Information Flow model Information item element as a definition of a link to the previous element is generated from the Enterprise Model.
- Step 11: UML Information flow elements Information item and Information Flow are linked.
- Step 12: UML Information flow elements Information Flow and Class are linked.

- Step 13: There is checking if there are more Information flows in the Enterprise Model related to the UML Information Flow model. In case, there are, the algorithm goes back to step 5.
- Step 14: UML Information flow elements Class and Actor are linked.
- Step 15: There is checking if there are more Processes in the Enterprise Model related to the UML Information Flow model. In case, there are, the algorithm goes back to Step 3.
- Step 16: UML Information flow element Actor is updated.
- Step 17: There is checking if more Actors in Enterprise Model are related to the UML Information Flow model. In case, there are, the algorithm goes back to step 1.
- Step 18: Else, all UML Information Flow model elements and links are generated from the Enterprise Model.

The generation of the UML Information Model is illustrated with the example of the Scheduled workflow for Ultrasound examination for the pet in a Veterinary clinic [99A]. The subject domain information of this example is stored in the Enterprise Model. The example shows how the pet owner registers his pet in the veterinary clinic for a veterinary appointment in order to get an ultrasound examination, surgeon evaluation and veterinary consultation. Firstly, the pet owner registers his pet in the Veterinary clinic registration system, orders the ultrasound examination in the ultrasound information system, then follows the process of the examination, data storage and examination data sending to the surgeon. The surgeon analyses examination data, writes the diagnosis using the reviewing and evaluating system and sends it to the veterinary through the reviewing and evaluating system, which gives the result to the pet owner [99A].

Detailed stages of Veterinary clinic example processes stored in the Enterprise Model are described [99A]:

- Stage 1 – The pet owner registers his pet in the veterinary clinic registration system. The information system manages pet owner registration and service ordering and is responsible for updating information.
- Stage 2 – Pet registration information from the veterinary clinic registration system is connected to the ultrasound

examination registration system, and the system manages examination order scheduling.

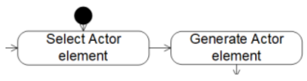
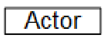
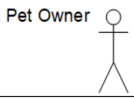
- Stage 3 – Data gaining system acquires and creates medical data while a pet is present (in the example: ultrasound, tomography and so on).
- Stage 4 – Data storage system manages examination data storage and sharing inside the Veterinary clinic.
- Stage 5 – The surgeon gets data from the data storage system, evaluates it by reviewing and evaluating the system and prepares a diagnosis response.
- Stage 6 – Veterinary gets the diagnosis response prepared by the surgeon by reviewing and evaluating the system.
- Stage 7 – The pet owner gets diagnosis information during the appointment with the veterinarian.

Transformation algorithm of UML Information Flow model generation of Stage 1 of Scheduled workflow for Ultrasound examination for the pet in Veterinary clinic example from the Enterprise Model process is illustrated by the following steps [99A]:

- Step 1: Selected initial element for the UML Information Flow model is the Actor element.
- Step 2: The Actor element of the UML Information Flow model is generated from the Enterprise Model; in a particular example, the first actor is the Pet owner.

The first two steps of the transformation algorithm are presented in Table 12.

Table 12. Steps 1, 2 in the UML Information Flow Model Generation Process

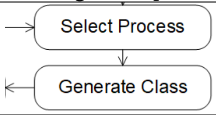
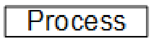
Transformation algorithm part	**Enterprise Model element**	**Generated UML Information Flow model element**
		

Source: created by the author [99A]

- Step 3: The process element from Enterprise Model, related to the initial actor element, is selected.
- Step 4: The Class element of the UML Information Flow model is generated from the Enterprise Model; in a particular example, the first class is Pet registration.

The other two steps of the transformation algorithm are presented in Table 13.

Table 13. Steps 3, 4 in the UML Information Flow model generation process

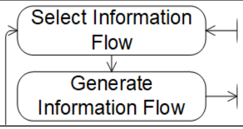
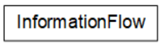
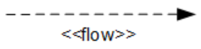
Transformation algorithm part	Enterprise Model element	Generated UML Information Flow model element
		Pet registration

Source: created by the author [99A]

- Step 5: The Information Flow element as a link to other elements from the Enterprise Model related to the process element is selected.
- Step 6: The Information Flow element of the UML Information Flow model, as a link of other elements, is generated from the Enterprise Model; in a particular example, the first Information flow is between the Pet owner and Pet registration.

The other two steps of the transformation algorithm are presented in Table 14.

Table 14. Steps 5, 6 in the UML Information Flow Model Generation Process

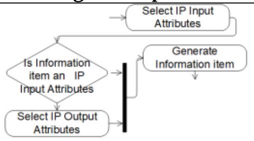
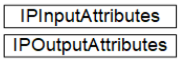
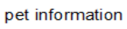
Transformation algorithm part	Enterprise Model element	Generated UML Information Flow model element
		

Source: created by the author [99A]

- Step 7: Information processing Input Attributes element as a definition of link element from Enterprise Model related to the process element is selected.
- Step 8: If the UML Information Flow model Information item element is a definition of a link to the next element, then it is generated from Enterprise Model; in a particular example, the first Information item is Pet information.
- Step 9: Else, the Information processing Output Attributes element as a definition of a link to the previous element from the Enterprise Model is selected.

The following two (in another case – three) steps of the transformation algorithm are presented in Table 15.

Table 15. Steps 7, 8, 9 in UML Information Flow Model Generation Process

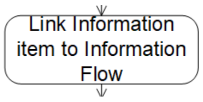
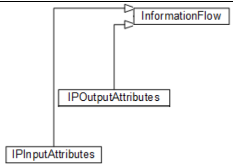
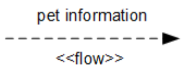
Transformation algorithm part	Enterprise Model element	Generated UML Information Flow model element
 Select IP Input Attributes Generate Information item Is Information item an IP Input Attributes Select IP Output Attributes		

Source: created by the author [99A]

- Step 10: The UML Information Flow model Information item element as a definition of a link to the previous element is generated from the Enterprise Model.
- Step 11: UML Information flow elements – Information item and Information Flow – are linked.

The following two steps of the transformation algorithm are presented in Table 16.

Table 16. Steps 10, 11 in UML Information Flow Model Generation Process

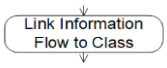
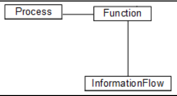
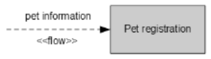
Transformation algorithm part	Enterprise Model element	Generated UML Information Flow model element
		

Source: created by the author [99A]

- Step 12: The UML Information flow elements – Information Flow and Class – are linked.

Step 12 of the transformation algorithm is presented in Table 17.

Table 17. Step 12 in UML Information Flow Model Generation Process

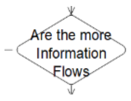
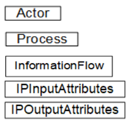
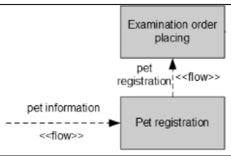
Transformation algorithm part	Enterprise Model element	Generated UML Information Flow model element
		

Source: created by the author [99A]

- Step 13: There is checking if there are more Information flows in the Enterprise Model related to the UML Information Flow model. In case, there are, the algorithm goes back to step 5, and all steps from the 5 are repeated.

Step 13 of the transformation algorithm is presented in Table 18, showing the result after repetition steps from step 5.

Table 18 Step 13 in UML Information Flow Model Generation Process

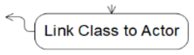
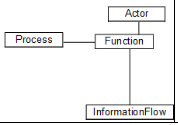
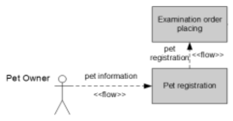
Transformation algorithm part	Enterprise Model element	Generated UML Information Flow model element
	 Actor Process InformationFlow IPInputAttributes IPOutputAttributes	 Examination order placing pet registration <<flow>> pet information <<flow>> Pet registration

Source: created by the author [99A]

- Step 14: The Class and Actor elements of UML Information flow are linked; in this particular example, the Pet owner is linked to pet registration.

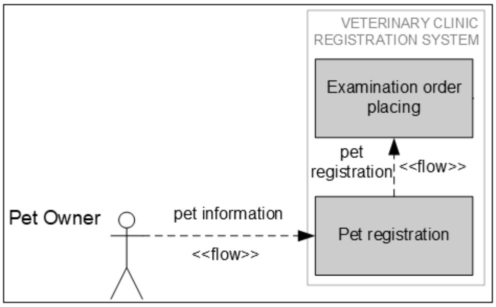
Step 14 of the transformation algorithm is presented in Table 19, showing the result after repetition steps from step 5.

Table 19. Step 14 in the UML Information Flow model generation process

Transformation algorithm part	Enterprise Model element	Generated UML Information Flow model element
		 Examination order placing pet registration <<flow>> Pet Owner pet information <<flow>> Pet registration

Source: created by the author [99A]

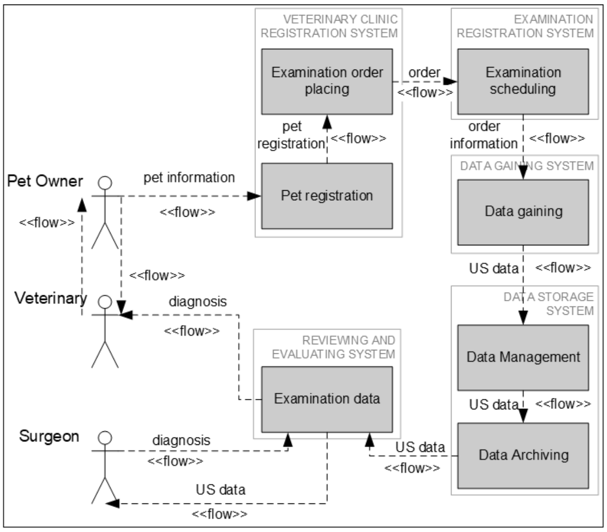
After 14 steps of the transformation algorithm generating of Scheduled workflow for Ultrasound examination for the pet in Veterinary clinic data from Enterprise Model, Stage 1 – pet owner registers his pet in the veterinary clinic registration system. The information system managing pet owners' registration and services order is responsible for updating information, and it is shown in Figure 32.



Source: created by the author [99A]

Figure 32. 1 Stage of the Scheduled Workflow for Ultrasound Examination for the Pet in the Veterinary Clinic Example Presented as the UML Information Model Generated from the Enterprise Model

The full UML Information flow model after all steps of the transformation algorithm generating Scheduled workflow for Ultrasound examination for the pet in the Veterinary clinic example is shown in Figure 33.



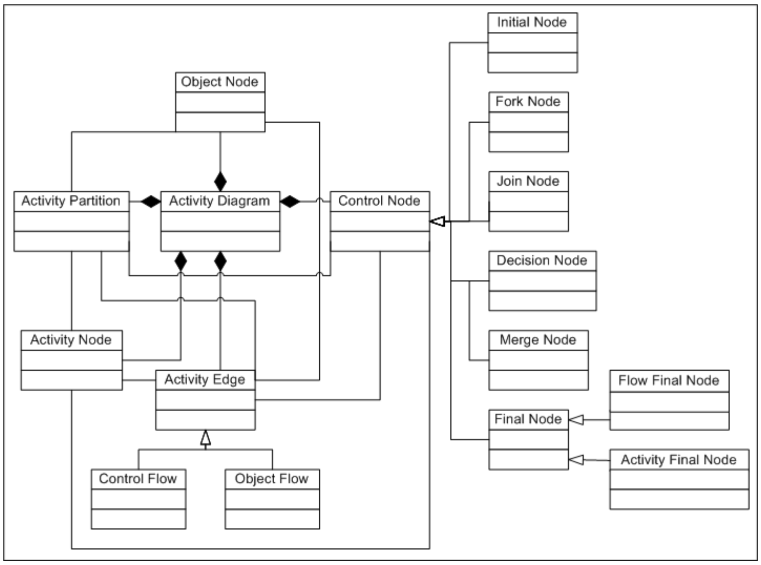
Source: created by the author [99A]

Figure 33. Complete UML Information Model Generated from Enterprise Model of Scheduled Workflow for Ultrasound Examination for the Pet in Veterinary Clinic

After the implementation of all the steps of the transformation algorithm, it can be undoubtedly declared that the chosen example perfectly illustrates the accuracy of the UML Information flow elements generated from the Enterprise Model.

3.3. UML Activity Model Generation

The Activity model is described in the previous section, and the UML Activity diagram meta-model is presented in Figure 34. UML Activity diagram is essentially an advanced version of a flow chart that models the flow from one activity to another activity [64A][91][94A][102A][103A][107A].



Source: created by the author according to [94A]

Figure 34. UML Activity Diagram Metamodel

The following elements are typically drawn on UML Activity diagrams [75][91][94A]:

- Activity node – a node can be the execution of a subordinate behaviour, such as arithmetic computation, a call to an operation, or manipulation of object contents. It also includes the flow of control constructs, such as synchronization, decision, and concurrency control. Activities may form invocation hierarchies invoking other activities, ultimately resolving to individual actions. In an object-oriented model, activities are usually invoked indirectly as methods bound to directly invoked operations.
- Activity partition – an activity group for actions that have some common characteristics. Partitions often correspond to organisational units or business actors in a business model.
- Object node – indicates an instance of a particular classifier, possibly in a particular state and may be available at a particular point in the activity. Object nodes can be used in various ways, depending on where the objects are flowing from and to, as described in the semantics sub-clause.
- Control node – coordinating the flows between other nodes.

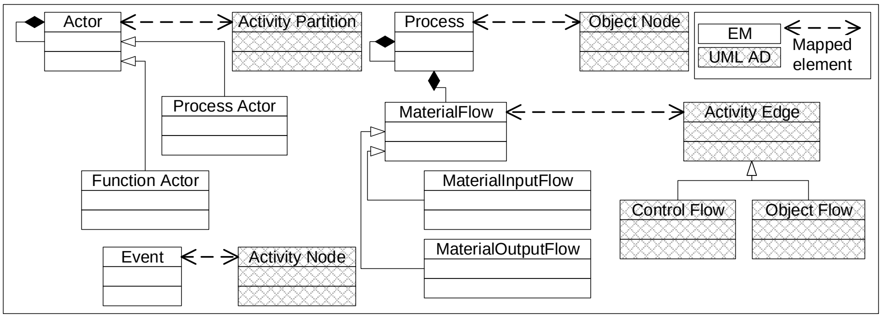
- Activity edge – an abstract class for the directed connections along which tokens or data objects flow between activity nodes. It includes control edges and object flow edges. The source and target of the edge must be in the same activity as the edge.

Table 20. Generated EM Elements to UML Activity Model Elements

UML Activity Model / EM					Control Nodes							Activity Edge		
										Final Node				
			Activity Node	Activity Partition	Initial Node	Fork Node	Join Node	Decision Node	Merge Node	Flow Final Node	Activity Final Node	Control Flow	Object Flow	Object Node
Actor		Process Actor		+										
		Function Actor		+										
Event			+											
Process		Material Input Flow											+	+
		Material Output Flow											+	+
Function	Business Rules	Interpretation Rule			+	+	+	+	+	+	+			
		Realization Rule				+	+	+	+	+	+			
		Information Processing Rule				+	+	+	+	+	+			
	Information Flow	Process Input Attributes										+		
		Process Output Attributes										+		
		IP Input Attributes										+		
		IP Output Attributes										+		
	Information Activity	Interpretation										+		
		Realization										+		
		Information Processing										+		

Source: created by the author [64A][91][94A][102A][103A]

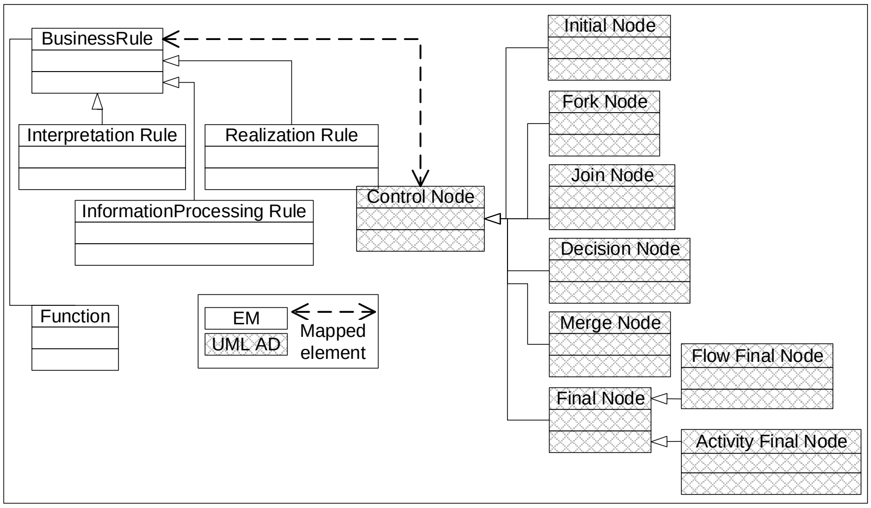
Table 20 presents the generated EM elements to UML Activity model elements. The Activity partition element in the UML Activity model maps the actor element in EM as well as the activity node maps the event, the object node and object flow map the process, control nodes map business rules [94A].



Source: created by the author [94A]

Figure 35. Mapping of EM Elements to UML Activity Model Elements (1)

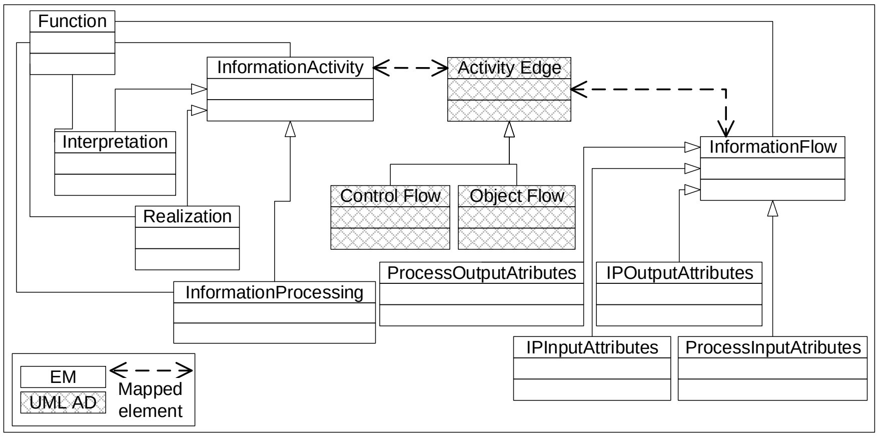
Figure 35 presents part of the generated EM elements to UML Activity model elements. EM actor element is mapped as an activity partition element of the UML Activity model. EM event element is mapped as an activity node element of the UML Activity model. EM process element is mapped as an object node element of the UML Activity model. And EM material flow element is mapped as an activity edge element of the UML Activity model [94A].



Source: created by the author [94A]

Figure 36. Mapping of EM elements to UML Activity model elements (2)

Figure 36 presents the second part of the generated EM elements to UML Activity model elements. EM business rule element is mapped as the control node element of the UML Activity model.



Source: created by the author [94A]

Figure 37. Mapping of EM Elements to UML Activity Diagram Elements (3)

Figure 37 presents the third part of the generated EM elements to UML Activity model elements, where EM information Activity element and information flow are mapped as activity edge element of the UML Activity model [94A].

The transformation algorithm presented in Figure 19 is an algorithm for Enterprise Meta–Model-based UML Activity model generating process. The main steps for generating processes are identifying starting elements, selecting all related elements, mapping Enterprise Model elements to UML model elements and generating selected UML Activity model elements [4].

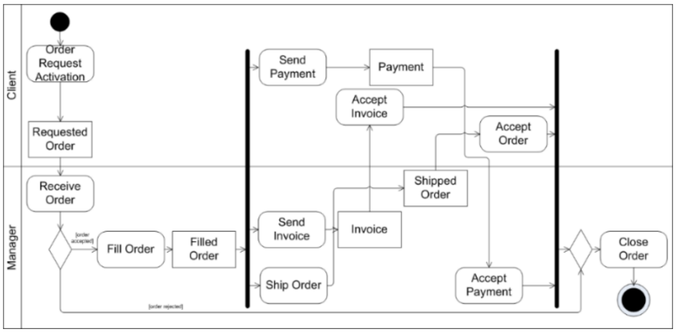
Table 21. An Example of EM and UML Activity Model Element Mapping

EM \ UML Activity Model Example		Manager	Client	Initial Node	Order Request Activation	Requested Order	Receive Order	Decision Node	Fill Order	Filled Order	Fork Node	Send Invoice	Send Payment	Payment	Invoice	Ship Order	Shipped Order	Accept Payment	Accept Invoice	Accept Order	Join Node	Merge Node	Close Order	Activity Final Node
Actor		+	+																					
Event					+		+		+			+	+			+		+	+	+			+	
Process	Material Input											+	+	+	+	+	+							

	Material Output													+	+		+	+	+	+				
Function	Business Rules			+				+			+										+	+		+
	Information Flow					+				+														
	Information Activity				+		+		+														+	

Source: created by the author [6A][94A]

Table 21 presents the example in which Enterprise Model elements are mapped as UML Activity model. Business activity of order request and its closure is used as an example for the UML Activity model generation process. Order requested activation is starting elements from the client side, and the requested order is the input parameter of the activity. After the order is accepted and all the required information is filled in, the payment is accepted, and the order is shipped. Note that this business flow allows order shipment before the invoice is sent or the payment is confirmed.



Source: created by the author [91][94A][107A]

Figure 38. An Example of Generated UML Activity Model

Figure 38 presents an example of a generated UML Activity model from the Enterprise Model. The necessary elements were received from the CASE tool’s knowledge-based subsystem’s Enterprise Model, where all subject domain knowledge is stored [107A].

3.4. UML Sequence Model Generation

UML Sequence model is the most common type of interaction model, which focuses on the message interchange between actors, objects (lifelines). The Sequence model shows how the objects interact with others in a particular scenario of a use case [91][96A][101A][102A][103A][104A].

There is a given formalized UML Sequence model description. UML Sequence model also can be described as Malcev algebra-based algebra system [96A]:

M2=<K, R>; (3)

where M2 – UML Sequence model as algebra system;

K – elements set of M2 system

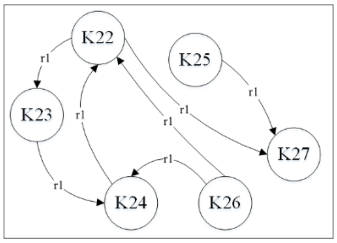
K={K22, K23,…, K27}, where K22,....K27 UML Sequence model meta-classes

R – set of relationships between elements, where R={r1, r2, r3}.

UML Sequence Model M2 composition is as follows:

M2=<{K22, K23,...,K27}, {r1}>; (4)

where: K22– meta-class Message, K23 – meta-class Execution, K24 – meta-class Lifeline, K25 – meta-class State Invariant, K26 – meta-class Combined Fragment, K27 – meta-class Destruction Occurrence, r1 – Aggregation. UML Sequence Model graphical schema is presented in Figure 39 [96A]:



Source: created by the author [96A]

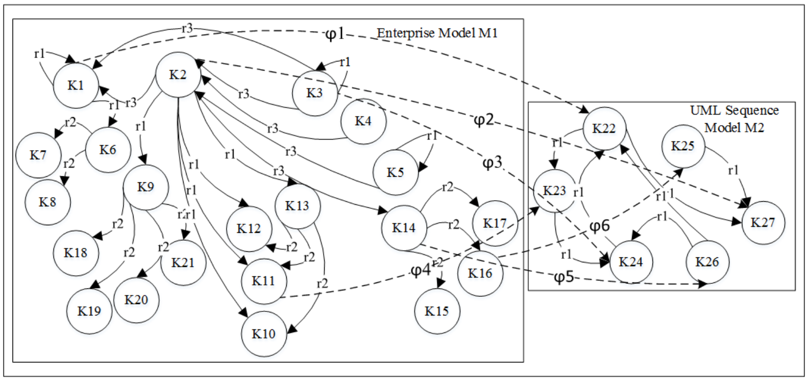
Figure 39. Graphical Schema of UML Sequence Model Based on Malcev Algebra

The intersection between Enterprise Model and UML Sequence model elements is presented in Table 22, and the graphical scheme is in Figure 40.

Table 22. The Intersection Between EM and UML Sequence Model Elements

Enterprise Model set element	**UML Sequence Model set element**	**Formal description**
Process (K1)	Message (K22)	φ1: K1→K22
Function (K2)	Destruction Occurrence (K27)	φ2: K2→K27
Actor (K3)	Lifeline (K24)	φ3: K3→K24
Data Processing and Solution Making (K11)	Execution (K23)	φ4: K11→K23
Business Rules (K14)	Combined Fragment (K26)	φ5: K14→K26
Data Processing and Solution-Making Business Rules (K16)	State Invariant (K25)	φ6: K16→K25

Source: created by the author [96A]



Source: created by the author [96A]

Figure 40. Graphical Scheme of Intersection Between Enterprise Model and UML Sequence Model Elements Based on Malcev Algebra

The transformation algorithm presented in Figure 23 is an algorithm for Enterprise Meta–model-based UML Sequence model generating process. The main steps for the generating process are identifying and selecting the UML Sequence model for the generating process, identifying starting elements, selecting elements types, selecting relationships between elements types, selecting all related elements, mapping Enterprise Model elements to UML model elements and generating selected UML sequence model elements [96A].

3.5. UML Timing Model Generation

One of the UML dynamic models is the Timing model. Timing models are UML interaction models used to show interactions when the model's primary purpose is to reason about time. Timing model focus on conditions changing within and among lifelines along a linear time axis. Timing models describe the behaviour of both individual classifiers and interactions of classifiers, focusing attention on the time of events causing changes in the modelled conditions of the lifelines [91][97A][101A][102A][103A][104A].

There is a given formalized UML Timing model description (Figure 41). UML Timing model also can be described as Malcev algebra-based algebra system [97A]:

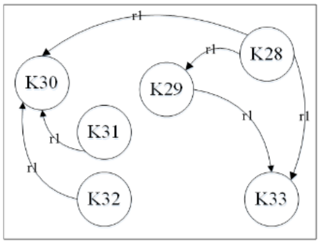
$$M3=<K,R> \qquad (5)$$

where M2 – UML Timing model as algebra system; K – elements set of M2 system; K={K28, K29,…, K33}, where K28,....K32 UML Timing model meta-classes; R – set of relationships between elements, where R={r1, r2, r3}.

UML Timing model M2 composition is as follows:

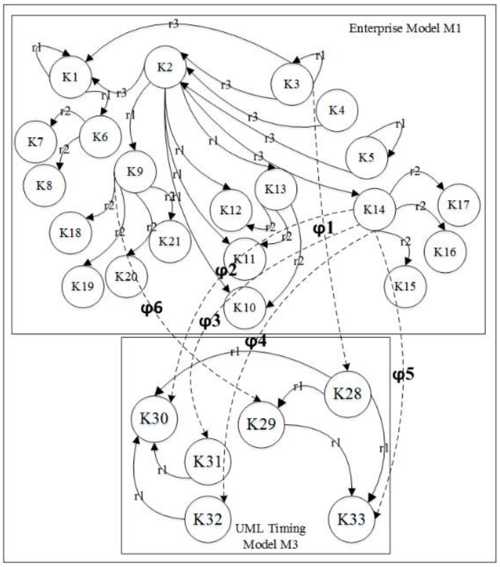
M3={K28, K29,…, K33},{r1}> (6)

where: K28– meta-class Lifeline, K29 – meta-class State or Condition Timeline, K30 – meta-class Interval Constraint, K31 – meta-class Duration Constraint, K32 – meta-class Time Constraint, K33 – meta-class Destruction Occurrence, r1 – Aggregation [97A].



Source: created by the author [97A]

Figure 41. Graphical Schema of UML Timing Model Based on Malcev Algebra



Source: created by the author [97A]

Figure 42. Graphical scheme of intersection between Enterprise Model and UML Timing Model elements based on Malcev algebra

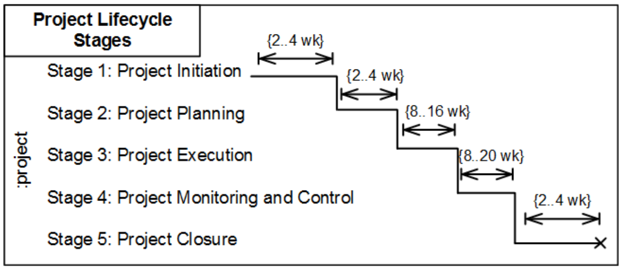
According to Figure 42, it is clear that Enterprise Model elements: Actor, Function and Business rules can be generated as UML Timing model elements: Lifeline, Destruction Occurrence, Intervals Constrains, Durations Constraint, Time Constraint and State or Condition Timeline [97A].

Table 23. Intersection Between Enterprise Model and UML Timing Model Elements

Enterprise Model set element	UML Timing model set element	Formal description
Actor (K3)	Lifeline (K28)	φ1: K3→K28
Business Rules (K14)	Interval Constraint (K30)	φ2: K14→K30
Business Rules (K14)	Duration Constraint (K31)	φ3: K14→K31
Business Rules (K14)	Time Constraint (K32)	φ4: K14→K32
Business Rules (K14)	Destruction Occurrence (K33)	φ5: K14→K33
Information Flow (K9)	State or Condition Timeline (K29)	φ6: K9→K29

Source: created by the author [97A]

Table 23 presents the intersection between Enterprise Model and UML Timing model elements, where a formal description of Enterprise Model elements generated to UML Timing model elements according to Malcev algebra can be found.


Project Lifecycle Stages
:project
Stage 1: Project Initiation
Stage 2: Project Planning
Stage 3: Project Execution
Stage 4: Project Monitoring and Control
Stage 5: Project Closure
{2..4 wk}
{2..4 wk}
{8..16 wk}
{8..20 wk}
{2..4 wk}

Source: created by the author [97A]

Figure 43. An Example of UML Timing Model, Project Life cycle Phases

In the example (Figure 43) of the UML Timing model, which presents project life cycle phases and their duration, certain elements generated from the Enterprise Model can be found: lifeline – project, state or condition – project phases, timelines, duration and timing constraints and destruction occurrence, where destruction event is depicted by a cross in the form of an X at the end of a timeline.

3.6. UML Interaction Overview Model Generation

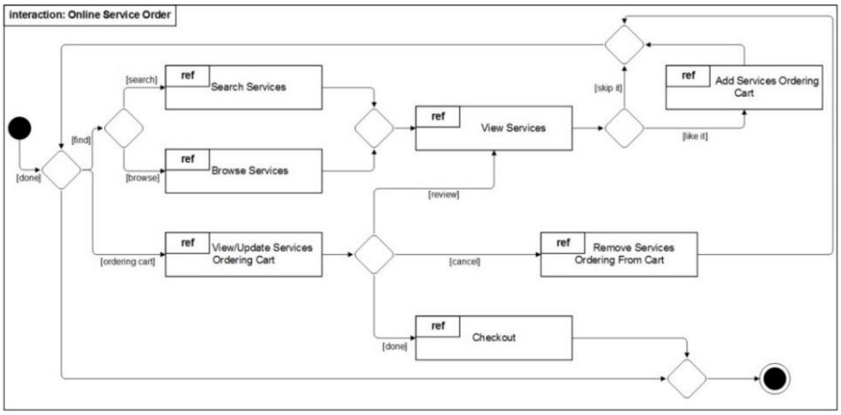
UML Interaction Overview model determines interactions through a variant of activity models in a manner that maintains an overview of the control flow. The interaction Overview model concentrates on the overview of the flow of control where the nodes are interactions or interaction uses. The lifelines and the messages do not perform at this overview level. As it is mentioned earlier, the UML Interaction Overview model combines elements from activity and interaction models [65A][91][101A][102A][103A][104A]. UML Interaction Overview model generation from Enterprise Model transformation algorithm presented in Figure 25.

Table 24 presents UML Interaction Overview model elements generated from the Enterprise Model. Frame as Interaction model element is generated from EM Actor element. Interaction Use as Interaction model element is generated from EM Information Activity, Initial Node, Decision Node, Merge Node, Final Node as Activity model elements are generated from EM Business Rules elements and Decision Guard as Activity model element is generated from EM Information Flow element [65A].

Table 24. EM and Online Service Ordering UML Interaction Overview Model Elements

EM \ **UML Interaction Overview model**		Frame (Interaction model element)	Interaction Use (Interaction model element)	Initial Node (Activity model element)	Decision Node (Activity model element)	Merge Node (Activity model element)	Final Node (Activity model element)	Decision Guard (Activity model element)
Actor		+						
Event								
Process	Material Input							
	Material Output							
Function	Business Rules			+	+	+	+	
	Information Flow							+
	Information Activity		+					

Source: created by the author [65A]



Source: created by the author [65A]

Figure 44. An Example of UML Interaction Overview Model: Online Service Order

Figure 44 presents an example of the UML Interaction Overview model. The necessary elements through transformation algorithms were received from the CASE tool's knowledge-based subsystem's Enterprise Model, where all knowledge of the subject domain is stored. In this figure, all necessary UML Interaction Overview model elements generated from the Enterprise Model are clearly seen [65A].

3.7. Generated UML Models of Ticket Buying Process Example

This section deals with a detailed explanation of the Ticket buying process and how this process can be designed by using a knowledge-based Enterprise Model, where all subject domain knowledge related to the described example is stored. It is also explained what knowledge is used for the generation of particular UML models through specific transformation algorithms created for each UML model generation process [101A][103A][104A][108A]. The UML Use Case, Sequence, State and Activity models generated from Enterprise Model are described.

The process of Ticket buying may seem very simple, but if this process were analyzed from different perspectives in the information systems design phase, this process would be projected and designed for the fulfilment of all possible functions. It would take a lot of time and effort from an analyst, designer and so on [108A].

In IS development life cycle design phase, all the details must be estimated. These details, this knowledge, is stored in the previously described Enterprise Model, and they are already verified and validated [108A].

3.7.1. UML Use Case Model of Ticket Buying Process Example

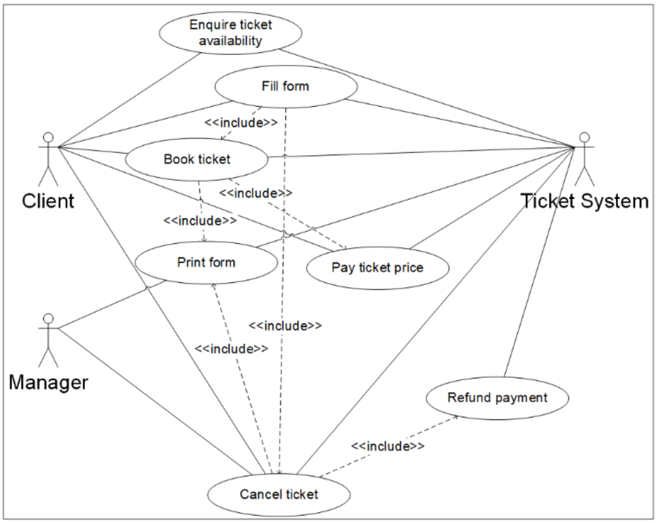
The UML Use Case model is the primary form of system requirements for a new IS underdeveloped. Use cases specify the expected behaviour – what, and not the precise method of making it take place – how? A key concept of use case modelling is that it assists in designing a system from the end user's perspective [108A].

Table 25. UML Use Case Model Elements Generated from Enterprise Model of Ticket Buying Example

Enterprise Model element	**UML Use Case Model element**	**Ticket Buying Example**	**Description**
Actor	Actor	Client	There are three actors, each of them is a behavioural classifier which defines a role played in a particular example.
		Manager	
		Ticket system	
Process, Function	Use Case	Enquire ticket availability	There are three use cases; each use case is a type of behavioural classifier that describes a unit of functionality performed by three actors.
		Fill form	
		Book ticket	
		Pay ticket price	
		Print form	
		Refund payment	
		Cancel ticket	
Business Rule	Include	Six include elements	There are six elements included; each is a directed relationship between two use cases used to demonstrate that the included use case's behaviour is inserted into the included use case.

Source: created by the author [108A]

Table 25 presents UML Use Case model elements generated from the Enterprise Model of Ticket buying example. In the Enterprise Model, all information related to actors, their functions and relationships between these functions is stored. There are three actors: Client, Manager and Ticket System. Ticket System as an actor is associated with all seven functions – use cases: Enquire ticket availability, Fill the form, which includes use case of ticket booking or ticket cancelling, ticket booking includes ticket price payment and form printing, this includes ticket cancelling and this includes payment refunding. Client as an actor is associated with all functions except payment refunding because it is the Ticket system's function. Manager as an actor is associated only with two functions – use cases: form printing and ticket cancelling [108A].



Source: created by the author [108A]

Figure 45. UML Use Case Model of Ticket Buying Example

Figure 45 presents the UML Use Case model of Ticket buying example generated step by step from the Enterprise Model through the UML Use Case transformation algorithm [108A].

3.7.2. UML Sequence Model of Ticket Buying Process Example

UML Sequence model is an interaction model that details how operations are implemented. This model captures the interaction between objects in the context of a collaboration. [108A].

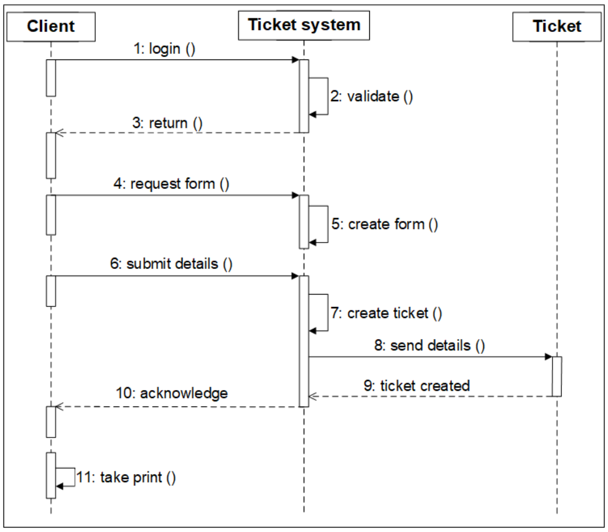
Table 26. UML Sequence Model Elements Generated from Enterprise Model of Ticket Buying Example

Enterprise Model element	**UML Sequence Model element**	**Ticket Buying Example**	**Description**
Actor	Lifeline	Client	There are three actors in the UML Sequence model three Lifelines, which are shown using a symbol that consists of a rectangle forming its “head” followed by a vertical line, and these lines represent the lifetime of the actor–participant of the process.
		Ticket system	

		Ticket	
Process, Function	Message	Login ()	Eleven messages are related to actors and define communication between them.
		Validate ()	
		Return ()	
		Request form ()	
		Create form ()	
		Submit details ()	
		Create ticket ()	
		Send details ()	
		Ticket created	
		Acknowledge	
		Take print ()	
Business Rules	Execution Specification	Ten Execution Specifications	Each of the ten execution specification elements represents a period in the actor's lifetime.

Source: created by the author [108A]

Table 26 presents UML Sequence model elements generated from the Enterprise Model of Ticket buying example. All information related to actors and their collaboration is stored in the Enterprise Model. There are three actors – process participants, which are called Lifelines in the UML Sequence model: person – Client, subject – Ticket system, object – Ticket. The Ticket has one execution specification, receives one message with details and sends one message of the created ticket; Ticket system has three execution specifications; one is assigned for validation after the Client logs in, it returns the result; the second is assigned for form creation and third for ticket creation; all these are related to messages from Client. The Client logs in, requests a form, submits details, prints a ticket – the client sends four messages and receives two: validate login and acknowledgement of all requests of this particular process [108A].



Source: created by the author [108A]

Figure 46. UML Sequence Model of Ticket Buying Example

Figure 46 presents the UML Sequence model of the Ticket buying example generated step by step from the Enterprise Model through the UML Sequence transformation algorithm [108A].

3.7.3. UML State Model of Ticket Buying Process Example

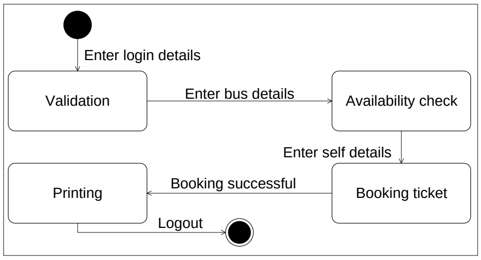
UML State Model shows the different states of an entity. The State model can also demonstrate how an entity responds to various events by changing from one state to another [97A].

Table 27. UML State Model Elements Generated from Enterprise Model of Ticket Buying Example

Enterprise Model element	**UML State Model element**	**Ticket Buying Example**	**Description**
Process, Function	Behavioural State Machine	Enter login details	Five states are used to specify the discrete behaviour of a part of the designed system through finite state transitions.
		Enter bus details	
		Enter self details	
		Booking successful	
		Logout	
Information Flow	Composite State	Validation	Four states of an entity are defined as a state that has substates.
		Availability check	
		Booking Ticket	
		Printing	

Source: created by the author [108A]

Table 27 presents UML State model elements generated from the Enterprise Model of Ticket buying example. In the Enterprise Model, all information related to processes, functions and their state is stored. This model is from the Client's perspective. There are four information flows – composite states of a Client entity: validation, availability check, ticket booking and printing. All these composite states are conducted by a particular behavioural state machine: Enter log in details, Enter bus details, Enter self details, Booking successful, Logout.



Source: created by the author [108A]

Figure 47. UML State Model of Ticket Buying Example

Figure 47 presents the UML State model of the Ticket buying example generated step by step from the Enterprise Model through the UML State transformation algorithm [108A].

3.7.4. UML Activity Model of Ticket Buying Process Example

UML Activity model describes how activities are coordinated to provide a service which can be at different levels of abstraction [108A].

Table 28. UML Activity Model Elements Generated from Enterprise Model of Ticket Buying Example

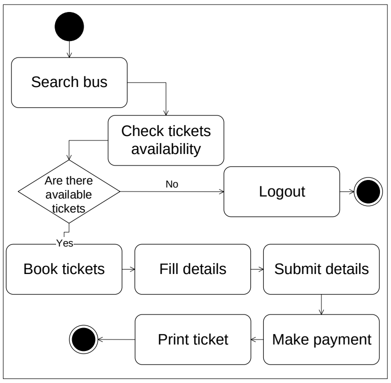
Enterprise Model element	**UML Activity Model element**	**Ticket Buying Example**	**Description**
Actor	Partition	Client	There is one partition, and all activities are directly related to that actor.
Function, Process	Activity	Search bus	There are eight activities directly related to one partition – Client. They represent a parameterized behaviour as a coordinated flow of actions.
		Check tickets availability	
		Book tickets	
		Fill details	
		Submit details	
		Make payment	
		Print ticket	

		Logout	
Business Rules	Control Nodes	Initial node, two activity final nodes, decision node – are there available tickets	There are four control nodes: one node – The initial node in the beginning; one decision node, regarding which process finishes in the success or otherwise; two activity final nodes: one in case of a successful process, another in case of an unsuccessful process. Basically, control nodes are used to coordinate the flows between other nodes.

Source: created by the author [108A]

Table 28 presents UML Activity model elements generated from the Enterprise Model of Ticket buying example. In the Enterprise Model, all information related to actors, their activities and relationships between these functions is stored. There is one business rule – the control node, related to the process beginning, the initial node. In this case, there is only one actor – one partition – the Client. There are two Client activities before the decision node: bus searching and checking ticket availability. In case there are no available tickets, the process finishes unsuccessfully with one of the final activity nodes. In another case, if there are available tickets, the Client books tickets, fills details, submits details, makes payment and prints the ticket. This process finishes as successful with the second final activity node [108A].

Figure 48 presents the UML Activity model of Ticket buying example generated step by step from the Enterprise Model through the UML Activity transformation algorithm [108A].



Source: created by the author [108A]

Figure 48. UML Activity Model of Ticket Buying Example

All four UML dynamic models: Use case, Sequence, State and Activity of one Ticket buying process example, are generated from Enterprise Model according to the subject domain knowledge stored inside. These four UML models define the same example but from diverse perspectives by showing different actors’ activities, states and use cases. A knowledge-based Enterprise Model is sufficient storage of data, which is necessary for UML model generation by specific transformation algorithms of each UML model.

3.8. Generated UML Models of Hospital Information Management Process Example

The Hospital intended to manage outside patients is the object of the presented example. In this institution, a doctor is only associated with one specialized hospital department (cardiology, paediatrics, etc.) at a time. Each doctor has a visiting time and day in a week [109A].

The patient data is entered at the reception, and the necessary fees are also taken. The patient is tracked on the basis of the ID number, which is generated automatically [109A].

Usually, a patient can visit the doctors in two possible ways: directly selecting a doctor or getting admitted to the hospital [109A].

A doctor can prescribe tests based on the patient’s described condition. The patient visits the laboratory to get done the tests prescribed by the doctor. The reports of the tests are given to the patient. The payments related to the tests are made at the reception. According to the reports, the doctor prescribes the patient medicine or further tests if needed or asks to admit the patient to the hospital [109A].

If available, a patient is admitted into a ward of a particular department as per the doctor’s prescription. The number of available wards is limited, and the patient’s admission is rescheduled if there is no free ward [109A].

Also, in case of the doctor’s prescription, the patient is operated on a scheduled date and time as decided by the doctor responsible for the operation [109A].

After finishing the treatment, a patient may get discharged on the advice of his doctor and upon the full payment of all due charges at the reception. On payment of total dues, the receptionist generates a discharge card for the patient [109A].

All data of a particular subject domain, in this case, Hospital Information Management data, is stored in Enterprise Model described previously. The stored information in Enterprise Model is already verified and validated by experts and analysts, so it is ready to be used for UML model generation [109A].

3.8.1. UML Use Case Model of Hospital Information Management Process Example

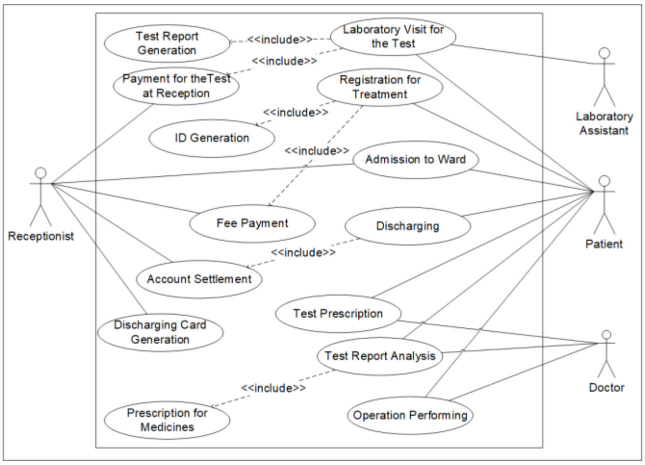
UML Use Case model is the initial form to identify and present system requirements for a new IS. Use cases identify the expected behaviour – what should be done, not the exact method of how it should be done. It is a powerful technique for communicating system behaviour in the user's conditions by specifying all externally visible system behaviour [98A].

Table 29. UML Use Case Model Elements Generated from Enterprise Model of Hospital Information Management Process Example

Enterprise Model element	**UML Use Case Model element**	**Hospital Information Management process example**	**Description**
Actor	Actor	Patient	There are four actors, each of them is a behavioural classifier which defines a role played in a particular example.
		Doctor	
		Receptionist	
		Laboratory Assistant	
Process, Function	Use Case	Laboratory Visit for the Test	There are fourteen use cases; each use case is a type of behavioural classifier that describes a unit of functionality performed by three actors.
		Test Report Generation	
		Payment for the Test at Reception	
		Registration for Treatment	
		ID Generation	
		Fee Payment	
		Admission to Ward	
		Discharging	
		Account Settlement	
		Discharging Card Generation	
		Test Prescription	
		Test Report Analysis	
		Prescription for Medicines	
		Operation Performing	
Business Rule	Include	Six include elements	There are six included elements; each includes a directed relationship between two use cases which is used to demonstrate that behaviour of the included use case is inserted into the behaviour of the included use case.

Source: created by the author [109A]

Table 29 presents UML Use Case model elements generated from the Enterprise Model of Hospital Information Management process example. In Enterprise Model, all information related to actors, their functions and relationships between these functions is stored. There are four actors: Patient, Doctor, Receptionist and Laboratory Assistant; Receptionist is related to five use cases; Laboratory Assistant – with one use case; Doctor is related to three use cases and the Patient is related to seven use cases. Four use cases include some additional use cases, six relationships in total. These elements and their relationships are presented in the following figure.



Source: created by the author [109A]

Figure 49. UML Use Case Model of Hospital Information Management Process Example

Figure 49 presents the UML Use Case model of the Hospital Information Management process example generated step by step from the Enterprise Model through the UML Use Case transformation algorithm.

3.8.2. UML Activity Models of Hospital Information Management Process Example

The UML Activity model describes how activities are coordinated, dependent on the actor or previous activity. Usually, an event needs to be gained by some operations, mainly where the operation is intended to gain a number of different things that require coordination or how the events in a single use case relate to one another, especially use cases where activities may overlap and require coordination [104A][109A].

According to the previously described UML Use Case model, there is possible to identify at least five different UML Activity models: Patient Registration, Ward Assignation, Medical Tests, Treatment Process and Discharging.

UML Activity Model: Patient Registration

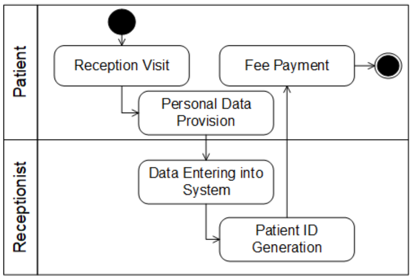
First UML Activity Model generated from EM is Patient Registration, where two participants – actors take part: the Patient and the Receptionist.

Table 30. UML Activity Model Elements Generated from Enterprise Model of Hospital Information Management Process Example, Registration Part

Enterprise Model element	**UML Activity Model element**	**Hospital Information Management process example**	**Description**
Actor	Partition	Patient	There are two partitions, and activities are related to these actors.
		Receptionist	
Function, Process	Activity	Reception Visit	There are five activities directly related to two partitions: Patient – three activities, Receptionist – two. They represent a parameterized behaviour as the coordinated flow of actions.
		Personal Data Provision	
		Data Entering into the System	
		Patient ID Generation	
		Fee Payment	
Business Rules	Control Nodes	Initial Node	There are two control nodes: one node – the initial node in the beginning, the final node at the end of the process.
		Final Node	

Source: created by the author [109A]

Table 30 presents UML Activity model elements generated from the Enterprise Model of Hospital Information Management process example, Registration part. Actor – first UML Activity model partition Patient starts registration process: visits reception, provides personal data, Actor – second partition Receptionist enters patient's data and provides patient's ID number, last activity Fee Payment is related with first partition, Patient pays the fee and registration process ends.



Source: created by the author [109A]

Figure 50. UML Activity Model of Hospital Information Management Process Example: Registration

Figure 50 presents the UML Activity Model of Hospital Information Management process example, the Registration part generated step by step from the Enterprise Model through the UML Activity model transformation algorithm [104A][109A].

UML Activity Model: Ward Assignment

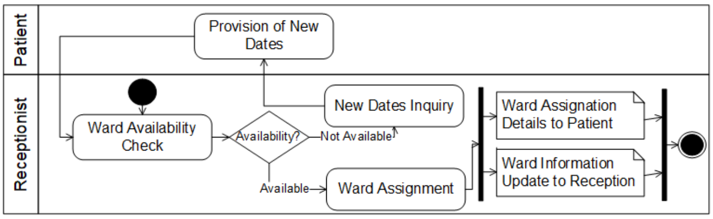
The second UML Activity Model generated from EM is Ward Assignment, where two participants – actors take part: Patient and Receptionist [109A].

Table 31. UML Activity Model Elements Generated from Enterprise Model of Hospital Information Management Process Example: Ward Assignment

Enterprise Model element	**UML Activity Model element**	**Hospital Information Management process example**	**Description**
Actor	Partition	Patient	There are two partitions, and activities are related to these actors.
		Receptionist	
Function, Process	Activity	Ward Availability Check	Four activities are directly related with two partitions: Patient – one activity, Receptionist – three. They represent a parameterized behaviour as the coordinated flow of actions.
		Provision of New Dates	
		Ward Assignment	
		New Dates Inquiry	
Information Flow	Object Flow Edge	Ward Assignation details to the Patient	There are two Object Flow Edges which are activity edges used to show a data flow between activities
		Ward Information Update to Reception	
Business Rules	Control Nodes	Initial Node	There are five control nodes: one node – initial node in the beginning; decision node – for ward availability check; join and fork nodes – to relate Object Flow Edges, the final node at the end of the process.
		Decision Node	
		Join Node	
		Fork Node	
		Final Node	

Source: created by the author [109A]

Table 31 presents UML Activity model elements generated from the Enterprise Model of Hospital Information Management process example, Ward Assignation part. Actor – first UML Activity model partition Receptionist starts Ward Assignment process: Checks ward availability, assigns it, or inquires for new dates because there are no free wards, Actor – second partition Patient provides a new date for the ward assignment, last activities are related with first partition, Receptionist prepares information for the patient and updates information in Reception and process ends [109A].



Source: created by the author [109A]

Figure 51. UML Activity Model of Hospital Information Management Process Example: Ward Assignment

Figure 51 presents the UML Activity Model of Hospital Information Management process example, Ward Assignment part generated step by step from the Enterprise Model through the UML Activity model transformation algorithm [104A][109A].

UML Activity Model: Medical Tests

The third UML Activity Model generated from EM is Medical Tests, where three participants – actors take part: Patient, Laboratory Assistant and Receptionist.

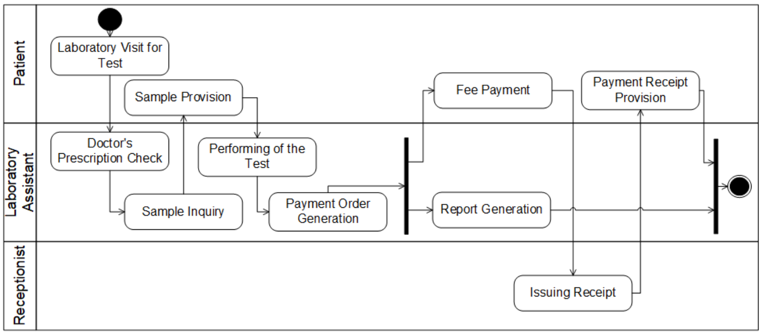
Table 32. UML Activity Model Elements Generated from Enterprise Model of Hospital Information Management Process Example: Medical Tests

Enterprise Model element	**UML Activity Model element**	**Hospital Information Management process example**	**Description**
Actor	Partition	Patient	There are three partitions, and activities are related to these actors.
		Laboratory Assistant	
		Receptionist	
Function, Process	Activity	Laboratory Visit for Test	There are ten activities directly related with three partitions: Patient – four activities, Laboratory Assistant – five
		Doctor’s Prescription Check	
		Sample Inquiry	

		Sample Provision	activities, Receptionist – one. They represent a parameterized behaviour as the coordinated flow of actions.
		Performing the Test	
		Payment Order Generation	
		Report Generation	
		Fee Payment	
		Issuing Receipt	
		Payment Receipt Provision	
Business Rules	Control Nodes	Initial Node	There are four control nodes: one node – the initial node in the beginning; join and fork nodes – to relate additional activities; the final node at the end of the process.
		Join Node	
		Fork Node	
		Final Node	

Source: created by the author [109A]

Table 32 presents UML Activity model elements generated from the Enterprise Model of Hospital Information Management process example, Medical Tests part. Actor – first UML Activity model partition Patient starts Medical Tests process: visits laboratory and provides sample after inquiry, Actor – second partition Laboratory checks doctor's prescription, inquires for sample, performs the test, generates payment order and prepares the report for the doctor; Actor – third partition Receptionist confirms payment form the patient and provides receipt; Patient makes payment and after payment confirmation receives receipt and process ends [109A].



Source: created by the author [109A]

Figure 52. UML Activity Model of Hospital Information Management Process Example: Medical Tests

Figure 52 presents the UML Activity Model of Hospital Information Management process example, Medical Tests part generated step by step from the Enterprise Model through the UML Activity model transformation algorithm [104A][109A].

UML Activity Model: Treatment Process

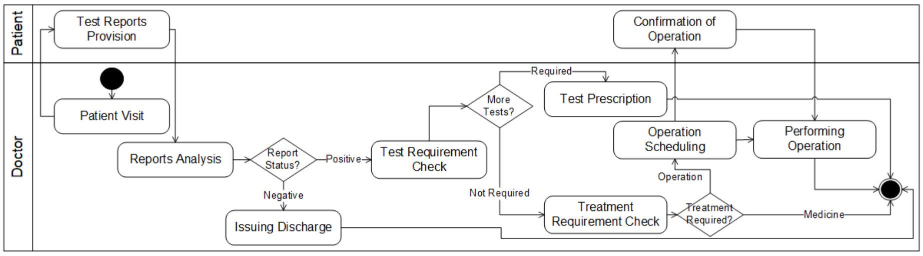
The fourth UML Activity Model generated from EM is Medical Tests, where two participants – actors take part: Patient and Doctor.

Table 33. UML Activity Model Elements Generated from Enterprise Model of Hospital Information Management Process Example: Treatment Process

Enterprise Model element	**UML Activity Model element**	**Hospital Information Management process example**	**Description**
Actor	Partition	Patient	There are two partitions, and activities are related to these actors.
		Doctor	
Function, Process	Activity	Patient Visit	Ten activities are directly related with two partitions: Patient – two activities, Doctor – eight. They represent a parameterized behaviour as the coordinated flow of actions.
		Test Report Provision	
		Report Analysis	
		Issuing Discharge	
		Test Requirements Check	
		Test Prescription	
		Treatment Requirement Check	
		Operation Scheduling	
		Confirmation of Operation	
		Performing Operation	
Business Rules	Control Nodes	Initial Node	There are five control nodes: one node – initial node in the beginning; three decision nodes – for test report status, for more tests possibility, for treatment type; the final node at the end of the process.
		Decision Nodes	
		Final Node	

Source: created by the author [109A]

Table 33 presents UML Activity model elements generated from the Enterprise Model of Hospital Information Management process example, Treatment Process part. Actor – first UML Activity model partition Doctor starts Treatment Process: meets the patient, analyses provided test reports, regarding test results, decides to discharge the patient or continue treatment process. The doctor decides if there is a need to do more tests or not, assigns treatment method medicine or operational intervention; after actor – second partition Patient confirmation, Doctor performs the operation, and the process ends [109A].



Source: created by the author [109A]

Figure 53. UML Activity Model of Hospital Information Management Process Example: Treatment Process

Figure 53 presents the UML Activity Model of the Hospital Information Management process example, the Treatment Process part generated step by step from the Enterprise Model through the UML Activity model transformation algorithm [104A][109A].

UML Activity Model: Discharging

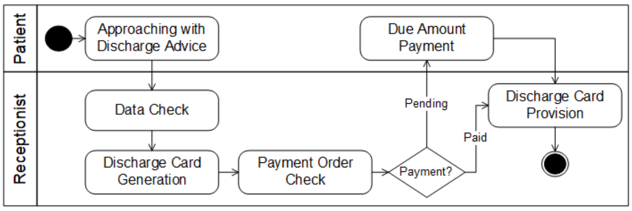
The fifth UML Activity Model generated from EM is Discharging, where two participants – actors take part: Patient and Receptionist.

Table 34. UML Activity Model Elements Generated from Enterprise Model of Hospital Information Management Process Example: Discharging

Enterprise Model element	**UML Activity Model element**	**Hospital Information Management process example**	**Description**
Actor	Partition	Patient	There are two partitions, and activities are related to these actors.
		Receptionist	
Function, Process	Activity	Approaching with Discharge Advice	Six activities are directly related with two partitions: Patient – two activities, Receptionist – four. They represent a parameterized behaviour as the coordinated flow of actions.
		Data Check	
		Discharge Card Generation	
		Payment Check Order	
		Due Amount Payment	
		Discharge Card Provision	
Business Rules	Control Nodes	Initial Node	There are two control nodes: one node – the initial node in the beginning; the decision node – for payment status final node at the end of the process.
		Decision node	
		Final Node	

Source: created by the author [109A]

Table 34 presents UML Activity model elements generated from the Enterprise Model of Hospital Information Management process example, Discharging part. Actor – first UML Activity model partition Patient starts discharging process: approaches with discharge advice from the doctor, Actor – second partition Receptionist checks data, generates discharge card, checks payment status after a patient makes the payment, provides discharge card and process ends [109A].



Source: created by the author [109A]

Figure 54. UML Activity Model of Hospital Information Management Process Example: Discharging

Figure 54 presents the UML Activity Model of Hospital Information Management process example, Discharging part generated step by step from Enterprise Model through the UML Activity model transformation algorithm [104A][109A].

3.8.3. UML Sequence Models of Hospital Information Management Process Example

The UML Sequence model is time focused and visually shows the order of the interaction by using the model's vertical axis to deliver time, what messages are sent and when [104A][109A].

According to the previously described UML Use Case and UML Activity models, there is possible to identify at least three different UML Sequence models: Patient Admission, Tests and Treatment, and Discharging.

UML Sequence Model: Patient Admission

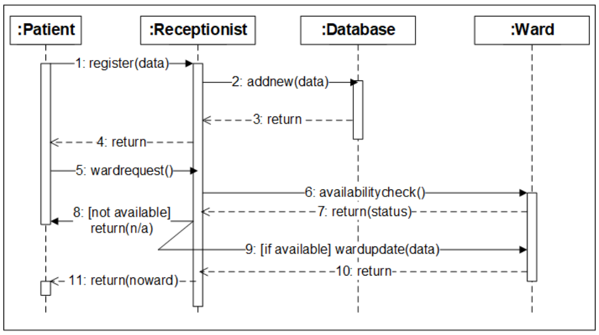
The first UML Sequence Model generated from EM is Patient Admission, where four participants – Lifelines take part: Patient, Receptionist, Database and Ward.

Table 35. UML Sequence Model Elements Generated from Enterprise Model of Hospital Information Management Process Example: Patient Admission

Enterprise Model element	**UML Sequence Model element**	**Hospital Information Management process example**	**Description**
Actor	Lifeline	Patient	There are four actors in the UML Sequence model four Lifelines, which are shown using a symbol that consists of a rectangle forming its "head" followed by a vertical line, and these lines represent the lifetime of the actor–participant of the process.
		Receptionist	
		Database	
		Ward	
Process, Function	Message	Register(data)	Eleven messages are related to actors, and they define communication between them.
		Addnew(data)	
		Return	
		Return	
		Wardrequest()	
		Availabilitycheck()	
		Return(status)	
		[not available] return(n/a)	
		[if available] ward update(data)	
		Return	
		Return(noward)	
Business Rules	Execution Specification	Five Execution Specifications	Each of the five execution specification elements represents a period in the actor's lifetime.

Source: created by the author [109A]

Table 35 presents the UML Sequence model elements generated from the Enterprise Model of Hospital Information Management process example, Patient Admission part. In Enterprise Model, all information related to actors and their collaboration is stored. There are four actors – process participants, which are called Lifelines in the UML Sequence model: persons – Patient, Receptionist, subject – Database, object – Ward. The Patient registers to the hospital; the Receptionist enters gathered data; the Patient requests the ward, the Receptionist checks availability and confirms or denies ward availability [109A].



Source: created by the author [109A]

Figure 55. UML Sequence Model of Hospital Information Management Process Example: Patient Admission

Figure 55 presents the UML Sequence model of the Hospital Information Management process example, the Patient Admission part generated step by step from the Enterprise Model through the UML Sequence model transformation algorithm [104A][109A].

UML Sequence Model: Tests and Treatment

The second UML Sequence Model generated from EM is Tests and Treatment, where four participants – Lifelines take part: Patient, Doctor, Operation and Test.

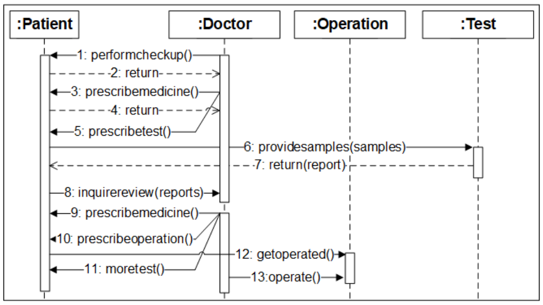
Table 36. UML Sequence Model Elements Generated from Enterprise Model of Hospital Information Management Process Example: Test and Treatment

Enterprise Model element	**UML Sequence Model element**	**Hospital Information Management process example**	**Description**
Actor	Lifeline	Patient	There are four actors in the UML Sequence model four Lifelines, which are shown using a symbol that consists of a rectangle forming its “head” followed by a vertical line, and these lines represent the lifetime of the actor–participant of the process.
		Doctor	
		Operation	
		Test	
Process, Function	Message	Performcheckup()	There are thirteen messages related to actors that define communication between them.
		Return	
		Prescribemedicine()	
		Return	
		Prescribetest()	

		Providesamples(samples)	
		Return(report)	
		Inquirereview(reports)	
		Prescribemedicine()	
		Prescribeoperation()	
		Moretest()	
		Getoperated()	
		Operate()	
Business Rules	Execution Specification	Five Execution Specifications	Each of the five execution specification elements represents a period in the actor's lifetime.

Source: created by the author [109A]

Table 36 presents the UML Sequence model elements generated from the Enterprise Model of Hospital Information Management process example, namely, the Test and Treatment part. In EM, all information related to actors-lifelines and their collaboration is stored. There are four actors – process participants, which are called Lifelines in the UML Sequence model: persons – Patient, Receptionist, objects – Operation, Test. The Doctor performs check-ups and prescribes medicine if necessary, prescribes tests, the Patient provides samples and gets reports, the Doctor reviews reports and prescribes more medicine or prescribes operation if necessary, prescribes more tests and operates [109A].



Source: created by the author [109A]

Figure 56. UML Sequence Model of Hospital Information Management Process Example: Test and Treatment

Figure 56 presents the UML Sequence model of the Hospital Information Management process example, the Test and Treatment part generated step by step from the Enterprise Model through the UML Sequence model transformation algorithm [104A][109A].

UML Sequence Model: Discharging

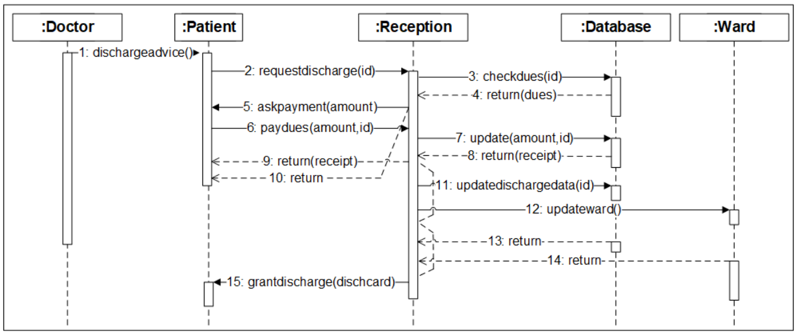
The third UML Sequence Model generated from EM is Discharging, where five participants – Lifelines take part: Doctor, Patient, Reception, Database and Ward.

Table 37. UML Sequence Model Elements Generated from Enterprise Model of Hospital Information Management Process Example: Discharging

Enterprise Model element	**UML Sequence Model element**	**Hospital Information Management process example**	**Description**
Actor	Lifeline	Doctor	There are five actors in the UML Sequence model with four Lifelines, which are shown using a symbol that consists of a rectangle forming its "head" followed by a vertical line, and these lines represent the lifetime of the actor–participant of the process.
		Patient	
		Reception	
		Database	
		Ward	
Process, Function	Message	Dischargeadvice()	There are fifteen messages related to actors that define communication between them.
		Requestdiscgarge()	
		Checkdues(patientid)	
		Return(dues)	
		Askpayment(amount)	
		Paydues(amount,patientid)	
		Update(amount,patientid)	
		Return(receipt)	
		Return(receipt)	
		Return	
		Updatedischargedata(patientid)	
		Updateward()	
		Return	
		Return	
		Grantdicharge (dichargecard)	
Business Rules	Execution Specification	Ten Execution Specifications	Each of the ten execution specification elements represents a period in the actor's lifetime. Each parallel defines potentially parallel execution of behaviours of the operands of the combined fragment.
	Parallel	Two Parallels	

Source: created by the author [109A]

Table 37 presents the UML Sequence model elements generated from the Enterprise Model of Hospital Information Management process example, Discharging part. In EM, all information related to lifelines and their cooperation is stored. There are five actors – process participants, which are called Lifelines in the UML Sequence model: persons – Patient, Doctor, subjects – Reception, Database, object – Ward. The Doctor provides discharge advice, the Patient requests for discharge, Reception checks information related to payments, inquires for payment, the Patient makes the payment, Reception updates financial information in Database, provides Receipt, Reception updates discharge information and information related to the ward, in the end, Reception provides Discharge card [109A].



Source: created by the author [109A]

Figure 57. UML Sequence Model of Hospital Information Management Process Example: Discharging

Figure 57 presents the UML Sequence model of the Hospital Information Management process example, Discharging part generated step by step from Enterprise Model through the UML Sequence model transformation algorithm [104A][109A].

3.8.4. UML State Models of Hospital Information Management Process Example

UML State Model demonstrates the diverse states of an entity [98A]. According to previously described UML models, it is possible to identify at least three different UML State model description states: Patient, Doctor and Ward.

UML State Model: Patient

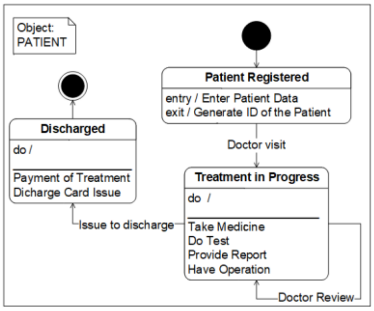
The first UML State Model generated from EM describes the states of the Patient.

Table 38. UML State Model Elements Generated from Enterprise Model of Hospital Information Management Process Example: Patient

Enterprise Model element	**UML State Model element**	**Hospital Information Management process example**	**Description**
Process, Function	Transition	Doctor Visit	Transitions from one state to the next respond to the activities, events, what causes the state's change.
		Doctor Review	
		Issue to Discharge	
Information Flow	Simple State	Patient Registered	The internal activities compartment holds a list of internal actions or state (do) activities (behaviours) performed while the element is in the state.
		Treatment in Progress	
		Discharged	
Business Rules	Initial and Final states	Initial State	Initial and final states are particular states signifying the beginning and closing processes of defined states.
		Final State	

Source: created by the author [109A]

Table 38 presents the UML State model elements generated from the Enterprise Model of Hospital Information Management process example, Patient part. In Enterprise Model, all information related to processes, functions and their states is stored. This model is from the Patient's perspective. In the model, these elements are presented: initial state, which starts the process; the first state – Patient registered; its state changes after the doctors' visit: a patient receives treatment, additional doctor's visit, after which the doctor advises discharge procedure and patient's state changes again, the patient is discharged, the process ends with final state [109A].



Source: created by the author [109A]

Figure 58. UML State Model of Hospital Information Management Process Example: Patient

Figure 58 presents the UML State model of the Hospital Information Management process example, Patient part.

UML State Model: Doctor

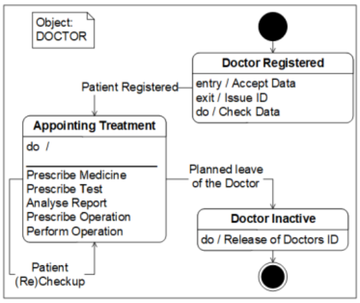
The second UML State Model generated from EM describes the states of the Doctor.

Table 39. UML State Model Elements Generated from Enterprise Model of Hospital Information Management Process Example: Doctor

Enterprise Model element	**UML State Model element**	**Hospital Information Management process example**	**Description**
Process, Function	Transition	Patient Registered	Transitions from one state to the next respond to the activities, events, what causes the state's change.
		Patient (re)Checkup	
		Planned leave of the Doctor	
Information Flow	Simple State	Doctor Registered	The internal activities compartment holds a list of internal actions or state (do) activities (behaviours) performed while the element is in the state.
		Appointing Treatment	
		Doctor Inactive	
Business Rules	Initial and Final states	Initial State	Initial and final states are particular states signifying the beginning and closing processes of defined states.
		Final State	

Source: created by the author [109A]

Table 39 presents UML State model elements generated from the Enterprise Model of Hospital Information Management process example, Doctor part. In Enterprise Model, all information related to processes, functions and their states is stored. This model is from Doctor's perspective. In the model, these elements are presented: the initial state, which starts the process; the first state Doctor registered; its state changes after the patient registers for the visit: Doctor prescribes treatment, checks up on treatment results, after the patient's discharge procedure Doctor's state changes, he is not needed for this particular patient, the process ends with the final state [109A].



Source: created by the author [109A]

Figure 59. UML State Model of Hospital Information Management Process Example: Patient

Figure 59 presents the UML State model of the Hospital Information Management process example, the Doctor part.

UML State Model: Ward

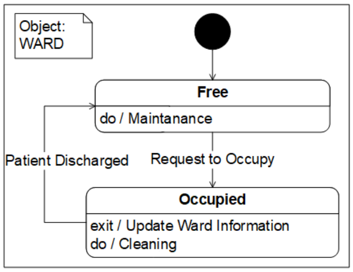
The third UML State Model generated from EM describes the states of Ward.

Table 40. UML State Model Elements Generated from Enterprise Model of Hospital Information Management Process Example: Ward

Enterprise Model element	**UML State Model element**	**Hospital Information Management process example**	**Description**
Process, Function	Transition	Request to Occupy	Transitions from one state to the next respond to the activities, events, what causes the state's change.
		Patient Discharged	
Informatio n Flow	Simple State	Free	The internal activities compartment holds a list of internal actions or state (do) activities (behaviours) performed while the element is in the state.
		Occupied	
Business Rules	Initial and Final states	Initial State	The initial state is a particular state signifying the beginning process of defined states.

Source: created by the author [109A]

Table 40 presents UML State model elements generated from the Enterprise Model of Hospital Information Management process example, Ward part. In Enterprise Model, all information related to processes, functions and their state is stored. This model is from Ward's perspective. In the model, these elements are presented: an initial state which starts the process; the first state means the ward is free; its state changes after the request to occupy; after the patient is discharged ward state changes again to free [109A].



Source: created by the author [109A]

Figure 60. UML State Model of Hospital Information Management Process Example: Ward

Figure 60 presents the UML State model of the Hospital Information Management process example, Ward part.

Having analysed the Hospital Information Management process example, the results of four UML models are the following: Use Case, Activity, Sequence and State generation from Enterprise Model through transformation algorithms are presented straightforwardly; all models define the same example but from different perspectives. In almost every subsection of the described example, there is more than one model of the same type presented: generated UML Use Case model presents all participants (Actors) who are involved in the Hospital Information Management process and their functions/processes (Use Cases); generated UML Activity models illustrate different activities from different perspectives (Registration, Ward Assignation, Medical Tests, Treatment process and Discharging) of the same example, and it is not the final list of possible models of the same type; generated UML Sequence model also defines sequence processes and functions sequences from different perspectives (Patient Admission, Tests and Treatment, Discharging) of the same example, and it is also not the final list of possible UML Sequence models; generated UML State model describe different states from the perspectives of objects (Patient, Doctor and Ward), and states of more objects of the same example can be generated [109A].

Provided example of the Hospital Information Management process shows and confirms that it is not the final amount of UML models which can be generated from EM; there are more different perspectives for UML model generation of the same example. As stated previously, the knowledge-based Enterprise Model, which stores verified and validated data of a specific subject domain, is enough data storage for the generation of various UML models [109A].

3.9. Proposed Method Principals' Application in Financial Data Analysis

Proposed method principals were applied in the project "Enterprise Financial Performance Data Analysis Tools Platform" activities. The aim of this project was to develop a finance and audit analytics system to automate business, finance and internal audit activities using artificial intelligence (AI), machine learning (ML) and big data technologies.

The proposed method was used to create dynamic models, by adapting the transformation algorithms presented in the thesis to the project specifics. The created dynamic models were applied for financial data analysis, for example Altman Z score indicator and behavioural change indicators (BCI's) calculation [66A] and identification of irregular financial operations [83A]. The workflow and decision automation platform Camunda, supporting Decision Model and Notation (DMN), and Natural Language Processing NLP-based algorithms were used to reach a part of the project interim goals. The project was successfully completed on time and the solutions have been applied in practice.

3.10. Evaluation of UML Dynamic Models Generation Method Based on Presented Examples Results

As mentioned before, traditional IS engineering is based on the analyst's experience. The analyst participates during the entire IS development life cycle process by analysing the subject domain, modelling and designing all project models, in our case UML models, needed, using different tools for this purpose, relying only on his personal experience and good practices. The analyst's activities may be described as analysis of the enterprise situation, identification of opportunities for improvements, design of an information system which will add value to the organisation. The analyst gathers subject domain information, identifies all requirements, searches for suitable meta-models, verifies and validates data, and starts implementing IS by designing UML models. There is always a risk of new problems, information or requirements appearing, which cause complicated project models updating and improving the process. The duration of IS developing process is increasing, and the number of errors is growing.

Using Enterprise Model as the core subject domain knowledge repository in IS engineering process ensures the corectness and quality of generated IS project models after any possible subject domain data update.

Table 41. Comparison of IS Analyst's Activities by Criteria

Criteria	**Traditional IS engineering**	**Enterprise Knowledge-Based UML Dynamic Models Generation Method**
Gathering subject domain data	The analyst gathers information from stakeholders, subject domain analysis, previous systems by using different methodologies	The analyst gathers information from stakeholders, subject domain analysis, previous systems by using different methodologies
Requirements identification	The analyst collects, identifies and specifies the requirements of the particular subject domain	The analyst collects, identifies and specifies the requirements of the particular subject domain

Data preparation for modelling	The analyst prepares data for modelling, uses any possible modelling tools of his own choice, relying on his own empirical experience.	The analyst prepares data for modelling, uses the enterprise model, which is formalised by enterprise meta-model; verifies and validates knowledge of the particular subject domain and fills the enterprise model with necessary data
Project models design	The analyst creates design models by using design tools according to his empirical experience	Necessary UML dynamic models are generated from the enterprise model by using transformation algorithms
Subject domain data update	The analyst approves the necessary changes and updates the subject domain data	The analyst verifies and validates the necessary changes and updates subject domain data in the enterprise model
Project model design improvement	The analyst re-creates or improves design models by using design tools	Necessary UML dynamic models are generated from enterprise model with updated data by using transformation algorithms
Increased IS development process duration	For design models improvement, additional time is needed	UML dynamic models are generated from enterprise model faster than without transformation algorithms
Increased number of errors	The possibility of getting an increased number of errors is higher	There is less possibility of getting an increased number of errors because subject domain data was verified and validated in advance

Source: created by the author

By using Enterprise Model in IS engineering process, the analysts enter all gathered subject domain data into EM. Subject domain knowledge stored in EM is used for UML model generation through transformation algorithms. After any possible new data upload to the EM, it is re-used and new UML models based on improved data are generated. The analyst does not need to re-do the entire project model design process. The main difficulties can be related to a lack of analyst experience during the process of subject domain analysis, gathering necessary data and completing the Enterprise model.

3.11. Third Part Conclusions

The third part represents how the presented Enterprise knowledge-based UML dynamic model generation method can be used for different subject domains. Different examples of different UML dynamic models and two more complex examples are presented: Ticket Buying Process and Hospital Information Management Process. Two complex examples demonstrate the wide range of UML models that can be created during the IS development process. This variety of models is not final; there is a possibility to create more models in the design phase and present the system under development in a more detailed way. There are also defined advantages of the proposed method from the perspective of analyst role evaluation in traditional and knowledge-based IS engineering process.

CONCLUSIONS

1. The analysis of information systems engineering methods and standards showed that existing methods and CASE tools created on their basis can generate IS project fragments in UML notation, such as: user interface prototype, database specifications, program code fragments, but no solutions for generating UML dynamic models of the design phase of IS development have been found. The application of the proposed method improves the efficiency of the IS design phase.
2. The transformation algorithms developed in the work and experimentally verified prove that the composition of the enterprise model (EM) is sufficient to generate UML dynamic models (use case model, activity model, state machine model, sequence model, communication model, timing model, interaction overview model).
3. Seven transformation algorithms developed in the work were used in the experiment, and they were used to generate dynamic UML models from activity models of eight subject areas. The generated dynamic models meet the requirements of the UML specification, because all elements of the composition of each type of UML dynamic model and the relationships between them are generated.
4. The proposed UML dynamic model generation method was evaluated by comparing the performance of an IS analyst in traditional and knowledge-based IS engineering. The proposed method is superior in performing dynamic model corrections (semi-automatic) and has advantages in the process of creating primary models. The results of the comparison are presented in table 41.
5. The results of this work provide a basis for the development of CASE tool plug-ins enabling the generation of dynamic models of an IS project from the subject domain knowledge stored in the CASE tool's knowledge repository.

REFERENCES

1. Aier, S., Gleichauf, B., Winter, R.: Understanding enterprise architecture management design - an empirical analysis. *Wirtschaftsinformatik Proceedings*, 2011.
2. Aksit M., Kindler E., McNeile A., Roubtsova E., (2009) Behaviour Modelling in Model Driven Architecture. *CTIT Workshop Proceedings Series* WP09-04, ISSN 0929-0672, Enschede, the Netherlands.
3. Alouini, W., Guedhami, O., Hammoudi, S., Gammoudi, M., Lopes, D.: Semi-Automatic Generation of Transformation Rules in Model Driven Engineering: The Challenge and First Steps. *International Journal of Software Engineering and Its Applications*. Vol. 5 No. 1, January, (2011).
4. Asadi, M., Ramsin, R. (2008) MDA-based Methodologies: *An Analytic Survey. Proceedings of the 4th European conference on Model Driven Architecture: Foundations and Applications.* pp. 419-431, Berlin
5. Bente, S., Bombosch, U., Langade, S., Collaborative Enterprise Architecture: Enriching EA with Lean, Agile, and Enterprise 2.0 Practices. *Elsevier*, Inc., 2012.
6. Bousetta B., El Beggar O.,, Gadi T., (2013) A methodology for CIM modelling and its transformation to PIM. *Journal of Information Engineering and Applications* www.iiste.org ISSN 2224-5782 (print) Vol.3, No.2, 2013.
7. Buckl, S., A Design Theory Nexus for Situational Enterprise Architecture Management. *Proceedings of the 14th International IEEE Enterprise Distributed Object Computing Conference*. IEEE Computer Society. Vitoria, Brazil, 2010, pp. 3–8.
8. Butleris R., Lopata. A., Ambraziunas M.; Gudas. S. (2012) The Main Principles of Knowledge-Based Information Systems Engineering. *Electronics And Electrical Engineering*, Vol. 120, No 4 (2012), pp. 99-102, ISSN 1392-1215
9. Cabot, J.; Pau, R.; Raventos, R. From UML/OCL to SBVR specifications: A challenging transformation. *Information Systems*, 2008, p. 1–24.
10. Chen, C.K., Construct Model of the Knowledge-based Economy Indicators, *Transformations in Business & Economics*, Vol. 7, No 2(14), 2008.
11. Chen, R., Sun, Ch., Helms, M., Jihd, W., Aligning information technology and business strategy with a dynamic capabilities

perspective: A longitudinal study of a Taiwanese Semiconductor Company. *International Journal of Information Management* 28 366–378, 2008.

12. Cuenca, L., Boza, A., Ortiz, A. Architecting Business and IS/IT Strategic Alignment for extended Enterprises. *Studies in Informatics and Control*, Vol. 20, No. 1, March 2011.
13. Dobing, B., Parsons, J., How UML is used. *Communications of the ACM - Two decades of the language-action perspective*. Volume 49 Issue 5, May 2006 Pages 109-113. ACM New York, NY, USA, 2006.
14. Dunkel J., Bruns R., Model-Driven Architecture for Mobile Applications. *Proceedings of the 10th Inter-national Conference on Business Information Systems* (BIS), Vol. 4439/2007, pp. 464–477, 2007.
15. Eichelberger H., Eldogan, Y., Schmid K. (2011) A Comprehensive Analysis of UML Tools, their Capabilities and Compliance. *Software Systems Engineering*. Universität Hildesheim. August 2011.
16. FFIECC. Development Procedures. IT Examination Handbook InfoBase. Available from [Accessed May 2013]: http://ithandbook.ffiec.gov/it-booklets/development-and-acquisition/development-procedures.aspx
17. Fouad A., Phalp K., Kanyaru J. M., Jeary S., Embedding requirements within Model-Driven Architecture. *Software Quality Journal*, Vol. 19, No 2, pp. 411-430, 2011.
18. Freiberg, M., Baumeister, J., Puppe, F., Interaction Pattern Categories — Pragmatic Engineering of Knowledge-Based Systems. *6th Workshop on Knowledge Engineering and Software Engineering (KESE6) at the 33rd German Conference on Artificial Intelligence* September 21, Karlsruhe, Germany, 2010.
19. Gerber A., le Roux P., Kearney C., van der Merwe A. The Zachman Framework for Enterprise Architecture: An Explanatory IS Theory. In: Hattingh M., Matthee M., Smuts H., Pappas I., Dwivedi Y., Mäntymäki M. (eds) *Responsible Design, Implementation and Use of Information and Communication Technology*. I3E 2020. Lecture Notes in Computer Science, vol 12066. Springer, Cham. 2020.
20. Gruber, T.R. Toward Principles for the Design of Ontologies Used for Knowledge Sharing. Technical report, KSL-93-04. *Knowledge Systems Laboratory*, Stanford University, 1993, p. 1–23.
21. Gudas S., Žiniomis grindžiamos informacijos sistemų inžinerijos procesų modeliai. Habilitacijos procedūrai teikiamų mokslo darbų apžvalga Kaunas: Kauno technologijos universitetas, 2005.

22. Gudas S. Enterprise knowledge modelling: Domains and aspects. Technological and economic development of Economy. *Baltic Journal on Sustainability*, 281–293, 2009.
23. Gudas S. Žiniomis grindžiamos IS inžinerijos metodų principai. - *Konferencijos pranešimų medžiaga „Informacinės technologijos 2004“,* T.2, Kaunas, Technologija, 2004, p.713-717 – ISBN 9955-09-788-4
24. Gudas S., Informacijos sistemų inžinerijos teorijos pagrindai. Vilniaus universiteto leidykla ISBN 978-609-459-075-7, 2012.
25. Gudas S., Architecture of Knowledge-Based Enterprise Management Systems: a Control View, *Proceedings of the 13th world multiconference on systemics, cybernetics and informatics* (WMSCI2009),) July 10 – 13, 2009, Orlando, Florida, USA, Vol. III, p.161-266 ISBN -10: 1-9934272-61-2 (Volume III).ISBN -13: 978-1-9934272-61-9 (Volume III).
26. Gudas S., Lopata A. Informacijos išteklių identifikavimas veiklos modelio pagrindu. Vilniaus universiteto leidykla, 2001.
27. Gudas S., Lopata A. Vartotojo poreikių modelio generavimas veiklos modelio pagrindu. *Informacijos mokslai* 2003 , p. 134- 140. ISSN 1392-0561.
28. Gudas S., Lopata A. Žiniomis grindžiama informacijos sistemų inžinerija. *Informacijos mokslai, Mokslo darbai*, T.30, Vilnius, Vilniaus Universiteto leidykla, 2004, p.90-98 ISSN 1392-0561.
29. Gudas S., Lopata A. Žiniomis grindžiama sistemų inžinerija. Mokomoji medžiaga. Vilniaus universiteto Kauni humanitarinis fakultetas. 2011.
30. Gudas S., Lopata A., Skersys T. Framework for knowledge-based IS engineering. *Advances in Information Systems: Third International Conference*, ADVIS 2004, Izmir, Turkey, October 20-22, 2004, p. 512
31. Gudas S., Lopata A., Skersys T. Domain Knowledge Integration For Information Systems Engineering. *Informacinės technologijos verslui 2002: konferencijos pranešimų medžiaga*. Kaunas: Technologija 2002. p.56-59
32. Gudas, S., Brundzaite R. Knowledge-Based Enterprise Modelling Framework. *Advances in Information Systems. Lecture Notes in Computer Science*, 2006, Volume 4243/2006, 334-343, DOI: 10.1007/11890393_35
33. Gudas, S., Lopata, A.; Skersys, T. Approach to Enterprise Modelling for Information Systems Engineering. *Informatica*. no. 2, 175-192, DOI 10.15388/Informatica.2005.092

34. Henderson, J., Venkatraman, N., Strategic Alignment: Leveraging Information Technology for Transforming Organizations, *IBM Systems Journal*, Vol. 38, No 2,3, pp.472-484, 1999.
35. Hevner, A.R.; March, S.T.; Park, J.; Ram, S. Design Science in Information Systems Research. *MIS Quarterly*, 28 (1), 2004.
36. Hinkelmann, K., Gerber, A., Karagiannis, D., Thoenssen, B., van der Merwe, A., Woitsch, R.: A new paradigm for the continuous alignment of business and IT: combining enterprise architecture modelling and enterprise ontology. *Computers in Industry*. 79, 77–86, 2016.
37. Hofstede, A.H.M.T.; Proper, H.A. How to formalize it? Formalization Principles for Information Systems Development Methods. *Information and Software Technology*, 1998, 40(10), p. 519–540.
38. IEEE Computer Society. Guide to the Software Engineering Body of Knowledge SWEBOK. Version 3.0. Paperback ISBN-13: 978-0-7695-5166-1., 2014.
39. ISO/IEC 12207. Systems and software engineering — Software life cycle processes. Reference number ISO/IEC 12207:2008(E) IEEE Std 12207-2008.
40. ISO/IEC/IEEE 15288:2015. Systems and software engineering — System life cycle processes. Reference number ISO/IEC 15288:2015(en)
41. ISO/IEC 19505-1, 19505-2. Information technology - Object Management Group Unified Modelling Language (OMG UML), Infrastructure. ISO/IEC19505-1,-2:2012(E).
42. ISO/IEC 29148. Systems and software engineering – Life cycle processes - Requirements engineering. Reference number ISO/IEC/IEEE 29148:2011(E).
43. ISO/IEC 19501:2005 Information technology — Open Distributed Processing — Unified Modeling Language (UML) Version 1.4.2
44. ISO/IEC JTC 1. Information technology standards. Available from [Accessed May 2014] https://www.iso.org/committee/45086.html
45. Jenney J. Modern Methods of Systems Engineering: With an Introduction to Pattern and Model Based Methods. ISBN-13:978-1463777357, 2010
46. Kappelman, L.A., Zachman, J.A.: The enterprise and its architecture: ontology and challenges. *Journal of Computer Information Systems Summer* 2013(4):87-95, June 2013.

47. Kardoš, M., Drezdová, M. Analytical Method of CIM to PIM Transformation in Model Driven Architecture (MDA). *Journal of Information & Organizational Sciences*; 2010, Vol. 34 Issue 1, p89.
48. Kerzazi, N., Lavallée, M., Robillard, P.N., A Knowledge-Based Perspective for Software Process Modelling. e-Informatica *Software Engineering Journal*, Volume 7., 2013.
49. Koch, Ch., Introduction to Information Technology, *Scientific e-Resources*, 2018, ISBN 1839472405, 9781839472404
50. Kirikova M., Finke A., Grundspenki J., What is CIM: an information system perspective, *Advances in Databases and Information Systems*, Vol. 5968, pp. 169-176., 2010.
51. Küster, J.M., Sendall, S., & Wahler, M., Comparing Two Model Transformation Approaches, 2004.
52. Lamsweerde A., Requirements engineering: from system goals to UML models to software specifications. Chichester Wiley, 2011.
53. Lapalme, J., Gerber, A., van der Merwe, A., de Vries, M., Hinkelmann, K.: Exploring the future of enterprise architecture: a Zachman perspective. *Computers in Industry*. 79, 103–113, 2016.
54. Lin, F., Dyck, H.: The value of implementing enterprise architecture in organizations. *Journal of International Technology and Information Management* 19, 2010.
55. Lönnfors, S., Theoretical and practical Requirements Engineering. *Degree Thesis*, 2012.
56. Lopata A., Veiklos modelių sudėties analizė. *Informacinės technologijos ir valdymas*, 2000.
57. Lopata A. Veiklos modeliu grindžiamas kompiuterizuotas funkcinių vartotojo reikalavimų specifikavimo metodas. *Disertacija*, 2004.
58. 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. 2012
59. Lopata A., Ambraziūnas M., Gudas S., Butleris R., The Main Principles of Knowledge-Based Information Systems Engineering, *Electronics and Electrical Engineering*, Vol. 11, No 1 (25), pp. 99-102, ISSN 2029-5731, 2012
60. Lopata A.; Gudas S. Meta-Model Based Development of Use Case Model for Business Function, *Information Technology and Control*, vol. 36, No. 324 2007

61. Lopata, A., Ambraziūnas, M., Gudas, S., Butleris, R. The main principles of knowledge-based information systems engineering. *Electronics and Electrical Engineering*, 2012, 4(120) 99-102. ISSN 1392-1215.
62. Lopata, A., Ambraziūnas, M., Veitaitė, I., Masteika, S., Butleris, R., SysML and UML models usage in knowledge based MDA process, *Elektronika ir elektrotechnika*. Kaunas : KTU. ISSN 1392-1215. eISSN 2029-5731. 2015, vol. 21, no. 2, p. 50-57 (A – author).
63. Lopata, A., Veitaitė, I., UML diagrams generation process by using knowledge-based subsystem, *Business Information Systems Workshop* 2013, Poznan, Poland, June 2013 / Witold Abramowicz (ed.). Berlin : Springer, 2013. ISBN 9783642416866. p. 53-60. (A – author).
64. Lopata, A., Veitaitė, I., Gudas, S., Butleris, R., Case tool component - knowledge-based subsystem UML diagrams generation process. *Transformations in business & economics*. Vilnius University, Brno University of Technology, University of Latvia. Brno, Kaunas, Riga, Vilnius, : Vilniaus universitetas. ISSN 1648-4460. 2014, Vol. 13, No. 2B (32B), p. 676-696. (A – author).
65. Lopata, A., Veitaitė, I., Žemaitytė, N., Enterprise model based UML interaction overview model generation process. *Business information systems workshops : BIS 2016 international workshops*, Leipzig, Germany, July 6-8, 2016 : revised papers / Editors: Abramowicz, Witold, Alt, Rainer, Bogdan, Franczyk . - Series: Lecture notes in business information processing. Vol. 263. ISSN: 1865-1348. Berlin : Springer International Publishing, 2017. ISBN 9783319524634. eISBN 9783319524641. p. 69-78. (A – author).
66. Lopata, Audrius; Gudas, Saulius; Butleris, Rimantas; Rudžionis, Vytautas Evaldas; Žioba, Liutauras; Veitaitė, Ilona; Dilijonas, Darius; Grišius, Evaldas; Zwitserloot, Maarten. Financial data anomaly discovery using behavioral change indicators // *Electronics*. Basel : MDPI. ISSN 2079-9292. 2022, vol. 11, iss. 10, art. no. 1598, p. 1-14. DOI: 10.3390/electronics11101598. (A – author).
67. Luftman, J., Assesing Business-IT Alginment Maturity. *Communications of the Association for Information Systems* vol.4(14), 2000.
68. Luftman, J., Lewis, P.R., Oldach, S.H. Transforming the Enterprise: The Alignment of Business and Information Technology Strategies. *IBM Systems Journal*, vol.32(1), p.198, 1993.

69. Melouk M., Rhazali Y., Youssef H. (2020) An Approach for Transforming CIM to PIM up To PSM in MDA. *Procedia Computer Science*. Volume 170, 2020, Pages 869-874.
70. Morkevicius A., Gudas S., Enterprise Knowledge Based Software Requirements Elicitation, *Information Technology and Control*, Vol. 40, No 3, pp. 181-190, 1392 – 124X, 2011.
71. Ndie T., Tangha C., Ekwoge F., MDA (Model-Driven Architecture) as a Software Industrialization Pattern: An Approach for a Pragmatic Software Factories, *Journal of Software Engineering and Applications*, Vol. 3 No. 6, pp. 561-571, ISSN 1945-3124, 2010.
72. Nemuraite, L.; Ceponiene, L.; Vedrickas, G. Representation of Business Rules in UML&OCL Models for Developing Information Systems. In: *The Practice of Enterprise Modeling*. LNBIP, vol. 15, 2008. Springer Berlin Heidelberg, Germany, ISBN 978-3-540-89217-5, p. 182-196.
73. Normantas, K. Research on Business Knowledge Extraction from Existing Software Systems. *Doctoral thesis*, 2013. Vilnius Gediminas Technical University.
74. OMG MOF. Meta Object Facility. Available from [Accessed May 2013]: https://www.omg.org/spec/MOF/2.5/Beta1/About-MOF/
75. OMG UML. Unified Modelling Language version 2.5.1 Unified Modelling. Available from [Accessed May 2020]: https://www.omg.org/spec/UML/About-UML/
76. Peak, D., Guynes, C., Prybutok, V., Xu, Ch., Aligning information technology with business strategy: An action research approach. *JITCAR*, Volume 13, Number 1, 2011.
77. Perjons, E., Model-Driven Process Design. Aligning Value Networks, Enterprise Goals, Services and IT Systems. Department of Computer and Systems Sciences, Stockholm University. Sweden by US-AB, Stockholm ISBN 978-91-7447-249-3, 2011.
78. Plazaola, L., Flores, J., Vargas, N., Ekstedt, M., Strategic Business and IT Alignment Assessment: A Case Study Applying an Enterprise Architecture-based Metamodel. *Proceedings of the 41st Hawaii International Conference on System Sciences*. 1530-1605/08, 2008.
79. Poole, J. D. Model-Driven Architecture: Vision, Standarts And Emerging Technologies. European Conference on Object-Oriented Programming *Workshop on Metamodeling and Adaptive Object Models*, 2001.

80. Rahimi, F., Gotze, J., Moller, C.: Enterprise architecture management: toward a taxonomy of applications. *Communications of the Association for Information Systems*. 40, 120–166, 2017.
81. Rosen, S., A glimpse into the current state of UML use. *Architexa – Working with Large Codebases*, 2010
82. Rouhani, B.D., Mahrin, M.N., Nikpay, F., Ahmad, R.B., Nikfard, P.: A systematic literature review on enterprise architecture implementation methodologies. *Information and Software Technology*. 62, 1–20, 2015.
83. Rudžionis, Vytautas; Lopata, Audrius; Gudas, Saulius; Butleris, Rimantas; Veitaitė, Ilona; Dilijonas, Darius; Grišius, Evaldas; Zwitserloot, Maarten; Rudzioniene, Kristina. Identifying irregular financial operations using accountant comments and natural language processing techniques // *Applied sciences*. Basel : MDPI. ISSN 2076-3417. 2022, vol. 12, iss. 17, art. no. 8558, p. 1-15. DOI: 10.3390/app12178558. (A – author).
84. Rumbaugh, J., Jacobson, I., Booch, G., The Unified Software Development Process, Addison Wesley Professional, p. 496, 1999.
85. Sajja, P.S., Akerkar, R., Knowledge-Based Systems for Development. Advanced Knowledge Based Systems: Model, *Applications & Research*, Vol. 1, 2010.
86. Sharifi, H. R., Mohsenzadeh M. (2012) A New Method for Generating CIM Using Business and Requirement Models. *World of Computer Science and Information Technology Journal (WCSIT)*, ISSN: 2221-0741, Vol. 2, No. 1, p.p. 8-12.
87. Silvius, G., Business & IT alignment in theory and practice, *IEEE proceedings of the 40th Hawaii international conference on System Science*, 2007.
88. Soares, M.S., Vrancken, J., Model-Driven User Requirements Specification using SysML, *Journal of Software*, Vol. 3, No. 6, pp. 57-68, ISSN 1796-217X, 2008
89. Sommerville, I., Software engineering — 9th ed. Pearson Education, Inc., publishing as Addison-Wesley. ISBN-13: 978-0-13-703515-1, 2011
90. Ulrich W.M., Information systems transformation: architecture-driven modernization case studies. *Newcomb Amsterdam. Elsevier*. Morgan Kaufmann xix, 2010
91. UML Diagrams. Unified Modeling Language website, Available from [Accessed February 2020]: www.uml-diagrams.org

92. Valatavičius, A., Gudas, S.: Toward the deep, knowledge-based interoperability of applica-tions. *Information sciences*, Vol. 79, 2017.
93. Vargas Chevez, N., A Unified Strategic Business and IT Alignment Model: A Study in the public universities of Nicaragua. *Licentiate Thesis in Computer and Systems Sciences Royal Institute of Technology*, KTH Stockholm, Sweden, ISBN 978-91-7415-646-1, 2010.
94. Veitaitė I., Lopata A. Veiklos modeliu grindžiamas UML diagramų generavimas. *XVII tarpuniversitetinė magistrantų ir doktorantų konferencija. Konferencijos pranešimų medžiaga*. Vilniaus universitetas. ISSN 2029-249X, 2012. (A – author).
95. Veitaitė, I., Ambraziūnas, M., Lopata, A., Enterprise model and ISO standards based information system's development process. *Business information systems: 2014 international workshops*, Larnaca, Cyprus, May 22-23, 2014: Revised Papers: proceedings.- Series: Lecture notes in business information processing (ISSN 1865-1348), Vol. 183/ Editors : Witold Abramowicz, Angelika Kokkinaki. Berlin : Springer, 2014. ISBN 9783319114590. p. 73-79. (A – author).
96. Veitaitė, I., Lopata, A., Additional knowledge based MOF architecture layer for UML models generation process, *Business information systems: 2015 international workshops: revised papers: proceedings*.- Series: Lecture notes in business information processing (ISSN 1865-1348), Vol. 226 / Editors : Witold Abramowicz, Angelika Kokkinaki. Berlin : Springer International Publishing, 2015. ISBN 9783319267616. eISBN 9783319267623. p. 56-63. (A – author).
97. Veitaitė, I., Lopata, A., Enterprise knowledge based UML timing model generation process. *XXII tarpuniversitetinė magistrantų ir doktorantų konferencija Informacinė visuomenė ir universitetinės studijos (IVUS2017 : pranešimų medžiaga*. Kaunas : Technologija. ISSN 2029-249X. 2017, p. 189-193. (A – author).
98. Veitaitė, I., Lopata, A., Enterprise model, MOF and ISO standards based information system’s development process. *Informacinės technologijos 2015: XX tarpuniversitetinė magistrantų ir doktorantų konferencija : Konferencijos pranešimų medžiaga*. Kaunas : Technologija. ISSN 2029-249X. 2015, p. 77-81. (A – author).
99. Veitaitė, I., Lopata, A., Knowledge based UML information flow model transformation algorithm. *ICYRIME 2018: proceedings of the international conference for young researchers in informatics,*

mathematics, and engineering, Kaunas, Lithuania, April 27, 2018. Aachen : CEUR-WS. 2018, p. 30-36. (A – author).

100. Veitaitė, I., Lopata, A., Knowledge-based transformation algorithms of UML dynamic models generation from enterprise model. *Data science: new issues, challenges and applications*. Dzemyda, Gintautas, Bernatavičienė, Jolita, Kacprzyk, Janusz (Eds.). Cham : Springer Nature, 2020. ISBN 9783030392499. eISBN 9783030392505. p. 43-59.
101. Veitaitė, I., Lopata, A., Knowledge-based UML models generation from Enterprise Model technique. *Information and software technologies : 23rd international conference, ICIST 2017, Druskininkai, October12-14, 2017 : proceedings* / editors: Robertas Damaševičius, Vilma Mikašytė . - Book series : Communications in Computer and Information Science. Vol 756. Cham : Springer Nature. ISSN 1865-0929. eISSN 1865-0937. 2017, p. 314-325. (A – author).
102. Veitaitė, I., Lopata, A., Knowledge-based UML use case model transformation algorithm. *Business information systems workshops: BIS 2019 international workshops,* Seville, Spain, June 26–28, 2019, revised papers / eds.: Witold Abramowicz, Rafael Corchuelo. Cham : Springer, 2019. ISBN 9783030366902. eISBN 9783030366919. p. 39-48. (A – author).
103. Veitaitė, I., Lopata, A., Problem domain knowledge driven generation of UML models. *Information and software technologies: 24th International Conference, ICIST 2018, Vilnius, Lithuania, October 4–6, 2018 : proceedings,* edited by: Robertas Damaševičius, Giedrė Vasiljevienė. Cham : Springer, 2018. ISBN 9783319999715. eISBN 9783319999722. p. 178-186. (A – author).
104. Veitaitė, I., Lopata, A., Transformation algorithms of knowledge based UML dynamic models generation. *Business information systems workshops BIS 2017,* Poznan, Poland, 28-30 June / editor Witold Abramowicz . - Series : Lecture notes in business information processing. Vol 303. Cham : Springer International Publishing, 2017. ISBN 9783319690223. eISBN 9783319690230. p. 59-68. (A – author).
105. Veitaitė, I., Lopata, A., Veiklos modelio taikymas informacijos sistemų inžinerijos reikalavimų specifikavimo etape. *Informacinės technologijos: 19-oji tarpuniversitetinė magistrantų ir doktorantų konferencija Informacinė visuomenė ir universitetinės studijos (IVUS 2014): pranešimų medžiaga.* Kaunas : Technologija. ISSN 2029-4832. 2014, p. 40-46. (A – author).

106. Veitaitė, I., Lopata, A., Enterprise knowledge based UML timing model generation process. *CEUR workshop proceedings [e.source]: IVUS 2017: Proceedings of the IVUS International Conference on Information Technology*, Kaunas, Lithuania, April 28, 2017, Edited by Robertas Damaševičius, ... [et al]. Aachen : CEUR-WS. ISSN 1613-0073. 2017, Vol.1856, p. 75-79. (A – author).
107. Veitaitė, I., Lopata, A., Knowledge-Based UML Activity Model Transformation Algorithm. CEUR workshop proceedings [e.source]: *IVUS 2020: Proceedings of the Information Society and University Studies 2020*, Kaunas, Lithuania, April 23, 2020.: CEUR-WS. ISSN 1613-0073. 2020, Vol.2698, p. 114-120. (A – author).
108. Veitaite I., Lopata A. (2020) Knowledge-Based Generation of the UML Dynamic Models from the Enterprise Model Illustrated by the Ticket Buying Process Example. In: Lopata A., Butkienė R., Gudonienė D., Sukackė V. (eds*) Information and Software Technologies. ICIST 2020. Communications in Computer and Information Science*, vol 1283. Springer, Cham. https://doi.org/10.1007/978-3-030-59506-7_3. (A – author).
109. Veitaitė, I., Lopata, A., Knowledge-Based UML Dynamic Models Generation from Enterprise Model in Hospital Information *Management Process Example Intelligent Systems for Sustainable Person-Centred Healthcare*. Kriksciuniene, Dalia, Sakalauskas, Virgilijus, (Eds.).: Springer Book of Chapters, 2022. (A – author).
110. Vitiutinas R., Silingas D., Telksnys L., Model-Driven Plug-In Development for UML Based Modelling Systems, *Information Technology and Control*, Vol. 40, No 3, pp. 191-201, ISSN 1392 – 124X, 2011.
111. Yamin, M., Zuna, V., Bugami A. (2010) Requirements Analysis and Traceability at CIM Level. *Journal of Software Engineering and Applications,* ISSN 1945-3124, Vol.3 No.9, p.p. 845-851.
112. Zachman, J.A., A Framework for Information Systems Architecture. *IBM Systems Journal*, Vol. 26, No. 3, 1987. (The same article was reprinted in 1999 in a special double issue of the IBM Systems Journal that is easier to locate: Vol. 38, Nos 2&3, 1999.)

SUMMARY IN LITHUANIAN

ĮVADAS

Verslo ir IT suderinamumas yra reikšminga valdymo proceso dalis ir jau daugiau nei du dešimtmečius tai išlieka pagrindiniu organizacijų tikslu. Aukštesnio lygio strateginis verslo ir IT strategijos derinimas turėtų padėti pasiekti geresnius rezultatus, palyginus su žemesnio lygio strateginio derinimo etapais. Pastaruoju metu intensyviai diskutuojama apie sąvoką „verslo ir IT suderinamumas“ ir šio proceso poveikį [12]. Kai kurie teigia, kad derinimas yra neesminis ir paprastas tikslas, kuris gali būti pasiekiamas tik nedaugelio organizacijų, kai teikiama pirmenybė techniniam palaikymui ir turimas pakankamas IT biudžetas tokio tipo projektams finansuoti. Kai kurie sutinka, kad verslo ir IT suderinamumas užtikrina tinkamą visos organizacijos valdymą [10][11]. Profesionalai teigia, kad pasiektas strateginis verslo ir IT suderinamumas suteikia organizacijoms reikšmingą konkurencinį pranašumą. Taip pat strateginis verslo ir IT suderinamumas turi teigiamos įtakos verslo rezultatams ir IT efektyvumui [11][12].

Šiandieninėse organizacijose egzistuoja gana didelis atotrūkis tarp verslo ir IT. Informacinių technologijų strategijos planavimas yra daugiapakopis procesas, o informacinių sistemų kūrimas priklauso nuo kiekvieno IS gyvavimo ciklo etapo įgyvendinimo. Pagrindiniai organizacijos verslo tikslai ir IS plėtros tikslai dažnai neturi tiesioginio ryšio. Todėl norint užtikrinti verslo ir IT suderinamumą, svarbu rasti sąsajas tarp šių dviejų pusių. Valdymo procesų įgyvendinimas suteikia aiškumo priimant sprendimus, taip pat įvertinama, kuriuose procesuose būtina naudoti IT ir kaip IT padeda juos optimizuoti. Organizacijų vadovai turi suprasti, kurie IT sprendimai svarbūs ir turi įtakos pagrindiniams verslo sprendimams, atitinkamai juos priimant turi dalyvauti tai suprantantys dalyviai [11][12][34]. Kitaip tariant,, verslo pusės dalyviai deleguoja verslo atvejus atitinkamai darbo grupei. Toks bendravimas dažnai sutvirtina abiejų pusių – verslo ir IT – dalyvių santykius, o jų bendradarbiavimas gali padėti nustatyti bendrus skirtingų vartotojų grupių poreikius. Dalijantis žiniomis, pagerėja bendravimas ir visi proceso dalyviai tampa labiau informuoti apie iššūkius, su kuriais susiduria IT, bandydamos patenkinti vartotojo reikalavimus [12][34].

Siekiant paskatinti komunikaciją ir supratimą, daugelyje organizacijų reglamentuojami su verslu susiję vaidmenys, nustatant atsakomybę už ryšių užmezgimą ir palaikymą tarp IT ir verslo pusių dalyvių. Tokiu būdu stengiamasi užtikrinti technologijų ir verslo vizijos pusiausvyrą, galimybę vienodai aiškiai priimti ir įvertinti verslo ir technologijų iššūkius. Gana dažnai vienai pusei aiškus ateities matymas ir supratimas, kaip pritaikyti naujas technologijas versle, ir tokios informacijos perteikimas visiškai nesuprantamas kitai pusei [12][34]. Svarbu, kalbant apie iššūkius, nevartoti vien techninių terminų. Būtina tinkamai paaiškinti, ko reikia verslui ir kaip bus patenkinti visų dalyvių poreikiai, o ne tik nušviesti tai, kaip pokyčiai padarys efektyvesnį IT pusės dalyvių darbą.

Motyvacija. Žiniomis grįsta IS inžinerija pasiūlė sistemos modeliavimo ir sprendimų priėmimo metodus ir priemones, kurios padeda sukurti tikslesnius ir išsamesnius, dalykinę sritį atitinkančius projektus [18][31][29]. IS projektų kūrėjui ir (ar) programuotojui suteikiama galimybė naudotis ne tik projektų žiniomis, kurios saugomos tradicinių CASE priemonių saugykloje, bet ir žinių saugykla, kurioje saugomos formaliais kriterijais patikrintos dalykinės srities žinios [21][23]. Tam yra sukurta daug standartų ir verslo modeliavimo metodikų [30][31][47].

UML yra vienas iš labiausiai paplitusių programinės įrangos specifikavimo standartų. Tai universali IS modeliavimo kalba, kurią taiko daugelis metodologų ir kuri vartojama populiariausiose modeliavimo priemonėse, pavyzdžiui, „Enterprise Architect“, „System Architect“, „MagicDraw“ ir kitose [13][15]. Darbe pateikiamas UML diagramų generavimo iš veiklos modelio metodas įgyvendina žiniomis grįstą IS kūrimo ciklo projektavimo etapą [26][27][28].

Tyrimo objektas – UML dinaminių modelių generavimo iš veiklos modelio (EM) procesas, pagrįstas žiniomis grįsta IS inžinerija, integruojant modeliais grindžiamą architektūrą (MDA), ISO standartus ir MOF.

Tyrimo apimamos sritys:

- Žiniomis grįstos informacinės sistemos inžinerija.
- ISO standartai, kurie teikia sprendimus ir užtikrina naudą beveik visiems veiklos sektoriams. Šie standartai – tai formalūs susitarimai, kuriuose pateikiamos techninės specifikacijos ir (ar) kiti tikslūs kriterijai, nuosekliai naudojami kaip taisyklės, gairės ar charakteristikų apibrėžimai, siekiant užtikrinti, kad medžiagos, produktai, procesai ir paslaugos atitiktų savo paskirtį [38][44].
- MOF. Tai Object Management Group (OMG) standartas, skirtas modeliais grįstai inžinerijai. Jo tikslas – pateikti esybių tipų sistemą ir sąsajų rinkinį, per kurį tuos tipus galima kurti ir jais

manipuliuoti. MOF suteikia tik kalbos priemones duomenų struktūrai arba abstrakčiai sintaksei apibrėžti [74].

- MDA, arba modeliais grįsta architektūra. Ttai OMG programinės įrangos projektavimo, kūrimo ir diegimo metodas. MDA pateikia gaires, kaip struktūrizuoti programinės įrangos specifikacijas, kurios išreiškiamos kaip modeliai.

Darbo problema. Tradiciniai IS inžinerijos etapai – nuo naudotojo reikalavimų specifikavimo iki projekto kūrimo – yra empiriškai pagrįsti, o tradicinės CASE priemonės įgyvendina empirinį UML grįstą IS projektų kūrimo procesą. Vadinasi, IS projekto kūrimas daugiausia grindžiamas analitikų žiniomis ir patirtimi [26][27][57][60].

Vienas iš daugelio kompiuterizuotų IS inžinerijos metodų trūkumų yra tas, kad IS projektavimo modeliai sukuriami tik iš dalies, nes projektavimo etape projektuotojas formuoja projektavimo modelius, remdamasis asmenine ar analitiko patirtimi, užuot taikęs žinių, kurios saugomos veiklos modelyje, generavimo principus. Šias problemas galima išspręsti integruojant veiklos modelį, kaip pagrindinę dalykinių žinių saugyklą, į IS projektavimo procesą. Žiniomis grindžiama posistemė, kaip CASE įrankio komponentas su įtrauktais veiklos metamodeliu ir veiklos modeliu, gali būti tokios problemos sprendimas.

Tikslas ir uždaviniai. Šio darbo tikslas – išplėsti galimybes generuoti sistemos projektavimo etapo UML dinaminius modelius iš veiklos modelio, naudojant transformacijos algoritmus.

Tyrimo uždaviniai:

1. Išanalizuoti įvairius UML dinaminių modelių naudojimo IS inžinerijoje atvejus.
2. Sukurti UML dinaminių modelių iš veiklos modelio (EM) transformacijos algoritmus žiniomis grįstoje IS inžinerijoje.
3. Eksperimentiškai patikrinti transformacijų algoritmus realių dalykinių sričių pavyzdžiais.
4. Įvertinti žiniomis grįsto UML dinaminių modelių generavimo metodo privalumus.

Tyrimo metodika. Dabartinė situacija ir tyrimo aktualumas buvo įvertinti analizuojant mokslinę literatūrą, susijusią su IS inžinerija, IS gyvavimo ciklo etapais, veiklos modeliavimu, UML, ISO standartais, MOF architektūra, modeliais grįstu kūrimu ir kitomis sritimis. Tyrimas atliktas taikant Design Science Research metodiką [31]. Siekiant užtikrinti efektyvesnį ir labiau kvalifikuotą generavimo procesą bei mažesnį klaidų ir paklaidų skaičių galutiniame IS kūrimo etape, buvo sukurtas UML dinaminių modelių generavimo iš veiklos modelio metodas, taikant transformacijos algoritmus. Siekiant įvertinti naujojo metodo taikymo galimybes ir įrodyti jo tinkamumą, pateikiami keli atskiri pavyzdžiai.

Ginamieji teiginiai:

1. Veiklos modelis (EM), atitinkantis veiklos metamodelį (EMM), yra pakankamas dalykinės srities žinių šaltinis generuoti UML specifikacijoje apibrėžtus dinaminius modelius.
2. Sukurti transformacijos algoritmai užtikrina UML dinaminių modelių generavimą iš veiklos modelio (EM) skirtingose dalykinėse srityse.

Pagrindinis šio darbo rezultatas – organizacijos žiniomis grįstas UML dinaminių modelių generavimo iš veiklos modelio metodas. Žiniomis grįstas generavimo metodas sujungia pagrindinius žiniomis grindžiamų metodų, ISO standartų, MOF ir MDA architektūrų principus. Šis metodas suteikia galimybę kurti pažangias programinės įrangos kūrimo technikas. Atliekant tyrimą buvo išanalizuotas esamas veiklos metamodelis (EMM) [23][32][57] ir panaudotas įvairių dalykinių sričių UML dinaminiams modeliams generuoti. Siekiant įgyvendinti UML dinaminių modelių generavimą iš veiklos modelio, buvo sukurta naujų modelių transformacijos algoritmų.

Pagrindinė praktinė šio darbo reikšmė – tai žingsnis link to, kad modeliavimo procesas taptų suprantamesnis ir naudingesnis visiems IS kūrimo ciklo dalyviams. Taikant pasiūlytą žiniomis grįstą generavimo metodą atsiranda galimybė naudoti papildomus modelių tinkamumo metodus, kuriuos apibrėžia veiklos metamodelis (EMM).

Rezultatų aprobavimas. Tyrimų rezultatai buvo pristatyti 14 tarptautinių ir 3 Lietuvos konferencijose, 6 tarptautiniuose seminaruose (stendiniai pranešimai) ir 1 doktorantūros simpoziume. 4 straipsniai paskelbti periodiniuose mokslo leidiniuose, žurnaluose, nurodytuose Clarivate Analytics Web of Science duomenų bazės leidiniuose su citavimo indeksu. 9 straipsniai paskelbti konferencijų pranešimuose, referuojamuose Clarivate Analytics Web of Science duomenų bazėje. 9 straipsniai buvo paskelbti recenzuojamuose konferencijų leidiniuose. 2 straipsniai kaip knygos skyriai publikuoti Springer knygų serijose, kurios priskiriamos kitiems recenzuojamiems periodiniams, tęstiniams ar vienkartiniams mokslo leidiniams. Pabaigoje pateikiamas išsamus publikacijų sąrašas.

Disertacijos apimtis ir struktūra. Pirmąją dalį sudaro mokslinės literatūros analizė, kurioje daugiausia dėmesio skiriama verslo ir IT suderinamumo iššūkiams, ISO standartams, MOF, MDA architektūrų naudojimui, tradicinei ir žiniomis grindžiamai IS inžinerijai, veiklos modeliavimo sprendimams ir IS kūrimo proceso etapams. Antrojoje dalyje pristatomas žiniomis grįstas generavimo metodas. Šioje dalyje pateikiamas išsamus metodo aprašymas ir sudėtis, aprašant UML dinaminių modelių kompozicijas, transformacijos taisykles ir pateikiant įvairių transformacijos iš veiklos modelio algoritmų. Trečiojoje dalyje pateikiami atskiri ir du kombinuoti UML dinaminių modelių generavimo pavyzdžiai, taikant pasiūlytus transformacijos algoritmus. Šioje dalyje patikrinama ir įrodoma, kad veiklos modelis (EM) tinka UML dinaminiams modeliams generuoti ir gali būti naudojamas taikant sukurtus transformacijos algoritmus.

1. SUSIJUSIŲ DARBŲ ANALIZĖ

Pirmąjį skyrių sudaro mokslinės literatūros analizė, kurioje daugiausia dėmesio skiriama verslo ir IT suderinamumo iššūkiams, tradicinei ir žiniomis grįstai IS inžinerijai, veiklos modeliavimo sprendimams ir IS kūrimo procesų etapams.

Labiausiai paplitusį ir priimtiną konceptualų suderinamumo modelį pasiūlė Henderson ir Venkatraman [34]. Šis teorinis modelis dar vadinamas strateginio suderinamumo modeliu (SAM). Strateginio suderinamumo koncepcija grindžiama dviem dimensijomis: pirma – strateginiu atitikimu tarp išorinių veiksnių, nukreiptų į verslo aplinką, ir vidinių veiksnių, nukreiptų į infrastruktūrą ir procesus, antra – funkcine verslo ir IT integracija. Strateginis atitikimas reiškia vidaus ir išorės sričių suderinamumą. Funkcinė integracija – tai IT strategijos įtraukimas į verslo strategiją, ypač vidinių IT strategijų integravimas į vidines organizacijos procedūras ir strategijas. Iš viso modelyje apibrėžiamos keturios sritys, kurios turi būti suderintos, kad būtų pasiektas suderinamumas [11][34][76][78]. Modelį sudaro keturios sritys: verslo strategijos domenas, verslo infrastruktūros domenas, IT strategijos domenas, IT infrastruktūros domenas, ir ryšiai tarp jų [34][67].

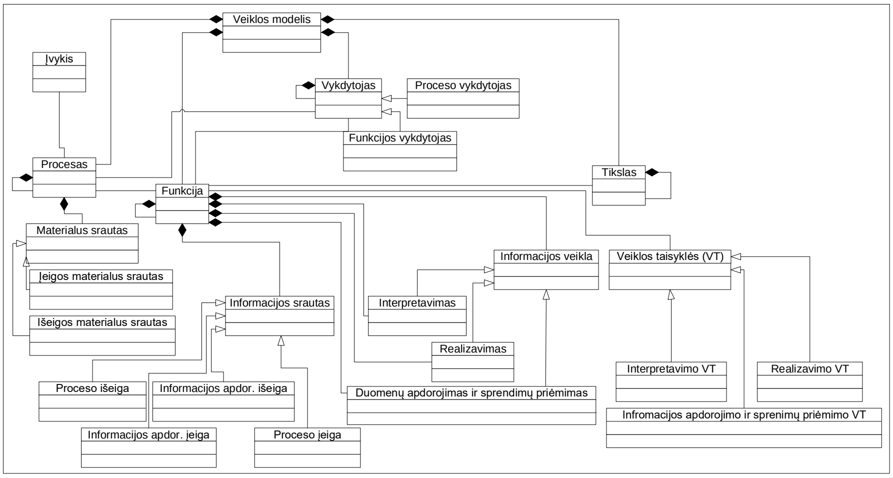
Žiniomis grįsta CASE įrankių posistemė verslo ir IT valdyme gali būti naudojama kaip pagrindinis tam tikrų sistemų šaltinis. Šiose sistemose nurodomos visos svarbios organizacijos struktūros, įskaitant verslą, taikomąsias technologijas, duomenis ir jų ryšius, reikalingus verslui vykdyti [22][23][24]. Verslo ir IT derinimo procesams reikalingi tam tikri specifiniai duomenys, kurie naudojami žinių posistemėje. Verslo strategijos sritis žinių bazei pateikia verslo tikslus. Verslo tikslus aprašo verslo vadovai ir juos surenka iš verslo aplinkos, o IT tikslus aprašo IT vadovai. Verslo infrastruktūros domenas pateikia verslo taisykles, apribojimus, procesus, funkcijas ir kitus susijusius duomenis, o IT infrastruktūros domenas pateikia informaciją apie IT infrastruktūrą, kurioje aprašoma esama programinė, techninė įranga. Visa ši informacija saugoma žinių bazėje ir gali būti naudojama taikant veiklos modelį, kuris patvirtinamas pagal veiklos metamodelį [63A][64A][65A].

Kitas galimas organizacinis sprendimas įmonėse – tam tikros sistemos pritaikymas pagal poreikius. „Enterprise Architecture“ sistema yra dvimatė klasifikavimo schema. MDG (angl. Model Driven Generation) Technology sukurtas Zachman Framework metodas dar labiau išplečia „Enterprise Architect“ diagramų (modelių) rinkinį [53][54][80][112]. Kiekvienam Zachman sistemos elementui gali būti naudojamos skirtingų notacijų diagramos arba modeliai. Gana daug sistemos elementų galima padengti UML diagramomis. Tai dar kartą patvirtina, kad UML diagramų vaidmuo IS inžinerijoje yra svarbus [53][82][112].

Informacinių sistemų inžinerija – tai informacinės sistemos kūrimo procesas, kurio vykdymas įgyvendinamas per IS kūrimo gyvavimo ciklo etapus. Kiekviena IS kūrimo gyvavimo ciklo fazė anksčiau buvo įgyvendinama nepriklausomai nuo kitų, nes nebuvo kompiuterizuotos žinių saugyklos, į kurią būtų įtrauktos visos gyvavimo ciklo fazės. Analitikai ir projektuotojai taikė įprastinius arba į objektus orientuotus IS inžinerijos metodus [19][29]. IS projektai buvo įgyvendinami nekompiuterizuotai arba naudojant specialiai sukurtus kompiuterizuotus įrankius, skirtus konkrečioms vartotojų reikalavimų specifikacijoms spręsti [27][33]. Vėliau pradėti taikyti kompiuterizuoti IS kūrimo metodai. Gerinant IS kūrimą buvo tobulinamos kompiuterizuotos IS kūrimo priemonės – CASE sistemos, kurios apėmė dalį IS kūrimo gyvavimo ciklo etapų – projektavimą, dokumentavimą ir kodavimą. Analitikai ir projektuotojai naudojo CASE sistemos IS kūrimo priemones koncepcinio ir detaliojo projektavimo etapuose, o praktinio realizavimo etape programuotojas naudojo kodui generuoti skirtas priemones. Šiuo atveju kompiuterizuota sistema buvo grindžiama tradiciniais IS inžinerijos ir (arba) į objektus orientuotos IS inžinerijos metodais [28][59][61].

Veiklos metamodelis (EMM) yra formaliai apibrėžta veiklos modelio struktūra, kurią sudaro formalizuotas veiklos modelis, atitinkantis bendruosius valdymo teorijos principus. Veiklos modelis yra pagrindinis IS inžinerijos ir IS reinžinerijos procesams reikalingų žinių apie konkrečią dalykinę sritį šaltinis [23][26][27][28][58][70].

Veiklos metamodelis valdo veiklos modelio sudėtį. Veiklos modelyje saugomos tik IS kūrimo procesui reikalingos dalykinės srities žinios, kurios paskui naudojamos visuose IS kūrimo gyvavimo ciklo etapuose [26][27][57]. Veiklos modelis ir veiklos metamodelis yra pagrindiniai CASE įrankio žinių posistemės komponentai. Būtų tikslinga surinkti reikalingas žinias į konkrečios CASE priemonės žiniomis grindžiamą posistemę. Ši posistemė yra pagrindinis žinių, reikalingų projektavimo etapo modeliams (įskaitant UML diagramas) ir pirminio kodo generavimo procesui, šaltinis.



Šaltinis: sudaryta autorės pagal [26][27][50]

1 pav. Veiklos metamodelio klasių modelis

Veiklos modelio pagrindu modeliuojami organizacijoje vykstantys procesai, įvertinant organizacijos aplinkos poveikį pačiai organizacijai. Veiklos modeliavimas apima šias sudėtines dalis: funkcijas, elgseną, išteklius ir jų valdymą. IS inžinerijoje svarbu unifikuoti veiklos modeliavimą, nes kiti funkcionuojantys veiklos modeliai neatitinka veiklos modeliavimo reikalavimų IS inžinerijos srityje [21]. Todėl IS inžinerijoje yra siekiama naudoti formalizuotą žinių apie veiklos valdymą struktūrą – veiklos metamodelį. Veiklos metamodelyje integruoti skirtingi veiklos modeliavimo požiūriai ir metodikos. Išskiriami du esminiai veiklos metamodelio elementai: funkcija ir procesas. Procesas apibūdina materialiąją, o funkcija – informacinę veiklos pusę.

Žiniomis grindžiamos CASE sistemos pagrindiniai komponentai padeda organizuoti žinias, naudojant žinių posistemės žinių bazę, kurios esminiai elementai yra veiklos metamodelio specifikacija ir tam tikros dalykinės srities veiklos modelis [23][65A][71][79].

Tradicinė IS inžinerija ir žiniomis grindžiama IS inžinerija turi kokybinių skirtumų. Tradicinės IS inžinerijos atveju tai empirinė IS inžinerija, kai individualus naudotojas neapima visos organizacijos procesų. Žiniomis grindžiama IS inžinerija apima visos organizacijos specifikacijas (veiklos modelis), nes joje nurodomos esminės organizacijos charakteristikos. Formalizuotas veiklos modelis grindžiamas teoriniais organizacijos veiklos valdymo principais [22][25].

Tradicinės IS inžinerijos atveju informacinė sistema kuriama empiriškai, taikant tradicinę ir (ar) kompiuterizuotą IS inžineriją. Žiniomis grindžiamoje kompiuterizuotoje IS inžinerijoje informacinė sistema kuriama naudojant veiklos žinių saugyklą, kurioje saugomos reikalingos ir pakankamos kompiuterizuotos dalykinės srities žinios [23][25][26][57].

Žiniomis grindžiamoje kompiuterizuotoje IS inžinerijoje visi projektiniai modeliai gali būti generuojami interaktyviai, generavimui naudojant transformavimo algoritmus, jei reikiamos žinios sukauptos žinių saugykloje. Trūkstamam žinių įkėlimui užtikrinti būtinas minimalus analitiko ir projektuotojo dalyvavimas. Žinių išsamumas tikrinamas; tai daroma siekiant užtikrinti automatiškai generuojamų projektinių modelių ir programinio kodo kokybę žinių rinkimo į žinių saugyklą etape [26][51][57][73][85].

2. MODELIŲ TRANSFORMAVIMO IŠ EM Į UML METODAS

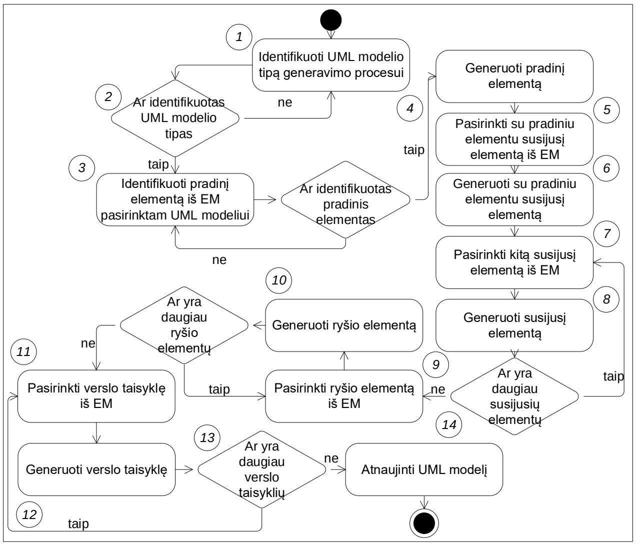
Antrajame skyriuje pateikiamas žiniomis grindžiamas generavimo metodas. Šioje dalyje pateikiamas išsamus pateikto metodo aprašymas ir sudėtis, aprašant UML dinaminių modelių sudėtį, transformacijos taisykles ir pateikiant transformacijos algoritmus generavimui iš veiklos modelio [5][24][81].

UML modeliavimo kalba, skirta programinės įrangos projektavimo sprendimams aprašyti. Tai modeliavimo kalba, apibrėžianti įvairių programinės įrangos architektūros modeliavimo aspektų grafinę notaciją [77][92]. Nuo (1997 m. gruodžio) OMG (angl. Object Management Group) paskelbė UML kaip standartą, kuris daro didžiulę įtaką IS projektavimui. Pirmaisiais šios kalbos gyvavimo metais ji buvo vartojama informacinėms sistemoms modeliuoti, tačiau ji tinka ir verslo procesams modeliuoti, tad sugebėjo stipriai įsitvirtinti tarp verslo analitikų. UML geba apibrėžti atitinkamus verslo struktūrinius ir elgsenos taisyklių aspektus. Vartojant UML kalbą galima užtikrinti dokumentacijos stabilumą ir palengvinti projektuotojų ir verslo kūrėjų bendravimą [75][84][91].

UML diagramų ir veiklos modelio sąveika įgyvendinama naudojant transformacijos algoritmus. Veiklos modelyje saugomi verslo elementai gali būti generuojami kaip UML modelių elementai naudojant transformacijos algoritmus. Žinių posistemės ir veiklos modelio naudojimo integracija leidžia generuoti formaliųjų kriterijų patikrintas žinias į UML modelius. Veiklos modelio naudojimas ir žiniomis grindžiamo IS kūrimo proceso automatizavimas taupo projektuotojų ir užsakovų darbo valandas.

UML dinaminiai modeliai gali būti sugeneruoti iš veiklos modelio naudojant transformacijos algoritmus. Pirma, reikia pasirinkti tam tikrą UML modelį generavimo procesui, tada iš veiklos modelio identifikuojamas pradinis – pagrindinis šio UML modelio – elementas. Antra, pagal pradinį elementą turi būti parinkti visi susiję elementai ir visi jie turi būti susieti vadovaujantis verslo taisyklėse esančiais apribojimais, būtinais anksčiau pasirinktam UML modelio tipui generuoti [64A][98A].

Pateiktas transformavimo algoritmas yra aukščiausio lygio algoritmas, skirtas veiklos metamodeliu grindžiamam UML modelių generavimo procesui. Pagrindiniai generavimo proceso etapai yra šie: UML modelio identifikavimas ir parinkimas generavimo procesui, pasirinkto UML modelio pradinių elementų identifikavimas ir visų susijusių elementų parinkimas, veiklos modelio elementų atvaizdavimas į UML modelio elementus ir pasirinkto UML modelio generavimas [101A][103A].


1
Identifikuoti UML modelio tipą generavimo procesui
2
Ar identifikuotas UML modelio tipas
ne
taip
3
Identifikuoti pradinį elementą iš EM pasirinktam UML modeliui
Ar identifikuotas pradinis elementas
ne
taip
4
Generuoti pradinį elementą
5
Pasirinkti su pradiniu elementu susijusį elementą iš EM
6
Generuoti su pradiniu elementu susijusį elementą
7
Pasirinkti kitą susijusį elementą iš EM
8
Generuoti susijusį elementą
Ar yra daugiau susijusių elementų
taip
9
ne
14
Pasirinkti ryšio elementą iš EM
10
Generuoti ryšio elementą
Ar yra daugiau ryšio elementų
taip
ne
11
Pasirinkti verslo taisyklę iš EM
Generuoti verslo taisyklę
12
taip
13
Ar yra daugiau verslo taisyklių
ne
Atnaujinti UML modelį

Šaltinis: sudaryta autorės [91]

2 pav. Aukščiausio lygio UML modelių generavimo iš veiklos modelio transformavimo algoritmas

Toliau aprašomi skirtingų UML dinaminių modelių tipų transformavimo algoritmai.

UML taikomųjų uždavinių diagramos paprastai vadinamos dinaminiais modeliais, naudojamais veiksmų, kuriuos tam tikra sistema ar sistemos turėtų ar gali atlikti bendradarbiaudamos su vienu ar keliais išoriniais sistemos naudotojais. Kiekvienas taikomasis uždavinys turėtų suteikti tam tikrus matomus ir vertingus rezultatus sistemos veikėjams ar kitiems dalyviams [101A][102A][103A][104A].

UML taikomųjų uždavinių modelio generavimo iš EM procese pradinis elementas yra veikėjas, sugeneravus veikėjo elementą, pasirenkamas taikomojo uždavinio elementas, tada pasirenkami įtraukimo, išplėtimo ir susiejimo ryšiai. UML taikomųjų uždavinių modelio generavimo iš veiklos modelio procese pradinis elementas yra veikėjas, arba subjektas, sugeneravus šį elementą, pasirenkamas veiklos modelio elementas – procesas arba funkcija, ir generuojamas taikomojo uždavinio elementas. Sugeneruoti šių dviejų tipų elementai turi būti susiejami tarpusavyje tam tikro tipo ryšiais: asociacija, išplėtimu arba įtraukimu, kurie apibrėžiami pagal veiklos modelio elemento verslo taisyklę. Sukūrus visus šiuos elementus, atnaujinamas veikėjo, arba subjekto, elementas ir patikrinama, ar veiklos modelyje liko daugiau veikėjo elementų [103A][104A].

UML veiklos diagrama – tai UML dinaminis modelis, kuriame vaizduojamas valdymo arba objektų srautas, akcentuojant srauto seką ir sąlygas. Veiksmai arba veiklos, kurias koordinuoja veiklos modeliai, gali būti inicijuojami dėl to, kad kiti veiksmai arba veiklos baigiami vykdyti, kadangi kiti objektai ir duomenys tampa prieinami arba dėl to, kad įvyksta tam tikrų srautui nepriklausančių įvykių [94A][101A][102A][103A][104A].

Generuojant UML veiklos modelį iš veiklos modelio, pradinis elementas yra veikėjas arba veikėjo skiltis, sugeneravus šį elementą pasirenkamas veiklos modelio elementas: procesas arba funkcija, ir generuojamas veiklos elementas. Sugeneruoti šių dviejų tipų elementai turi būti susiejami tarpusavyje. Veiklos modeliuose objektų mazgai generuojami iš veiklos modelio žinių arba informacijos srauto elementų. Visi šie sugeneruoti elementai sujungiami per valdymo mazgus, kurie grindžiami veiklos modelio verslo taisyklių elementais. Vėliau visi šie elementai sukuriami, atnaujinamas veikėjo elementas ir patikrinama, ar veiklos modelyje liko daugiau veikėjo elementų [91][101A][102A][103A][104A].

UML būsenų diagrama naudojama elgsenai modeliuoti taikant baigtinių būsenų perėjimus. Būsenos gali būti naudojamos ne tik sistemos dalies elgsenai, bet ir sistemos dalies naudojimo protokolui išreikšti. [91][101A][102A][103A][104A].

UML būsenų modelio generavime iš veiklos modelio pradinis elementas yra procesas arba funkcija, taigi iš šių veiklos modelio elementų generuojamas būsenos elementas. Vėliau pagal šį elementą generuojamas antrasis susijęs elementas – paprasta būsena arba sudėtinė būsena, kuriama iš informacijos srauto. Be to, pirmieji du elementai susiejami tarpusavyje ir su pseudobūsenos elementu, generuojamu iš verslo taisyklės. Paskui atnaujinamas pradinis elementas ir patikrinama, ar veiklos modelyje liko daugiau proceso elementų [101A][102A][103A][104A].

Antrasis būsenų modelio tipas yra UML protokolo būsenos modelis, kuris apibrėžia naudojimo protokolą arba tam tikro klasifikatoriaus gyvavimo ciklą [14].

UML protokolo būsenų modelio generavime iš veiklos modelio pradinis elementas taip pat yra procesas arba funkcija, iš šių veiklos modelio elementų sukuriamas protokolo būsenos elementas. Sugeneravus šį elementą pasirenkamas informacijos srautas ir generuojamas protokolo būsenos elementas. Šie du elementai susiejami tarpusavyje ir su protokolo perėjimo elementu, generuojamus iš verslo taisyklės. Paskui atnaujinamas pradinis elementas ir patikrinama, ar veiklos modelyje liko daugiau proceso elementų [91][101A][102A][103A][104A].

UML sekų diagrama yra labiausiai paplitusi sąveikos modelių rūšis, joje daugiausia dėmesio skiriama pranešimų mainams tarp veikėjų, objektų (gyvavimo linijų). Sekų modelis parodo, kaip objektai sąveikauja su kitais objektais konkrečiame naudojimo scenarijuje [91][101A][102A][103A][104A].

Generuojant UML sekų modelį iš veiklos modelio pradinis elementas yra veikėjas-gyvavimo linija, sugeneravus šį elementoą pasirenkamas veiklos modelio elementas: procesas arba funkcija, ir pranešimo elementas. Sugeneruoti šių dviejų tipų elementai turi būti susiejami tarpusavyje. Sekos modelių vykdymo specifikacijoje iš veiklos modelio verslo taisyklių elementų generuojami kombinuotas fragmentas, sąveikos naudojimas, būsena ir pabaigos įvykis. Vėliau, kai visi šie elementai sugeneruojami, atnaujinamas veikėjo-gyvavimo linijos elementas ir patikrinama, ar veiklos modelyje liko daugiau veikėjo-gyvavimo linijos elementų.

UML komunikacijos diagrama (UML 1.x versijoje vadinama bendradarbiavimo diagrama) – tai UML sąveikos modelio tipas, kuris parodo objektų ir (arba) dalių (vaizduojamų kaip veikėjas-gyvavimo linija) sąveiką naudojant nuoseklius laisvos formos pranešimus [91][101A][102A][103A][104A].

Generuojant UML komunikacijos modelį iš veiklos modelio, pradinis elementas yra veikėjas-gyvavimo linija, o sugeneravus šį elementą pasirenkamas veiklos modelio elementas: procesas arba funkcija, ir sukuriamas struktūros elementas. Sugeneruoti šių dviejų tipų elementai paskui turi būti susiejami tarpusavyje. Sąveikos modeliuose pranešimų elementai generuojami iš informacijos srauto elemento. Kai visi šie elementai sugeneruojami, atnaujinamas veikėjo-gyvavimo linijos elementas ir patikrinama, ar veiklos modelyje liko daugiau veikėjo-gyvavimo linijos elementų [101A][102A][103A][104A].

UML laiko diagrama – tai sąveikos modelis, kuriame vaizduojama sąveika, kai pagrindinė modelio taikymo sritis yra laiko pagrindimas. Laiko modelyje dėmesys sutelkiamas į veikėjų-gyvavimo linijų terminų vidinę ir tarpusavio kaitą pagal linijinę laiko ašį. Laiko modeliai apibrėžia tiek atskirų klasifikatorių, tiek klasifikatorių sąveikos elgseną, sutelkdami dėmesį į įvykių, sukeliančių modeliuojamų veikėjų-gyvavimo linijų terminų pokyčius, laiką [91][101A][102A][103A][104A].

UML laiko modelio generavimo iš veiklos modelio pradinis elementas yra veikėjas-gyvavimo linija, sugeneravus šį elementą pasirenkamas veiklos modelio elemento informacijos srautas ir generuojamas laiko juostos arba trukmės apribojimo elementas. Vėliau sugeneruoti šių dviejų tipų elementai turi būti susiejami tarpusavyje. Laiko modeliuose laiko apribojimo ir pabaigos atsiradimo elementai generuojami iš verslo taisyklės elemento. Vėliau visi šie elementai sugeneruojami, atnaujinamas veikėjo-gyvavimo linijos elementas ir patikrinama, ar veiklos modelyje liko daugiau veikėjo-gyvavimo linijos elementų [101A][102A][103A][104A].

UML sąveikos diagramoje sąveikos nustatomos naudojant veiklos modelių variantą taip, kad būtų galima apžvelgti valdymo srautą. Sąveikos modelyje dėmesys sutelkiamas į valdymo srauto apžvalgą, kur mazgai yra sąveikos arba sąveikos naudojimo būdai. Veikėjai-gyvavimo linijos ir pranešimai neatitinka šio apžvalgos lygio. UML sąveikos modelis koordinuoja veiklos ir sąveikos modelių elementus [91][101A][102A][103A][104A]:

- iš veiklos modelio: pradinis mazgas, srauto galutinis mazgas, veiklos galutinis mazgas, sprendimo mazgas, susijungimo mazgas, šakutės mazgas, prisijungimo mazgas;

- iš sąveikos modelių: sąveika, sąveikos naudojimas, trukmės apribojimas, laiko apribojimas.

Generuojant UML sąveikos modelį iš veiklos modelio pradinis elementas yra procesas arba funkcija, vadinasi, iš šių veiklos modelio elementų generuojamas struktūros elementas. Visi kiti elementai: trukmės apribojimas, laiko apribojimas, sąveikos naudojimo ir valdymo mazgai, yra susiję su pradiniu struktūros elementu ir priklauso nuo veiklos modelio verslo taisyklės elemento.

3. METODO PRISTATYMAS PAVYZDŽIAIS

Trečiajame skyriuje pateikiama atskirų UML modelių ir vieno jungtinio UML modelių generavimo pavyzdžių santrauka ir vienas detalus UML dinaminių modelių generavimo pavyzdys naudojant transformacijos algoritmus. Šiame skyriuje patikrinama ir įrodoma, kad veiklos modelis (EM) tinka UML dinaminiams modeliams generuoti ir gali būti taikomas naudojant sukurtus transformacijos algoritmus.

Įprastai informacinių sistemų projektavimo ir kūrimo metodai nurodo sistemų inžinerijos veiksmų seką, t. y. kaip, kokia tvarka ir kokius UML modelius naudoti projektavimo procese ir kaip šį procesą įgyvendinti [44][58][59][61][63A]. UML modelių ir veiklos modelio sąveika įgyvendinama taikant transformacijos algoritmus. [64A][99A][100A].

Veiklos modelis, kaip organizacijos žinių saugykla, leidžia generuoti UML modelius per transformacijos algoritmus. Veiklos metamodelis apima esminius verslo modeliavimo metodikų ir technikų elementus, kurie užtikrina tinkamą UML modelių generavimo procesą [63A][64A].

Atskirų dalykinių sričių UML modelių generavimo pavyzdžiai

Šiame skyrelyje pateikiami atskiri UML dinaminių modelių generavimo pavyzdžiai:

- UML taikomųjų uždavinių modelio generavimo atveju pateikiamas veiklos modelio ir jame saugomų elementų bei UML taikomųjų uždavinių modelio generuojamų elementų sąryšis.
- UML informacijos srautų modelio generavimas iliustruojamas konkrečiu suplanuoto augintinio ultragarsinio tyrimo eigos per vizitą veterinarijos klinikoje pavyzdžiu. Pažingsniui aprašoma UML

informacijos srautų modelio generavimo pagal ansktesniame skyriuje aprašytą transformacijos algoritmą eiga.

- UML veiklos modelio generavimas aiškinamas parodant veiklos modelyje saugomų elementų ir UML veiklos modelio generuojamų elementų sąryšį. Šis sąryšis iliustruojamas prekės užsakymo pavyzdžiu: užsakymo užklausos aktyvinimas yra kliento teikiamas pradinis elementas , o užsakymas yra veiklos įvesties parametras. Priėmus užsakymą ir užpildžius visą reikiamą informaciją, priimamas mokėjimas ir užsakymas išsiunčiamas, šis veiklos srautas leidžia išsiųsti užsakymą prieš išsiunčiant sąskaitą faktūrą arba patvirtinant mokėjimą.
- UML sekų modelio generavimo atveju, naudojant Malcev algebrą ir grafiškai atvaizuojant, pateikiamas veiklos modelio ir jame saugomų elementų bei UML sekų modelio generuojamų elementų sąryšis..
- UML laiko modelio generavimas pateikiamas grafiškai atvaizuojant veiklos modelyje saugomų elmentų ir UML laiko modelio generuojamų elementų sąryšį bei iliustruojant projekto gyvavimo ciklo pavyzdžiu, kuriame pateikiamos projekto gyvavimo ciklo fazės ir jų trukmė. Iš veiklos modelio galima sugeneruoti šiuos elementus: gyvavimo linija – projektas, būsena arba būklė – projekto fazės, terminai, trukmė, laiko apribojimai ir pabaigos įvykis laiko juostos pabaigoje.
- UML sąveikos modelio generavimas pateikiamas per paslaugos užsakymo internetu pavyzdį, iliustruojant, kurie veiklos elementai per transformacijos algoritmą sugeneruojami į konkretaus UML sąveikos modelio elementus.

Bilieto pirkimo proceso UML modelių generavimo pavyzdžiai

Šiame skyrelyje išsamiai paaiškinamas bilietų pirkimo procesas ir tai, kaip šį procesą galima suprojektuoti naudojant žiniomis grindžiamą veiklos modelį, kuriame saugomos visos su anksčiau minėtu pavyzdžiu susijusios žinios. Taip pat paaiškinama, kokios žinios taikomos generuojant konkrečius UML modelius naudojant tam tikrus transformacijos algoritmus, sukurtus kiekvienam UML modelio generavimo procesui [101A][103A][104A][108A].

Bilietų pirkimo procesas gali atrodyti labai paprastas, tačiau jei informacinių sistemų projektavimo etape šis procesas būtų analizuojamas įvairiais aspektais iš įvairių perspektyvų, jei šis procesas būtų projektuojamas ir kuriamas taip, kad atliktų visas įmanomas funkcijas, tai pareikalautų daug laiko ir analitiko, dizainerio bei kitų proceso dalyvių pastangų [108A].

IS gyvavimo ciklo projektavimo etape turi būti įvertintos visos detalės. Šios detalės, šios žinios saugomos anksčiau aprašytame veiklos modelyje ir jau yra patikrintos bei patvirtintos analitiko [108A].

Šiame skyrelyje aprašomi keturių tipų UML modeliai, sugeneruoti iš veiklos modelio: UML taikomųjų uždavinių, UML sekų, UML būsenų ir UML veiklos modeliai.

Veiklos modelyje saugoma visa informacija, susijusi su dalyviais, jų funkcijomis ir ryšiais tarp šių funkcijų, reikalingų UML taikomųjų uždavinių modeliui generuoti. Yra trys veikėjai : klientas, vadybininkas ir bilietų sistema. Bilietų sistema kaip veikėjas yra susijęs su visomis septyniomis funkcijomis (taikomaisiais uždaviniais): bilietų užsakymas apima bilieto kainos apmokėjimą ir formos spausdinimą, taip pat bilieto atšaukimą ir mokėjimo grąžinimą. Klientas kaip veikėjas yra susijęs su visomis funkcijomis, išskyrus mokėjimo grąžinimą, nes tai yra bilietų sistemos funkcija. Vadybininkas kaip veikėjas yra susijęs tik su dviem funkcijomis (taikomaisiais uždaviniais): formos spausdinimo ir bilieto atšaukimo [108A].

Veiklos modelyje taip pat saugoma visa su dalyviais ir jų bendradarbiavimu susijusi informacija, reikalinga UML sekų modeliui generuoti. Yra trys veikėjai – proceso dalyviai, kurie UML sekų modelyje vadinami veikėjais-gyvavimo linijomis: asmuo – klientas, subjektas – bilietų sistema, objektas – bilietas. Bilietas turi vieną vykdymo specifikaciją, gauna vieną pranešimą su informacija ir išsiunčia vieną sukurto bilieto pranešimą; bilietų sistema turi tris vykdymo specifikacijas, viena skirta patvirtinimui, kai klientas prisijungia; antroji skirta formai sukurti, o trečioji – bilietui sukurti; visos jos susijusios su pranešimais iš kliento. Klientas prisijungia, užsako formą, pateikia duomenis, atspausdina bilietą. Klientas išsiunčia keturis pranešimus ir gauna du: prisijungimo patvirtinimo ir visų šio konkretaus proceso užklausų patvirtinimo [108A].

Veiklos modelyje saugoma visa informacija, susijusi su procesais, funkcijomis ir jų būsenomis, kurių reikia UML būsenų modeliui generuoti. Šis modelis pateikiamas iš kliento perspektyvos. Yra keturi informacijos srautai – sudėtinės kliento esybės būsenos: patvirtinimas, prieinamumo patikrinimas, bilietų užsakymas ir spausdinimas. Esybių būsenas keičia elgsenos būsenos: įvesti prisijungimo duomenis; įvesti autobuso duomenis; įvesti savo duomenis; užsakymas sėkmingas; atsijungimas.

Galiausiai veiklos modelyje saugoma visa informacija, susijusi su dalyviais, jų veikla ir UML veiklos modeliui generuoti reikalingais ryšiais tarp šių funkcijų. Yra viena verslo taisyklė – valdymo mazgas, susijęs su proceso pradžia, pradiniu mazgu. Šiuo atveju yra tik vienas veikėjas – klientas. Prieš sprendimo mazgą yra dvi kliento veiklos: autobusų paieška ir bilietų prieinamumo tikrinimas. Jei laisvų bilietų nėra, procesas baigiamas nesėkmingai vienu iš veiklos pabaigos mazgų. Jei laisvų bilietų yra, klientas rezervuoja bilietus, užpildo duomenis, pateikia duomenis, atlieka mokėjimą ir atspausdina bilietą. Šis procesas baigiamas sėkmingai antruoju veiklos pabaigos mazgu [108A].

Visų keturių tipų UML dinaminiai modeliai: taikomųjų uždavinių, sekų, būsenų ir veiklos, vieno bilieto pirkimo proceso pavyzdyje yra generuojami iš veiklos modelio. Šie keturi UML modeliai apibrėžia tą patį pavyzdį, tačiau iš skirtingų perspektyvų, parodydami skirtingas dalyvių veiklas, būsenas ir taikomuosius uždavinius. Pavyzdys parodo, kad žiniomis grindžiamas veiklos modelis yra pakankama duomenų, reikalingų UML modeliams generuoti, saugykla.

Ligoninės informacijos valdymo proceso UML modelių generavimo pavyzdžiai

Pateikiamo pavyzdžio objektas – ligoninės informacijos valdymo proceso sistema, skirta pacientų srautams valdyti. Šioje įstaigoje gydytojas vienu metu susijęs tik su vienu specializuotu ligoninės skyriumi (kardiologijos, pediatrijos ir kt.). Kiekvienam gydytojui nustatytas vizitų laikas ir savaitės diena [109A].

Registratūroje įvedami paciento duomenys ir priimami reikiami apmokėjimai. Pacientas stebimas pagal automatiškai sugeneruojamą identifikacinį numerį [109A].

Paprastai pacientas pas gydytojus patenka dviem galimais būdais: tiesiogiai pasirinkdamas vizitą pas gydytoją arba patekdamas į ligoninę per priimamąjį [109A].

Gydytojas gali paskirti tyrimus pagal aprašytą paciento būklę. Pacientas apsilanko laboratorijoje, kad būtų atlikti gydytojo paskirti tyrimai. Pacientui pateikiamos tyrimų ataskaitos. Su tyrimais susiję apmokėjimai atliekami registratūroje. Remdamasis tyrimų rezultatais, gydytojas skiria pacientui vaistų ar papildomų tyrimų, jei jų reikia, arba skiria gydymą ligoninėje [109A].

Jei yra galimybė, pacientas guldomas į tam tikro skyriaus palatą pagal gydytojo paskyrimą. Skyriuje palatų skaičius ribotas, o jei laisvų nėra, pacientas paguldomas į kitą skyrių [109A].

Jei reikia,, pacientas operuojamas numatytą dieną paskirtu laiku, kaip nusprendžia už operaciją atsakingas gydytojas [109A].

Pasibaigus gydymui, gydytojui rekomendavus, pacientas gali būti išrašytas, kai sumoka visus mokesčius registratūroje. Sumokėjus registratūra pacientui išduoda išrašymo kortelę [109A].

Visi konkrečios dalykinės srities, šiuo atveju ligoninės informacijos valdymo proceso, duomenys saugomi aukščiau aprašytame veiklos modelyje. Veiklos modelyje saugoma informacija jau yra patikrinta ir patvirtinta eksperto ir analitiko, todėl ją galima naudoti UML modeliams kurti [109A].

Generuojami UML taikomųjų uždavinių modelio elementai iš ligoninės informacijos valdymo proceso veiklos modelio. Veiklos modelyje saugoma visa informacija, susijusi su dalyviais, jų funkcijomis ir ryšiais tarp šių funkcijų. Yra keturi dalyviai: pacientas, gydytojas, registratorius ir laboratorijos asistentas; registratorius susijęs su penkiais taikomaisiais uždaviniais; laboratorijos asistentas – su vienu taikomuoju uždaviniu; gydytojas susijęs su trimis taikomaisiais uždaviniais, o pacientas – su septyniais taikomaisiais uždaviniais. Keturi taikomieji uždaviniai apima keletą papildomų taikomųjų uždavinių, iš viso šeši ryšiai [109A].

Pagal aukščiau aprašytą UML taikomųjų uždavinių modelį galima nustatyti bent penkis skirtingus UML veiklos modelius: pacientų registracija, palatos paskyrimas, medicininiai tyrimai, gydymo procesas ir išrašymas [109A].

Generuojami UML veiklos modelio elementai iš veiklos modelio ligoninės informacijos valdymo proceso pacientų registracijos dalies. Veikėjas pacientas – pirmasis UML veiklos modelio elementas – pradeda registracijos procesą: apsilanko registratūroje, pateikia asmens duomenis, veikėjas registratorius – antrasis UML veiklos modelio elementas – įveda paciento duomenis ir nurodo paciento identifikacinį numerį. Paskutinė veikla, mokesčio sumokėjimas, susijusi su pirmuoju veikėju: pacientas sumoka mokestį, ir registracijos procesas baigiasi [109A].

Generuojami UML veiklos modelio elementai iš veiklos modelio ligoninės informacijos valdymo proceso palatos priskyrimo dalies. Veikėjas registratorius – pirmasis UML veiklos modelio elementas – pradeda palatos paskyrimo procesą: veikėjas pacientas – antrasis elementas – pateikia naują palatos paskyrimo datą, paskutiniai veiksmai yra susiję su pirmuoju elementu, registratorius parengia informaciją pacientui ir atnaujina informaciją registratūroje, ir procesas baigiasi [109A].

Generuojami UML veiklos modelio elementai iš veiklos modelio ligoninės informacijos valdymo proceso medicininių tyrimų dalies. Veikėjas pacientas – pirmasis UML veiklos modelio elementas – pradeda medicininių tyrimų procesą: apsilanko laboratorijoje ir po užklausos pateikia mėginį, veikėjas laboratorija – antrasis elementas – patikrina gydytojo receptą, užklausia dėl mėginio, atlieka tyrimą, suformuoja mokėjimo nurodymą ir parengia ataskaitą gydytojui; veikėjas registratorius – trečiasis elementas – patvirtina paciento mokėjimą ir pateikia kvitą; pacientas atlieka mokėjimą, patvirtinus mokėjimą gauna kvitą, ir procesas baigiasi [109A].

Generuojami UML veiklos modelio elementai iš veiklos modelio ligoninės informacijos valdymo proceso gydymo proceso dalies. Veikėjas gydytojas – pirmasis UML veiklos modelio elementas – pradeda gydymo procesą: susitinka su pacientu, analizuoja pateiktas tyrimų ataskaitas, atsižvelgdamas į tyrimų rezultatus nusprendžia, ar išrašyti pacientą, ar tęsti gydymą. Gydytojas nusprendžia, ar reikia atlikti daugiau tyrimų, ar ne, paskiria gydymo metodą – medikamentus arba operacinę intervenciją, o kai veikėjas pacientas – antrasis UML veiklos modelio elementas – patvirtina, gydytojas atlieka operaciją, ir procesas baigiasi [109A].

Generuojami UML veiklos modelio elementai iš veiklos modelio ligoninės informacijos valdymo proceso išrašymo dalies. Veikėjas pacientas – pirmasis UML veiklos modelio elementas – pradeda išrašymo procesą: kreipiasi į gydytoją dėl rekomendacijos išrašyti, veikėjas registratorius – antrasis elementas – patikrina duomenis, suformuoja išrašymo kortelę, patikrina mokėjimo būklę, pacientui sumokėjus, pateikia išrašymo kortelę, ir procesas baigiasi [109A].

Pagal aukščiau aprašytus UML taikomųjų uždavinių ir UML veiklos modelius galima nustatyti bent tris skirtingus UML sekų modelius: pacientų priėmimas, tyrimai ir gydymas bei išrašymas [109A].

Generuojami UML sekų modelio elementai iš veiklos modelio ligoninės informacijos valdymo proceso pavyzdžio pacientų priėmimo dalies. Veiklos modelyje saugoma visa su dalyviais ir jų bendradarbiavimu susijusi informacija. Yra keturi veikėjai – proceso dalyviai, kurie UML sekų modelyje vadinami veikėjais-gyvavimo linijomis: asmenys – pacientas, registratorius, subjektas – duomenų bazė, objektas – palata. Pacientas užsiregistruoja ligoninėje, registratorius įveda surinktus duomenis, pacientas prašo palatos, registratorius patikrina, ar yra laisvų vietų, ir patvirtina arba paneigia, kad palata yra laisva [109A].

Generuojami UML sekų modelio elementai iš veiklos modelio ligoninės informacijos valdymo proceso pavyzdžio tyrimo ir gydymo dalies. Veiklos modelyje saugoma visa informacija, susijusi su veikėjais-gyvavimo linijomis ir jų bendradarbiavimu. Yra keturi veikėjai – proceso dalyviai, kurie UML sekų modelyje vadinami veikėjais-gyvavimo linijomis: asmenys – pacientas, registratorius, objektai – operacija, tyrimas. Gydytojas atlieka apžiūrą ir skiria vaistų, prireikus skiria tyrimą, pacientas pateikia mėginius ir gauna ataskaitas, gydytojas peržiūri ataskaitas ir skiria daugiau vaistų, jei reikia, – operaciją ar daugiau tyrimų ir operuoja [109A].

Generuojami UML sekų modelio elementai iš veiklos modelio ligoninės informacijos valdymo proceso pavyzdžio išrašymo dalies. Veiklos modelyje saugoma visa su veikėjais-gyvavimo linijomis ir jų bendradarbiavimu susijusi informacija. Yra penki veikėjai – proceso dalyviai, kurie UML sekų modelyje vadinami veikėjais-gyvavimo linijomis: asmenys – pacientas, gydytojas, subjektai – registratūra, duomenų bazė, objektas – palata. Gydytojas teikia išrašymo rekomendacijas, pacientas prašo išrašyti, registratūra tikrina su mokėjimais susijusią informaciją, teiraujasi dėl mokėjimo, pacientas atlieka mokėjimą, registratūra atnaujina finansinę informaciją duomenų bazėje, pateikia kvitą, taip pat atnaujina išrašymo informaciją ir su palata susijusią informaciją, galiausiai pateikia išrašymo kortelę [109A].

Pagal aukščiau aprašytus UML dinaminius modelius galima išskirti bent tris skirtingus UML būsenų modelius, apibūdinančius būsenas: pacientas, gydytojas ir palata [109A].

Generuojami UML būsenų modelio elementai iš veiklos modelio ligoninės informacijos valdymo proceso pacientų dalies. Veiklos modelyje saugoma visa su procesais, funkcijomis ir jų būsenomis susijusi informacija. Šis modelis pateikiamas iš paciento perspektyvos. Modelyje pateikiami šie elementai: pradinė būsena, kuria pradedamas procesas, pirmoji būsena – pacientas užregistruotas, jo būsena keičiasi apsilankius gydytojui – pacientas gauna gydymą, papildomai apsilankęs gydytojas pataria išrašymo procedūrą, ir paciento būsena vėl keičiasi, pacientas išrašomas, procesas baigiasi galutine būsena [109A].

Generuojami UML būsenų modelio elementai iš veiklos modelio ligoninės informacijos valdymo proceso gydytojo dalies. Veiklos modelyje saugoma visa su procesais, funkcijomis ir jų būsenomis susijusi informacija. Šis modelis pateikiamas iš gydytojo perspektyvos. Modelyje pateikiami šie elementai: pradinė būsena, nuo kurios prasideda procesas, pirmoji būsena gydytojas užsiregistravo, ji pasikeičia pacientui užsiregistravus vizitui: gydytojas skiria gydymą, patikrina gydymo rezultatus; pacientą išrašius gydytojo būsena pasikeičia, jis nebereikalingas šiam konkrečiam pacientui, procesas baigiasi galutine būsena [109A].

Generuojami UML būsenų modelio elementai iš veiklos modelio ligoninės informacijos valdymo proceso palatos dalies. Veiklos modelyje saugoma visa informacija, susijusi su procesais, funkcijomis ir jų būkle. Šis modelis pateikiamas iš palatos perspektyvos. Modelyje pateikiami šie elementai: pradinė būsena, kuria pradedamas procesas, pirmoji būsena – palata yra laisva, jos būsena pasikeičia pateikus prašymą užimti; išrašius pacientą palatos būsena vėl pasikeičia į laisvą [109A].

Naudojant ligoninės informacijos valdymo proceso pavyzdį sugeneruoti keturių UML tipų modeliai. Visi sugeneruoti modeliai apibrėžia tą patį dalykinės srities pavyzdį, tačiau iš skirtingų perspektyvų. Beveik kiekviename aprašyto pavyzdžio poskyryje pateikiamas daugiau nei vienas to paties tipo modelis: sugeneruotas UML taikomųjų uždavinių modelis pristato visus dalyvius (veikėjus), kurie dalyvauja ligoninės informacijos valdymo procese, ir jų funkcijas (procesus); sugeneruoti UML veiklos modeliai iliustruoja skirtingas to paties pavyzdžio veiklas iš skirtingų perspektyvų (registracija, palatos paskyrimas, medicininiai tyrimai, gydymo procesas ir išrašymas), ir tai nėra galutinis galimų UML veiklos modelių sąrašas; sugeneruoti UML sekų modeliai taip pat apibrėžia tos pačios dalykinės srities procesų ir funkcijų sekas iš skirtingų perspektyvų (paciento priėmimas, tyrimai ir gydymas, išrašymas), ir tai taip pat nėra galutinis galimų UML sekų modelių sąrašas; sugeneruoti UML būsenų modeliai apibūdina skirtingas būsenas iš objektų (paciento, gydytojo ir palatos) perspektyvų, ir galima sugeneruoti daugiau tos pačios dalykinės srities objektų būsenų modelių [109A].

Pateiktas ligoninės informacijos valdymo proceso pavyzdys rodo ir patvirtina, kad tai nėra galutinis UML dinaminių modelių, kuriuos galima sugeneruoti iš veiklos modelio, kiekis, yra ir daugiau skirtingų UML dinaminių modelių tai pačiai dalykinei sričiai projektuoti. Kaip minėta aukščiau, žiniomis grįstas veiklos modelis, kuriame saugomi patikrinti ir patvirtinti konkrečios dalykinės srities duomenys, yra pakankama duomenų saugykla UML modeliams generuoti [109A].

Gautų rezultatų vertinimas

Kaip minėta aukščiau, tradicinė IS inžinerija grindžiama analitiko patirtimi. Analitikas dalyvauja visame IS kūrimo gyvavimo ciklo procese, analizuodamas dalykinę sritį, modeliuodamas ir kurdamas visus reikalingus projekto modelius, nagrinėjamu atveju – UML modelius, naudodamas įvairias tam skirtas priemones, pasikliaudamas tik savo asmenine patirtimi ir gerąja praktika. Analitiko veiklą galima apibūdinti taip: veiklos situacijos analizė, tobulinimo galimybių nustatymas, informacinės sistemos, kuri sukurs pridėtinę vertę organizacijai, projektavimas. Analitikas surenka informaciją apie dalykinę sritį, nustato visus reikalavimus, ieško tinkamo metamodelio, patikrina ir patvirtina duomenis ir pradeda kurti IS projektinius UML modelius. Visada išlieka rizika, kad atsiras nauja problema, daugiau informacijos ar reikalavimų, todėl projekto modelių atnaujinimo ir tobulinimo procesas labai sudėtingas. IS kūrimo proceso trukmė ilgėja, o klaidų skaičius didėja.

Naudojant veiklos modelį (EM) kaip pagrindinę dalykinės srities žinių saugyklą IS inžinerijos procese, užtikrinamas sukurtų IS projektinių modelių teisingumas ir kokybė, atnaujinus bet kokius dalykinės srities duomenis.

1 lentelė. IS analitiko veiklos palyginimas pagal kriterijus

Kriterijus	Tradicinė IS inžinerija	Žiniomis grindžiamas UML modelių generavimo iš veiklos modelio metodas
Dalykinės srities duomenų surinkimas	Analitikas renka informaciją iš suinteresuotųjų šalių, dalykinės srities analizės, ankstesnių sistemų, taikydamas įvairias metodikas	Analitikas renka informaciją iš suinteresuotųjų šalių, dalykinės srities analizės, ankstesnių sistemų, taikydamas įvairias metodikas
Reikalavimų identifikavimas	Analitikas renka, nustato ir apibrėžia tam tikros dalykinės srities reikalavimus	Analitikas renka, nustato ir apibrėžia tam tikros dalykinės srities reikalavimus
Duomenų paruošimas modeliavimui	Analitikas parengia duomenis modeliavimui, savo nuožiūra naudoja visas galimas modeliavimo priemones, remdamasis savo empirine patirtimi	Analitikas parengia duomenis modeliavimui, naudoja veiklos modelį, kuris formalizuotas vadovaujantis veiklos metamodeliu; patikrina ir patvirtina tam tikros dalykinės srities žinias ir užpildo veiklos modelį reikalingais duomenimis
Projektinių modelių kūrimas	Analitikas kuria projektavimo modelius naudodamasis projektavimo įrankiais, remdamasis savo empirine patirtimi	Reikiami UML dinaminiai modeliai generuojami iš veiklos modelio naudojant transformacijos algoritmus
Dalykinės srities duomenų atnaujinimas	Analitikas patvirtina būtinus pakeitimus ir atnaujina dalykinės srities duomenis	Analitikas patikrina ir patvirtina būtinus pakeitimus ir atnaujina dalykinės srities duomenis veiklos modelyje
Projektinių modelių atnaujinimas	Analitikas iš naujo sukuria arba patobulina projektinius modelius naudodamasis projektavimo įrankiais	Reikiami UML dinaminiai modeliai generuojami iš veiklos modelio su atnaujintais duomenimis naudojant transformacijos algoritmus
Padidėjusi IS kūrimo proceso trukmė	Projektinių modelių tobulinimui reikalingas papildomas laikas	UML dinaminiai modeliai iš veiklos modelio sukuriami greičiau nei be transformacijos algoritmų

Padidėjęs klaidų skaičius	Didesnė tikimybė, kad bus didesnis klaidų skaičius	Mažesnė tikimybė, kad klaidų skaičius padidės, nes dalykinės srities duomenys iš anksto patikrinami ir patvirtinami

Šaltinis: sudaryta autorės

Naudojant veiklos modelį IS inžinerijos procese analitikai į veiklos modelį įkelia visus surinktus dalykinės srities duomenis. Veiklos modelyje saugomos dalykinės srities žinios naudojamos UML modeliams generuoti naudojant transformacijos algoritmus. Į veiklos modelį įkėlus bet kokius galimus naujus duomenis, vėl panaudojami transformacijos algoritmai ir pagal atnaujintus duomenis generuojami nauji UML modeliai. Analitikui nereikia iš naujo atlikti viso projektinių modelių kūrimo proceso.

IŠVADOS

1. Informacinių sistemų inžinerijos metodų ir standartų analizė parodė, kad rinkoje egzistuojantys metodai ir jų pagrindu sukurti CASE įrankiai gali generuoti IS projekto fragmentus UML notacija, tokius kaip: vartotojo sąsajos prototipas, duomenų bazės specifikacijas, programinio kodo fragmentus, tačiau nebuvo aptikta IS kūrimo projektavimo etapo UML dinaminių modelių generavimo sprendimų. Pasiūlyto metodo taikymas gerina IS projektavimo etapo efektyvumą.
2. Sukurti ir eksperimentiškai patikrinti transformacijos algoritmai įrodo, kad veiklos modelio (EM) sudėtis yra pakankama generuoti UML dinaminius (taikomųjų uždavinių, veiklos, būsenų, sekų, komunikacijos, laiko, sąveikos) modelius.
3. Eksperimente patikrinti darbe sukurti septyni transformacijos algoritmai, jie panaudoti generuoti dinaminius UML modelius iš aštuonių dalykinių sričių veiklos modelių. Sugeneruoti dinaminiai modeliai tenkina UML specifikacijos reikalavimus, nes sugeneruojami visi kiekvieno tipo UML dinaminio modelio sudėties elementai bei ryšiai tarp jų.
4. Sukurtas darbe Organizacijos veiklos žiniomis grįstas UML dinaminių modelių generavimo metodas buvo vertinamas lyginant IS analitiko veiklą tradicinėje ir žiniomis grįstoje IS inžinerijoje. Pasiūlytas metodas yra pranašesnis atliekant dinaminių modelių korekcijas (pusiau-automatinis) ir turi pranašumų pirminių modelių kūrimo procese. Palyginimo rezultatai pateikti 41 lentelėje.
5. Šio darbo rezultatai suteikia pagrindą kurti CASE įrankių įskiepius įgalinančius IS projekto dinaminių modelių generavimą iš CASE įrankio žinių saugykloje saugomų dalykinės srities žinių.

APPENDICES

LIST OF PUBLICATIONS

Journals listed in Web of Science database with citation index

1. Rudžionis, Vytautas; Lopata, Audrius; Gudas, Saulius; Butleris, Rimantas; Veitaitė, Ilona; Dilijonas, Darius; Grišius, Evaldas; Zwitserloot, Maarten; Rudzioniene, Kristina. Identifying irregular financial operations using accountant comments and natural language processing techniques // Applied sciences. Basel : MDPI. ISSN 2076-3417. 2022, vol. 12, iss. 17, art. no. 8558, p. 1-15. DOI: 10.3390/app12178558. [Science Citation Index Expanded (Web of Science); Scopus; Dimensions] [IF: 2,838; AIF: 5,795; IF/AIF: 0,490; Q2 (2021, InCites JCR SCIE)] [CiteScore: 3,70; SNIP: 1,026; SJR: 0,507; Q2 (2021, Scopus Sources)] [S.fld.: T 007, S 004] [Contribution: 0,111]
2. Lopata, Audrius; Gudas, Saulius; Butleris, Rimantas; Rudžionis, Vytautas Evaldas; Žioba, Liutauras; Veitaitė, Ilona; Dilijonas, Darius; Grišius, Evaldas; Zwitserloot, Maarten. Financial data anomaly discovery using behavioral change indicators // Electronics. Basel: MDPI. ISSN 2079-9292. 2022, vol. 11, iss. 10, art. no. 1598, p. 1-14. DOI: 10.3390/electronics11101598. [Science Citation Index Expanded (Web of Science); Scopus; Dimensions] [IF: 2,690; AIF: 4,505; IF/AIF: 0,597; Q3 (2021, InCites JCR SCIE)] [CiteScore: 3,70; SNIP: 1,013; SJR: 0,590; Q2 (2021, Scopus Sources)] [S.fld.: T 007] [Contribution: 0,111]
3. Lopata, Audrius; Ambraziūnas, Martas; Veitaitė, Ilona; Masteika, Saulius; Butleris, Rimantas. SysML and UML models usage in knowledge based MDA process // Elektronika ir elektrotechnika. Kaunas: KTU. ISSN 1392-1215. eISSN 2029-5731. 2015, vol. 21, no. 2, p. 50-57. DOI: 10.5755/j01.eee.21.2.5629. [Science Citation Index Expanded (Web of Science); Scopus; INSPEC] [IF: 0,389; AIF: 1,847; IF/AIF: 0,211; Q4 (2015, InCites JCR SCIE)] [CiteScore: 1,60; SNIP: 0,588; SJR: 0,337; Q3 (2015, Scopus Sources)] [S.fld.: T 007] [Contribution: 0,200]
4. Lopata, Audrius; Veitaite, Ilona; Gudas, Saulius; Butleris, Rimantas. Case tool component - knowledge-based subsystem UML diagrams generation process // Transformations in business & economics = Verslo ir ekonomikos transformacijos. Vilnius : Vilniaus universitetas. ISSN 1648-4460. 2014, vol. 13, no. 2B (32B), p. 676-696. [Social Sciences

Citation Index (Web of Science); Scopus; EconLit with Full Text] [IF: 0,374; AIF: 1,514; IF/AIF: 0,247; Q4 (2014, InCites JCR SSCI)] [CiteScore: 0,50; SNIP: 0,303; SJR: 0,275; Q3 (2014, Scopus Sources)] [S.fld.: T 007, S 003] [Contribution: 0,250]

ARTICLES IN SCIENTIFIC CONFERENCE PROCEEDINGS

Conference proceedings indexed by Web of Science database

1. Veitaite, Ilona; Lopata, Audrius. Problem domain example of knowledge-based enterprise model usage for different UML behavioral models generation // Business information systems workshops: BIS 2021 international workshops virtual event, June 14–17, 2021: revised selected papers / W. Abramowicz, S. Auer, M. Stróżyna (eds.). Cham : Springer, 2022. ISBN 9783031042157. eISBN 9783031042164. p. 45-55. (Lecture notes in business information processing, ISSN 1865-1348, eISSN 1865-1356; Vol. 444). DOI: 10.1007/978-3-031-04216-4_5. [Conference Proceedings Citation Index - Science (Web of Science); Scopus] [S.fld.: T 007] [Contribution: 0,500] [Indėlis autoriniais lankais: 0,393]
2. Lopata, Audrius; Butleris, Rimantas; Gudas, Saulius; Rudžionis, Vytautas; Rudžionienė, Kristina; Žioba, Liutauras; Veitaitė, Ilona; Dilijonas, Darius; Grišius, Evaldas; Zwitserloot, Maarten. Financial data preprocessing issues // Information and software technologies: 27th international conference, ICIST 2021, Kaunas, Lithuania, October 14–16, 2021: proceedings / A. Lopata, D. Gudonienė, R. Butkienė (eds.). Cham: Springer Nature, 2021. ISBN 9783030883034. eISBN 9783030883041. p. 60-71. (Communications in computer and information science, ISSN 1865-0929, eISSN 1865-0937; vol. 1486). DOI: 10.1007/978-3-030-88304-1_5. [Conference Proceedings Citation Index - Science (Web of Science); Scopus] [S.fld.: T 007] [Contribution: 0,100] [Indėlis autoriniais lankais: 0,086]
3. Veitaitė, Ilona; Lopata, Audrius. Knowledge-based UML use case model transformation algorithm // Business information systems workshops: BIS 2019 international workshops, Seville, Spain, June 26–28, 2019: revised papers / W. Abramowicz, R. Corchuelo (eds.). Cham: Springer, 2019. ISBN 9783030366902. eISBN 9783030366919. p. 39-48. (Lecture notes in business information processing, ISSN 1865-1348, eISSN 1865-1356; Vol. 373). DOI: 10.1007/978-3-030-36691-9. [Conference

Proceedings Citation Index - Science (Web of Science); Scopus] [S.fld.: T 007] [Contribution: 0,500] [Indėlis autoriniais lankais: 0,357]

4. Veitaitė, Ilona; Lopata, Audrius. Problem domain knowledge driven generation of UML models // Information and software technologies: 24th International Conference, ICIST 2018, Vilnius, Lithuania, October 4–6, 2018: proceedings / edited by: Robertas Damaševičius, Giedrė Vasiljevienė. Cham: Springer, 2018. ISBN 9783319999715. eISBN 9783319999722. p. 178-186. (Communications in Computer and Information Science, ISSN 1865-0929, eISSN 1865-0937; vol. 920). DOI: 10.1007/978-3-319-99972-2. [Conference Proceedings Citation Index - Science (Web of Science); Scopus] [S.fld.: T 007] [Contribution: 0,500] [Indėlis autoriniais lankais: 0,322]
5. Lopata, Audrius; Veitaitė, Ilona; Žemaitytė, Neringa. Enterprise model based UML interaction overview model generation process // Business information systems workshops: BIS 2016 international workshops, Leipzig, Germany, July 6-8, 2016: revised papers / Editors: Abramowicz, Witold, Alt, Rainer, Bogdan, Franczyk . - Series: Lecture notes in business information processing. Vol. 263. ISSN: 1865-1348. Berlin : Springer International Publishing, 2017. ISBN 9783319524634. eISBN 9783319524641. p. 69-78. DOI: 10.1007/978-3-319-52464-1_7. [Conference Proceedings Citation Index - Science (Web of Science); Scopus; DBLP (Computer Science Bibliography)] [S.fld.: T 007] [Contribution: 0,333] [Indėlis autoriniais lankais: 0,238]
6. Veitaitė, Ilona; Lopata, Audrius. Knowledge-based UML models generation from enterprise model technique // Information and software technologies: 23rd international conference, ICIST 2017, Druskininkai, October12-14, 2017: proceedings / editors: Robertas Damaševičius, Vilma Mikašytė . - Book series: Communications in Computer and Information Science. Vol 756. Cham: Springer Nature. ISSN 1865-0929. eISSN 1865-0937. 2017, p. 314-325. DOI: 10.1007/978-3-319-67642-5_26. [Conference Proceedings Citation Index - Science (Web of Science)] [S.fld.: T 007] [Contribution: 0,500] [Indėlis autoriniais lankais: 0,429]
7. Veitaitė, Ilona; Lopata, Audrius. Transformation algorithms of knowledge based UML dynamic models generation // Business information systems workshops BIS 2017, Poznan, Poland, 28-30 June / editor Witold Abramowicz. - Series: Lecture notes in business information processing. Vol 303. Cham: Springer International

Publishing, 2017. ISBN 9783319690223. eISBN 9783319690230. p. 59-68. DOI: 10.1007/978-3-319-69023-0. [Conference Proceedings Citation Index - Science (Web of Science); Scopus] [S.fld.: T 007] [Contribution: 0,500] [Indėlis autoriniais lankais: 0,357]

8. Veitaitė, Ilona; Lopata, Audrius. Additional knowledge based MOF architecture layer for UML models generation process // Business information systems: 2015 international workshops: revised papers: proceedings.- Series: Lecture notes in business information processing (ISSN 1865-1348), Vol. 226 / Editors: Witold Abramowicz, Angelika Kokkinaki. Berlin: Springer International Publishing, 2015. ISBN 9783319267616. eISBN 9783319267623. p. 56-63. DOI: 10.1007/978-3-319-26762-3_6. [Scopus; Conference Proceedings Citation Index] [S.fld.: T 007] [Contribution: 0,500] [Indėlis autoriniais lankais: 0,286]
9. Veitaitė, Ilona; Ambraziūnas, Martas; Lopata, Audrius. Enterprise model and ISO standards based information system's development process // Business information systems: 2014 international workshops, Larnaca, Cyprus, May 22-23, 2014: Revised Papers: proceedings.- Series: Lecture notes in business information processing (ISSN 1865-1348), Vol. 183/ Editors : Witold Abramowicz, Angelika Kokkinaki. Berlin: Springer, 2014. ISBN 9783319114590. p. 73-79. DOI: 10.1007/978-3-319-11460-6_7. [Scopus; Conference Proceedings Citation Index; SpringerLink] [S.fld.: T 007] [Contribution: 0,334] [Indėlis autoriniais lankais: 0,024]
10. Lopata, Audrius; Veitaitė, Ilona. UML diagrams generation process by using knowledge-based subsystem // Business Information Systems Workshop 2013, Poznan, Poland, June 2013 / Witold Abramowicz (ed.). Berlin : Springer, 2013. ISBN 9783642416866. p. 53-60. [Conference Proceedings Citation Index] [S.fld.: T 007] [Contribution: 0,500] [Indėlis autoriniais lankais: 0,286]

Other peer-reviewed conference proceedings

1. Lopata, Audrius; Butleris, Rimantas; Gudas, Saulius; Rudžionienė, Kristina; Žioba, Liutauras; Veitaitė, Ilona; Dilijonas, Darius; Grišius, Evaldas; Zwitserloot, Maarten. Financial process mining characteristics // Information and software technologies: 28th international conference, ICIST 2022, Kaunas, Lithuania, October 13–15, 2022: proceedings / A. Lopata, D. Gudonienė, R. Butkienė (eds.). Cham: Springer, 2022. ISBN 9783031163012. eISBN 9783031163029. p. 209-220. (Communications

in computer and information science, ISSN 1865-0929, eISSN 1865-0937; vol. 1665). DOI: 10.1007/978-3-031-16302-9_16. [S.fld.: T 007] [Contribution: 0,111] [Indėlis autoriniais lankais: 0,095]

2. Veitaitė, Ilona; Lopata, Audrius. Knowledge-based UML use case and UML activity models generation from enterprise model. school of languages case study // CEUR workshop proceedings: IVUS 2021: Information society and university studies 2021: Proceedings of the 26th international conference on information society and university studies (IVUS 2021), Kaunas, Lithuania, April 23, 2021 / edited by: I. Veitaitė, A. Lopata, T. Krilavičius, M. Woźniak. Aachen: CEUR-WS. ISSN 1613-0073. 2021, vol. 2915, art. no. 13, p. 112-121. [Scopus] [S.fld.: N 009] [Contribution: 0,500] [Indėlis autoriniais lankais: 0,357]
3. Veitaitė, Ilona; Lopata, Audrius. Knowledge-based UML activity model transformation algorithm // CEUR workshop proceedings: IVUS 2020: Information society and university studies 2020: proceedings of the information society and university studies, 2020, Kaunas, Lithuania, April 23, 2020 / edited by: A. Lopata, V. Sukackė, T. Krilavičius, I. Veitaitė, M. Woźniak. Aachen: CEUR-WS. ISSN 1613-0073. 2020, vol. 2698, p. 114-120. [Scopus; Compendex] [S.fld.: T 007] [Contribution: 0,500] [Indėlis autoriniais lankais: 0,250]
4. Veitaitė, Ilona; Lopata, Audrius. Knowledge-based generation of the UML dynamic models from the enterprise model illustrated by the ticket buying process example // Information and software technologies: 26th international conference, ICIST 2020, Kaunas, Lithuania, October 15–17, 2020: proceedings / eds.: Audrius Lopata, Rita Butkienė, Daina Gudonienė, Vilma Sukackė. Cham : Springer, 2020. ISBN 9783030595050. eISBN 9783030595067. p. 26-38. (Communications in Computer and Information Science, ISSN 1865-0929, eISSN 1865-0937; vol. 1283). DOI: 10.1007/978-3-030-59506-7_3. [Scopus; SpringerLink] [S.fld.: T 007] [Contribution: 0,500] [Indėlis autoriniais lankais: 0,465]
5. Veitaitė, Ilona; Lopata, Audrius. Knowledge based UML information flow model transformation algorithm // ICYRIME 2018: proceedings of the international conference for young researchers in informatics, mathematics, and engineering, Kaunas, Lithuania, April 27, 2018. Aachen: CEUR-WS. 2018, p. 30-36. (CEUR workshop proceedings, ISSN 1613-0073; vol. 2152). [Scopus] [S.fld.: T 007] [Contribution: 0,500] [Indėlis autoriniais lankais: 0,250]

6. Veitaitė, Ilona; Lopata, Audrius. Enterprise knowledge based UML timing model generation process // Informacinės technologijos 2017: XXII tarpuniversitetinė magistrantų ir doktorantų konferencija "Informacinė visuomenė ir universitetinės studijos" (IVUS2017): pranešimų medžiaga. Kaunas: Technologija. ISSN 2029-249X. 2017, p. 189-193. [S.fld.: T 007] [Contribution: 0,500] [Indėlis autoriniais lankais: 0,179]
7. Veitaitė, Ilona; Lopata, Audrius. Enterprise knowledge based UML timing model generation process // CEUR workshop proceedings: IVUS 2017: Proceedings of the IVUS International Conference on Information Technology, Kaunas, Lithuania, April 28, 2017 / Edited by Robertas Damaševičius, ... [et al]. Aachen: CEUR-WS. ISSN 1613-0073. 2017, Vol.1856, p. 75-79. [Scopus] [S.fld.: T 007] [Contribution: 0,500] [Indėlis autoriniais lankais: 0,179]
8. Veitaitė, Ilona; Lopata, Audrius. Enterprise model, MOF and ISO standards based information system’s development process // Informacinės technologijos 2015: XX tarpuniversitetinė magistrantų ir doktorantų konferencija: Konferencijos pranešimų medžiaga. Kaunas: Technologija. ISSN 2029-249X. 2015, p. 77-81. [S.fld.: T 007] [Contribution: 0,500] [Indėlis autoriniais lankais: 0,179]
9. Veitaitė, Ilona; Lopata, Audrius. Veiklos modelio taikymas informacijos sistemų inžinerijos reikalavimų specifikavimo etape // Informacinės technologijos : 19-oji tarpuniversitetinė magistrantų ir doktorantų konferencija "Informacinė visuomenė ir universitetinės studijos" (IVUS 2014) : pranešimų medžiaga. Kaunas: Technologija. ISSN 2029-4832. 2014, p. 40-46. [S.fld.: T 007] [Contribution: 0,500] [Indėlis autoriniais lankais: 0,250]
10. Veitaitė, Ilona; Lopata, Audrius. Veiklos modeliu grindžiamas UML diagramų generavimas // Informacinės technologijos 2012: 17-osios tarpuniversitetinės magistrantų ir doktorantų konferencijos pranešimų medžiaga. Vilnius. ISSN 2029-249X. 2012, p. 109-114. [S.fld.: N 009] [Contribution: 0,500] [Indėlis autoriniais lankais: 0,215]

ABSTRACTS IN CONFERENCE PROCEEDINGS

1. Lopata, Audrius; Veitaitė, Ilona. Transformation algorithms of UML models generation process. UML dynamic models generation examples // 11th international workshop on data analysis methods for software

systems, Druskininkai, Lithuania, November 28-30, 2019 / Lithuanian Computer Society, Vilnius University Institute of Data Science and Digital Technologies, Lithuanian Academy of Sciences. Vilnius : Vilnius University, 2019. ISBN 9786090703243. eISBN 9786090703250. p. 50. [S.fld.: T 007]

2. Lopata, Audrius; Veitaitė, Ilona. Knowledge-based UML models transformation algorithm // DAMSS 2018 : 10th international workshop on "Data analysis methods for software systems", Druskininkai, Lithuania, November 29 - December 1, 2018: [abstract book]. Vilnius: Vilniaus universitetas, 2018. ISBN 9786090700433. p. 55. [S.fld.: T 007]
3. Veitaitė, Ilona; Lopata, Audrius. Knowledge-based UML models generation transformation algorithms from enterprise model // 9th International workshop on Data Analysis Methods for Software Systems (DAMSS), Druskininkai, Lithuania, November 30 - December 2, 2017. Vilnius : Vilniaus universitetas, 2017. ISBN 9789986680642. p. 53. DOI: 10.15388/DAMSS.2017. [S.fld.: T 007]
4. Veitaitė, Ilona; Lopata, Audrius. Knowledge-based UML dynamic models generation method usage in IS lifecycle design stage // Data analysis methods for software systems: 8th international workshop on data analysis methods for software systems, Druskininkai, December 1-3, 2016. Vilnius: Vilniaus universiteto leidykla, 2016. ISBN 9789986680611. p. 65-66. [S.fld.: T 007]
5. Veitaitė, Ilona; Lopata, Audrius. Knowledge based UML dynamic models generation method // Data analysis methods for software systems: 7th international workshop, Druskininkai, December 3-5, 2015: [abstracts book]. Vilnius : Vilniaus universiteto Matematikos ir informatikos institutas, 2015. ISBN 9789986680581. p. 52-53. [S.fld.: T 007]
6. Veitaitė, Ilona; Lopata, Audrius. Enterprise knowledge based UML dynamic models generation process // Data analysis methods for software systems : 6th International Workshop: [abstracts book], Druskininkai, Lithuania, December 4-6, 2014. [Vilnius], 2014. ISBN 9789986680505. p. 57. [S.fld.: T 007]

PARTS OF BOOKS

Parts of scientific publications

(parts of scientific monographs, studies, synthesizing scientific works)

1. Veitaite, Ilona; Lopata, Audrius. Knowledge-based UML dynamic models generation from enterprise model in hospital information management process example // Intelligent systems for sustainable person-centered healthcare: [collective monograph] / Dalia Kriksciuniene, Virgilijus Sakalauskas (eds.). Cham: Springer, 2022, ch. 12. ISBN 9783030793524. eISBN 9783030793531. p. 225-250. (Intelligent systems reference library, ISSN 1868-4394, eISSN 1868-4408; vol. 205). DOI: 10.1007/978-3-030-79353-1_12. [Scopus] [M.kr.: T 007]

Parts of others books

(parts of science, art promotion and other books)

1. Veitaitė, Ilona; Lopata, Audrius. Knowledge-based transformation algorithms of UML dynamic models generation from Enterprise Model // Data science: new issues, challenges and applications / Dzemyda, Gintautas, Bernatavičienė, Jolita, Kacprzyk, Janusz (Eds.). Cham: Springer Nature, 2020. ISBN 9783030392499. eISBN 9783030392505. p. 43-59. (Studies in computational intelligence, ISSN 1860-949X, eISSN 1860-9503 ; vol. 869). DOI: 10.1007/978-3-030-39250-5_3. [Scopus] [M.kr.: T 007] [Indėlis: 0,500]

LIST OF CONFERENCES AND SCIENTIFIC EVENTS

1. 17th Inter University Conference for master and phd students on Information Technology IT2012, Kaunas, Lithuania, April 2012.
2. International Conference Business Information Systems BIS2013, Poznan, Poland, June 2013.
3. 19th Inter University Conference for master and phd students Information Society and University Studies IVUS2014, Kaunas, Lithuania, April 2014.
4. International Conference Business Information Systems BIS2014, Larnaca, Cyprus, May 2014.
5. 6th International Workshop Data Analysis Methods for Software Systems DAMSS2014, Druskininkai, Lithuania, December 2014.
6. 20th Inter University Conference for master and phd students on Information Technologies IT2015, Kaunas, Lithuania, April 2015.
7. International Conference Business Information Systems BIS2015, Poznan, Poland, May 2015.
8. 7th International Workshop Conference Data Analysis Methods for Software Systems DAMSS2015, Druskininkai, Lithuania, December 2015.
9. International Conference Business Information Systems BIS2016, , Leipzig, Germany, July 2016.
10. 8th International Workshop Conference Data Analysis Methods for Software Systems DAMSS2016, Druskininkai, Lithuania, December 2016.
11. 22nd International Conference for master, phd students and young researcherrs, Information Society and University Studies IVUS2017, Kaunas, Lithuania, April 2017.
12. International Conference Business Information Systems BIS2017, Poznan, Poland, June 2017.
13. 23rd International Conference Information and Software Technologies, ICIST2017, Druskininkai, Lithuania, October 2017.
14. 9th International Workshop Conference Data Analysis Methods for Software Systems DAMSS2017, Druskininkai, Lithuania, December 2017.
15. 2nd International Conference for young researchers in informatics, mathematics, and engineering ICYRIME 2018, part of 23rd International Conference for master, phd students and young researcherrs, Information

Society and University Studies IVUS2018, Kaunas, Lithuania, April, 2018.

16. 24th International Conference Information and Software Technologies, ICIST2018, Vilnius, Lithuania, October 2018.
17. 10th International Workshop Conference Data Analysis Methods for Software Systems DAMSS2018, Druskininkai, Lithuania, December 2018.
18. International Conference Business Information Systems BIS2019, Seville, Spain, June 2019.
19. 11th International Workshop Conference Data Analysis Methods for Software Systems DAMSS2019, Druskininkai, Lithuania, November 2019.
20. 25th International Conference for master, phd students and young researcherrs, Information Society and University Studies IVUS2020, Kaunas, Lithuania, April, 2020.
21. 26th International Conference Information and Software Technologies, ICIST2020, Kaunas, Lithuania, October 2020.
22. 26th International Conference for master, phd students and young researcherrs, Information Society and University Studies IVUS2021, Kaunas, Lithuania, April, 2021.
23. 27th International Conference Information and Software Technologies, ICIST2021, Kaunas, Lithuania, October 2022.
24. 28th International Conference Information and Software Technologies, ICIST2022, Kaunas, Lithuania, October 2023.

Ilona Veitaitė

Enterprise Knowledge-Based UML Dynamic Models Generation Method

DOCTORAL DISSERTATION
Technological sciences,
Informatics Engineering [T 007]
Thesis Editor: UAB „Bella Verba“, info@bellaverba.lt, +370 655 69981

Ilona Veitaitė

Organizacijos žiniomis pagrįstas UML dinaminių modelių generavimo metodas

DAKTARO DISERTACIJA
Technologijos mokslai
Informatikos inžinerija [T 007]
Santraukos redaktorius: UAB „Bella Verba“, info@bellaverba.lt, +370 655 69981

Vilniaus universiteto leidykla
Saulėtekio al. 9, III rūmai, LT-10222 Vilnius
El. p. info@leidykla.vu.lt, www.leidykla.vu.lt
Tiražas 20 egz.