https://doi.org/10.15388/vu.thesis.663
https://orcid.org/0000-0003-4062-2149

VILNIUS UNIVERSITY

Karolis Noreika

# Application Software Development Process Using an Enhanced Agile Project Management Method

**DOCTORAL DISSERTATION**

Technological Sciences,
Informatics Engineering (T 007)

VILNIUS 2024

The dissertation was prepared in 2019 – 2023 at Vilnius University.

**Scientific supervisor:** Prof. Dr. Saulius Gudas (Vilnius University, Technological Sciences, Informatics Engineering – T 007).

This doctoral dissertation will be defended in a public meeting of the Dissertation Defence Panel:

**Chairman** – Prof. Habil. Dr. Gintautas Dzemyda (Vilnius University, Technological Sciences, Informatics Engineering – T 007).
**Members:**
Prof. Dr. Rimantas Butleris (Kaunas University of Technology, Technological Sciences, Informatics Engineering – T 007),
Prof. Dr. Jānis Grabis (Riga Technical University, Latvia, Technological Sciences, Informatics Engineering – T 007),
Prof. Dr. Tomas Krilavičius (Vytautas Magnus University, Technological Sciences, Informatics Engineering – T 007),
Prof. Dr. Dalius Navakauskas (Vilnius Gediminas Technical University, Technological Sciences, Informatics Engineering – T 007).

The dissertation shall be defended at a public meeting of the Dissertation Defence Panel at 12:00 on the 27th of September, 2024 in Room 203 of the Institute of Data Science and Digital Technologies of Vilnius University.
Address: Akademijos str. 4, LT-08412, Vilnius, Lithuania.

The text of this dissertation can be accessed at the libraries of Vilnius University as well as on the website of Vilnius University:
www.vu.lt/lt/naujienos/ivykiu-kalendorius

https://doi.org/10.15388/vu.thesis.663
https://orcid.org/0000-0003-4062-2149

VILNIAUS UNIVERSITETAS

Karolis Noreika

# Taikomųjų programų kūrimo procesas naudojant modifikuotą Agile projektų valdymo metodą

**DAKTARO DISERTACIJA**

Technologijos mokslai,
Informatikos inžinerija (T 007)

VILNIUS 2024

Disertacija parengta 2019 – 2023 metais Vilniaus universitete.

**Mokslinis vadovas:** prof. dr. Saulius Gudas (Vilniaus universitetas, technologijos mokslai, informatikos inžinerija – T 007).

Gynimo taryba:
**Pirmininkas** – prof. habil. dr. Gintautas Dzemyda (Vilniaus universitetas, technologijos mokslai, informatikos inžinerija – T 007).
**Nariai:**
prof. dr. Rimantas Butleris (Kauno technologijos universitetas, technologijos mokslai, informatikos inžinerija – T 007),
prof. dr. Jānis Grabis (Rygos technikos universitetas, Latvija, technologijos mokslai, informatikos inžinerija – T 007),
prof. dr. Tomas Krilavičius (Vytauto Didžiojo universitetas, technologijos mokslai, informatikos inžinerija – T 007),
prof. dr. Dalius Navakauskas (Vilniaus Gedimino technikos universitetas, technologijos mokslai, informatikos inžinerija – T 007).

Disertacija ginama viešame Gynimo tarybos posėdyje 2024 m. rugsėjo mėn. 27 d. 12 val. Vilniaus universiteto Duomenų mokslo ir skaitmeninių technologijų instituto 203 auditorijoje.
Adresas: Akademijos g. 4, LT-08412 Vilnius, Lietuva.

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

# ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my supervisor Prof. Saulius Gudas for his unshakable belief in me and unwavering support during my PhD studies and preparation of this thesis. Your guidance helped me to keep going through the toughest moments.

Also, I would like to thank the Institute of Data Science and Digital Technologies at Vilnius University which provided creative conditions and encouraged doctoral students to improve, all scientists and professors of this and other universities, reviewers, fellow PhD students, and everyone who provided valuable comments, insights and advice that allowed me to make this thesis better.

Lastly, I would like to thank my family, especially my wife Raminta for her support and understanding while working on this thesis. Without her support, it would not have been done.

# ACRONYMS

AIPM – Australian Institute of Project Management
AMDD – Agile Model-Driven Development
ASD – Adaptive Software Development
CASE – Computer-Aided Software Engineering
CD – Continuous Delivery
CES – Cyber-Enterprise Systems
CI – Continuous Integration
CIM – Computation Independent Model
CMMI – Capability Maturity Model Integration
CPS – Cyber-Psychical Systems
CPSS – Cyber-Physical Social Systems
DKM – Domain Knowledge Model
DODAF – Department of Defence Architecture Framework
DSL – Domain Specific Language
DSDM – Dynamic System Development Method
EAS – Enterprise Application Software
EA – Enterprise Architecture
EAF – Enterprise Architecture Frameworks
EEML – Extended Enterprise Modelling Language
ERP – Enterprise Resource Planning
IM – Internal Model
IPMA – International Project Management Association
IS – Information System
KB – Knowledge Base
KPMG – Klynveld Peat Marwick Goerdeler
LeSS – Large Scale Scrum
MDA – Model Driven Architecture
MDD – Model Driven Development
MODAF – Ministry of Defence Architecture Framework
MT – Management Transaction
NAF **–** NATO Architecture Framework
NFR – Non-Functional Requirements
OKRs – Objectives and Key Results
OMG – Object Management Group
PDCA – Plan Do Check Act
PIM – Platform Independent Model
PSM – Platform Specific Model
RICE – Reach Impact Confidence Effort

ROI – Return on Investment
SAFe – Scaled Agile Framework
SAM – Strategic Alignment Model
SME – Subject Matter Expert
SPI – Software Process Improvement
SySML – Systems Modelling Language
TCN - Temporal Convolutional Network
TIES – Theme, Initiative, Epic, Story (user story)
TOGAF – Open Group Architecture Framework
UEML – User Experience Modelling Language
UML – Unified Modelling Language
VCM – Value Chain Model
XP – eXtreme Programming

# GLOSSARY

Agile – a value-oriented mindset and approach to delivering EAS projects.
Agile activity – a description of EAS requirements on different levels of Agile hierarchy.
Agile hierarchy – a set of activities managed using Agile framework to deliver enterprise application software (EAS).
Agile Model Driven Development (AMDD) – an Agile oriented Model Driven Development approach.
Causal modelling – a type of modelling, based on a white box approach where the cause and effect relation is defined using a formal structure.
Capability Maturity Model Integration – a model to evaluate the level of the maturity of processes in the organization.
Enterprise application software – a computer system used to manage, support and monitor organizational processes and activities.
Enterprise architecture – a well-defined approach for conducting enterprise analysis, application design, and implementation of relevant IT necessary to execute business strategies.
Enterprise architecture framework – the collection of processes, templates and tools that software teams use to plan and build large, enterprise-grade application architecture systems.
Epic – a large piece of software development project work that is too broad or complex to be completed in one development iteration that allows to deliver significant value.
External modelling – a type of modelling based on input-output result analysis.
Initiative – a collection of epics that drive toward a common goal and provide business and IT perspectives on achieving business goals.
Internal modelling – a type of modelling, based on behavioural/structural patterns.
Ministry of Defence Architecture Framework – enterprise architecture framework developed by the British Ministry of Defence.
Model Driven Development – software development approach based on developing domain business problem models and transforming models into working code.
Object Management Group (OMG) – an international, open membership, not-for-profit technology standards consortium.
Product backlog – a set of activities to deliver a product (EAS system in the context of this thesis).

RAG indicator – a classical project management indicator, representing a situation in the project, typically R-red (critical situation), A-amber (situation requires attention, but it is not critical), G-green (project goes according to plan).
Subject Matter Expert – a person with specialized knowledge in business processes in a specific area.
Scrum – a lightweight Agile framework, where decisions on product development are based on observation, experience and experimentation and that describes a set of events, artifacts and roles for efficient project delivery.
Scrum artifact – a tangible outcome of project delivery, expressed as a set of requirements or valuable pieces of software.
Sprint – timebox in Scrum Agile framework, typically from one to four weeks dedicated to developing valuable increment (a piece of software).
Sprint backlog – a set of activities that deliver a working piece of software that eventually becomes the final version of the product (EAS system in the context of this thesis).
Theme – an aggregation of user stories to show business value delivered and to help with prioritization as well as show planned product delivery at a high level.
User story – a description of the business problem that is not too detailed and not too wide and that solves a business problem.

TABLE OF CONTENTS

# LIST OF FIGURES

LIST OF TABLES

# INTRODUCTION

Nowadays, many businesses rely on enterprise application software systems (EAS) that support, orchestrate, and manage the processes in the enterprise. Such systems are created and improved using different software development management methodologies that highly influence the way development of enterprise software projects happens. Adopting a relevant methodology for enterprise application software development can determine to what extent business goals such as enterprise strategy, cost, quality, and timeliness are achieved once the software development project is completed.

Society is living in the times of the fourth industrial revolution. Its key features are the blurred boundaries between the physical and digital spheres of the real world where state-of-the-art systems, cyber-systems and Artificial Intelligence (AI) technologies support interactions with people, technical devices and the environment.

The cyber-physical system is a subgroup of the Cyber system group, with Cyber-physical social systems (CPSS), Cyber-enterprise systems (CES), and Cyber-biological systems. The term "cyber-physical systems" emerged around 2006, when it was coined by Helen Gill at the National Science Foundation in the United States [1].

AI technologies are used in everyday life from customer service to transportation and organizational decision-making [2, 3]. More new types of cyber systems such as "digital twins" are emerging. Such systems are of high complexity as they represent real-world causal dependencies by using an internal business domain model. The business domain model is used as a source of knowledge about the internal dependencies of the domain, which is necessary to form the logic of cyber system actions.

The research domain in this thesis includes enterprise management, enterprise application software engineering, and an Agile approach toward EAS project management. From the point of view of this thesis, all these activities are ultimately related and considered complex systems with self-managing capabilities, i.e. having closed-loop interactions between internal activities (management transactions). The formal basis of such complex systems composition is defined by a "good regulator" theorem by Conant et al. [4] and the internal model (IM) principle by Francis [5]. An IM is considered a causal knowledge model of the subject domain and is a mandatory component (by Francis) of any enterprise management (control) system as well as cyber systems (or intelligent systems with feedback control). In systems analysis, IM is considered as a white box (or grey box) model, that describes the discovered and known causal dependencies of the problem

domain. The awareness of the specific domain causality is the prerequisite for discovering deep knowledge (i.e. regularities, laws) in a given domain.

Causal modelling is aimed at discovering causal dependencies of processes and information attributes in various real-world domains. Causal knowledge is defined by Zack as a "description of causal links among a set of factors … which provides a means for organizations … how best to achieve some goal" [6]. Thus, the condition and basis for the creation of the advanced complex systems is internal modelling which is aimed to reveal causal dependencies of the problem domain, to know the causality specific to a particular domain. Internal modelling is a paradigm that corresponds to the causal modelling approach.

The „digital twins" systems are an example of modern cyber systems with an internal model at the core. The "digital twins" concept was introduced in 2014 by Grieves [7] to "visualize and simulate complex systems and systems of systems, including their physical behaviour, in real-time and with acceptable compute costs". The improvement of this type of systems engineering and their project management requires new methods that can be developed on the basis of the causal modelling paradigm.

The causal modelling approach helps to create a virtual real-world domain knowledge model (domain causality model) that is close to identical to the real-world environment to simulate some process before heavily investing in the actual implementation of the real-world process. Therefore, it is relevant to review traditional software engineering methods, domain modelling languages, project management methods, and tools from the perspective of the causal modelling paradigm.

The Agile approach has emerged as a philosophy, a way to address the shortages of application software development projects where software features were developed but never used, extensively long project durations, gaps in the knowledge-sharing process throughout the IT system development process and eventually to address the issues of misalignment between enterprise application software developed and business objectives that the software was aimed to achieve. Origins of the Agile approach and iterative methods trace back to the 1930s [8]. Agile frameworks for software development such as Scrum [9, 10] in 1995, DSDM [11] in 1997, Crystal [12, 13] in 1998, ASD [14] and XP [15] in 1999 and work management system Kanban [16] were all predecessors and helped to formulate what is known as Manifesto for Agile Software Development [17] or more commonly referred to as Agile Manifesto. It was formulated in 2001 by a group of IT systems developers who wanted to address the aforementioned shortages of application software development projects and is the first structured document

that is recognized globally and describes the Agile approach. One of the key aspects of Agile is the iterative and incremental approach to IT project delivery by ensuring continuous development of the IT system with regular feedback from the customers of the system.

As Portman [18] described in his book "Scaling Agile in Organizations - Guide for Project Managers and Agile Leaders" Agile has over 15 different frameworks. The most popular Agile software development frameworks for team level are Scrum [9, 10] and Kanban [16]. For enterprise or large-scale Agile software development Large Scale Scrum (LeSS) [19], Scaled Agile Framework SAF'e [20, 21] and others are used that are inspired by Lean software development described in [22]. Scrum Body of Knowledge (SBoK) is a "comprehensive guide to Scrum methodologies and practices" [23] that in the latest 4th edition contains sections about scaling Scrum for large projects and the enterprise. The usage of Agile frameworks for software development is increasing, but is often seen as sporadic and difficult to measure in terms of success, or the classical project management constraints of time, scope, and cost.

In the thesis, the Agile methodology, used for EAS project management is reviewed from the point of view of the causal modelling paradigm as this is the basis of which the most effective project management methods (including both business management and software engineering management activities) are created. The specific Agile methods that were used in this thesis are Scrum for team-level activities management and attributes from the Scaled Agile Framework. The team composition and organizational structures are based on the Spotify model [24].

The thesis presents a modified model of Agile project management based on causal modelling, which identifies the content (and meaning) of the hierarchical interactions of the project work defined as the Agile activities. This allows the formation of new indicators of the project state evaluation, which are not currently available in Agile project management tools (such as Atlassian Jira).

Evaluation of the enterprise application software state is only available if the content of the Agile activities interactions is specified, and this is only possible if the internal modelling paradigm (causal modelling) is chosen.

Traditional software engineering methods such as business process analysis using BPMN[25], UML[26], or other notations are aimed to represent the business domain from an external modelling perspective. Requirements traceability [27] is a traditional method to ensure the alignment of business objectives to requirements for enterprise application software development. However, these methods, waterfall and Agile project management tools such

as Atlassian Jira do not contain an internal modelling-based causality perspective.

This thesis presents a new approach to Agile software development project management, based on causality modelling, including integration with enterprise strategic objectives and EAS project content status assessment. The model of causal vertical and horizontal interactions of the Agile activities hierarchy together with the realization design of this model to enhance Agile project management tools is presented. The basic concepts and processes for the effective evaluation of the EAS development project content state are presented.

A major part of this thesis presents the literature review on the causality aspect of various EAS project management methods, including various Agile project management methods (Scrum, SAF'e, AMDD, etc.). Most popular tools for Agile project management are also analysed from the causality perspective. The relationship between Agile activities specified using various Agile project management methods and enterprise strategic goals is also examined. The literature review is aimed at the identification of the major issues in software development process monitoring and progress evaluation and to gather the requirements for a software development process measurement and content evaluation method newly proposed by the author. Based on the results of the literature review, a measurement and evaluation model for Agile software development management is formulated. The objective of the model is to provide the groundwork for a software process measurement and evaluation framework. The model is deemed to be applicable in a broad spectrum of scenarios by providing concepts that are independent of specific software process improvement initiatives and tools.

The domain causality modelling has different accuracy levels:

1. Cause and effect modelling in organizational systems is based on business rules IF ... THEN ... ELSE … [28] and is used in enterprise management modelling.
2. Workflow models integrate a set of cause-and-effect states into a coherent chain, a unified pattern that has a well-defined beginning and end. These are models of sequential processes, the management of which is external, not modelled [29].
3. Models comprising feedback loop as mandatory component aimed for domain causation description/specification such as action workflow [30], Causal Loop Diagram (CLD) [31], Deming's PDCA cycle [32], transactional workflows [33]. The formal basis of such models is control theory. By adapting control theory for enterprise management modelling from the perspective of information/knowledge

transformations the management transaction construct was developed in [34] that is used in this thesis to model the interactions of the Agile activities.

### Motivation

Enterprise Application Software (EAS) solutions are used to monitor and manage business processes and operations in most modern organizations: to help companies deploy their products to the market, collaborate with partners, achieve new strategic goals, expand, and grow. EAS is always expected to fully support the business management processes, thus ensuring the alignment between enterprise strategic needs, business operations, and IS capabilities.
At the same time, the enterprise carries out further EAS development projects, which need to be managed and the compatibility of their results with the existing EAS and business strategies must be ensured.

However, the alignment of business management needs and the functionality of EAS solutions is usually the result of human communication and collaboration at different levels of the organization‘s management hierarchy, which is not systematically orderly and not supported by any formal method of ensuring business requirements and IS functionality alignment.
This leads to lesser satisfaction of end-users of the EAS development projects and worse performance results for the entire organization. Standish Group, a leading project management statistics provider, shares publicly available results from 2011 to 2015 that successful projects only constituted up to 31% [35]. The definition of “successful” included the time, scope and budget constraints and also that the stakeholders are satisfied with the outcome of the project. Project management research by KPMG, AIPM, and IPMA [36] indicate that only up to 44% of EAS projects are likely to be delivered that meet the original goal and business intent, up to 36% on budget and up to 30% on time.

As Pikkarainen et al. state in [37] “Companies are becoming Agile in order to improve the productivity of product development teams”. Business development teams are also making product development decisions based on the Agile methodology approach. For a company to become Agile means changing the mindset of employees or orienting them towards accepting emerging changes instead of strictly following product development plans or roadmaps. It also means that employees at all levels of the organization need to adapt to the new way of working, which is getting the results of their daily duties evaluated much faster than in the traditional way of working. However, when “going Agile”, the overall goals of the organization are not always

supported by an organizational change. Some researchers emphasize the importance of supporting the Agile way of working from an organizational perspective (providing an appropriate physical atmosphere and work environment that encourages creativity) [38, 39].

Although Agile methods are becoming more popular, using only Agile methods still proves not to be enough to improve the success rates of IT project delivery. The link between the task or user story in the EAS project and the strategic objectives of the enterprise is not clearly defined. This is also not resolved by following the traditional requirements traceability approach [27] as it does not cover the business context or domain causality. This means that business and IT alignment is still an important field of research.

Experience shows that Agile project management tools such as Atlassian Jira have limited project progress monitoring capabilities, capturing the status of EAS projects (some project implementation parameters at the moment) by relying solely on the ability and opinion of the expert, which is not supported by any virtual (internal) knowledge model.

Project management tools with an internal model of the ongoing process (domain) could provide support for decision-making, also could automatically evaluate a wide range of project parameters, i.e. perform analysis of EAS design and development processes to ensure better business and IT alignment. This is one of the reasons why only using the Agile project management approach it still does not provide sufficient EAS project delivery results.

Currently used methods for business needs and IT alignment require a significant time investment to evaluate the alignment of EAS project requirements to the organizational process models and do not thoroughly define the link between strategic business objectives, the capabilities of the organization, and EAS project requirements. The methods are as follows: business and EAS development alignment Guidelines Regarding Architecture Alignment (GRAAL) [40], Business IT Alignment Method (BITAM) [41], Service-Oriented Business and Information Systems Alignment Method (SBISAF) [42], and enterprise architecture frameworks: The Ministry of Defence Architecture Framework (MODAF) [43], The Open Group Architecture Framework (TOGAF) [44], ArchiMate [45], NAF [46], UAF [47].

Traditional EAS development defines the business needs for EAS project development manually and later updates it in order for them to match the organizational strategy, goals, and capabilities. A different approach to support the links between strategic business objectives and Agile management hierarchy concepts of themes, initiatives, epics, and user stories [48] is required.

From the causal modelling perspective, elicitation of business needs for the development of EAS requires modelling methods, focused on the understanding of causality in the particular subject domain [34, 49, 50].

This study covers Agile hierarchy activities and their interactions. In the typical Agile management hierarchy, only the fact of the connection (is or is not) of Agile activities is recorded. The content of information transactions between Agile hierarchy activities (causal interactions of themes, initiatives, epics, user stories, etc.) is not specified. This is an example of a black box approach in an Agile management tool when only the fact of interaction is specified, but content is not available. Such Agile project management tools as Atlassian Jira allows to specify the interaction type (outgoing: “blocks”, “clones”, “duplicates”, “causes”, etc., or incoming: “is blocked by”, “is cloned by”, “is duplicated by”, “is caused by”, etc.).

From the perspective of causal modelling, the content of information transactions between Agile hierarchy activities (causal interactions of themes, initiatives, epics, user stories, etc.) must be specified.

This describes the qualitative difference between the content of causal models (white box) and traditional (black box) models.

The conducted research allows us to assert that in order to create an intelligent environment for EAS project management, project state monitoring, and evaluation, a modified Agile hierarchy management model is needed, based on the causal modelling methodology and integrated with the enterprise strategy models.

### Object and Scope of Research

Alignment of Enterprise Application Software (EAS) design solutions to enterprise strategy-driven capabilities in an Agile project management environment.

### Problem Statement

The Agile approach is popular for the management of Enterprise Application Software (EAS) development projects. However, the Agile methods (Scrum, Kanban, SAFe, LeSS) do not contain business strategy modelling. This results in a gap between IS functionality and business needs. The research focuses on the issue of inconsistency between strategic business objectives, Agile project content and the functionality of the software to be developed. Agile project management tools lack the functionality of enterprise strategic objectives modelling as goals of the projects and their execution lacks alignment with

strategic business objectives. Current Agile project management tools are missing EAS project activities‘ content coordination and evaluation functionality.

This research aims to rethink the Agile project management method using the causal modelling approach. The causal modelling allows for capturing the causal interactions between activities in different levels of Agile hierarchy revealing the informational content of the interactions. Causal modelling is based on the formal specification of causal interactions and gives the basis for formalizing the analysis of Agile activities structure and mutual transactions. In this thesis, the business domain and EAS project activities’ causal interactions are specified as management transactions (MTs). MT is used as a unified structure for the specification of the Agile activities at all levels of hierarchy. This allowed for modifying the traditional Agile project management method and to develop a causal Agile management model. The causal Agile management model helps to consistently track the integrity of EAS project content against the strategic business objectives of the enterprise.

Research Goal

The goal is to develop a modified Agile project management method and a project management system architecture using a causal modelling method that ensures EAS project solution alignment to business strategy and allows the evaluation of the EAS project specification status against the enterprise's strategic objectives.

Research Objectives

1. To investigate Agile project management tools and methods for EAS project management that evaluate the perspective of business strategy execution.
2. To investigate possibilities to integrate causal modelling and enterprise architecture frameworks for the identification of enterprise strategic objectives as the basis for the functional requirements of EAS.
3. To develop a modified Agile project management method based on the business domain causality model specified as management transaction including business strategy modelling and the coordination of Agile activities content.
4. To align the strategic business objectives of the enterprise with the top-level Agile activities and identify functional requirements of the EAS

projects by using enterprise architecture frameworks for the decomposition of strategic business objectives.

5. To compose a taxonomy of the mutual horizontal and vertical interactions of the activities of the modified Agile hierarchy based on which EAS project execution state could be evaluated against strategic business objectives of the enterprise using the developed qualitative evaluation parameters.
6. To design the architecture of a modified Agile project management method environment, to develop a prototype of a modified Agile project management system with the functionality of monitoring and evaluating EAS design solutions and conduct an experimental evaluation of the proposed method.

## Research Methodology

The research goal and methodology are formulated by summarizing extensive experience in EAS development project management and more than six years of experience using the Agile management method and tools for the implementation of business projects in various fields.

The focus of the research is to improve the alignment of application software development solutions with enterprise (business) strategic goals using an Agile project management approach and tools, specifying the main components of intelligent project management system architecture.

This is achieved by shifting real-world domain modelling and EAS development to a causal modelling paradigm [34, 49, 50] and thus incorporating the business domain causal knowledge into the Agile project management process. The business domain interactions and dependencies between Agile management activities are explored using a causal knowledge framework, namely the management transaction (MT) construct.

External/Internal modelling meets the "grey system" model in Grey System Theory [51, 52, 53]. The methodologies used in the grey box approach are of a different level of uncertainty (greyness) and some examples as business rules or ontologies are dedicated to moving from grey towards white box modelling. The boundary case that turns grey into a white box is the usage of causal knowledge in modelling [54].

The concept of causality was scientifically researched for over a hundred years as in [55]. Schurz and Gebharter [56] provided a philosophical justification of causality and used Bayes nets (TCN) as a theory supporting it. Causality is an important concept in modern science and advanced

technologies, it helps to reveal the domain properties hidden to the outside observer and to construct relevant models and IT supported systems [57].

Applying the MT framework reveals the structure and information content of Agile management interactions, thus giving the basis for a taxonomy of coordination inside Agile hierarchy and the specification of project state evaluation indicators.

Another component of research methodology is to integrate the enterprise architecture framework for the consistent decomposition of strategy into a set of capabilities [43, 46], which specify high-level requirements for the strategy-related business activities and workflow of processes (operational models). The next step is to map an identified capability set to the causal Agile hierarchy.

The conceptual model of the overall causal Agile management approach is based on the FRISCO tetrahedron [58].

### Defended Propositions

Propositions to be defended by the research:

1. Enhancing the Agile activities hierarchy modelling of enterprise strategies creates the prerequisites for analysing the compatibility of the development solutions of the ongoing EAS project with business management needs in a virtual environment.
2. The modified Agile process model, based on a unified (normalised) internal structure of all Agile activities (theme, initiative, epic, user story), defined as a management transaction MT, allows the unification of specification forms of Agile activities. This enables software development to analyse EAS project content, i.e., evaluate the state of EAS project specifications without human involvement.
3. The modified Agile process activities (theme, initiative, epic, user story) have a unified internal structure and are aligned with models of enterprise strategies and operational processes (defined in the enterprise architecture framework), which gives the possibility to specify the knowledge base of the intelligent project management tool.

### Major Contributions and Novelty

1. A modified Agile process was developed using the causal modelling method, defining EAS project management activities as a hierarchy of management transactions MT = (F, P, A, V).

2. The modified Agile method is supplemented with structural activity models, linking EAS project activities with enterprise strategy and operational models using enterprise architecture framework MODAF, thus ensuring the alignment of enterprise management strategies with EAS project solutions and automatic project management monitoring based on the knowledge models of the business domain.
3. The activities of the modified Agile process (theme, initiative, epic, user story) are described as management transactions with a typical internal structure MT = (F, P, A, V), which allowed defining the EAS project states (types of coordination) and quantification parameters.
4. Software elements (models) created, necessary to implement a knowledge-based Agile project management component based on the modified Agile process:
    - user interface layout – data panel (dashboard) elements;
    - knowledge base (model) for specifying the elements of the modified Agile process.
5. Indicators were introduced to evaluate the relative EAS project complexity based on typical EAS project specifications. The indicators allow for evaluating the EAS project complexity based on the interaction specification content quality and their average and normalized values are presented.

Practical Significance

Based on the developed classification of interactions of Agile activities, the quantitative evaluation metrics for EAS project alignment to business management requirements were defined.

An Agile project management prototype component to monitor EAS project requirements alignment to the business strategy objectives was developed.

Scientific Approval

Research results were presented at five international and five national conferences. The section “List of publications” contains a detailed list of the published publications on the topic of the dissertation; three articles were published in peer-reviewed conference proceedings:

1. Noreika, Karolis. Improving enterprise application software development management with MODAF. Joint proceedings of the BIR

2021 workshops and doctoral consortium co-located with the 20th International Conference on Perspectives in Business Informatics Research (BIR 2021), Vienna, Austria, September 22-24, 2021. Vienna. 2021, p. 141-152.

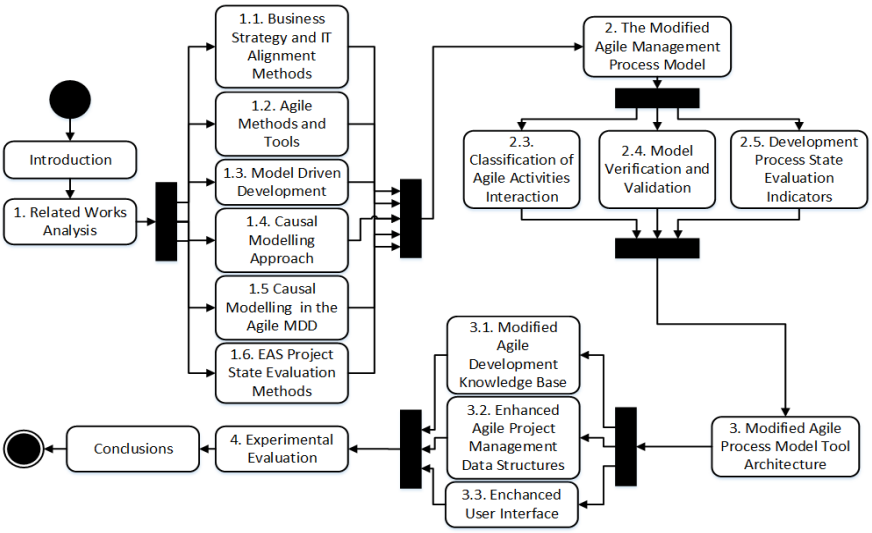
2. Noreika, Karolis; Gudas, Saulius. Modelling the alignment between Agile application development and business strategies. Joint proceedings of the BIR 2021 workshops and doctoral consortium co-located with the 20th International Conference on Perspectives in Business Informatics Research (BIR 2021), Vienna, Austria, September 22-24, 2021. Vienna. 2021, p. 59-73.
3. Noreika, Karolis. Business capabilities utilization enhancement using Archimate for EAS projects delivery in an agile environment. Baltic DB&IS 2020 Conference Forum and Doctoral Consortium co-located with the 14th International Baltic Conference on Databases and Information Systems (BalticDB&IS 2020), Tallinn, Estonia, June 16-19, 2020: proceedings. Tartu: University of Tartu. 2020, p. 49-56.

One article was published as a chapter in the Springer book series classified as other peer-reviewed periodicals, continuous or one-time scientific publications.

Two articles were published in journals referred to in Clarivate Analytics Web of Science database publications with a citation index.

Thesis Structure

The thesis structure and main parts are provided in the research schema in Figure 1.




Source: Created by the author

Figure 1. Research schema

Section 1 contains related works focusing on business and IT alignment methods, Agile methods and tools, model-driven development, causal modelling and EAS project state evaluation methods. Section 2 describes the causal modelling-driven Agile management method. This section presents the theoretical background of the research, the modified Agile hierarchy, EAS project state evaluation metrics and model verification and validation. In Section 3 the enhanced Agile management tool architecture is presented. In this section, the modified Agile management method knowledge base structure is presented together with the modified Agile project management data structures and an enhanced user interface to support the method is introduced. The thesis experimental evaluation on 4 EAS development projects is presented in Section 4.

This thesis contains 191 pages including a summary in Lithuanian on page 158. It includes 53 figures and 28 tables.

# 1. RELATED WORKS

Many attempts have been made to improve business management and IT capabilities alignment. Most notable of them are described in this section.

## 1.1. Business and IT Alignment Methods

Business management and IT capabilities alignment methods are generally focused on communication and leadership. They do not contain any formal structure and are based on the managerial aspect, written or verbal communication and "common sense" and "are not sufficient to ensure that the information defined as strategic objectives of the enterprise is successfully transferred to a software development task to ensure business and IT alignment" [59].

### 1.1.1. Business Management Methods

Strategic business management is generally based on management approaches, decisions of top leadership in the organization, and the ability to transfer these decisions from the top level throughout the hierarchy of departments and employees. Alkhafaji and Nelson [60] state that "Strategic management is the process of assessing the corporation and its environment in order to meet the firm's long-term objectives of adapting and adjusting to its environment through manipulation of opportunities and reduction of threats".

"Organizational capability" dates back to 1987 when Ulrich [61] described as "the firm's ability to manage people to gain a competitive advantage" that connects the financial, strategic, and technological capabilities of the organization. A business capability is an endeavour that helps an organization achieve its goals. The most efficient business strategy execution is to utilize key organizational capabilities often based on the level of competence in a department of the organization or individual domain knowledge of people in the organization.

Interdependencies between different tasks cause coordination problems. An early definition of coordination can be found in Van de Ven et al. [62], who defined coordination as "integrating or linking together different parts of an organization to accomplish a collective set of tasks". Taxen and Riedl [63] provide a theory of conceptualization of coordination in the IS domain based on a neurobiological perspective.

Classical organization theory includes the scientific management approach, Weber's [64] bureaucratic approach, and administrative theory.

The IT infrastructure and installed enterprise application software are dedicated to supporting business management activities.

The discussed business management methods and theories mostly rely on communication between different levels of managers in an enterprise and are not sufficient to ensure that the information defined as strategic objectives of the enterprise is successfully transferred to a software development task to ensure business and IT alignment.

### 1.1.2. Business and IT Interaction Modelling Methods

Enterprise modelling, business management modelling, and organizational systems theory are interrelated areas of research of complex systems with an active human component (goal-oriented behaviour).

Business and IT capabilities alignment has been a topic of research for a long time. A very early attempt to tackle the problem was the Business and IT strategic alignment model (SAM) by Henderson and Venkatraman [65]. SAM was proposed to represent business strategy alignment with IT strategy. SAM consists of four domain alignment perspectives (Business strategy, IT strategy, Organization infrastructure and processes, and Information Systems infrastructure and processes). Each domain alignment perspective focuses on a different aspect of the business and IT alignment. However, it is quite fuzzy and focuses mostly on the business management part of the problem. It should be noted that the Business and IT Strategic Alignment Model (SAM) is the early attempt to identify causal dependencies between two enterprise domains: business activities and IT-based activities.

One of the most well-known methods for business and IT interaction modelling is a conceptual framework for architecture alignment guidelines Guidelines Regarding Architecture Alignment (GRAAL). It is based on SAM. GRAAL is a conceptual framework providing a collection of concepts and relations among them [40]. Gerow et al. [66] identified 6 definitions of alignment, i.e.: Business alignment, Cross-domain alignment (business strategy to IT infrastructure and processes), Cross-domain alignment (IT strategy to business infrastructure and processes), Intellectual alignment, IT alignment, Operational alignment based on Henderson's and Venkatraman's SAM [65]. However, these methods are conceptual and not adapted to be used in most popular enterprise architecture modelling tools.

Enterprise modelling perspective includes such approaches as Service Oriented Architecture (SOA) [67], enterprise architecture frameworks

(DODAF, MoDAF [43], TOGAF [44]), business process modelling (UML, BPMN [68]), and decision modelling notation (DMN) [69]. Also SOA based methods like BITAM by Chen et al. [41]. BITAM uses a twelve-step process for managing, detecting, and correcting misalignment at the architecture level. Also, the SBISAF framework by Morkevicius et al. [42] is an SOA-based framework that has its implementation in the MagicDraw CASE tool using UPDM modelling language and proved to significantly reduce the misalignment between business requirements and IS solutions.

DMN is integrated with BPMN and aims to identify and specify decision-making logic (rules). The first version of the DMN specification was created in 2013, almost a decade later than BPMN (in 2005). From the systems analysis perspective, BPMN can be considered only as an external (black box) modelling approach. However, DMN can already be assigned to the internal modelling paradigm, which allows specifying grey box (white box) models.

It is evident that overall business processes and enterprise modelling standards are shifting from external modelling towards the internal modelling paradigm.

The purpose of the research is to create an effective EAS project solution alignment to business strategy method and tools that enables managing the project progress and allow us to evaluate the EAS project state against the strategic objectives of the organization.

The mentioned methods are not adapted for use in the most popular enterprise architecture modelling or software development management tools, so complex work with a very large amount of information falls to a human (expert).

The business analysis models still need to be translated to project requirements or the requirements themselves should be adjusted manually. The Agile management concepts are also not taken into account; therefore, a different method is required to address these issues. Furthermore, it takes a lot of time to assess discrepancies between the EAS design solutions and strategic business needs, thus causing a high risk of missing critical situations.

### 1.1.3. Enterprise Architecture-Based Software Engineering

Enterprise Architecture (EA) is one of the most widely recognized methods to ensure business strategic needs and IS functionality alignment. Dahalin et al. [70] state that EA provides “a blueprint for how an organization achieves the current and future business objectives using information technologies”. Enterprise architecture development is a well-defined approach for conducting enterprise analysis, application design, and implementation of

relevant IT necessary to execute business strategies, to guide organizations through the business and technology changes. The history of "Enterprise Architecture" started in 1987 when J. Zachman coined the term [71].

Different enterprise architecture frameworks (DODAF, MODAF, NAF, ArchiMate, etc.) cover slightly different perspectives and aspects of enterprise activities and supporting systems. Recent researches show the topicality of enterprise architecture and a need for even further research [72].

Kim et al. [73] describe enterprise architecture as "a holistic understanding of all aspects of a business, its drivers and surrounding environments, its business processes, organizational structures, information flows, IT systems, and technical infrastructures". A properly defined EA allows businesses to identify and utilize their capabilities in executing business strategy. Enterprise architecture framework models are developed using various enterprise modelling languages like EEML, UEML, and ArchiMate. An approach to business and IT alignment in an Agile environment using concepts from Scrum, Scaled Agile frameworks, and ArchiMate was presented in [74].

The British Ministry of Defence Architecture (MODAF) is an Enterprise Architecture framework used for developing complex systems: strategic and operational models, and supporting software applications. MODAF is the most widely known and established framework. MODAF uses the concepts of views (Strategic, Operational, System, Acquisitions, Technical Standards, and All views) together with a set of models that help model the entire enterprise and its operations. One of the key concepts is the "capability" concept. The strategic views are focused primarily on capability identification. The term "capability" is defined in EA frameworks (DODAF, MODAF, NAF, ArchiMate, TOGAF). Capabilities are named and defined in business terms. They may be composite, including lower-level sub-capabilities. In MODAF, a "capability" is a high-level specification of the enterprise's ability [43]. Capabilities rarely change over time; the main underlying idea is that capabilities enable the organization to act and succeed in the preferred domain based on the skill set and abilities the enterprise has.

A publication describing the state-of-the-art works in the field of Enterprise Architecture mining including usage of Enterprise Architecture for business and IT alignment was published by Perez-Castillo et al. [75]. The research demonstrated that currently developed Enterprise Architecture Frameworks (EAFs) do not resolve the problem of business and IT alignment and still validate that the issue being researched is relevant.

An important observation is that business domain knowledge (including strategies and objectives) is modelled in a systematic and orderly manner using EAFs. The disadvantage is that traditional EA frameworks are based on

external modelling methods, empirical models are created, and therefore causal domain knowledge is not discovered as stated by Gudas [76].

From the perspective of causal modelling, it is necessary to move to an internal modelling paradigm, use different methods, and supplement EAF with new causal knowledge modelling constructs. One such method to supplement EAF with causal modelling constructs is described in [76].

### 1.1.4. Causal Modelling of Agile Activities

Causality is an important concept, it denotes the inherent properties (regularities) of the domain hidden from the outside observer. The internal modelling paradigm is a prerequisite for discovering causality in various real-world domains and developing advanced (smart, intelligent) systems of all types: physical systems, enterprise systems, biological systems, etc.

The paradigm of internal modelling requires new modelling methods to reveal domain internal (causal) dependencies. Discovered domain causal dependencies is a causal knowledge. Causal knowledge can be described formally as a causal model in mathematical expressions or as a conceptual model (framework, meta-model).

The notion of causality is considered a system of cause-effect relationships relevant to real-world regularity in the specific domain as described in [76].

Different levels of causal knowledge can be defined as [76]:

1. A cause-effect relationship model;
2. A chain of cause-effect relationships (a workflow model);
3. A circular causality model (model with a feedback loop, transaction)

Usually, the notion of causality is considered as a single cause-effect relationship or as a chain of cause-effect relationships relevant to the real-world regularity in the specific domain.

A cause-effect relationship model is a very simple type of causal model, a kind of (Input, Process, Output) model. A cause-effect relationship is a conceptualisation of the causal dependence of the real world: it is a relationship in which one event (process, activity, i.e. the cause) makes another event (process, activity) happen (the effect). One cause can have several effects. A separate cause-effect relationship is not yet “causality” in the full sense (scientific law, scientific explanation), it is only a fragment of the chain, an item of the causal knowledge. A chain of cause-effect relationships (or workflow model) is a more advanced understanding of causation, it is called a „one-dimensional cause-and-effect chain“ [77].

However, this type of model (workflow model) is not sufficient to describe systems that are characterized by the self-management feature, because this

feature is based on the feedback loop interaction between system elements. In the case of an organizational system (enterprise), it is needed to use such modelling methods that can identify the characteristics of circular causality.

An important idea was that proper modelling of complex systems with self-organization features (i.e. CPSS, CES) requires “the simple notion of one-dimensional cause-and-effect chains” to be replaced by “the two-dimensional notion of a circular process” [77].

Circular causality is a key feature of some well-known business management models: Deming’s PDCA cycle (Plan, Do, Check, Act) [32], interactions of primary activities and support activities in Porter’s Value Chain Model [78], Harmon’s business management model [79] and Rummler-Brache model of enterprise management [80]. Thus, to achieve the goals of this research, a more complex causal model is required – a circular causality model.

Circular causality is abstracted in the concept of “transaction” in systems engineering and software engineering. Circular causality is a very basic concept in the science and research of real-world phenomena. In control theory, a circular causality is formalized and represented as a feedback control loop of the control system. “A transaction in the context of a database, is a logical unit that is independently executed for data retrieval or updates” [81]. A transaction is a key concept for discovering deep properties (causality) of the subject domain. A management transaction in the context of enterprise causal modelling, is a logical unit that is an independent (management and control) cycle of activities interaction [76].

The subject domain in this thesis is organizational systems, which are a type of complex system, also called "enterprise" in software engineering methodologies. An enterprise is defined as “one or more organizations sharing a definite mission, goals, and objectives to offer an output such as a product or a service” by ISO 15704:2000 standard [82].

The understanding of causality modelling is summarized taking into account that the subject domain is enterprise. The causal model of enterprise is constructed using a deep knowledge of the inherent properties of that domain type.

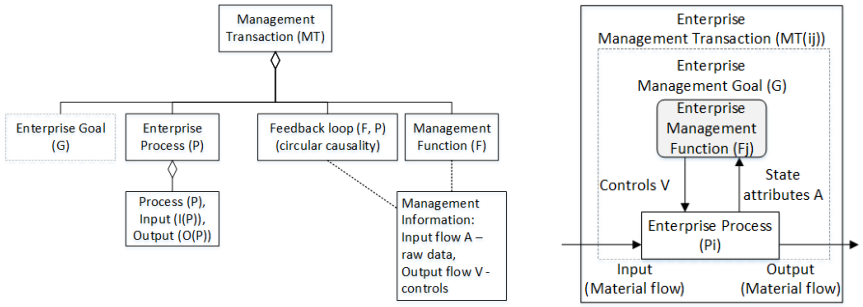
The research is based on the causal modelling methodology of enterprise management described in [34, 49]. The main causal modelling concept adopted in the thesis for Agile activities modelling is the management transaction (MT) [34].

Two levels of enterprise causality modelling were introduced in [34, 49] The first level is the presentation of the discovered causation using the Management Transaction (MT) framework. At the second level, a deep

knowledge structure of MT is revealed in a more detailed framework called the Elementary Management Cycle (EMC) [34]. In this thesis, only MT will be used.

MT framework is used only as a unified component of causal knowledge (deep knowledge) for an enterprise management model. MT is relevant to some enterprise goal (G) and captures knowledge on enterprise process (P), a feedback loop (F, P), informational input flow (A), informational output flow (V), and management function (F): MT = (F, P, A, V).

The conceptual structure of MT is presented in Figure 2 (a). The conceptual causal model presented in Figure 2 (b) reveals the internal steps and information flows of MT and also shows a feedback loop of information transferring in-between F and P.




a) b)

Source: [34]

Figure 2. Management transaction: a) conceptual structure, b) internal model of management transaction

The goal of this thesis is to apply the taxonomy of enterprise coordination situations and to define the semantics of management interactions based on the content of the information being transmitted. The content of self-managed Agile activities is conceptualized as the MT, which specifies the internal causation of activities.

## 1.2. Agile Methods and Tools

Agile software development management is also considered part of business management methods as they aim to ensure customer value through regular software delivery. The software development projects are aimed at implementing business strategy. Over the years, Agile frameworks such as Scrum, Extreme Programming (XP), ScrumBan, where Scrum is mixed with Kanban framework, Scaled Agile Framework, Spotify model or hybrid

versions of these and other methods gained popularity because of adaptability to change and reduction of project costs [83].

EAS projects managed using Agile methods have distinct definitions that represent the hierarchy of tasks in an EAS project to ensure the business needs are aligned with the development of enterprise application software. Agile software project management professionals and vendors use various namings for the elements in the hierarchical structure. Usually, there are four levels used to define the task size and each different level has a distinct definition The levels in the hierarchy from themes to user stories are also known as TIES, standing for Themes, Initiatives, Epics, and user Stories [48, 84].

The lowest level of granularity is defined by the term "user story". User stories provide not too many details and also not too wide a description of the business problem. It was originated by Connextra in the United Kingdom and popularized by Cohn [85]. User stories should span no longer than the single development iteration (1 to 4 weeks). The next level in the Agile software project management hierarchy is epic. Epic is bigger in terms of the development capacity required than the available capacity for development iteration [86], [87]. Epics should span no longer than 12 weeks. The next level is initiative. Atlassian, the top vendor of tools for software development management, describes the initiative as "a collection of epics that drive toward a common goal" [87]. Initiatives provide business and IT perspective on achieving business goals and should not span more than 1 year. Theme is the top level in Agile software project management. It is defined as "an aggregation of user stories to show business value delivered and to help with prioritization as well as show planned product delivery at a high level" [88]. Themes can be considered as the level between initiatives and strategic business objectives. Themes usually take up to two years to implement.

The links between TIES structure elements and the strategic business objectives are currently defined by communication between project managers, Agile leaders, and business managers. To ensure business and IT alignment, an additional definition of coordination between business and IT management is required. A large number of informational units need to be coordinated during the Agile software development project to achieve the strategic business objectives. As an example, information elements distribution from a real-world Agile EAS development project is presented in Table 1.

Table 1. Typical Agile software project management hierarchy

| **Hierarchy level** | **1** | **2** | **3** | **4** | **5** |
|---|---|---|---|---|---|
| Agile concept | Strategic business objective | Theme | Initiative | Epic | User story |
| Description | Short statement explaining long term business goal and having a link to the vision and mission of the company. | High level objective with time constraint and usually monetary goals that allow to achieve strategic business objective. | Specific activity to contribute to achieving the goals on the theme level. | Functionality description to achieve goals of linked initiative. | Task representing business need or problem. |
| Number of objects | 1 | 1-5 | 5-15 | >300 | >3000 |

Source: Created by the author according to [48][87][88]

Agile methods provide a good way to manage changes and provide transparency to the enterprise application software development process, but it does not on its own ensure that development tasks are aligned with the strategic objectives of the enterprise.

Many tools support the Agile software development project management process. Atlassian Jira software is one of the leaders [83]. The current state of Jira only supports the themes, initiatives, epics, and user stories structure. It does not provide formalities to ensure that the links between the different activities in the levels of Agile hierarchy are properly identified and justified in the business context. It means that it is not ensured that each user story, epic, initiative, and theme links to one of the strategic business objectives. Links are determined by experts working on project delivery using their experience and knowledge.

Microsoft Azure DevOps is also one of the most popular Agile software project management tools, mainly due to its vast capabilities for managing software development itself – i.e. CI\CD, code writing, pipelines, etc. However, the structure for Agile software project management in Microsoft Azure DevOps is simple and does not contain the business context knowledge in any formal way. Microsoft Azure boards – a part of Microsoft Azure DevOps uses the structure of epics, features, user stories, and tasks.

The top five tools for Agile project management were investigated in order to understand how the structure of Agile project hierarchies linking them to strategic business objectives are defined and are presented later.

### 1.2.1. Scaled Agile Frameworks

Scaled Agile frameworks are primarily dedicated to ensuring alignment and coordination across the enterprise and introduce practices for multiple team collaboration using Agile frameworks. The most popular Agile frameworks for scaling are Large Scale Scrum (LeSS) and Scaled Agile Framework (SAFe).

In the case of the Scaled Agile Framework, the responsibility of ensuring that strategic objectives are aligned with team deliverables lies with the role of the release train engineer. This person is responsible for "aligning and facilitating an ART" (team-of-Agile Teams) [21]. However, such an explanation does not contain formalities and information structure specifications to ensure the alignment between business and IT.

In the case of LeSS [19] ensuring that development tasks adhere to the strategic business objectives lies with communication as coordination is defined as "Just Talk, Communicate in Code, Travelers, Open Space, and Communities." It means that one has to talk as needed and only about what matters at the time (Open Space) and otherwise promote cross-team interactions through sitting with another team (Travelers) and forming communities around shared interests" [84].

However, both of these methods still lack the formal specification to ensure Agile activities are contributing to a specific long-term goal of the organization and the coordination aspect is left to human communication.

Current most popular Agile project management tools (Jira, Azure DevOps, etc.) do not contain the metrics for evaluating the project state and content against the organizational strategic business needs expressed as enterprise models.

In Table 2, the comparison of the most popular agile frameworks is presented and analysed from the causality perspective. The relationships defined as MTs are based on the feedback loop consisting of state attributes and management control flows.

Table 2. Agile frameworks comparison from the causality perspective

| **Agile modelling construct types** | **Scrum** | **SAFe** | **LeSS** | **Causal Agile process** | **Comments** |
|---|---|---|---|---|---|
| Business strategy | Not defined | Not defined | Not defined | Defined | Decomposed as a capability set |
| Theme | Theme | Match | Match | Theme | |
| Initiative | Initiative | Match | Match | Initiative | |
| Epic | Epic | Match | Match | Epic | |
| User story | User story | Match | Match | User story | |
| Task | Task | Match | Match | Task | |
| Business domain model | Not defined | Not defined | Not defined | Yes | Linked to Agile constructs |
| **Vertical relationship types (control)** | | | | Yes | |
| Business strategy/ Theme | Not defined | Not defined | Not defined | Defined | Mapping capabilities to themes |
| Theme/ Initiative | No | Not defined | Not defined | Defined as MT | MT(T,I) = (F, P, A, V) |
| Initiative/Epic | No | Not defined | Not defined | Defined as MT | MT(I,E) = (F, P, A, V) |
| Epic/User story | No | Not defined | Not defined | Defined as MT | MT(E,U) = (F, P, A, V) |
| **Horizontal relationship types (Same-level coordination of activities )** | | | | Yes | |
| Theme/ Theme | Not defined | Not defined | Not defined | Defined | Coordination types defined |
| Initiative/ Initiative | Not defined | Not defined | Not defined | Defined | Coordination types |
| Epic/ Epic | Not defined | Not defined | Not defined | Defined | Coordination types |
| User story/ User story | Not defined | Not defined | Not defined | Defined | Coordination types |

Source: Created by the author according to [34][38][89]

Based on Table 2, the results were summarized and appended in Table 3.

Table 3. Agile project management tools capabilities comparison

| Tool | Agile structure | Linking tasks to strategic objectives | Business domain knowledge base |
|---|---|---|---|
| Atlassian Jira | Theme, initiative, epic, user story | – | – |
| Atlassian Jira Align | Theme, initiative/epic, feature, user story, task | – | – |
| Azure Boards | Epic, feature, user story, task | – | – |
| Enhanced Agile project management tool | Theme, initiative, epic, user story | Specified | Included |
| Microsoft Project | Not specified | – | – |

Source: Created by the author according to [83][87][90][91]

Originally, two more tools belong in this list based on the State of Agile report of the most popular Agile project management tools [83]:

- Mural/Miro are tools for whiteboard-style collaboration and practically any information can be put there, including links between tasks, but they need to be defined by the user (there is no specification in the tool itself).
- Microsoft Excel is primarily a calculation and data analysis tool. Despite having vast integration capabilities, including integration with Atlassian Jira, it still does not qualify as an Agile project management tool.

Having analysed all the Agile project management tools in Table 3, it was identified that the structure for Agile software project management is simple and does not contain the business context knowledge defined in any formal way.

Overall, the Agile methods and tools lack the definition of business context and therefore another approach to the current structure of Agile tools and methods is needed.

### 1.2.2. Traceability of Business Strategy to IT Development

Business strategy execution is a complex process of acting on the defined strategic plan in an effort to achieve strategic business goals that are derived from the vision and mission statement of the organization. It is an ongoing activity during the whole lifespan of the enterprise.

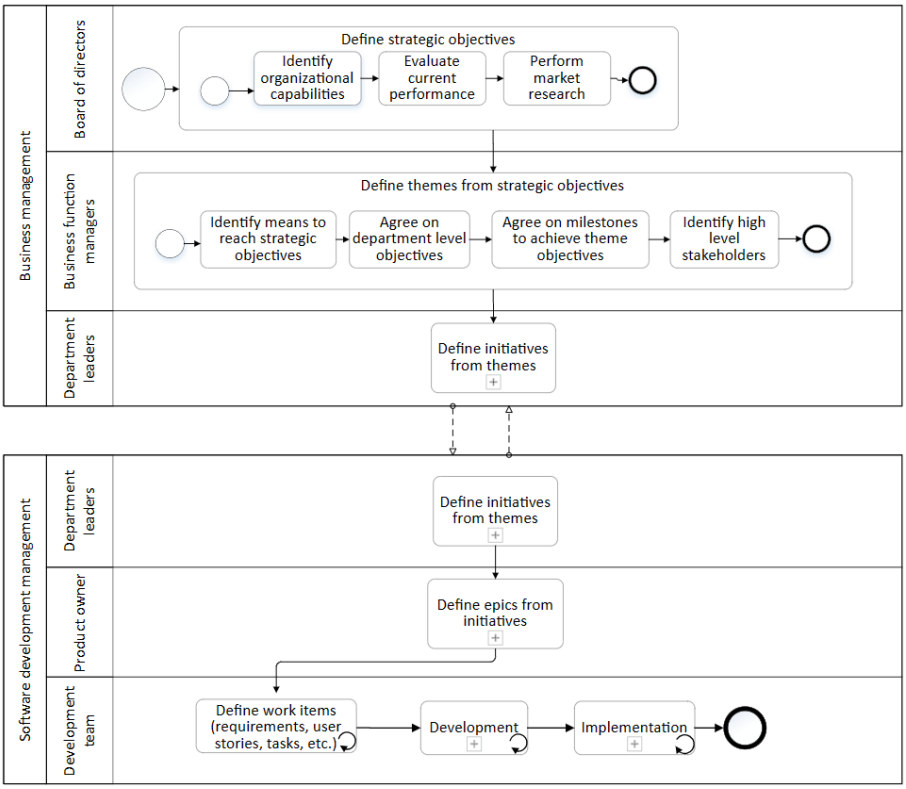
Coordination is a common problem in managing business and controlling the activities of an organization. It is considered a very high-efficiency factor for traditional organizations (enterprises), virtual organizations, and cyber-enterprise systems (CES).

One of the problems in Agile project management using the TIES approach is ensuring the indication of the business strategic objectives are implemented as monitoring of this step is not supported by Agile management tools (such as Jira). The implementation of strategic objectives is dependent on experts who analyse the content of the strategic objectives and specify the top-level Agile activities – themes.

When the business execution process goes from strategic objectives to themes in the Agile management TIES context, market research is conducted to ensure the planned themes are what the market is expecting from the enterprise. Also, enterprises evaluate their capabilities, and strengths to ensure these capabilities are utilized in order to achieve strategic objectives. Typically, in the business execution process achieving a strategic objective takes effort from various business functions like marketing, logistics, etc., and very often – IT capabilities. Improving existing or creating new IT solutions makes a direct impact on achieving strategic objectives as customers need service on-demand with fewer approvals or any other delays in their experience.

When the business execution process goes from themes to initiatives, business function managers evaluate the relevant capabilities and distribute the high-level tasks to appropriate department leaders. In the case of the IT department, this could be reducing costs through “sunsetting” – a list of legacy applications by either implementing features in more modern EAS or by optimizing the business process so that it does not rely on the legacy system. Once moving through the business strategy execution process from initiatives to epics this will be units of a smaller work effort like replacing a feature in some old legacy software with more modern tech stack tools. A single task to contribute to is usually called a user story and reflects the business need from the stakeholders' perspective. In the example case, this may be just moving some software components to a new tech stack.

The described business strategy execution process could be presented as a BPMN diagram (Figure 3)

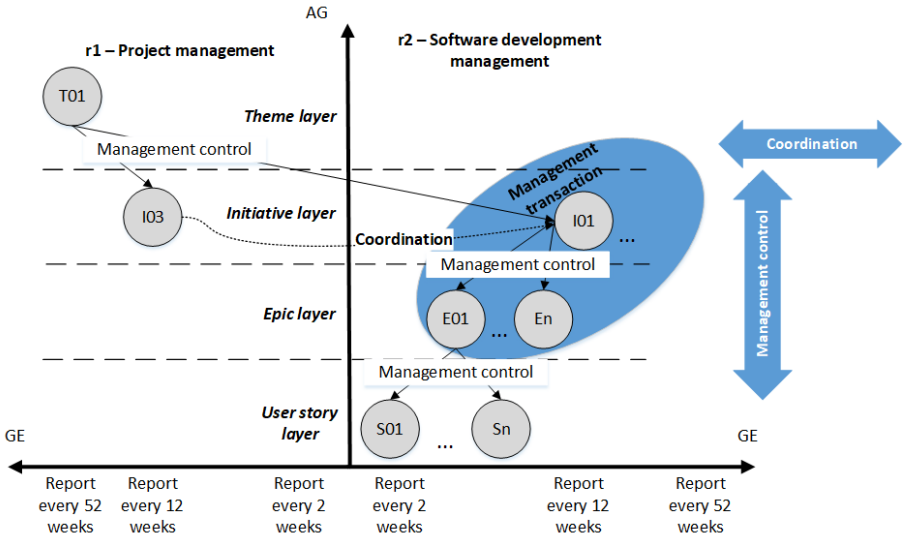



Source: Created by the author

Figure 3. Business strategy execution process transformation

"Development" in Figure 3 means the software development activities: engineering, coding, and testing the EAS or its components. "Implementation" in Figure 3 means following the established procedures to put the tested solution (the result of development) into a production environment that is readily available for end-users. The message flows between separate lanes indicate the interaction of information between different levels of Agile hierarchy and contain a feedback loop as presented in Figure 2. The Agile project management process can also be described as fuzzy.

The actual business and IT alignment is happening on the level of "Department leaders" as in Figure 3. This interaction between the two types of management functions requires additional coordination between business management and software development management activities as displayed in Figure 4.




Source: [50]

Figure 4. Business management and software development management coordination relationship

Business strategy execution takes place on the theme level depicted as T01. I03 is a business management initiative that defines assignment I01 for enterprise application development. A coordination link is required between the business management process in initiative I03 and the Agile software development management process in initiative I01. Agile activities I01, E01..En, and S01..Sn is the software development management task hierarchy TIES as described previously in this thesis.

The Agile activities hierarchy is a static structure in terms of its components: themes, initiatives, epics, and user stories, and it does not represent the dynamics of the system. In our approach enterprise architecture is also considered as a static structure that details Agile concepts in this step of transformation.

## 1.3. Model Driven Development

Modelling domain externally is based on model-driven development (MDD) and model-driven architecture (MDA). MDA describes a way how application system is developed by separating the domain knowledge acquisition stage (layer), requirements specifications, and system design layers from its implementation on a specific platform [92].

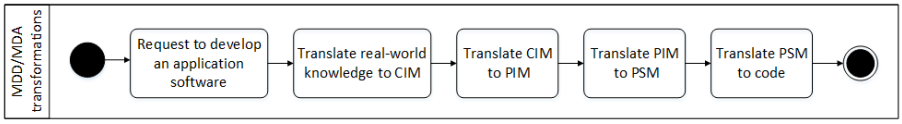
Model-driven architecture (MDA) is the OMG's particular approach to MDD. Although MDA addresses the aspects of MDD there is a big gap in its implementation as there is a lack of causality aspect being involved.

Modelling the domain externally is based on the model-driven development (MDD) paradigm. External modelling means addressing only input-output results without understanding the inner workings (patterns, rules) of the system. This requires causality modelling and thus causal-based MDD.

By using the defined OMG standard for MDD – MDA and enterprise architecture frameworks that are integrated to develop an intelligent MDD process.

It enables us to automate the development of EAS starting from the gathering of requirements that are transformed into models and the structure of the new system is generated, which can be implemented on different platforms. The method is largely based on the possibility of generating a system structure using the UML model, therefore full compatibility with UML models is required. However, this approach has limitations in understanding the laws behind the processes. The business domain must be understood more deeply than just by external observation (i.e. black box or grey box approach) in order to correctly represent the real-world processes and make them intelligent by bringing out the inherent properties of a business domain. Characteristics of a definite type of real-world domain (i.e. enterprise) must be understood and any virtual representation of it (model, simulation) must follow those characteristics. Such an approach is becoming more and more popular in physical or cyber-physical systems, and cyber-biological systems. The conceptual approach to MDA is illustrated in Figure 5.

The MDA approach uses the abstraction layers as follows: Computation Independent Model (CIM), Platform Independent Model (PIM), Platform Specific Model (PSM), and model transformations (inside each layer and between these layers) are defined. MDD is a model transformation process aimed at generating (semi-automatically) the target model from the source model [93].




Source: Created by the author

Figure 5. The conceptual scheme of model-driven architecture

MDD is a very promising methodology for the development of cyber-physical-social systems (CPSS), cyber-enterprise systems (CES), cyber-physical systems (CPS), and other types of complex systems.

Over the years, Kulkarni et al. [94] presented several solutions aiming to improve the development of their component-based development environment Mastercraft. Mastercraft is a model-driven development toolset for developing large database-centric business-critical applications. It contains a meta-modelling tool to specify an abstract view. This represents the capture of predefined knowledge (i.e. grey box approach). The MDA lies in one of the components of the Mastercraft tool – i.e. a set of code generators that transform each view instance to the desired implementation artifacts. Barat et al. have presented a configurable code generator meta-model to transform models into code [95]. Although some information that is required to smoothly conduct transformations is missing (i.e. "various non-functional concerns, namely, design strategies, technology platform, and architecture") the model described in [94] is well-defined to be used for the MDA approach.

The model transformation rules specify the mapping between MDA layers and lead to the last step, i.e. specification of the computer code for the implementation of the solution. The transformations between CIM, PIM, and PSM layers must comply with relevant meta-models.

Prior knowledge is defined as all the knowledge one has before modelling a particular domain. Deep prior knowledge and causal knowledge are knowledge related to causality that is more or less known or discovered.

External/Internal modelling refers to the intermediate area of domain modelling between the black box approach and white box approach and is therefore named the grey box approach. It is based on several modelling methodologies that aim to move the domain knowledge from fully observation-based (i.e. black box approach) to causality-based (i.e. white box approach).

The modelling notations and tools used for external/internal modelling are:

- Formal modelling, that is based on formalization of domain regularities through mathematics, i.e. using set theory, fuzzy sets, Petri nets, ontologies, etc.
- Knowledge-based modelling – cause and effect based. Examples: domain behaviour rules, patterns, meta-models (in software engineering this would be the analogue of ontology).

From the point of view of the causal modelling perspective, model-driven development (MDD) is based on the model-driven architecture (MDA) as external domain modelling (black box) approach. As external modelling is

based on input-output analysis only, it does not evaluate the inherent properties (causality) of the object under analysis. This prevents the analyst from understanding the patterns and properties of the system and, thus, does not allow to reflect the modelled object as closely as possible to the real-world object.

Internal modelling includes some prior knowledge of the system attributes, its behaviour, and internal dependencies. As causal modelling is considered to be a white box type of modelling that is aimed to identify and specify the inherent properties of the modelled object, this thesis aims to utilize such an approach.

### 1.3.1. Agile Model Driven Development

Agile Model Driven Development (AMDD) is an attempt to use the benefits of the fast pace and responsiveness to change Agile development and the focus is on the quality and stable structures of Model-Driven development. Ambler coined the term Agile Model Driven Development [96]. Examining the blended concepts of Agile and MDD it should be noted that MDD is a model-centric approach and values the thoroughness of models which take time for development. Agile, on the other hand, values working software, and the attention to models is little to none. Using the Agile approach, extensive models are considered as not required, and as Ambler states [97] even with the AMDD approach, “an agile model is a model that is just barely good enough”. Agile models need to exhibit a set of traits like fulfilling purpose, being sufficiently detailed, etc. However, it is not explained how the knowledge is obtained and there is no mention of performing model transformations as required by the MDD approach. This shows that model transformations are left to the MDD side of the AMDD approach. Furthermore, Ambler states that in order to perform modelling, including Agile modelling fundamental information-gathering skills are required and names observation as one of them [97]. This leads to the conclusion that Agile modelling is external observation-based (black box). Ambler also stated that “AMDD is <only> about modelling MDD models more effectively” [97].

Kulkarni et al. [98] introduced a Meta Sprint-based software development approach. Results show that there was an increase in turnaround time from requirement specification to delivery, the productivity of the development team, and less rework of their developed tool Mastercraft. However, this approach is very similar to the dual Scrum or dual Agile approach [99] which later evolved into the so-called “design sprint” [100]. The dual-track Agile and design sprint was inspired by the double diamond methodology [101]

formulated by the British Design Council. Furthermore, the approach by Kulkarni et al. does not describe the causal knowledge that the meta-model should explain.

Matinnejad [102] presented criteria for how to evaluate the attributes of the fast-paced and flexible approach of Agile and the thorough, well-thought, and complete approach of MDD and analysed the AMDD processes of Sage [103], Hybrid MDD [104], MDD-SLAP [105] and the AMDD high-level life cycle [97].

After thorough research, to the best of the author's knowledge, there is no attempt to add the component of intelligence (causal knowledge capture) to Agile MDD methods.

### 1.4. Causal Modelling Perspective in the AMDD Methods

The awareness of the theory of specific domain causality is the prerequisite for discovering deep knowledge (i.e. regularities, laws) in a given domain.

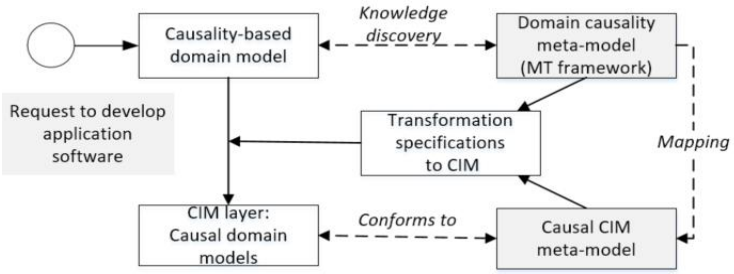
Causation modelling methods are common in statistics, econometrics, cybernetics, computer science, data science, and other complexity sciences to study cause-effect relationships and construct causal models in order to predict and control the possible dynamics of the systems.

The Grey system theory (GST) is a sub-type of systems theory for understanding, modelling, and incorporating uncertainty in systems analysis [51]. Grey relational analysis (GRA) is an impact evaluation model that measures the degree of similarity or difference between two sequences based on the grade of relation [52]. Over recent years it has been used in various domains ranging from healthcare to finance and climate change [53]. The term "grey" means unclear, not defined, or lacking information. Usually, systems, including EAS, are perceived in that way by someone externally trying to understand how the system works. Some business rules are clear but still, a lot of prior knowledge is built-in and it is not clear what patterns, and business models are used.

Both the traditional MDD and the AMDD approaches have flaws – they do not contain the aspect of the domain causality, therefore, it is missing the key feature of creating an intelligent software development management system. In order to improve this, the causal modelling approach is adopted for Agile activities modelling.

The MDA approach was moved to the causal modelling paradigm in [54] by introducing a new layer of domain knowledge discovery (next to the CIM layer) and a causal knowledge discovery (CKD) technique tailored for the enterprise domain. From the causal modelling viewpoint, an enterprise is a

subject domain, considered a self-managed system driven by internal needs (strategy, management, system goals). The top levels of the causal MDA/MDD principle scheme are presented in Figure 6 (PSM and coding layers not presented).




Source: [106]

Figure 6. Causal MDA/MDD transformations

Based on [55, 107], the elements on the left mean the creation of specific models (i.e. BPMN) based on the transformation specification defined by transformation specification MT to CIM. Here only the CIM to PIM transformation is shown, but the transformations PIM to PSM and PIM to code follow the same logic (using meta-models), and the full cycle is described in [54].

The meta-models for CIM and PIM are defined using i.e. BPMN notation that is supplemented with constraints required in the MT framework [54, 76]. Every metamodel is affected by the constraints of the causal knowledge metamodel.

Alfraihi and Lano have conducted a thorough systematic literature review on the topic of Agile MDA transformations [108]. The results were examined to determine whether there is a description of how to capture causal domain knowledge and are presented in Table 4. The papers that do not have MDA transformations specified were not included. DSL stands for "Domain-specific language". The conclusions are based on the causal modelling approach as described in [54, 89]. External modelling is considered a black box; external/internal – deeper knowledge is not still sufficiently captured – grey box, if the modelling is internal – white box.

Table 4. Causal perspective in the AMDD methods

| MDA transformation specification type | Domain model | Domain Meta-model | Domain knowledge type | | |
|---|---|---|---|---|---|
| | | | Observation-based/ black box | Rule-based/ grey box | Causal knowledge/ white box |
| Meta-model | Yes | Yes | – | [94] | – |
| Meta-model | – | Yes | [103], [109] | – | – |
| DSL | – | – | [110], [111] | [112] | – |
| Meta-model | Yes | – | – | [113], [114], [115] | – |
| Meta-model | Causal | Causal | – | – | [54], [76] |

Source: [106]

The causal Agile project management method presented in this thesis is based on the causal MDA process that allows the revealing of both business domain and also Agile activities interactions and their content as the causal knowledge model will be used. Based on this approach the causal model is saved in the knowledge repository, therefore, it is a virtual knowledge model.

Current EAS project management tools such as Atlassian Jira only evaluate the static state of the project – i.e. number of initiatives related to a theme, number of epics related to an initiative, and number of user stories related to an epic (their state evaluation) but do not take into account how much effort is required to complete. The parameters (attributes) in Jira that are saved reflect only the static information. The difference of domains (i.e. e-shop for leisure goods and banking system) makes it even more important to capture the inherent properties of the domain as each of the systems has very different patterns, and regularities. The causal model captures the semantics of the business domain. Therefore, the causal knowledge model is a basis to declare patterns of interaction content specific for each Agile hierarchy level: interaction theme-initiative, initiative-epic, epic-user story and the virtual causal model is saved in the knowledge repository. The MT framework is used to capture the informational flows of the different Agile activities throughout the Agile TIES hierarchy. The MT for each adjacent level theme-initiative, initiative-epic, and epic-user story is different as their informational flows contain different information (with different abstraction levels).

## 1.5. EAS Project State Evaluation Methods

In order to analyse the state-of-the-art research to EAS project state evaluation, it should be noted, that there are several different scientific streams to address the topicality. One of the most notable is software development maturity or software process improvement (SPI) models, which contain practices to ensure business and IT alignment.

### 1.5.1. EAS Project Success Evaluation Methods

Varajão et al. [116] have conducted thorough research on IS project success evaluation. They have examined 34 articles and the research showed that "most models and methods are theoretical exercises with only a few pieces of evidence of their use in practice, thus urging for more empirically based research." Furthermore, the evaluation process is generally not thoroughly described, hence the evaluations are subjective and not formal.

Bokovec et al. [117] developed a global efficiency factors methodology-based evaluation method to assess the ERP projects' complexity. The authors claim that by using their assessment, some of the benefits include:

- "The knowledge exchange in the corporation, which brings the knowledge from the top of the corporation to its companies and vice versa, thus, encapsulating the broadest view of the business;"
  and
- "Benefits from the synergy of individual projects and project programs in the corporation."

However, it is not specified in the research the measurement process of these benefits. Also, a formal assessment of the different variants of the complexity components is evaluated subjectively and lacks formalization.

### 1.5.2. Software Development Process Improvement Models

Many authors have researched the software process improvement models and developed their own. S. Komi-Sirvio [118] defined the critical success factors for software process improvement methods. While the author concludes that „software process improvement (SPI) initiatives need to be directed towards business goals" he does not specify formally how it should be done. The author conducted research in an organization whose goal was "to reduce software defects by increasing the knowledge transfer between different projects". This was done using semi-structured interviews and proved to be successful as "the customer project was served the required knowledge just in

time." However, this research does not specify what the knowledge base structure is, and using a semi-structured interview alone does not provide the sufficient level of details and much important information might have been left out due to a lack of formalization of information transfer (knowledge base) structure.

Cheng and Permadi [119] have done thorough research on the effective measurement and evaluation of the outcome of software process improvement and provided a theoretical evaluation model of SPI initiatives. It is based on evaluation areas, methods, timing of evaluation, the measurements themselves and a holistic evaluation approach. However, the data for evaluation is gathered using surveys. This approach does not contain a sufficient level of detalization of the knowledge gathered and is prone to the loss of important information that might reveal causal domain knowledge.

At a similar timeframe, when the SPI model in [118] was developed, a CMMI [120] evaluation method to assess an organization's maturity in its process management practices was presented. CMMI assessment has two basic versions: staged and continuous evaluation. In the staged representation, the organization is measured against five maturity levels and different practices need to be implemented to increase the overall organizational maturity level. In the case of continuous representation, the individual process areas (that the model specifies) are evaluated. The evaluation itself is evidence-based and provides a good overview of the organizational capabilities to sustainably deliver quality software. It is a typical practice to have a CMMI assessment done so that the company, that is providing software development services would indicate a high level of confidence in delivering quality software within time and budget constraints. Besides CMMI, the International Organization for Standardization had also identified the importance of software development lifecycle process assessment models and therefore introduced an ISO standard – ISO 15504 [121]. This standard "provides a detailed description of the structure and key components of the Process Assessment Model, which includes two dimensions: a process dimension and a capability dimension. It also introduces assessment indicators."

### 1.5.3. Agile Maturity Models

Agile maturity models emerged as early as 2001 and since then more than 40 different models have been created [122]. The Agile maturity models approach to support business and IT alignment stems from the Capability Maturity Model (CMM) [123] which is currently known as Capability Maturity Model Integration (CMMI) and was described above. To address the topicality of this thesis, the Agile maturity models, assessed in [124] were evaluated from a business and IT alignment perspective, focusing on investigating the formalities of ensuring business and IT alignment. Table 5 and Table 6 contains the selected practices. Text in *italics* indicates practices aimed at ensuring business and IT alignment.

Table 5. Agile capability maturity models practice for business and IT alignment in SAFeMM/AAF

| **Level 5** | **Level 4** | **Level 3** | **Level 2** | **Level 1** |
|---|---|---|---|---|
| Collaborative | Evolutionary | Effective | Adaptive | Encompassing |
| • *Low process ceremony*<br>• *Agile project estimation*<br>• Measuring business performance<br>• *Face-to-face interaction between develops and users*<br>• Impact on customers and operations | • *Client-driven iterations*<br>• *Continuous customer satisfaction feedback*<br>• Lean requirements at scale<br>• *Adaptive planning*<br>• *CRACK Customer immediately accessible* | • Regular reflection and adaptation<br>• *Risk-driven Iterations*<br>• *Planning features not tasks*<br>• Roadmap<br>• PSI/Release<br>• *Frequent face-to-face communication*<br>• Agile Release Train<br>• DevOps<br>• Vision, features | • *Evolutionary requirements*<br>• Smaller, more frequent releases<br>• Requirements Discovery<br>• Release Planning<br>• Tracking iteration progress<br>• Product Backlog | • *Collaborative Planning*<br>• User stories<br>• *Customer commitment to work with the development team* |

Source: Created by the author according to [124][125]

Table 6. Agile capability maturity models practices for business and IT alignment in Chandrasekaran's SAMM

| **Stage 6** | **Stage 5** | **Stage 4** | **Stage 3** | **Stage 2** | **Stage 1** |
|---|---|---|---|---|---|
| Portfolio level matured | Portfolio level streamlined | Program level matured | Program level streamlined | Team level matured | Team level streamlined |
| • S6P1 Program portfolio management | • S5P1 Strategic themes estimation<br>• S5P2 Measuring business objectives<br>• S5P3 Portfolio backlog | • S4P0 Stakeholder alignment | • S3P2 DevOps<br>• S3P4 Program Backlog<br>• S3P5 Release planning<br>• S3P6 Inspect and adapt workshop PO level | • S2P2 Increased collaboration<br>• S2P4 Acceptance-driven development<br>• S2P10 Tracking iteration progress<br>• S2P11 Stakeholder alignment<br>• S2P12 Adaptation sessions – Inspect and adapt workshops team level | • S1P12 Task boards<br>• S1P13 Sprint backlog<br>• S1P15 Automated Status reporting<br>• S1P16 Common language |

Source: Created by the author according to [124]

The aforementioned practices are communication-based, meaning that there is no formal information structure used to ensure business and IT alignment and such a way is error-prone.

### 1.5.4. Business and IT Alignment Evaluation Methods

GRAAL[40], BITAM[41], Service-Oriented Business and Information Systems Alignment Method (SBISAF) [42], SAM[65] and SOA[67] are described in section 1.1.2. of this thesis, but they do not include an EAS project content evaluation aspect of the business and IT alignment. To ensure that EAS project requirements content aligns with business strategy, a proper evaluation of the EAS project state must be conducted.

The "Balanced scorecard" method was introduced by Kaplan and Norton This method "enables the company to align its management process and focuses the entire organization on implementing long-term strategy" [126]. It focuses on the financial, customer, internal processes and learning and development of employees' perspectives. One of the steps in the "Balanced scorecard" method asks the managers to evaluate each perspective against the

business strategy to prioritize any initiatives needed to improve the key performance indicators. However, it is a communication-based method and lacks formalization of specific information to be transferred to each department, team and eventually individual in an organization.

A project management office (PMO) is a department typically found in large organizations that ensures governance of projects across the organization and keeps track of project progress in terms of timeline, budget and scope completed. As for product development to assess which feature should be worked on first, the projects in the organization might be prioritized according to various prioritization methods, e.g. RICE [127]. Using such frameworks promotes alignment, however, it is still communication-based [128].

Objectives and Key Results (OKRs) is another methodology aimed at business alignment across the organization, including business and IT alignment [129]. However, it is also communication-based [130] and the improvements in the methodology are focused on the ways of how to communicate strategic objectives, to measure key results, but not on specifying informational content among different departments.

### 1.5.5. Project State Evaluation Capabilities in Agile Project Management Tools

Jira tool by Atlassian is the most popular Agile project management tool [83] and various research has been done using it. Ortu et al. [131] used Jira repository dataset to understand the social aspects of software development and found out that comments stored on Jira issues provide valuable information on "how developers interact, as well as how they feel about the project and their peers". Krasniqi and Do [132] used Jira data for bug reports classification and provided a framework for "the direct tracing and visual mapping of quality concerns into codebase". Diamantopoulos et al. have presented a dataset of 656 projects with over a million Jira issues that could be used for automated bug triaging or determining the priority of issues [133]. Făgărăşan et al. [134] have introduced two KPI metrics categories "Increment Global Quality" and "Increment Scope" that consist of 5 new KPIs introduced by the authors and data from Jira was used for that. Those KPIs under the "Increment Global Quality" category include "Defect removal efficiency" measuring how many defects were identified before the product increment delivery "Major bugs which were not addressed before the increment delivery" and "Reopened defects" calculating the number of defects that were presumably resolved but got reopened later. The KPIs under the „Increment Scope" category include "Acceptance criteria definition" which calculates the

tasks that have clearly defined acceptance criteria and “Work breakdown definition” which calculates the total number of tasks over an eight-hour-worth estimate. However, these KPIs are rather subjective and open to interpretation. Also, they do not take into account the Agile activities content from the causality perspective.

The latest Jira software instance v9 has over 150 tables and over 600 fields [135]. As being highly customizable, Jira typically stores EAS project data such as:

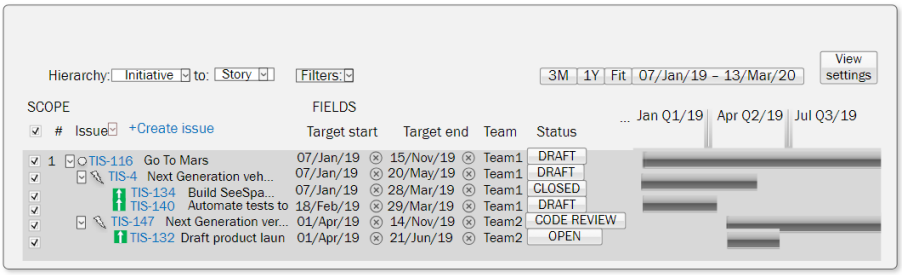
- Issue information (description, type, links, assignee, reporter, attachments, comments, etc.)
- Project hierarchy information (epic link, link to higher level Agile activity, status)
- Custom fields (from ROI information to business impact description, etc.).

Vavpotič et al. [136] have used the log data from Jira together with the survey data from the users (developers and product owners/managers) to analyse stakeholders' perception of the software development management process and in this way manage business and IT alignment. The authors have discovered that one of the identified difficulties is "adjusting prioritizations of user stories to ensure customer satisfaction when customer priorities changed". Their provided recommendation to address this difficulty is that "management needs to start to act as a connector that encourages and helps the team to establish direct collaborations between them and the customer's product owners” lacks formalization as it does not specify what particular information needs to be transferred between different parties of the management process (business development management and software development management).

In the context of this thesis, Jira was analysed to identify what are the fields in the Jira database dedicated to EAS project design specifications and evaluation of project content against the strategic business objectives. Despite vast customization options, the standard Jira software does not contain specific fields to specify an internal model of the domain and content-based EAS project specification cannot be evaluated. Furthermore, Jira does not specify any feedback loop which is mandatory for causality-based modelling of information.

Figures 7 to 10 illustrate the current composition of Jira windows used for an overview of the TIES structure and that are used for Agile project management. Figure 7 contains the portfolio view, i.e. a set of initiatives (e.g. “TIS-116: Go to Mars”), a set of linked epics (“TIS-4: Next Generation

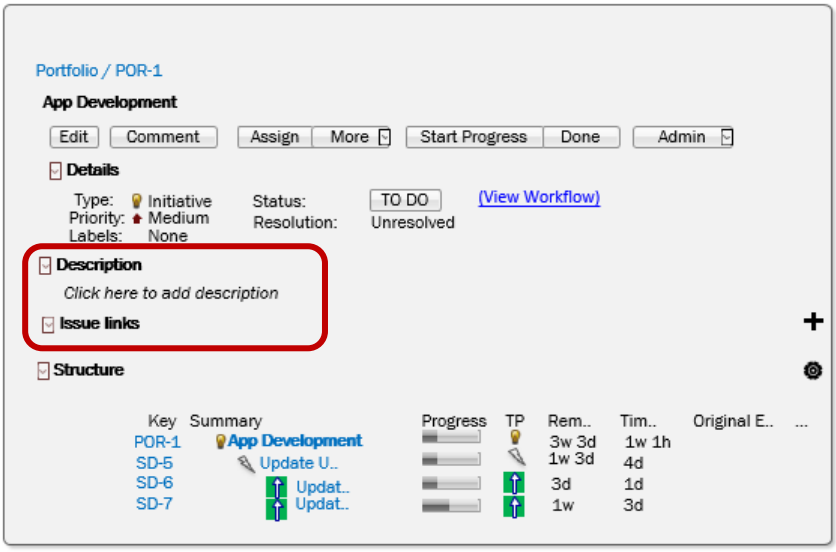
vehicle development", "TIS-147: Next generation version development") and some user stories (TIS-134, TIS-140 and TIS-132)




Source: Created by the author according to [84]

Figure 7. Jira portfolio view current composition

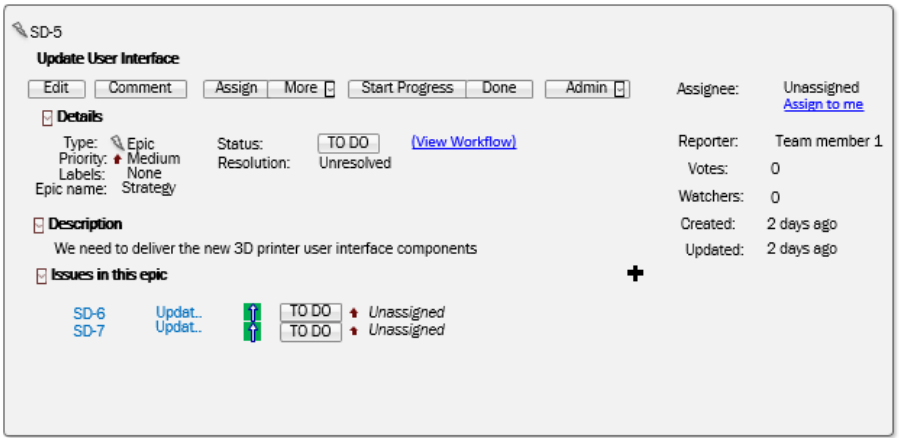
Figure 8 contains an initiative example. The marked sections ("Description" and "Issue links") refer to the default Jira fields that are currently empty and miss the required specification.




Source: Created by the author according to [84]

Figure 8. Jira initiative current view

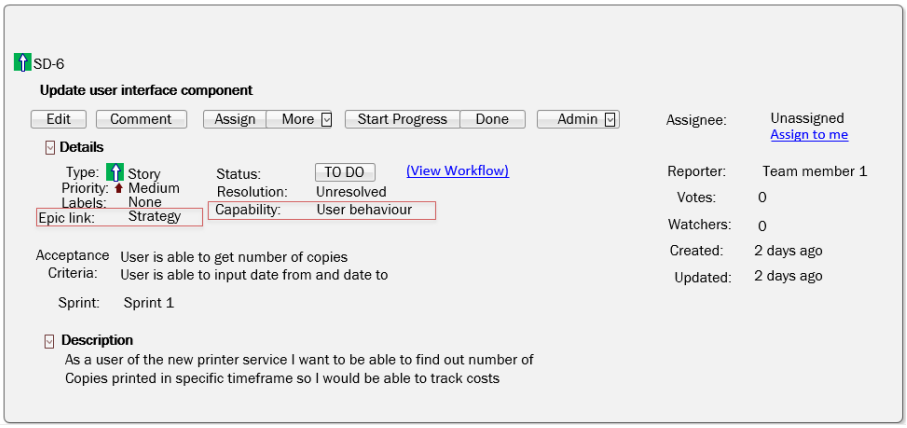
An example of an epic is displayed in Figure 9. In this case, the epic contains the required information: it is a part of the initiative POR-1 (Figure 8) and there are links related to user stories (SD-6 and SD-7).




Source: Created by the author according to [84]

Figure 9. Jira epic current view

Figure 10 indicates the user story example: the requirements information needed to do development work is presented (in the "Description" and "Acceptance Criteria" fields). The fields marked in red ("Epic link" and "Capability") are the fields proposed as part of the specification based on the MT structure. Epic link indicates the management function based on the MT structure. Capability indicates the capability of the enterprise based on the capability to Agile concepts mapping (Table 9).




Source: Created by the author according to [84]

Figure 10. Jira user story view with additional fields

However, this set of Jira screens shows that there are no project state evaluation indicators. The dashboards that are possible to be designed in Jira are based on the presented data and, therefore, do not contain the required specification of Agile hierarchy relations.

## 1.6. First Part Conclusions

Agile project management methods are the best alternative currently that allows to deal with complex EAS development problems. However Agile methods that are used in practice today have significant downsides, such as:

a) not containing sufficient detalization of linking EAS development solutions to business strategy execution (from the SAM perspective) and Agile project management activities;
b) in large projects using Agile methods (even with using Agile tools) it is time-consuming and requires additional manual work (because project leaders need to analyse long lists of records with many attributes) in order to understand the progress of overall project execution and this is done regularly;
c) the Agile tools used in practice today specify the fact of connection between different agile hierarchical levels (theme-initiative, initiative-epic, epic-user story), but this is not enough to make decisions, because the interaction content between Agile activities is not described;
d) in the Agile tools environment, no business domain model conveys mandatory business process relationships and attributes.

The analysis of existing business-IT alignment methods found that they lack the modelling of the internal dependencies of the domain activities, i.e. deep knowledge (causal interactions) models are not built. Furthermore, most of the Agile methods that are used in practice today take significant time to evaluate the misalignments, and although they provide metrics to track misalignments, models still need to be translated to project requirements, or the requirements should be adjusted manually, therefore, the more automated solution is demanded.

Therefore, it is claimed in this thesis that the current Agile hierarchy model needs to be improved by including domain causal knowledge, and formalizing specifications of Agile activities and hierarchical interactions. MDD approach is used to streamline the EAS development process and it is assumed that models sufficiently represent reality and different alternative paths of processes. However, the current approach does not cover the causality aspect sufficiently detailing the domain peculiarities that might impact the end result of the developed EAS.

Causal MDD enables us to move away from the external modelling paradigm developing the black box models towards the internal modelling paradigm (white box modelling), where the acquired domain knowledge is causal knowledge and serves as an enabler to develop solutions in a more efficient way.

The project success evaluation methods lack formalization and specification on how to ensure that business and IT alignment is achieved. They mainly focus on the classical constraints of time, budget and scope(quality), frequently taking into account the user satisfaction component that is also subjectively defined and, therefore, is prone to be adjusted artificially to demonstrate a better situation.

In summary, it can be concluded that it is necessary to create a modified Agile management method using causal modelling that would specify in detail the content of Agile activities and the content of their interactions, and would include knowledge about business domain processes and strategies.

# 2. CAUSALITY DRIVEN AGILE MANAGEMENT METHOD

## 2.1. Conceptual Models of EAS Development and Management

The conceptual model of the traditional EAS development and project management is presented in Figure 11. It is assumed in this thesis that the traditional MDA/MDD methodology and modelling techniques such as BPMN, UML, SysML, etc. are used [92]. The conceptual model includes the following nodes:

- M1 and M3 denote business domain – real-world objects (activities, processes, equipment, divisions):
- M1 corresponds to the management layer (support activities according to Porter's Value Chain Model [78]);
- M3 corresponds to the physical activities layer (primary activities according to Porter's Value Chain Model);
- The analyst's / designer's understanding of real-world domain (M1, M3) behaviour is based on observation (external modelling paradigm).
- M2 refers to models that are expressed (specified) in some modelling language and are further used for EAS development. Domain modelling and application system modelling (M2) have been performed using known specification languages (i.e. OMG specifications BPMN, DMN, UML, etc. [68]);

Current (traditional) EAS development models belong to the external modelling paradigm because the used modelling specifications allow the creation of black box models and their aggregated structures. Such modelling notations and methods do not seek to discover the causality of the business area, they only represent the results of external monitoring. Also, these notations used for modelling do not differentiate the managing function and the managed process and the content of the interaction between these two different elements.

Therefore, these modelling methods do not identify or form the circular feedback loop of causality that must be created to ensure that causality is captured.

The M2 node represents such a model(s) defined using BPMN or other modelling notations that M1 and M3 type activities are not identified as different type activities.

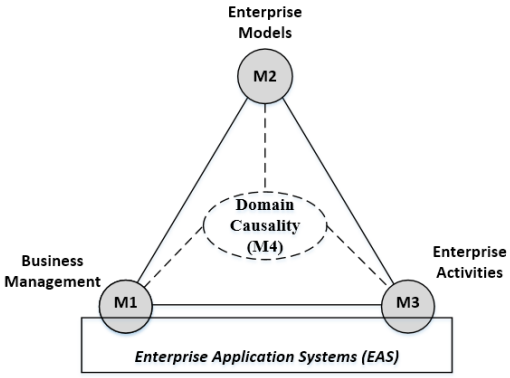
The interaction between the nodes is defined as follows:

1. Business and IT alignment process based on the experience: M1 – (M4) – M3, where (M4) denotes the business manager's understanding of

causality (experience, mental ideas), which is a mental model (not manifested in an expressive form), therefore marked in brackets.

2. Business and IT alignment process based on the Enterprise modelling and business process modelling methods (model-driven engineering(MDE) approach): M1 – M2 – (M4) – M3;
3. Business and IT alignment process, based on causal modelling. The domain causality model (M4) is created in an expressive form and implemented in the virtual environment to validate M1 – M2, M2 – M3, and M1 – M3 mapping (1):

$$\{M1 – (M4) – M2; M2 – (M4) – M3; M1 – (M4) – M3\} \quad (1)$$



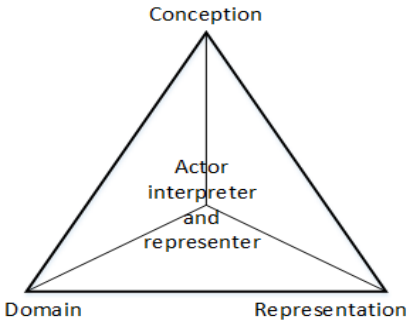

Source: [55]

Figure 11. Approach to business and IT interaction modelling

The business and IT alignment management activities in Figure 4 can be organised in several ways. The following subsections explain the approach to business and IT interaction modelling in more detail as shown in Figure 11.

### 2.1.1. EAS Development Management Approach to Business and IT Interaction Modelling

The conceptual diagram in Figure 11 is based on the FRISCO semiotic tetrahedron (Domain, Conception, Representation, Actor) [58]. The FRISCO approach is based on semiotics and tries to bridge the gap between the real world and its modelling concepts. FRISCO semiotic tetrahedron is presented in Figure 12.




Source: [58]

Figure 12. FRISCO semiotic tetrahedron

The conceptual diagrams in Figure 11 show the difference between non-IT management activities and computerized activities in terms of knowledge used to manage the activities.

Firstly, all causal MDA development [54] seeks to reveal the causality of the domain and specifies the captured causal knowledge using meta-modelling or ontology description methods.

The causal MDA/MDD scheme includes an additional layer named the Domain Knowledge Model (DKM) next to the CIM layer of the MDA approach [54], Figure 6.

Table 7 contains the mapping of the terms of the FRISCO semiotic tetrahedron to MDA methodology and the causal modelling concepts presented in this thesis to demonstrate the consistent transition from the formal FRISCO and MDA structures to the structure of the proposed method.

Table 7. FRISCO semiotic tetrahedron to causal modelling mapping

| **Terms of FRISCO semiotic tetrahedron** | **Terms of causal modelling method** | **Terms of causal MDA / MDD methodology** |
|---|---|---|
| Domain | Enterprise Activities (M3) | DKM |
| Conception | Enterprise Models (M2) | CIM, PIM |
| Representation | Business Management (M1) | DKM |
| Actor | Domain causality (M4) | Real World domain |

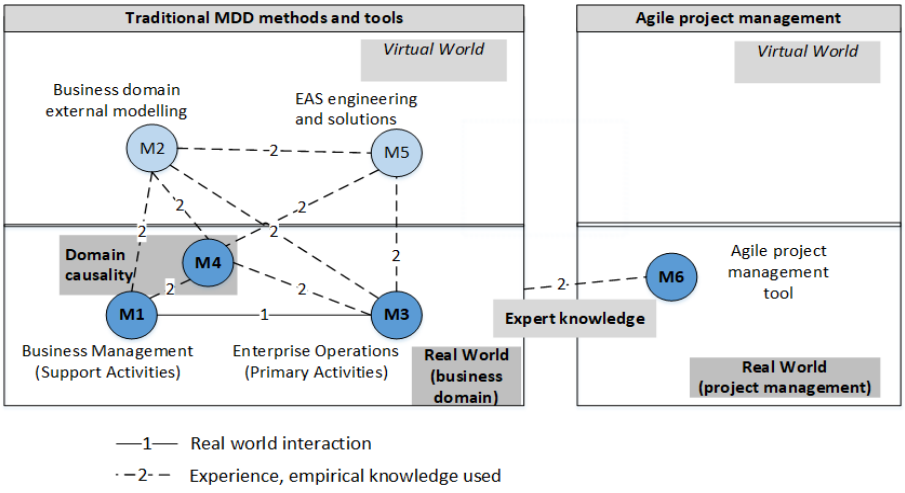
Source: Created by the author

The discovered causal knowledge is a solid foundation for building the knowledge base that underpins the EAS development tool with the features of artificial intelligence.

### 2.1.2.Conceptual Model of EAS Engineering and Agile Management

In this section, the conceptual models of EAS engineering, Agile management processes, and causal Agile management models are presented. These conceptual models are based on the FRISCO approach. It helps to reveal and define the interaction content between the elements of the EAS engineering processes. These conceptual models are aimed at explaining the importance of domain causality modelling and the impact of causal knowledge on EAS engineering solutions and management decisions.

Also, these models allow us to distinguish the traditional EAS engineering and knowledge base and explain the differences between traditional Agile and causal Agile methods. Traditional Agile methods (no particular framework) are based on the assumption that the required information to ensure business and IT alignment across all levels of Agile hierarchy is delivered by experts: managers, product owners, or their equivalent roles. However, this approach is error-prone as the beforementioned roles have different levels of contexts, experience, and domain understanding, i.e. expert knowledge (Figure 13).




Source: Created by the author according to [106]

Figure 13. Conceptual model of the traditional EAS development and Agile management

The interactions between nodes in Figure 13 are defined as follows:

- M4 stands for real-world domain causation (regularity) that is characteristic of a particular domain. In traditional MDA it is not expressed as a model in any form, but is perceived by experts as

experience, and influences the development of EAS, but is not captured as a causal knowledge model (in traditional EAS engineering).

- Traditional MDA methodology does not require knowing the regularities of the domain (M4). It is not explicitly clear whether only fragments of the whole system are observed or the whole components of the system and causal interactions are captured. In real-world EAS development projects, this is often the case because initially a lot of information about the processes is context-based, so the analyst or developer (external observer) cannot perceive the process in full, until it has sufficient knowledge of it and various exceptions, workarounds, etc. Only then proper modelling can be done.
- M5 refers to enterprise application software (EAS) to be developed;
- M6 represents the Agile project management tool (i.e. software tool used for EAS project management, i.e. JIRA, Azure Boards, Monday.com , Wrike, etc), including project management database;
- Transitions M3 – M4 and M1 – M4 mean that the processes are observed externally (black box approach) but it is not yet expressed as a model. The set of M1 – M4 – M3 represents the observation-based knowledge (i.e. empirical knowledge);
- Transition M1– M3 depicts the knowledge inside the minds of the employees working with the system (the empirically based causal knowledge accumulated throughout working on the process);

The link between the parts of the "Model World (Virtual)" and "Real World" as depicted by the relations of M2 – M4, M2 – M1, and M5 – M3, M5 – M1 illustrates the methodology of engineering, i.e. how knowledge from the real world is translated to the requirements for EAS, in this case, it is the MDA approach;

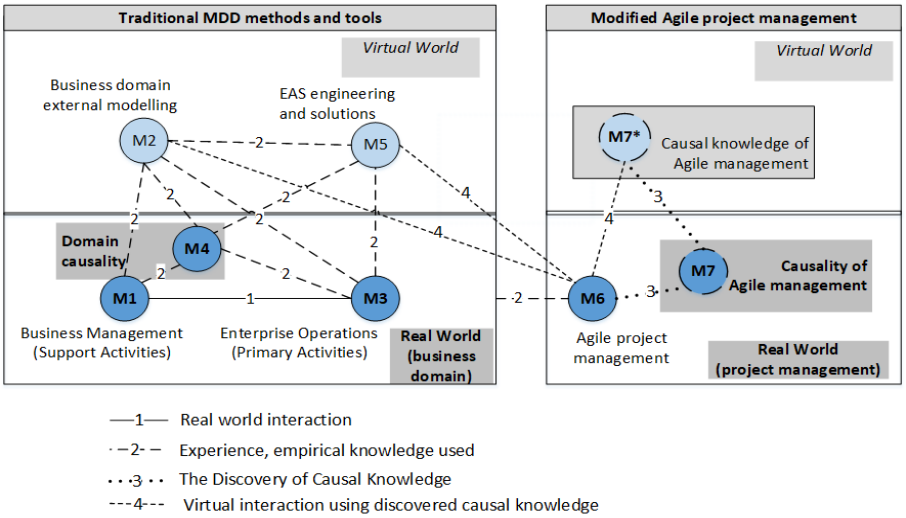
Thus, the conceptual model of traditional EAS development management includes M1 – M6 nodes.

On the other hand, the causal modelling-based Agile method (no particular framework) relies on the information specification requirements based on the management transaction framework across the levels of Agile hierarchy (Figure 15). Further in this research, the Agile method used is Scrum and a customized scaling method based on SAFe and other practices.

Based on the analysis of the traditional EAS development method (Figure 13), a modification is proposed that creates an intelligent EAS development process. In our approach towards intelligent Agile management tools development, domain causation is the regularity of the enterprise management, defined as a management transaction [34], and forms the basis

of the knowledge base M7. Management transaction (MT) is considered a causal knowledge unit. Recognized, discovered domain processes are specified by examining (verifying) whether they conform to the MT framework.

M7 refers to a knowledge base with captured domain knowledge and transformation rules that ensure intelligent EAS development management. This element has a dashed line meaning it does not exist currently but the aim is to integrate it into the current Agile project management tools (represented as M6);




Source: [106]

Figure 14. Conceptual model of the causal modelling-based approach to EAS development linked to Agile management activities

Further specifications for the knowledge base M7 and project management tool M6 database additions are provided that allow for building an intelligent project management tool. The purpose of displaying M7 here as a component of the EAS development management is to demonstrate that it is acknowledged that domain causality must be understood by the analysts, developers, and any personnel working with EAS development to make sure that the developed EAS meets the real-world processes.

The causal business domain model M4 is defined as Management Transaction (MT) [34, 49]. Management Transaction MT = (F, P, A, V) is relevant to some enterprise goal (G), captures knowledge on management function (F), enterprise process (P), informational input flow (state attributes A), informational output flow (controls V). MT includes a feedback loop between F and P composed of A and V flows. A conceptual management

transaction model is presented in Figure 2, estimating the enterprise goal (G), which is not specified explicitly in the MT definition.

Knowledge base (KB) is a crucial component of any intelligent system. KB contains the real-world knowledge captured by experts and is an internal model following the internal model principle defined by Francis [5]. The causal MDA approach uses predefined knowledge to ensure causality-based transformations (as depicted in Figure 6). The KB could be created to support MDA transformations for developing EAS in an Agile setup.

Table 8 contains the descriptions of the nodes from Figure 14:

Table 8. Causal Agile Development Management model nodes description

| Node ID | Name of node | Description |
|---|---|---|
| M1 | Business management activities | Real-world processes (support activities of Porter's VCM [78]). |
| M2 | Business domain external modelling | Virtual models M2 describe the M3 and M1 activities. |
| M3 | Enterprise operations | Real-world processes (primary activities of Porter's VCM [78]). |
| M4 | Domain causality | Real-world causation (regularity) that is inherent for the domain type. |
| M5 | EAS solutions | EAS development activities (development, testing, deployment) |
| M6 | Agile project management tool | Computerized tool, supporting the enterprise strategy alignment with EAS solutions. |
| M7 | Causality of Agile management | Real-world activities, causal interactions between Agile activities (theme, initiative, epic, user story). |
| M7* | Causal knowledge base of Agile management | Based on the causal models of the modified Agile hierarchy, and the management transaction (MT) framework. |

Source: Created by the author

The conceptual model in Figure 14 is divided into two different perspectives: real world (nodes M1, M3, M4, M6, and M7), and the virtual world (nodes M2, M5, and M7*). The link between the parts of the "Virtual World" and "Real World" as depicted by the relations of M2 – M3, M2 – M1, and M5 – M3 illustrates the methodology of engineering, i.e., how the knowledge from the real world is translated to the requirements for EAS – in our case – the MDA/MDD approach. Content of M2 and M5 based on partly discovered M4 (through the experience of M1 and M3).

The brief characteristics of the nodes, defined in Table 8 are as follows:

- M1 – Experience-based business management processes (activities and responsibilities), related to the enterprise strategy requirements, and operational goals. Corresponds to management functions (support activities of Porter's Value Chain Model).

- M2 – the virtual models, specified in some standard modelling notation (BPMN, UML, MODAF, etc.) and is further used;
- M3 – Real-world processes (primary activities of Porter's Value Chain Model), i.e. physical processes (manufacturing, development) of products or services. Controls from M1 have impact on M3;
- M4 – Real-world regularities (patterns) that could be known to a certain extent as causal knowledge (deep knowledge). Causal dependencies of M1 and M3;
- M5 – EAS solution including project solutions and code, based on empirical models (M2) and experience related to M3 and M1;
- M6 – Intelligent Agile management environment, based on the causal Agile management model, focused on the enterprise strategy alignment with EAS solutions. Contains the recorded state of the EAS development (project scope, status of completed features), including validation of project state against the causal model of Agile management interactions;
- M7 – The inherent interactions between items in Agile hierarchy (theme, initiative, epic, user story);
- M7* – Causal knowledge constructs related to Agile management, i.e. causal knowledge base consisting of:

a) meta-model of the modified Agile management hierarchy;
b) project state (data) mapping to modified Agile management hierarchy, i.e., project state evaluation (verification) records.

The causal knowledge base specifies causal knowledge as meta-models: management transaction (MT) meta-model [34], modified Agile hierarchy and project state evaluation metrics [50].

### 2.1.3. Interplay of Agile Management Hierarchy with Business Strategies and Activities

Agile project management practices and tools do not provide structured and formalized techniques (models) to define interactions in the TIES hierarchy, based on business domain models. Business strategy is essential but is not expressed formally and only explained verbally. It is not specified using modelling tools and therefore cannot be connected to the highest level of the typical Agile hierarchy – theme.

In MODAF, capability is a high-level specification of the enterprise's ability [43]. Capabilities rarely change over time; the main underlying idea is that capabilities enable the organization to act and succeed in the preferred domain based on the skill set and abilities of the enterprise.

Mapping of business strategy execution to well-defined (structural) application design models is carried out using MODAF products: Strategy view models (StV-1 Strategy View, StV-2 Capability Taxonomy, StV-4 Capability Dependencies, StV-5 Capability to Organization Deployment Mapping).

This ensures the transformation of the declared business strategies into well-defined structures, starting with the capability concept. This way, the content of Agile concepts (theme, initiative, epic, user story) is defined using the concept of capability and related MODAF models (Table 9).

Table 9. Alignment of Agile and MODAF concepts

| **Agile concepts** | **MODAF products and concepts** | |
|---|---|---|
| | **Elements** | **Views** |
| Theme | Capability | StV-1<br>StV-2<br>StV-6 |
| Initiative | Operational node | StV-6 |
| Epic | Operational activity<br>Operational activity flow | OV-2<br>OV-5 |
| User story | Operational activity<br>Operational performer<br>Operational role<br>Operational activity flow | OV-5 |

Source: [106]

Strategic views (StV-1, StV-2, StV-6) indicate the necessary capabilities of the enterprise (organization, business company), for which Operational View models OV-2, OV-5 are created, consisting of operational nodes and their mutual interactions flows. In the work context, the Strategic View option can mean specific needs of the business area (target features, new functionality, services).

Knowing the capability set (e.g. in banking) Operational View models are created, which specify in detail the processes of how these capabilities must be realized in projects (e.g. for enabling a self-service solution for business customers).

Agile management concepts mapping to MODAF products from Table 9 is required to specify the internal structure (content) of Agile concepts using StV-1, StV-2, StV-6 models. This will help to reveal the integrity of the interactions between the Agile process levels. Table 10 contains the description of MODAF elements in StV and OV models.

Table 10. Elements of MODAF StV and OV models

| MODAF view | Purpose | Elements |
|---|---|---|
| StV-1 - Enterprise Vision | Provides the high-level scope of the architecture and a strategic context for the capabilities it contains. | • Enterprise Vision.<br>• Enterprise Phase.<br>• Enterprise Goals.<br>• Capability.<br>• Enduring Task. |
| StV-2 - Capability Taxonomy | Models capability taxonomies in the context of an Enterprise Phase. | • Capability.<br>• Capability Specialisation (super-subtype relationship between capabilities).<br>• Capability Composition (whole-part relationship between capabilities). |
| StV-6 - Operational Activity to Capability Mapping | Specifies standard (e.g. doctrinal) activities, and traces them to the capabilities they support. | • Capability.<br>• Standard Operational Activity.<br>• Enduring Task. |
| OV-2 Operational Node Internal Relationship Description | • Definition of operational concepts.<br>• Elaboration of capability requirements.<br>• Definition of collaboration needs.<br>• 'Localising' capability.<br>• Problem space definition.<br>• Operational planning.<br>• Supply chain analysis | • Nodes ("Operational Nodes").<br>• Needlines (bundles of information exchanges).<br>• Logical Flows (of material, people or energy).<br>• Operational Activities.<br>• Locations. |
| OV-5 Operational Activity Model | • Description of business processes and workflows.<br>• Requirements capture (input to user requirements document (URD)).<br>• Definition of roles and responsibilities.<br>• Support task analysis to determine training needs.<br>• Problem space definition.<br>• Operational planning.<br>• Logistic support analysis.<br>• Information flow analysis | • Operational activities.<br>• Standard operational activities (originating in StV-6).<br>• Operational Activity Flow Objects.<br>• Swimlanes (each associated with a node). |

Source: Created by the author

Initiatives and themes are considered highly abstract and they need to be mapped to capability concept. The capability concept from MODAF is sufficient to represent the fuzzy information from the theme and a strategic objective level. The mapping is presented in Table 11.

Table 11. Mapping of Agile concepts to MT and MODAF concepts

| **Static view** | | **Dynamic view** |
|---|---|---|
| Agile activity mapping to MT elements | MODAF concepts mapping to MT | MT elements |
| Theme (T):<br>T := G; | StV concepts:<br>Enterprise Phase (EP),<br>Capability (C):<br>(EP, C) := (G) | Goal (G). |
| Initiative (I):<br>I := MT (F, P, (A,V)); | OV concepts:<br>Operational Node (N),<br>Operational Activity (O),<br>Operational Activity Flow (AF):<br>(N, O, AF) := (F, P, (A,V)); | Goal (G).<br>Management function (F).<br>Process (P) – (P1, P2, ...,Pn).<br>Information flows (A,V). |

Source: [106]

The mapping in Table 11 represents the logical links between different levels of Agile hierarchy. They do not represent any software development realization.

To detail further the content of the coordination link between Agile concepts initiatives I03 and I01 a mapping to management transaction concepts is required. The "MODAF concepts mapping to MT" column in Table 11 shows that the MT = (F, P, A, V) can be modelled in more detail. MT is a causality-based framework representing the grey/white-box approach which specifies the causal dependencies of the problem domain.

The Agile concepts hierarchy is a static structure in terms of its components: themes, initiatives, epics, and user stories, and it does not represent the management dynamics of the system. In our approach enterprise architecture is also considered as a static structure that details Agile concepts in this step of transformation. However, MT is a dynamic structure, capable of representing internal structure and interactions of items:

- MT represents an internal structure of the Agile concepts (theme, initiative, epic, user story), if concepts are defined as self-managed activities;
- MT represents the internal structure of the interactions between adjacent Agile hierarchy level activities vertically (theme – initiative; initiative – epic; epic – user story) if interactions between adjacent Agile hierarchy levels are considered self-managed activities. In this case, the content of interactions (feedback loop) between adjacent Agile hierarchy levels is revealed.

- MT represents the internal structure of the interactions between the same Agile hierarchy level activities horizontally if interactions between adjacent Agile activities on the same level are considered as self-managed activities. In this case, the content of interactions (feedback loop) between the same Agile hierarchy-level activities is revealed.

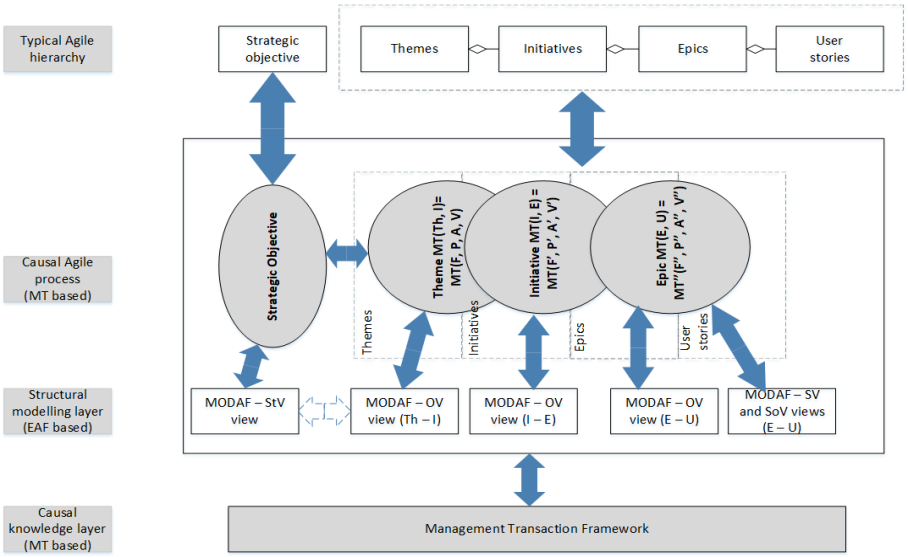
## 2.2. Modified Agile Hierarchy

A typical Agile management hierarchy is the TIES hierarchy based on the concepts of themes (T), initiatives (I), epics (E), and user stories (S) [48]. Basically, in the typical Agile activities hierarchy, the concepts of themes, initiatives, epics, and user stories represent the requirements for EAS development in different levels of project management. TIES concepts are considered project management activities, the implementation of which requires certain actions, such as data analysis, solution development, testing, and others. In the Agile activity hierarchy, there are vertical relationships between levels. A set of initiatives form a theme, a set of epics form an initiative and a set of user stories form an epic.

A special new feature of the modified (causal) Agile hierarchy is that it includes the relationship between the strategic goal (which is defined as a set of capabilities) and the themes (T) that match the identified capabilities. The typical Agile management hierarchy as well as the causal Agile management hierarchy are presented in Figure 15. The causal Agile hierarchy includes the specification of enterprise strategies using enterprise architecture frameworks and integration with the TIES hierarchy. The specification of enterprise strategies is a complex task, where business domain top-level management needs and specific strategic tasks are transformed into well-defined capabilities. To solve this task, the enterprise architecture framework MODAF is used. Structural modelling of enterprise strategies requires knowledge of the business domain, including the company's mission, vision, and higher-level goals, as well as the detailed knowledge of business processes causality.

Therefore, in Figure 15 there is a structural modelling layer (EAF-based layer) and a causal knowledge layer (MT-based). As mentioned, causal modelling is a critical aspect in understanding the behavioural patterns of the domain, and the inherent and implicit informational flows of how the organization operates. The modified Agile hierarchy ensures that every activity in Agile hierarchy contains a set of attributes required to ensure the relations between all the levels and ensures that every activity is beneficial for the organization. The MT framework as a causal modelling tool, together with

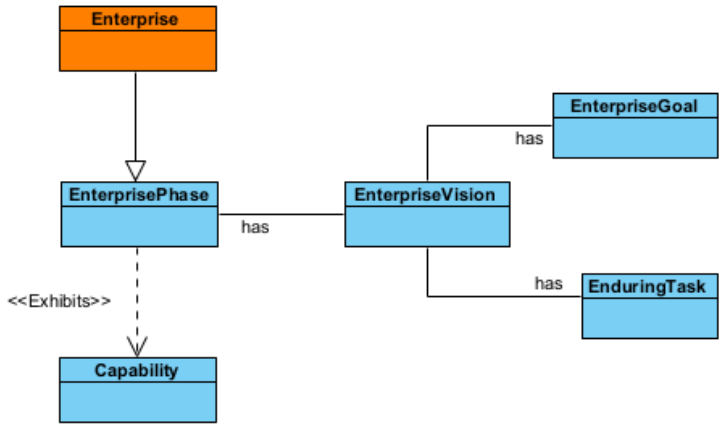
the TIES hierarchy, Scrum framework and some aspects of the customized SAFe framework form the foundation of the research presented in this thesis. Under the structural knowledge layer (EAF-based) the dotted arrow represents the mapping from StV to OV models as defined in MODAF [43]. The strategy is described but there is no link from the strategic objective to Agile hierarchy (as a theme).




Source: Created by the author

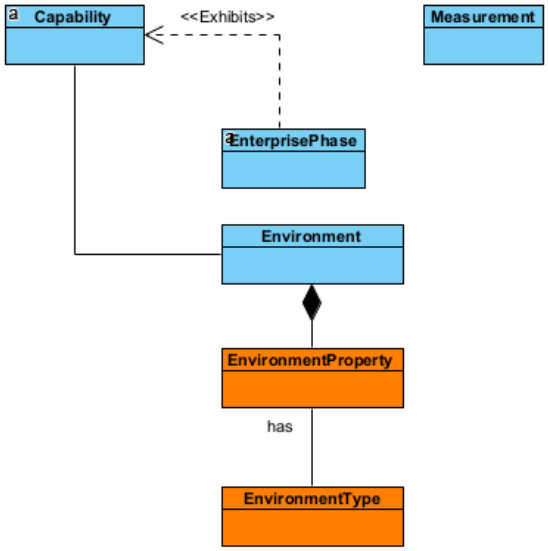
Figure 15. Conceptual scheme of the modified Agile management system based on the causal modelling

MODAF ensures the specification of organizational capability. In this thesis, capability means strategic objectives. The MODAF StV-1 (Enterprise vision) metamodel contains an Enterprise Phase entity which in turn contains capability specification in Figure 16. Figure 17 contains StV-2 (Capability taxonomy) metamodel. Figure 18 contains the OV-2 model of the business strategy execution process depicted in Figure 3.



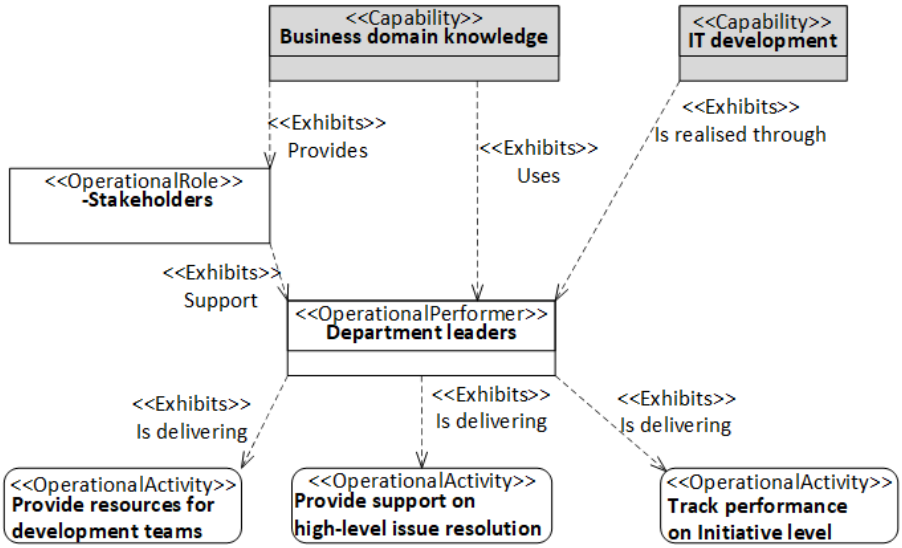

Source: [43][47]

Figure 16. Strategic view StV-1 metamodel (UML Class diagram notation)




Source: [43][47]

Figure 17. Strategic view StV-2 metamodel (UML Class diagram notation)




Source: Created by the author

Figure 18. OV-2 model of business strategy execution process (UML notation)

Analysis of typical Agile hierarchy (as defined in Table 1) from the causal modelling perspective revealed that:

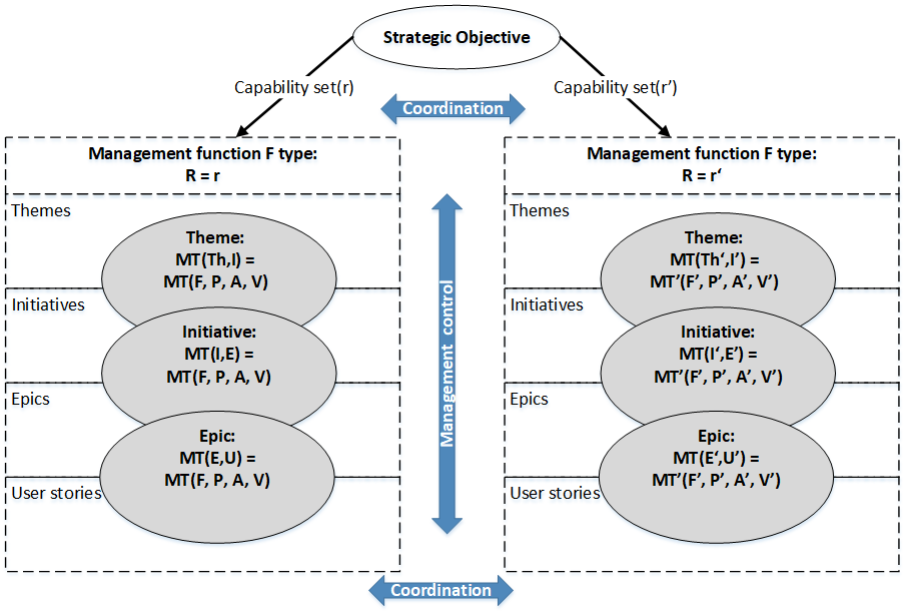
- the business strategic objectives and different enterprise management function types (i.e., finance management, human resource management, software development management, etc.) are not identified and specified in EAS development projects, therefore, the quality of the project execution and alignment to strategic business objectives suffers;
- there is a causal link between any two adjacent levels of activities of Agile hierarchy when data and information are transferred between activities;
- interaction of two activities from adjacent levels of Agile hierarchy is a complex process with a feedback loop (there is circular causality) where the mutual interaction between Agile activities is controlled by an expert (manager, project manager, product owner or developer)

To summarize, the typical Agile process is very little formalized (by design), and Agile tool support is limited to capturing and storing information about Agile activities. The EAS development project execution control, alignment to strategic business objectives and Agile activities coordination rely solely on experts working on a project which is a complex and error-prone process.

Therefore, a few new aspects need to be included in the Agile causal modelling-based EAS project management process as follows:

- the specifications of the relationship between the strategic goal and the themes (T);
- different enterprise management function types are identified and can be involved in EAS development (namely software development management and business development management);
- the interaction of activities in the two adjacent levels of Agile hierarchy is a complex process defined as a management transaction, that includes a feedback loop between activities from different hierarchy levels (there is predefined circular causality). Specification of the information flow content of feedback interaction between (themes – initiatives – epics – user stories) is normalized and includes two subsets: state attributes and control attributes.

By evaluating these shortages of a typical Agile process, a modified Agile hierarchy model is defined based on a causal modelling approach. The modified (causal) Agile hierarchy is presented in Figure 19:




Source: Created by the author based on [50]

Figure 19. Modified Agile activities hierarchy where activities are defined as management transactions (MTs)

It should be noted, that the list of themes is mapped to the set of "capabilities" derived from the modelling of the strategic business objectives using the MODAF approach [43]. Typically, strategic objectives are decomposed following the mission and vision of the enterprise (MODAF StV-1 model) and a set of enterprise capabilities is revealed (Figure 19). The premise of this approach to integrating business strategies with Agile activities is that identified capabilities define the highest level of Agile activities, a Theme set (Th). A feature of the proposed enhanced Agile method is that themes are classified according to the type R of management functions. For instance, theme Th in Figure 19 matches the specific capability "business management" (knowing customer needs, etc.), and so belongs to management function type r; but theme Th' matches a different capability (software development, software development management, etc.) and so belongs to management function type r'.

The following features are important in the modified (causal) Agile hierarchy:

1. Any Agile activity on each hierarchy level $q \in Q$ = (Theme, Initiative, Epic, User story) belongs to some management function type ($r \in R$) where R is a set of management function types. Examples of enterprise management function types are finance management, human resource management, software development management, etc.
2. Any Agile activity (themes, initiatives, epics, and user stories) on each Agile hierarchy level has a predefined structure since it is defined as a management transaction MT = (F, P, A, V). Any Agile activity (theme, initiative, epic, user story) can be specified as F or P, i.e., have a role F or P in the management transaction based on their levels in the modified Agile hierarchy. This relativity of the activity roles is described in detail in section 2.2.1.
3. Content of the vertical interactions between any causal Agile activity MT(r) = (F(r), P, A, V) and some other causal Agile activity MT′(r′)= (F′(r′), P′, A′, V′) can be revealed and classified by analysis of coordination type W = (W1, W2,..., Wn).
4. Content of the horizontal interactions between any causal Agile activity MT(r) = (F(r), P, A, V) and some other causal Agile activity MT(r′)= (F′(r′), P′, A′, V′) can be revealed and classified by analysis of coordination type C = (C1, C2,..., Cm).
5. On Agile hierarchy level q = "Themes" for every management function F of type r is applicable that:

a) any theme (Th) is defined as MTq(Th, I) = MTq(F(r), P, A, V), where theme (Th) is in the role of management function (F), causally related initiatives (I) are in the role of process (P), flow A denotes a set of state attributes of process P (e.g. initiative's state data), flow V denotes a set of controls of process P (feedback from theme (Th) to related initiatives (I)).
b) initiative (I) denotes a set of processes P = {P1, P2,..., Pm} managed by function F (a single theme Th). Flow A denotes a set of state attributes data A = {A1, A2,…, Am} matched with the corresponding set of processes P = {P1, P2,…, Pm}, where A1 is state attributes of P1 (e.g. initiative I1 state data), A2 is attributes data of P2, etc.
c) the theme (Th in the role of F) carries out a set of controls of the initiatives (a set of I = {I1, I2,...,In} in the role P) through the feedback flow V, sends control data (instructions, requirements) related to the corresponding initiatives. Flow V denotes a set of controls (feedback impacts) V = {V1, V2,…, Vn} matched with the corresponding Process P = {P1, P2,…, Pm} where V1 is management controls of P1 (e.g. impact to initiative I1), V2 – is management controls of P2, etc.
d) the causal interaction between theme and initiatives is defined as a management transaction (MT) in formula (2), which also includes the evaluation of the vertical interaction type (W) and the horizontal interaction type (C):

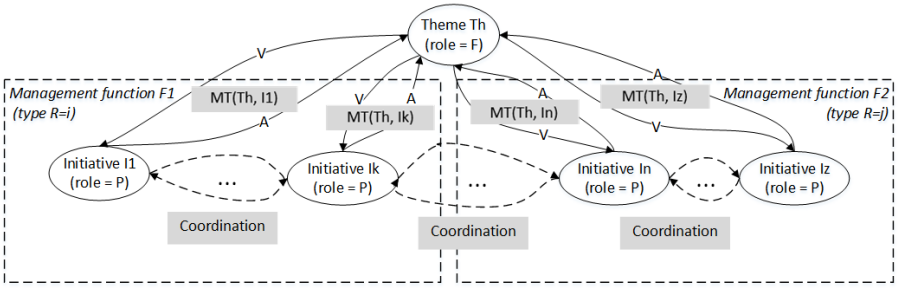
$$MTq(r, (Th, I), C, W) = MTq\{(r, F(Th), P(I1), A1, V1, C, W), (r, F(Th), P(I2), A2, V2, C, W), ..., (r, F(Th), P(In), An, Vn, C, W)\} \quad (2)$$

where:

- q – Agile hierarchy level;
- r – management function type;
- MT(Th, I) – specific MT – between theme and initiative(s);
- C – coordination flow type (horizontal interaction type of activities);
- W – management control flow type (vertical interaction type of activities);
- F(Th) – activity "theme" in the role of management function F;
- {P(I1), ..., P(In)} – a set of activities „initiatives" in the role of processes P;

- A – state attributes set A = {A1, A2, ..., An};
- V – management control set V = {V1, V2, ..., Vm}.

Figure 19a contains an illustration of vertical interactions in a modified Agile hierarchy where theme (Th) is in the role of management function F and controls a set of initiatives (I = {I1, …, Ik, …, In, …, Iz}). All initiatives from the set {I} are in the role of process P (i.e. all initiatives I are controlled by the same theme Th). Each interaction between theme Th and any of the related initiatives are management transactions {MT(Th, I1), ..., MT(Th, Ik), ..., MT(Th, In), ..., MT(Th, Iz)}. A vertical interaction in a modified Agile hierarchy where theme (Th) is in role F and controls a set of initiatives (I = {I1, …, Ik, …, In, …, Iz)) is called management control. Horizontal interaction of the initiatives is called coordination.




Source: Created by the author

Figure 19a. Modified Agile hierarchy: example of the management transactions of the single theme (Th) (in the role of F) with a set of initiatives (I) (in the role of P)

6. On Agile hierarchy level q = “Initiative” for every management function F of type r is applicable that:
   a) any initiative (I) is defined as MTq(I, E) = MTq(F(r), P, A, V), where initiative (I) is in the role of management function (F), causally related epics (E) are in the role of process (P), flow A denotes a set of state attributes of process P, flow V denotes a set of controls of process P (feedback from initiative (I) to epics (E)).
   b) epic (E) in MTq(I, E) denotes a set of processes P={P1, P2,..., Pm} managed by function F (a single initiative I). Flow A denotes a set of state attributes A = {A1, A2, …, Am} matched with the corresponding Process P={P1, P2,…, Pm}, where A1 is state attributes of P1 (e.g. epic E1 state data), A2 – state attributes of P2, etc.

c) the initiative (I in the role of F) carries out a set of controls of the epics (a set of E = {E1, E2, ..., En} in the role of P) through the feedback flow V, sends control data (instructions, requirements) related to the corresponding epics. Flow V denotes a set of controls (feedback impacts) V = {V1, V2, …, Vn} matched with the corresponding Process P = {P1, P2, …, Pm} where V1 is management controls of P1 (e.g. impact to epic E1), V2 – is management controls of P2, etc.

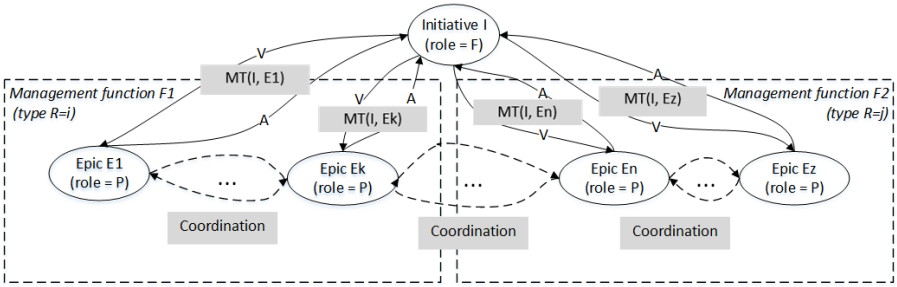
d) the causal interaction between initiative and epics is defined as a management transaction (MT) in formula (3), which also includes the evaluation of the vertical interaction type (W) and the horizontal interaction type (C):

$$MTq(r, (I, E), C, W) = MTq\{(r, F(I), P(E1), A1, V1, C, W), (r, F(I), P(E2), A2, V2, C, W), ..., (r, F(I), P(En), An, Vn, C, W)\} \quad (3)$$

where:

- q – Agile hierarchy level;
- r – management function type;
- MT(I, E) – specific MT – between initiative and epic(s);
- C – coordination flow type (horizontal interaction type of activities);
- W – control flow type (vertical interaction type of activities);
- F(I) – activity „initiative“ in the role of management function F;
- {P(E1), ..., P(En)} – a set of activities „epics“ in the role of process P;
- A – state attributes set A = {A1, A2, ..., An};
- V – management control set V = {V1, V2, ..., Vm}.

Figure 19b contains an illustration of vertical interactions in a modified Agile hierarchy where initiative (I) is in the role of management function F and controls a set of epics (E = {E1, …, Ek, …, En, …, Ez}). All epics from the set {E} are in the role of process P (i.e. all epics E are controlled by the same initiative I). Each interaction between initiative I and any of the related epics are management transactions {MT(I, E1), ..., MT(I, Ek), ..., MT(I, En), ..., MT(I, Ez)}. A vertical interaction in a modified Agile hierarchy where initiative (Th) is in role F and controls a set of epics (E = {E1, …, Ek, …, En, …, Ez}) is called management control. Horizontal interaction of the epics is called coordination.




Source: Created by the author

Figure 19b. Modified Agile hierarchy: example of the management transactions of the single initiative (I) (in the role of F) with a set of epics (E) (in the role of P)

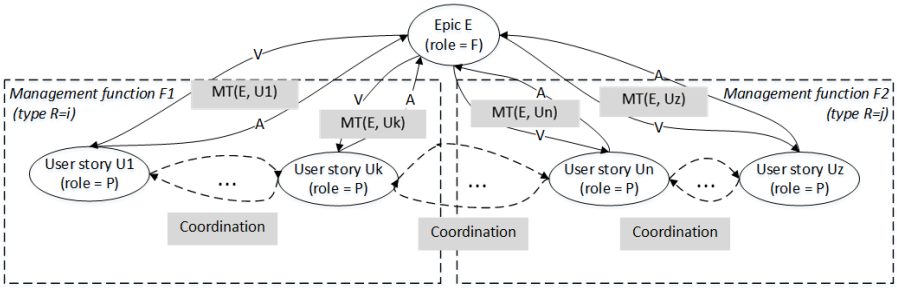
7. On Agile hierarchy level q = "Epics" for every management function F type r is applicable that:
   a) any epic (E) is defined as MTq(E, U) = MTq(F(r), P, A, V), where epic (E) is in the role of management function (F), causally related user stories (U) are in the role of process (P), flow A denotes a set of state attributes of process P (e.g. user stories state data), flow V denotes a set of controls of process P (feedback from epic (E) to related user stories (U) ).
   b) the user story (U) denotes a set of processes P = {P1, P2,..., Pm} managed by function F (a single epic E). Flow A denotes a set of state attributes data A = {A1, A2,…, Am} matched with the corresponding set of processes P = {P1, P2,…, Pm}, where A1 is state attributes of P1 (i.e. user story U1 state data), A2 – attributes data of P2, etc.
   c) the epic (E is in the role F) carries out a set of controls of the user stories (a set of U = {U1, U2,..., Un} in the role P) through the feedback flow V, sends control data (instructions, requirements) related to the corresponding user stories. Flow V denotes a set of controls (feedback impacts) V = {V1, V2,…, Vn} matched with the corresponding Process P = {P1, P2,…, Pm} where V1 is management controls of P1, V2 is management controls of P2, etc.
   d) the causal interaction between epic and user stories is defined as a management transaction (MT) in formula (4), which also includes the evaluation of the vertical interaction type (W) and the horizontal interaction type (C):

$$MTq(r, (E, U), C, W) = MTq\{(r, F(E), P(U1), A1, V1, C, W) \quad (4)$$
$$(r, (F(E), P(U2), A2, V2, C, W), ..., (r, (F(E), P(Un), An, Vn, C, W)\}$$

where:

- q – Agile hierarchy level;
- r – management function type;
- MT(E, U) – specific MT – between epics and user story(ies)
- C – coordination flow type (horizontal interaction type of activities);
- W – management control flow type (vertical interaction type of activities);
- F(E) – activity "epic" in the role of management function F;
- {P(I1), ..., P(In)} – a set of activities "user stories" in the role of process P;
- A – state attributes data set A = {A1, A2,.., An};
- V – management control data set V = {V1, V2, ...Vm}.

Figure 19c contains an illustration of vertical interactions in a modified Agile hierarchy where epic (E) is in the role of management function F and controls a set of user stories (U = {U1, …, Uk, …, Un, …, Uz}). All user stories from the set {U} are in the role of process P (i.e. all user stories U are controlled by the same epic E). Each interaction between epic E and any of the related user stories are management transactions {MT(E, U1), ..., MT(E, Uk), ..., MT(E, Un), ..., MT(E, Uz)}. A vertical interaction in a modified Agile hierarchy where epic (E) is in role F and controls a set of user stories (U = {U1, …, Uk, …, Un, …, Uz)) is called management control. Horizontal interaction of the user stories is called coordination.




Source: Created by the author

Figure 19c. Modified Agile hierarchy: example of the management transactions of the single epic (E) (in the role of F) with a set of user stories (U) (in the role of P)

Let us examine possible flow content classifications that are applied in practice when epic and user story interaction is analysed, which is defined as MT(F(E), P(U), A, V, C, W).

State flow A = {A1, A2, A3}, where:

- workflow status of user story U defined as flow A content A1 = {a1.1 = „To do“, a1.2 = „In progress“, a1.3 = „In review“, a1.4 = „In testing“, a1.5 = „Ready for deployment“, a1.6 = „Done“, a1.7 = „Cancelled“};
- user story U assignee defined as flow A content A2 = {a2.1 = „the same“; a2.2 = „changed“};
- time spent working on user story defined as flow A content A3 = {A3.1 = “within allocated time limit”, A3.2 = “close to allocated time limit”, A3.3 = “over the allocated time limit”“).

Therefore, the content of A1 flow in the MTq(E, U1)=MTq(r, F(E), P(U1), A1, V1, C, W) can be specified as follows:

- workflow status of user story U status is “In progress” (a1.2);
- user story U assignee defined as a specific developer (name) (a2.1);
- user story U time spent in status “In progress” is taking longer than defined.

The status of a controls flow V = {V1, V2, V3} can be as follows:

- optimal/expected duration of user story completion (based on team velocity and progress of the user story: is the user story expected to be completed within an expected time frame or not) – V1 = {v1.1 = “user story fits within an expected time frame”, v1.2 = “user story does not fit within an expected time frame”}.
- total time remaining dedicated to complete all user stories (original estimate of the epic) based on velocity if a new user story is added could indicate if it would fit or not into the remaining time to complete all the user stories under epic – V2 = {v2.1 = “added user story fits into originally estimated time”, v2.2 = “added user story does not fit into originally estimated time”}.

Therefore, the content of V1 flow in the MTq(E, U1)=MTq(r, F(E), P(U1), A1, V1, C, W) can be specified as follows:

- user story U1 flow V1 content is v1.1 – “user story fits within the expected time frame”.
- user story U5 flow V2 content is v2.2, meaning the added user story does not fit into the originally estimated time and the original

estimate should be reviewed by project experts (project managers, product owners).

By further defining the management control flow V or state attributes A specifications, the Agile project management tool could be even further specified determining missing information as the presented modified Agile management method allows to introduce a specification of informational flows and check the states of informational flows in a computerized way.

8. On Agile hierarchy level q = "User stories" for every management function F type r is applicable that:
   a) any user story (U) is defined as MTq(U, X) = MTq(F(r), P, A, V), where the user story (U) is in the role of management function (F), any lower level causally related activity (e.g. sub-task) (X) is in the role of process (P), flow A denotes a set of state attributes data of process P (e.g. lower level Agile activity state data), flow V denotes a set of controls of process P (feedback from the user story (U) to any related lower-level activity (X) ).
   b) the lower level activity (X) denotes a set of processes P = {P1, P2,..., Pm} managed by function F (a single user story U). Flow A denotes a set of state attributes data A = {A1, A2,…, Am} matched with the corresponding set of processes P = {P1, P2,…, Pm}, where A1 is state attributes data of P1, A2 – attributes data of P2, etc.
   c) the user story (U in the role F) carries out a set of controls of the lower level activities (a set of X = {X1, X2,..., Xn} in the role P) through the feedback flow V, sends control data (instructions, requirements) related to the corresponding lower level Agile activities. Flow V denotes a set of controls (feedback impacts) V = {V1, V2,…, Vn} matched with the corresponding Process P = {P1, P2,…, Pm} where V1 is management control data of P1, V2 is management control data of P2, etc.
   d) the causal interaction between user story and lower-level Agile activities is defined as a management transaction (MT) in formula (5), which also includes the evaluation of the vertical interaction type (W) and the horizontal interaction type (C):

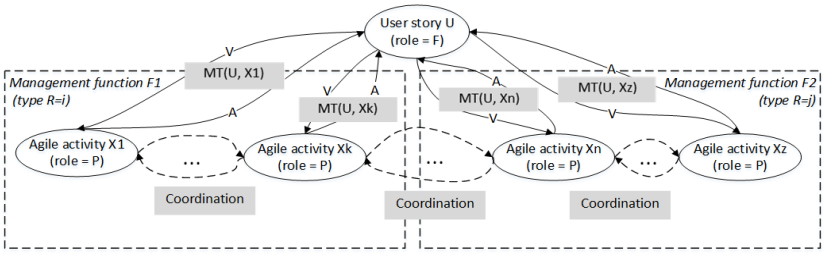
$$MTq(r, (U, X), C, W) = MTq\{(r, (F(U), P(X1), A1, V1, C, W), (r, (F(U), P(X2), A2, V2, C, W), ... (r, (F(U), P(Xn), An, Vn, C, W)\} \quad (5)$$

where:

- q – Agile hierarchy level;
- r – management function type;
- MT(U, X) – specific MT – between user story and lower-level agile activities;
- C – coordination flow type (horizontal interaction);
- W – management control flow type (vertical interaction type of activities);
- F(U) – activity „user story“ in the role of management function F;
- {P(X1), ..., P(Xn)} – a set of activities „lower level Agile activities“ in the role of processes P;
- A – state attributes data set A = {A1, A2,.., An};
- V – management control data set V = {V1, V2, ...Vm}.

Figure 19d contains an illustration of vertical interactions in a modified Agile hierarchy where user story (U) is in the role of management function F and controls a set of lower-level Agile activities (X = {X1, …, Xk, …, Xn, …, Xz}). All lower-level Agile activities from the set {X} are in the role of process P. Each interaction between user story U and any of the related lower level Agile activities are management transactions {MT(U, X1), ..., MT(U, Xk), ..., MT(U, Xn), ..., MT(U, Xz)}. A vertical interaction in a modified Agile hierarchy where the user story (U) is in role F and controls a set of lower-level Agile activities (X = {X1, …, Xk, …, Xn, …, Xz)) is called management control. Horizontal interaction of the lower level Agile activities is called coordination.

It should be noted that generally, the user story level is the final level in Agile hierarchy and in case of the need to use any lower-level activity the presented method allows us to take that into account as well.




Source: Created by the author

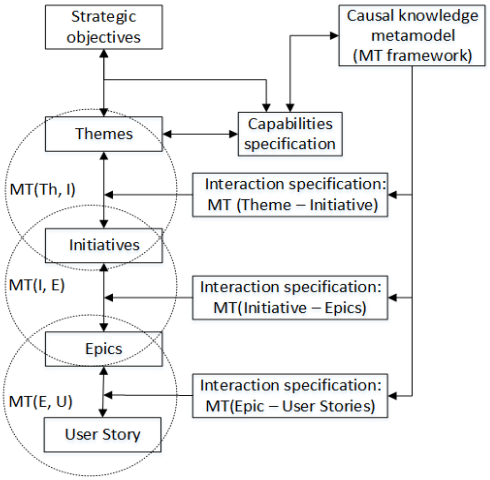
Figure 19d. Modified Agile hierarchy: example of the management transactions of the single user story (U) (in the role of F) with a set of lower level Agile activities (X) (in the role of P)

9. Any Agile activity on each level is considered as a white box in mutual (vertical and horizontal) interactions in the Agile hierarchy. The (informational) content of the mutual (coordination) interactions of these Agile activities is predefined by the definition of the internal structure of the management transaction MT (F, P, A, V). Therefore, it is possible to create classifications of interaction types (types of coordination) between different Agile activities.
10. Classifications of vertical and horizontal interaction types (types of coordination) give a background for formal analysis of the content of interactions.

Summarizing the enumerated features of the modified Agile hierarchy, the definition of the causal Agile activity follows:

a) Any causal Agile activity (of type q = (theme, initiative, epic, user story)) belongs to some management function type ($r \in R$) and has a predefined structure defined as management transaction: MTq(r) = (F(r), P, A, V);
b) The status of any causal Agile activity (at a moment in time) can be identified by analysis of the content of its elements (F, P, A, V);
c) The role of Agile hierarchy activities (theme, initiative, epic, user story) in the management transaction is relative: Agile hierarchy activities (theme, initiative, epic, user story) could be specified as F or P depending on their role in the management transaction based on their levels in the modified Agile hierarchy (as described in more detail in subsection "2.2.1. Relativity Principle of Assigning Roles");

The presented modified (causal) Agile hierarchy (formally defined in previous paragraphs as features from 1 to 8) is illustrated in detail in Figure 20. „Interaction specification" in Figure 20 corresponds to formulas (2) – (5). The strategic objectives are decomposed using the MODAF StV-1 model and a set of capabilities is revealed. Each capability matches the highest-level Agile activity, i.e. some theme. Specifying themes in alignment with capabilities is a more accurate way than defining themes directly based on experience or discussion. As presented above, in the next steps, a set of initiatives is derived from any single theme, a set of epics is derived from any single initiative and a set of user stories is derived from any single epic. The interaction between all levels of the modified Agile hierarchy is specified using the causal knowledge metamodel, defined as management transaction (MT).




Source: Created by the author according to [106]

Figure 20. Modified Agile hierarchy detailed view

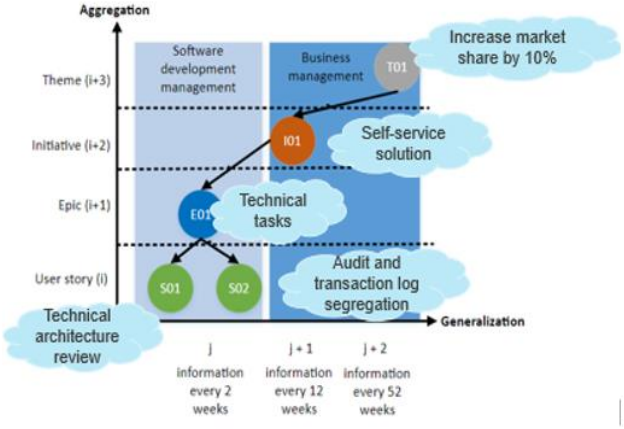
Each activity in the modified Agile hierarchy defined as a complex management transaction interacts with multiple (lower) level activities as multiple management transactions occur simultaneously at the same time. The complexity of the interactions can be quantitively evaluated by calculating the complexity indicators (presented in section 2.5)

An example of different activities in the modified Agile hierarchy is represented in Figure 21. Here the axis AG (aggregation) indicates the hierarchy of Agile activities (vertical interactions), and the axis GE (generalization) indicates the classification of Agile activities into types. Theme T01 in Figure 21 represents an example of themes in Figure 20, initiative I01 – initiatives, epic E01 – epics and user stories are marked as S01 and S0. The interaction of activities T01 and I01 is a management transaction that is a complex process and it is defined as a set of information flows specified in the element "Interaction specification: MT (Theme – Initiative)" (Figure 20) and so on respectively (I01 – E01, E01 – S01, E01 – S02).

An example of Agile hierarchy with the two types of management functions is depicted in Figure 21. The modified Agile hierarchy reveals the different management functions (software development management and business management) that are executed to achieve business objectives. Theme T01 to initiative I01 relation presents the details (interaction content) of business objectives needed to be achieved and high-level means of doing that (increase market share by 10% by creating a self-service solution). Relation initiative I01 to epic E01 specifies on a high level the effort needed

to implement the self-service solution. Epic E01 cover the development tasks under the software development management function that need to be implemented in order to develop the self-service solution. User stories S01 and S02 are examples of detailed requirements that enable us to complete epic E01 and eventually, complete initiative I01 and theme T01.

The interaction specification in Figure 20 between each adjacent level of Agile hierarchy indicates that in order to model Agile hierarchy from the causal modelling perspective, an MT specification MT = (F, P, A, V) needs to be provided. For example, the theme needs to contain information relevant to the execution of the initiative (duration, constraints, resources, informational attributes, financial goals, etc.). The initiative needs to contain information relevant to epics (i.e. duration of the initiative, duration of the epic, related teams involved in the EAS project, etc.). The interaction specification between the epic and user story should contain information relevant to the completion of the user story (NFRs, duration of the story, the sequence of how the user stories under the epic should be completed to make a streamlined process).




Source: Created by the author according to [106]

Figure 21. Example of Agile activity hierarchy

### 2.2.1.Relativity Principle of Assigning Roles

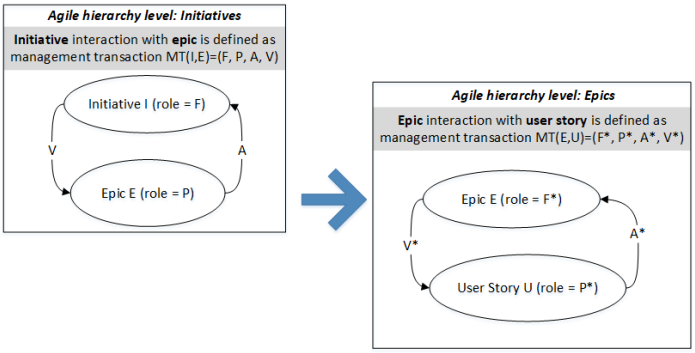
When specifying a causal Agile hierarchy, the assignment of type P (process) or type F (management function) to the Agile activity considered an MT element at some step of modelling is a relative matter.

For example, the interaction between initiative (defined as MT(I,E) = MT(F, P, A, V) and epic (defined as MT(E, U) = MT(F*, P*, A*, V*) ) is examined.

On the level of “initiatives” in Figure 22, initiative (I) is in the role of management function (F) and epic (E) is in the role of the process (P) in this interaction. A feedback loop (initiative, epic) includes flow A (state attributes of epic) and controls V (impact to epic). A content of flow V is the requirements focused on any element (F*, P*, A*, V*) of MT(E, U) or on the combination of these elements.

On the level of “epics”, epic (E) is in the role of management function (F) and user story (U) is in the role of the process (P) in this interaction. A feedback loop (epic, user story) includes flow A* (state attributes of user story) and controls V* (impact to user story). The impact of epic (E) on user story (U) is the content of flow V* (controls).

The user story could also be considered as a white box and defined as MT(U,X) = MT′(F′, P′, A′, V′) where X denotes any optional lower-level Agile activity.




Source: [50]

Figure 22. Example of the impact of initiative I on epic E

In the same way, the relative roles are assigned to the MT elements at other levels of Agile hierarchy. For example, in MT(Th,I) = MT(F, P, A, V), a lower-level activity “initiative” is considered a process (P) associated with a higher-level activity “theme”. The role of activity “theme” in this interaction is “management function (F)”. This allows for calculating EAS project complexity indicators based described in Section 2.5

## 2.3. Agile Activities Interaction Types Classification in the Modified Agile Hierarchy

In order to systematize the interaction types of different Agile activities, it is rational to define the interactions in the abstract Space of Processes (SP), introduced by Gudas [34].

The Space of Processes (SP) [34] is defined as an abstract 3D space with the coordinate axes (AG, GE, T): SP = (AG, GE, T), Here: AG – aggregation axis, GE – generalization axis, T – time axis, DE – axis of detailization (opposite direction to AG axis), CO – axis of concretization (opposite direction to GE axis). The Space of Processes (SP) is adapted here to represent interactions between Agile hierarchy activities, which are specified as management transactions (MT). Types of interactions between Agile hierarchy activities (defined as MTs) can be classified according to the mutual position of MTs in the Space of Processes (SP) [34] concerning the type of coordinate axes (AG, GE, T): SP = (AG, GE, T).

### 2.3.1.Formal Description of The Activity Interactions

Any Agile activity is defined as MT = MT(F, P, A, V) and is a point in the Space of Processes with indexes (i, j, t, r, p), here i – level of aggregation (AG), j – level of generalization, t – time, r – type of function F (activity type), p-type of process P (see Figure 1 for MT structure).

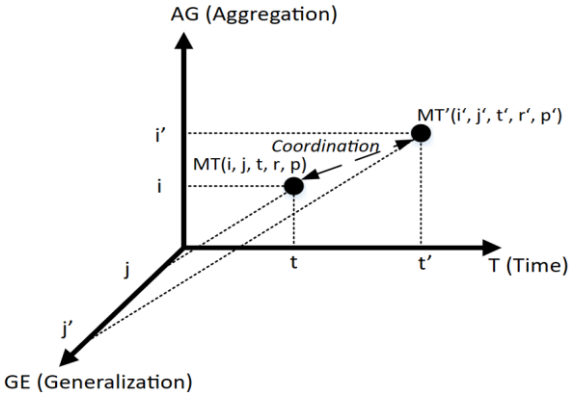
The semantics of interaction between different MTs (i.e. the content of information transferred from one MT(i, j, t, r) to another MT' (I', j', t', r') depends on the differences in the position (location defined by indexes (i, j, t, r, p)) of these MTs in the Space of Processes SP = (AG, GE, T) (Figure 23).

Two types of interaction between Agile activities are distinguished in the Space of Processes [34];

- vertical interaction (called management control);
- horizontal interaction (called coordination).

Vertical interaction between Agile activities which are considered as management transactions (MTs) is presented in Figure 23 and Figure 24. Vertical interaction is between Agile activities from different aggregation AG levels (i) and (i').

The general abstract case of vertical interaction from different aggregation levels (i) and (i') is depicted in Figure 23; here Agile activities are represented as black boxes – the dots MT(i, j, t, r, p) and MT' (I', j' ,t' ,r', p'). Their internal 'structure is not defined, as it is now in the Agile methodology.




Source: Created by the author based on [30]

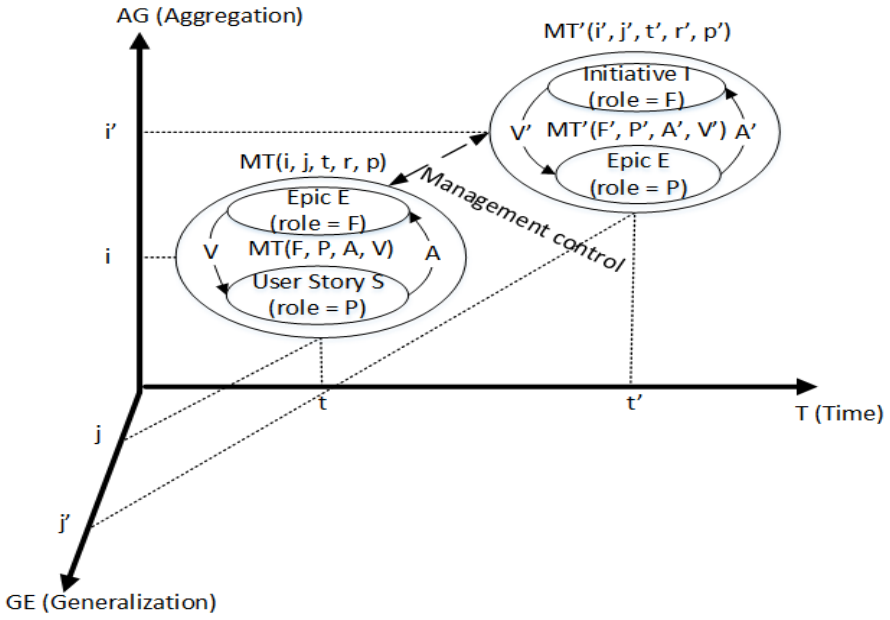
Figure 23. Vertical interaction between Agile activities MT and MT'

To illustrate the vertical interaction between Agile activities modelled as a white box (by following the MT structure) Figure 24 is presented by the example of initiative (I) and epic (E').

Here both Agile activities are defined as MT: the initiative (I) is defined as MT(I,E) and the epic (E) is defined as MT(E,U), e.g. represented as white boxes:

- epic (E) is defined as MT(E,U) = MT(F, P, A, V) and is located in the point (i, j, t, r, p) of the Process Space, here epic (E) is in the role of function (F), the user story (S) is in the role of the process (P) under control of F;
- initiative (I) is defined as MT(I,E) = MT' (F', P', A', V') and is located in the point (I' ,j' ,t' ,r' ,p') of the Process Space, here initiative (I) is in the role of function (F), and the same epic (E) is in the role of the process (P) under control of F.

The initiative I = MT' (F', P', A', V') and epic E = MT (F, P, A, V) are on the different levels of aggregation hierarchy AG ($i \neq I'$) and generalization GE level ($j \neq j'$), time is different ($t \neq t'$), and processes P are different ($p \neq p'$).




Source: Created by the author based on [30]

Figure 24. Vertical interaction between Agile activities defined as MTs

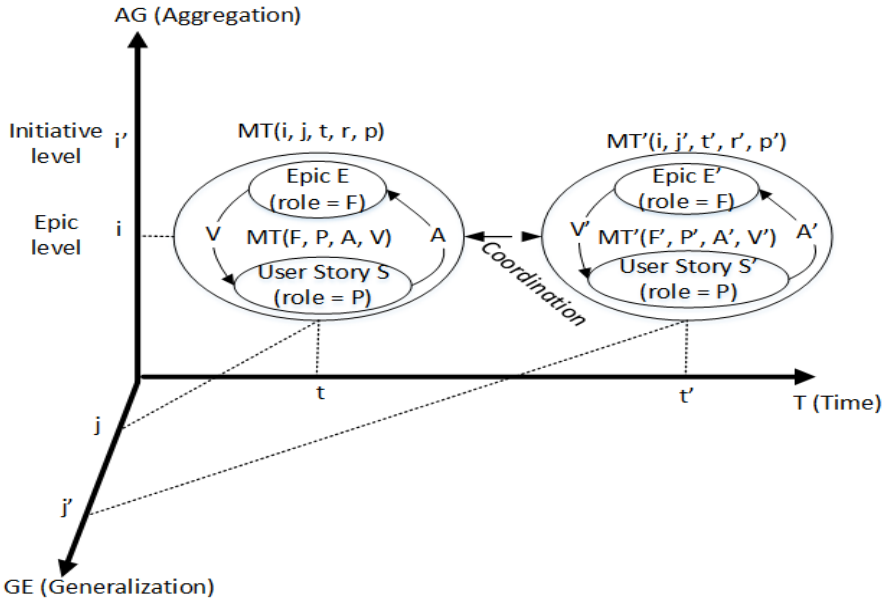
Horizontal interaction between Agile hierarchy activities at the same level of aggregation hierarchy (AG) is called coordination.

An example of coordination is depicted in Figure 25 by example of epic (E) and epic (E'). Here both Agile activities are defined as MT (represented as white boxes):

- epic (E) is defined as MT (F, P, A, V)) and is located in the point (i, j, t, r, p) of the Process Space,
- epic (E′) is defined as MT′ (F', P', A', V') and is located in the point (i', j', t', r' ,p') of the Process Space,
- epic E = MT and epic E' = MT′ are on the same level (i) of aggregation hierarchy AG.

In the case of horizontal interaction (coordination in Figure 24), the Agile activities are of the same type and are on the same hierarchy level (epic level i). An example of the coordination between two different epics E and E′ occurs in Figure 25.

The same principle applies to any type of the same level Agile activities – theme to theme(s), initiative to initiative(s), user story to user story(-ries).




Source: Created by the author based on [30]

Figure 25. Horizontal interaction between Agile activities defined as MTs

The mutual location of two Agile activities MT and MT' in the Process Space (PS = (AG, GE, T) can be described through the values of indexes (i, j, t, r, p):

- The level in the Aggregation axis (index i) is the same (i = i′);
- The level in the Aggregation axis (index i) is different (i ≠ i′);
- The level in the Generalization axis (index j) is the same (j = j′);
- The level in the Generalization axis (index j) is different (j ≠ j′);
- The type of the management function F is the same (r = r′);
- The types of the management functions are different (r ≠ r′);
- The process P of different MT and MT′ is the same (p = p′);
- Processes P of different MT and MT' are different (p ≠ p′);
- Managed processes are of the same period (t = t′);
- Managed processes are of different periods (t ≠ t′).

Different combinations of indexes i, j, t, r and p values are used to define MT coordination meta-types and types, as classified in Table 15.

A typical description of the requirements for Enterprise application project components (situation as-is) together with the normalized specification of Agile activity according to the MT definition includes the attributes in Table

12. As is shown, the normalized specification contains enterprise management function and MT attributes compared to the typical description.

Table 12. Typical and modified (normalized) Agile project description

| **Typical Agile project description** | **Modified Agile project description** | |
|---|---|---|
| --- | Enterprise management function | |
| **Agile activity type** | Agile activity type | |
| --- | **Management transaction content** | |
| Activity ID | Activity ID (role – Function F) | |
| Activity description | Activity description | |
| --- | Activity ID (role – Process P) | Description of P |
| --- | Flow A (ID) | Flow A description |
| --- | Flow V (ID) | Flow V description |
| Summary | Summary | |

Source: Created by the author

Considering TIES elements (theme, initiative, epic, user story) as self-managed activities (MTs) the internal content of the hierarchy elements could be specified properly – normalized using the definition of MT = (F, P, A, V).

A brief example, explaining the approach to normalized specification of Agile activities is presented in Tables 13 and 14.

Table 13 shows the mapping of the requirements (as-is description of the situation) for an enterprise application software project to defined normalized specification (standard view).

Table 13. Agile activity (as-is) specification example

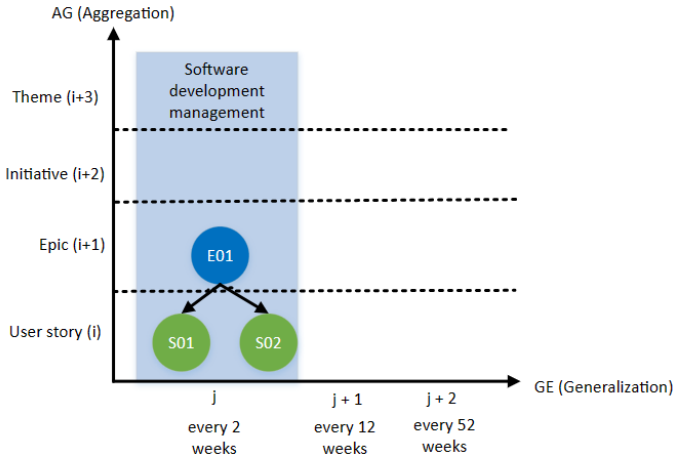
| **Enterprise management function** | **Agile activity type** | **Activity ID (Role – Function F)** | **Summary** | **Management transaction components (activity ID + description)** | | | |
|---|---|---|---|---|---|---|---|
| | | | | **Function F (ID + desc)** | **Process P (ID + desc)** | **Flow A (ID +desc.)** | **Flow V (ID + desc)** |
| N/A | Theme | T01 | N/A | N/A | N/A | N/A | N/A |
| N/A | Initiative | I01 | N/A | N/A | N/A | N/A | N/A |
| r100-software development management | Epic | E01 | Technical tasks | N/A | N/A | N/A | N/A |
| r100-software development management | User story | S01 | Technical architecture review | N/A | partially | N/A | N/A |
| r100-software development management | User story | S02 | As a developer, I want audit and transaction log segregation | N/A | N/A | N/A | N/A |

Source: Created by the author based on [34][137]

Legend:

- N/A – no information provided.
- Management transaction internal elements:
  - Function (F);
  - Process (P);
  - State attributes (flow A);
  - Controls (flow V).

The abovementioned transactions between the elements in the TIES structure could be mapped to the coordinate system in the Space of Processes where on the GE axis there is a management function and time elements are represented, and on the AG axis – the different elements of the TIES structure projecting to the strategic goals. The projection is displayed in Figure 26.




Source: Created by the author based on [34, 137]

Figure 26. Project as-is state in the Space of Processes

The columns in Table 13 are relevant to the components of the formal specification of MT = (F, P, A, V) where in order to ensure a causal relationship between Agile activities and in this way ensure business and IT alignment – the elements that are currently missing must be provided.

This is further done by adding these parts of information by the employees responsible for delivering the IT solution (business analysts, product owners, developers or other experts).

An example of the normalized specification (situation to-be) of requirements where business and IT alignment is ensured is displayed in Table 14 with the missing information added (to ensure a causal relationship between activities from different levels of Agile hierarchy).

Table 14. Example of Causal Agile hierarchy items normalized specification

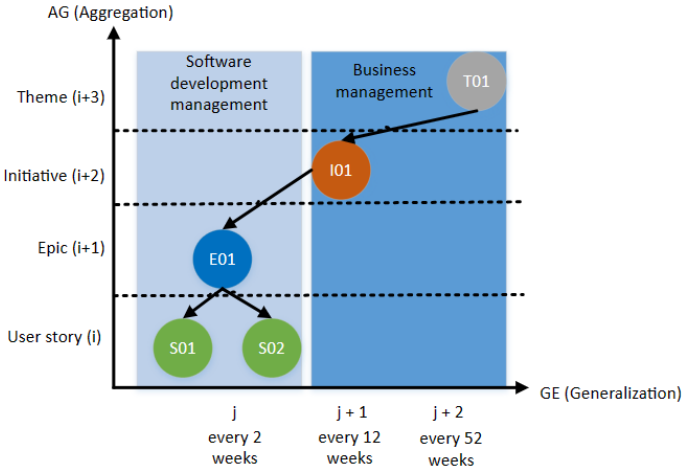
| **Enterprise management function** | **Agile activity type** | **Activity ID (Role – Function F)** | **Summary** | **Management transaction components (activity ID + description)** | | | |
|---|---|---|---|---|---|---|---|
| | | | | **Function F (ID + desc)** | **Process P (ID + desc)** | **Flow A (ID +desc.)** | **Flow V (ID + desc)** |
| r001 – Business management | Theme | T01 | Increase the market share of home loans by 10% by the end of 2022 | + | + | + | N/A |
| r001 – Business management | Initiative | I01 | Create a self-service solution for checking loan remortgage options | + | + | + | + |
| r100-software development management | Epic | E01 | Technical tasks | + | + | + | + |
| r100-software development management | User story | S01 | Technical architecture review | + | + | + | - |
| r100-software development management | User story | S02 | As a developer, I want audit and transaction log segregation | + | - | + | + |

Source: Created by the author based on [34][137]

Legend:

- Text in yellow – added after normalization.
- "–" – information does not exist.
- "+" – information provided.
- Management transaction internal elements:
  - Function (F);
  - Process (P);
  - State attributes (flow A);
  - Controls (flow V).

The information from Table 14 provides the all required information for the project requirements for the EAS project development situation that is displayed in Figure 27.




Source: Created by the author based on [34, 137]

Figure 27. Project to-be state in the Space of Processes

### 2.3.2.Coordination Meta-types and Types

Coordination of different Agile activities is based on the transferring of information from one (coordinating) activity (modelled as MT = (F, P, A, V)) to another (i.e. MT′= (F′, P′, A′, V′) ) when the content of the coordinated MT′ is altered.

It is assumed that the coordinating MT has the right to:

- Transfer the flow of information (attributes, data, requirements, constraints) from the coordinating MT to the coordinated MT′;
- The flow of information transmitted may affect the content of the MT′ elements (F′, P′, A′, V′).

Different combinations of indexes r, p, and t values are used to define MT coordination meta-types, as classified in Table 15 (here: 1 – "the same", 0 – "different"). For example, meta-type A denotes the interaction of two activities in a single time period t when the same process p is managed, and management function F type r is the same, meta-type C – when management function types are different. Meta-types of Coordination of Management Transactions are presented in Table 15.

This taxonomy of coordination meta-types introduced in [34] for specification interactions of business management activities is formally defined as the Elementary management cycle (EMC). The EMC is defined in [34] as a detailed structured model of the business activities interaction and is the basic construct of enterprise management modelling [49, 138].

This thesis examines MT, which does not detail the internal structure of management function F, as it is the case with EMC. Therefore, the formal classifications presented in [34] can be applied to models with a different structure and to analyse their interactions. The coordination of this type of transaction has not been examined until now. In the course of the research, there was an article was published [50] which describes individual cases of coordination between MTs.

Coordination classification is necessary for defining status indicators of Agile projects to reveal the underlying complexity of coordination demand.

Table 15. Coordination taxonomy (meta-types)

| **MT identifiers** | **Coordination meta-types** | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Identifiers (t, r, p) | A | B | C | D | F | G | H | L |
| Type of management function (r) | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| Process identifier (p) | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| Time period (t) | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

Source: [34]

### 2.3.3. Meta-types of the Coordination of Business Management Activities

The Agile management activities (themes, initiatives, epics, user stories) are considered here as management transactions (MTs), and are constructions of larger granularity, not as detailed as EMC in [34].

The coordination taxonomy of MTs has not been examined so far, therefore, it will be described below.

Coordination types correspond to the reciprocal arrangement of MTs in the Space of Processes SP = (AG, GE, T) (see Figure 23).

Based on the taxonomy of coordination meta-types in Table 15, coordination types of management transactions are classified in Table 16. MT

coordination types are defined by combining all possible combinations of indexes (i, j, t, r, and p) (here: 1 – "same", 0 – "different").

Table 16. Types of coordination

| Identifiers | Coordination type | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **A0** | **A1** | **A2** | **A3** | **B0** | **B1** | **B2** | **B3** | **C0** | **C1** | **C2** | **C3** | **D0** | **D1** | **D2** | **D3** |
| i | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| j | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| t | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| r | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| p | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

Source: [34]

Table 16. Types of coordination (continued)

| Identifiers | Coordination type | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **F0** | **F1** | **F2** | **F3** | **G0** | **G1** | **G2** | **G3** | **H0** | **H1** | **H2** | **H3** | **L0** | **L1** | **L2** | **L3** |
| i | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| j | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| t | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| r | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| p | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

Source: [34]

#### 2.3.3.1. Coordination Types in the Same Time Period

Coordination meta-types A, B, C, and D cover different coordination cases (types) in the same time period t when the values of the other MTs identifiers (i, j, r, and p) overlap or differ (Table 15) and are formally described in [34]. These types of coordination cover real-time monitoring and management of enterprise activity. To illustrate some of the peculiarities of the EAS project management in an Agile environment the coordination types A0, A1, A2, and A3 for the same period (where t = t') will be described in more detail.

Coordination meta-type A (t = t′, r = r′, p = p′) comprises of four coordination types A0, A1, A2 and A3:

1. Coordination type A0 (i = i′, j = j′, t = t′, r = r′, p = p′) – the location of the two MTs in the Space of Processes is identical (i = i′, j = j′, t = t′) and these MTs are associated with the same process (p = p′) and management function type (r = r′). Type A0 indicates two identical activities (duplication), one of which needs to be eliminated. A0 indicates a surplus of activities;
2. Coordination type A1 (i ≠ i′, j = j′, t = t′, r = r′, p = p′) – the level of two MTs on the AG axis is different (i ≠ i'); the values of the other identifiers are the same (overlap);

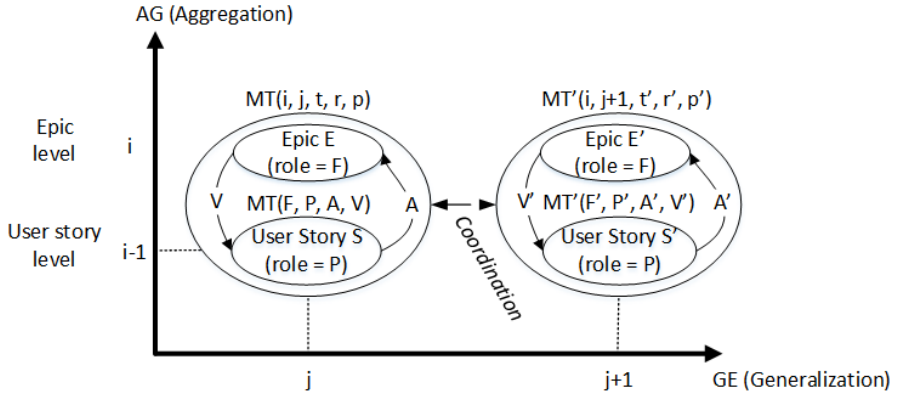
3. An abstract example of A1 (Figure 28): initiative (AG level i +1) and epic (AG level i).
4. Coordination type A2 (i = i′, j ≠ j′, t = t′, r = r′, p = p′) – the level of two MTs on the GE axis is different (j ≠ j′); the values of the other identifiers are the same (overlap);
5. An abstract example of A2 (Figure 29) initiatives (GE level j +1) and epic (GE level j).
6. Coordination type A3 (i ≠ i′, j ≠ j′, t = t′, r = r′, p = p′) – the level of two MTs on the AG axis and GE axis is different (i ≠ i′, j ≠ j′); the values of the other identifiers are the same (overlap).

Examples of the coordination type A1 in modified Agile management hierarchy occurring in the scope of one real project:

Coordination type A1 between theme Th1 (i+1, j, t, p, r) and initiative I1(i, j, t, p, r): theme Th1(i+1, j, t, p, r) represents a long-term objective: "in two years to reduce cost by X amount"; one of the linked initiatives I1 (i, j, t, p, r) "Stop using system Y, and transfer the features of the system to new system Z".

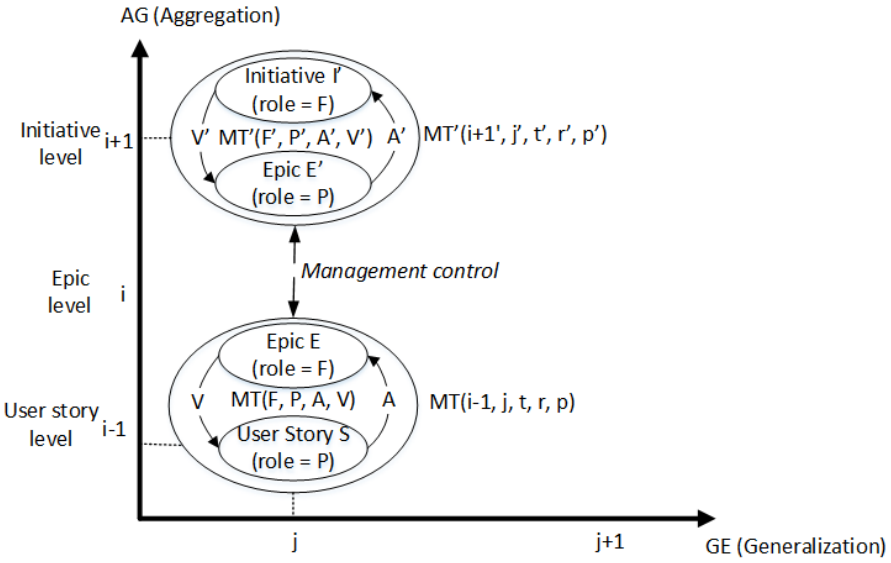
Coordination type A1 between initiative I1 (i, j, t, p, r) and epic E1(i-1, j, t, p, r): one of the linked epics E1(i-1, j, t, p, r) "Transfer system Y function N to system Z in 3 months";

Coordination type A1 between epic E1(i-1, j, t, p, r) and user story U1(i-2, j, t, p, r): one of the user stories u1(i-2, j, t, p, r) "in two weeks to transfer system Y function N component K to system Z".




Source: Created by the author based on [34]

Figure 28. Coordination type A1 in the Space of Processes




Source: Created by the author based on [34]

Figure 29. Coordination type A2 in the Space of Processes

### 2.3.3.2. Types of Coordination in Different Time Periods

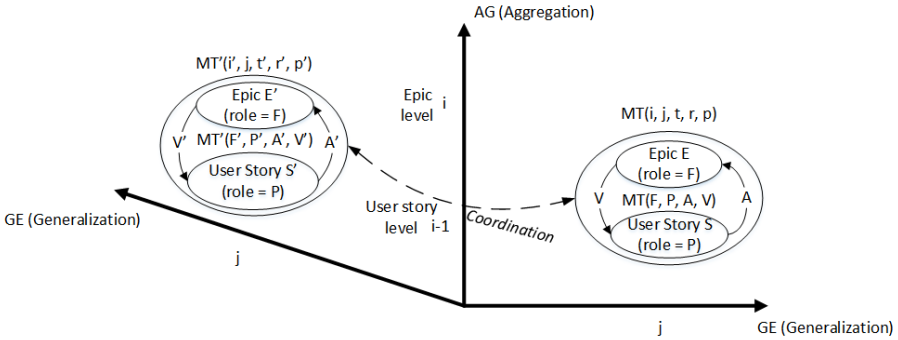
Coordination meta-types F, G, H, and L (Table 15) cover different cases (types) of MTs coordination in different time periods ($t \neq t'$) when the values of the other MTs identifiers (i, j, r, and p) are the same or different.

In enterprise governance practice, coordination of the activities from different time periods is called:

- planning or prediction: in case the implementation of one or more different MTs is to be performed from time t onwards, in the next period (t + 1);
- analysis: in case one or more different MTs are considered to be performed earlier, from time t back to time (t - 1).

In all these cases, the coordination should be performed by MT(i, j, t, r, p), which is implemented in the current period t, or by creating an additional coordinating MT*(i*, j*, t*, r*, p*) in period t. Here the coordination types H0, H1, H2 and H3 for different periods (where $t \neq t'$) will be described in more detail:' '

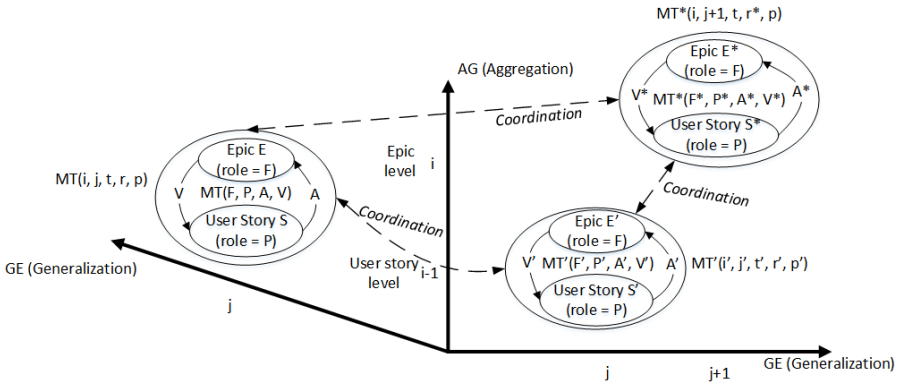
Coordination type H0 ($i = i'$, $j = j'$, $t \neq t'$, $r \neq r'$, $p = p'$) – coordination of MTs that have the same controlled process p during different time periods when management function type r is different, and the aggregation and the generalization levels are the same (Figure 30);




Source: Created by the author based on [34]

Figure 30. Coordination of type H0 in the Space of Processes

1. Coordination type H1 ($i = i', j \neq j', t \neq t', r \neq r', p = p'$) – coordination of MTs that have the same controlled process p during different time periods when management function types are different, the generalization levels are different, and the aggregation level is the same;
2. Coordination type H2 ($i \neq i', j = j', r \neq r', t \neq t', p = p'$) – coordination of MTs that have the same controlled process p during different time period when management function types and aggregation levels are different, and generalization level is the same;
3. Considering the difference between management function types ($r \neq r'$) of MTs, it is necessary to ensure their functional compatibility. This can be done using a common data warehouse or by coordination from the higher level MT* (Figure 31).




Source: Created by the author based on [34]

Figure 31. Coordination of type H2 in the Space of Processes

4. Coordination type H3 is a complicated case of coordination in different time periods ($t \neq t'$) when MTs that have the same controlled process (p

= p′), all the other identifiers of these MTs differ (since i ≠ i′, j ≠ j′, r ≠ r′, t ≠ t′).

Functional compatibility of different MTs can be done using a common data warehouse or coordination from the higher-level MT* (analogous to H2).

Modelling the content of the Agile activity interaction consists of two cases:

- Content of horizontal interaction between the same type of Agile activities: themes, initiatives, epics, and user stories;
- content of vertical interaction between the different types of Agile activities: themes and initiatives, initiatives and epics, epics and user stories.

### 2.3.4. Content of Horizontal Interactions

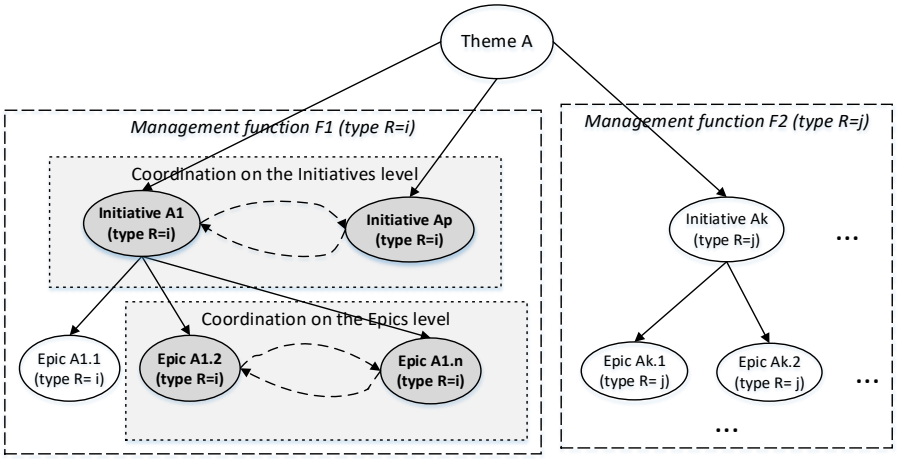
Horizontal interaction between the Agile activities is called coordination. Coordination occurs on the same level of Agile hierarchy between the Agile activities of the same type: themes, initiatives, epics, or user stories.

There are two different cases of horizontal interaction when the interaction is considered a collaboration of two self-managed activities using coordinating messages:

- a type of management function is the same (in Figure 32);
- a type of management function is different (in Figure 33 and Figure 34), i.e. type i = “business management”, type j = “software development management”.

In BPMN, this communication of independent processes when a message is transmitted is called “choreography”.

The third case (c) is more complicated when horizontal interaction is considered a management transaction MT(X, X*), where X and X* are activities on the same Agile hierarchy level (e.g. themes, initiatives, epics, user stories) also considered as management transactions (Figure 27).



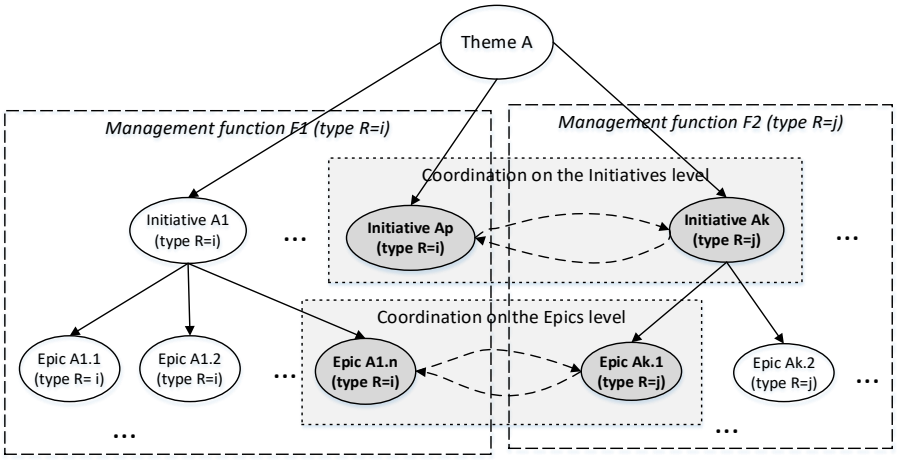

Source: [50]

Figure 32. Horizontal interaction (coordination) of the same management function F1 (type R = i)

In Figure 32, A1, Ap, Ak, etc. stands for the identifiers (id) of the initiative, A1.1, A1.n, Ak.1, Ak.2, etc. – for the id of the epic. Type R stands for the type of management function. In the example, the management function R is different (on the left R=i, on the right – R=j). The related Agile activities – epics (A1.1, etc.) represent a set of related epics to the initiative A1. Coordination occurs on the same level of Agile hierarchy within the same management function F1 by coordination message transmission. Coordination on the epic level in Figure 33 represents the software feature development policy impact within the same management function F1. It is decided which specific feature will be built under separate epics and how those components should be reused. Coordination on the initiative level in this example represents the business management impact, i.e. message under which specific initiative a certain system or project will be developed. This is necessary to ensure that different EAS development projects do not interfere with each other and that development efforts are not duplicated.

In case when the management function is different, i.e. in Figure 33, “A1”,..., “Ap”, “Ak” stands for the id’s of the initiatives, “A1.1”..”A1.n”, “Ak.1”, “Ak.2” – for the identifiers of the epics. In this example, the management function type R is different (on the left R=i, on the right – R=j). In this case, coordination occurs on the same level of Agile hierarchy within different management functions (for example, business development management F1 and software development management F2). This example shows the business development coordination with the development of application systems used by end customers. An example of such a case is the

launch of a new self-service portal for ordering banking products. On the IT side, i.e., the marketing area may provide some requirements to provide promotions and discounts for new customers, and the IT development area needs to implement the requirements. The marketing area needs to prepare marketing material for the launch of the system and the onboarding of customers. This needs to be coordinated with the IT area, as some instructions may be required.

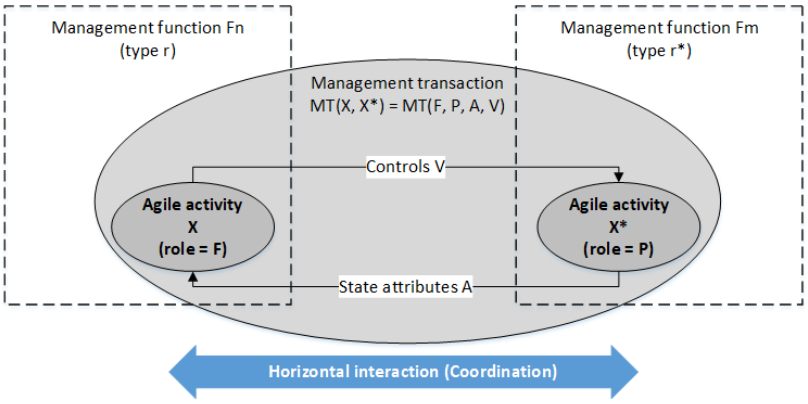



Source: [50]

Figure 33. Horizontal interaction (coordination) of Agile activities of different management function types

In the general case, horizontal interaction can be conceptualized as a management transaction MT(X, X*), where X and X* are activities of the same type (e.g. theme, initiative, epic, user story) also considered management transactions (Figure 34).

This assignment of Agile activity roles F and P inside MT(X, X*) is consistent with the strategic business and IT alignment model SAM, where certain activities are leading due to management decisions in a specific situation. Here X and X* are activities of the same type.




Source: [50]

Figure 34. Horizontal interaction conceptualized as a management transaction MT(X, X*)

In the general case, it is assumed that the main (leading) activity is X, so its role in the management transaction MT (X, X*) is F and the role of activity X* is P.

It should be noted that in turn, activities X and X* are also considered management transactions defined as follows: MT(X, Z) and MT(X*, W), where Z and W are the lower-level Agile activities (of a different type as X). Management transactions MT(X, Z) = MT(F", P", A", V") and MT(X*, Z) = MT(F", P", A", V") are composite, there are internal vertical interactions due to management control.

The content of horizontal interaction between the same type of the Agile activities: themes, initiatives, epics and user stories will be examined in more detail. Horizontal interaction (coordination) types are as follows:

a) one-way message transmission between activities of the same management function type (Figure 37);
b) one-way message transmission between activities of the different management function types (Figure 37);
c) horizontal interaction between activities is a feedback loop, i.e., is a management transaction (Figure 38);

The content of horizontal interaction (coordination) is specified using an example of the initiative (I) horizontal impact on another initiative (I*). A causal model of initiative (I) is defined as MT(I,E) = (F, P, A, V) and the other related initiative (I*) is defined as MT*(I*, E*) = (F*, P*, A*, V*). Possible content combinations of the coordination message (variants with different semantics) are listed in Table 17. „X" here denotes „applicable".

In cases (a) and (b) of horizontal interaction as a choreography the content of coordination flow C from initiative (I) could have an impact on the different elements (F*, P*, A*, V*) of coordinated initiative (I*). The instructions received in the message are information based on which the initiative (I*) makes its own decisions, and it is not obligatory to follow them.

In the case (C) of horizontal interaction as a management transaction the content variations of control flow V are the same, but the instructions are mandatory and must be followed.

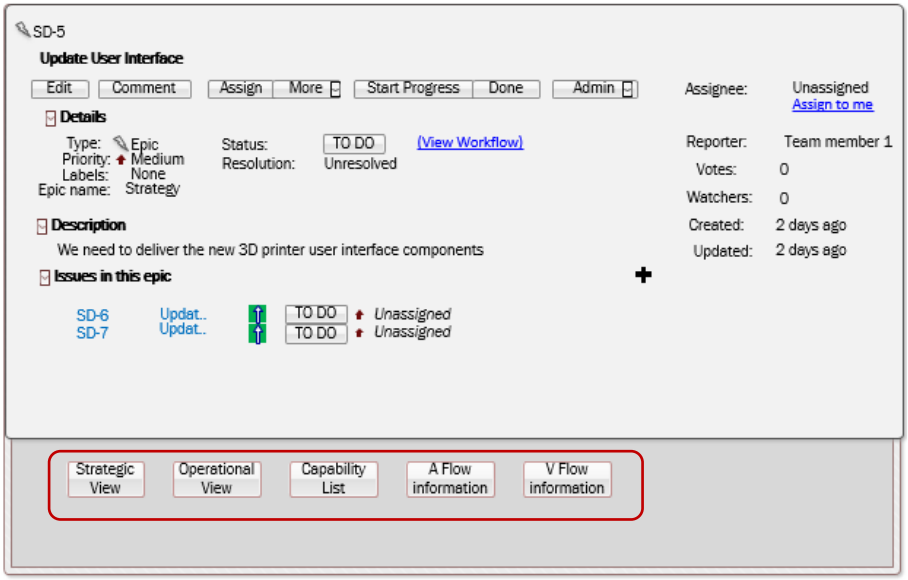
Table 17. Horizontal interaction content variants (C): by example of initiative (I) impact on the epic (E)

| **MT(I, E): flow C** | **MT*(I*, E*) = (F*, P*, A* V*)** | | | |
|---|---|---|---|---|
| | **F*** | **P*** | **A*** | **V*** |
| C1 | X | – | – | – |
| C2 | – | X | – | – |
| C3 | – | – | X | – |
| C4 | – | – | – | X |
| C5 | X | X | – | – |
| C6 | – | X | X | – |
| C7 | – | – | X | X |
| C8 | – | X | – | X |
| C9 | X | – | – | X |
| C10 | X | – | X | – |
| C11 | X | X | X | – |
| C12 | – | X | X | X |
| C13 | X | – | X | X |
| C14 | X | X | – | X |
| C15 | X | X | X | X |

Source: [50]

Based on the MT specification and the content of horizontal coordination cases (Table 17), the Jira screen that was presented in Figure 8 is updated with the new controls (Figure 40):

- The button “Strategic view” displays the MODAF model associated with the current epic and initiative that has the strategic objectives modelled as capabilities.
- The button “Operational view” displays the MODAF model that has the actors of the current initiative delivery (teams).
- The button “Capability list” displays a list of capabilities and their owners.
- “A flow” information presents a list of informational attributes based on the content in Table 17 (Figure 35)




Source: Created by the author

Figure 35. Jira epic enhanced view

Figure 36 Example of A flow content specification:

| Flow ID | MT component | Description | Value |
|---|---|---|---|
| C9 | A* | Specify the component | UX controls |
| | F* | Management function | Business development management |

Source: Created by the author

Figure 36. Flow content specification

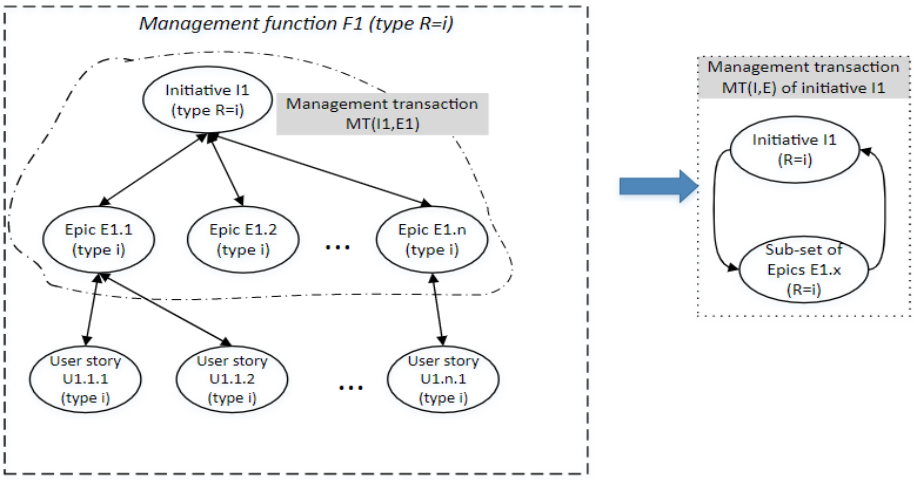
### 2.3.5. Content of Vertical Interactions

Vertical interaction of activities on any two adjacent levels in Agile hierarchy (Figure 41) is considered as Management Transaction MT = (F, P, A, V). Vertical interaction between activities is considered a self-managed activity MT, with an internal management control loop between activities in two levels of Agile hierarchy.

Therefore, each interaction of adjacent levels (theme, initiative), (initiative, epic), (epic, user story) is referred to as MT, where the role of the higher Agile hierarchy level activity is "management function F" and the role of the lower-level activity is "process P".

Each specific F is assigned a set of specific P's and a feedback loop is specified – flows A (state attributes of process P) and control instructions V (controls).

An example in Figure 37 represents identified MT(I,E) between initiative and epic levels. Here, “I1” stands for the id of the initiative, “E1” – for the id of the epic. Type R stands for the type of management function (i.e. software development management). R=i means that the management function is the same for several adjacent Agile activities. The related Agile activities – epics (E1.1, E1.2, etc.) represent a set of related epics to initiative I1. Accordingly, user stories (U1.1.1, U1.1.2, U1.n.1) represent a set of related user stories to initiative I1.




Source: [50]

Figure 37. Identification of MT(I, E) between initiative I1 and a set of epics E1.n

Vertical interaction of activities in Agile hierarchy (interaction between activities from different levels of hierarchy) is considered a management control process defined as management transaction MT(F, P, A, V).

Thus, the causal model of any Agile activity (theme, initiative, epic, user story) is defined as MT and includes internal elements as follows: management function (F), process (P) and their feedback loop interaction with information flows (A – state attributes, V – controls). For example, a causal model of theme is MT(Th,I) = MT(F, P, A, V) and a causal model of initiative (I) is MT(I,E) = MT(P*, F*, A*, V*).

Vertical interaction content variants are listed in Table 18 as an example of theme impact on the initiative. Variants of the vertical interaction content are defined in the same way as the rest of the adjacent Agile hierarchy levels: initiative – epic, epic – user story.

Table 18. Vertical interaction content variants (W): by example of theme impact on the initiative

| **MT(Th, I): management control W** | **Initiative I: MT(I,E) = (F*, P*, A* V*)** | | | |
|---|---|---|---|---|
| | **F*** | **P*** | **A*** | **V*** |
| W1 | X | – | – | – |
| W2 | – | X | – | – |
| W3 | – | – | X | – |
| W4 | – | – | – | X |
| W5 | X | X | – | – |
| W6 | – | X | X | – |
| W7 | – | – | X | X |
| W8 | – | X | – | X |
| W9 | X | – | – | X |
| W10 | X | – | X | – |
| W11 | X | X | X | – |
| W12 | – | X | X | X |
| W13 | X | – | X | X |
| W14 | X | X | – | X |
| W15 | X | X | X | X |

Source: [50]

Examples of the content of W1 – W15 impacts from theme to initiative: the content of W1 is the requirements to the state attributes A* of MT(I,E); the content of W2 is the requirements to the controls W* of MT(I, E); the content of W3 is the impact on process P* execution rules of MT(I, E); V4 includes requirements to management function F* (impact to management logic, rules) of MT(I, E); W5 – W15 are combinations of W1 – W4.

For example, content W1 in Table 18 indicates the impact of the epic state data, i.e., restrictions on the delivery date of projects. The impact W2 defines the impact initiative duration, i.e., the requirement that each initiative must be no longer than 9 months. The control flow W3 sets the business rules upon initiative management, i.e., the sequence of initiatives must be defined. The control flow W4 sets the requirements for the duration of the epics (3 months). The control flow W5 sets requirements for the content of both A* and V* flows of MT(I, E), i.e. forms an impact on the feedback between the initiative (managing function) and the epics that are controlled processes.

By further defining the management control flow V or state attributes A specifications, the Agile project management tool could be even further specified in determining missing information as the presented modified Agile management method allows the introduction of a specification of informational flows and checks the states of informational flows in a computerized way.

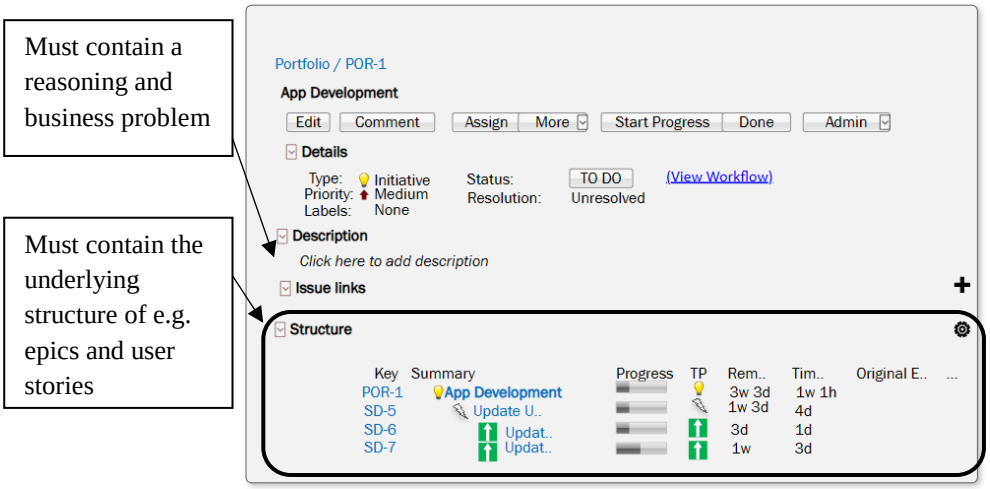
### 2.3.6. Practical Application in Jira

Agile activities must contain specific pieces of information, i.e. usually, epic or user stories contain as few details as it is possible about the context of the task. The context is described during refinements, and discussions in the Agile team, however, without proactive questioning, it is not presented why a specific piece of work needs to be done and the context is only kept in the head of product owner or some senior developer. This means that in order to analyse the data and compare it with a standard definition of the Agile activity (user story, epic, initiative, theme), the structure of describing the Agile activity should be in place. It is proposed to use these fields as mandatory:

1. As – is situation (including process model).
2. To – be situation (including process model).
3. User story format "As a "role", I want "behaviour of the system", so that "business problem" would be solved).
4. Link with Agile hierarchy items (vertical and horizontal).

It should be noted that on different levels of Agile hierarchy, the content and attributes of the structure change because from the bottom (user story) to top (theme) the level of abstraction increases. An example of an initiative description with additional requirements is presented in Figure 38.

This information specification (what needs to be provided) is a part of the organization project or business management knowledge, also known as governance policies. Organizations spend a lot of effort to ensure project delivery quality is assured, therefore, in order to utilize the knowledge obtained from previous projects each new project must adhere to these requirements. Some of the examples of poor EAS project delivery management include situations regarding the time dimension: some of the activities are done e.g. in the first month, but not all epic is completed, and then the remaining work is left to be completed after several months, resulting in a gap between delivery: changed environment and new changes to be implemented to address the situation that has changed in terms of technical and business environment.




Source: Created by the author

Figure 38. Jira Initiative description with additional mandatory fields

## 2.4. Model Verification and Validation

The MT framework is considered a proven method [34], [49], [138] for capturing domain causal knowledge. The composition of the MT framework is the basis for verifying the modified Agile hierarchy and validating its application in the enhanced Jira tool.

Model verification definition in [139] states that "model verification <is> the process of assuring that a computerized model is a sufficiently accurate representation of the corresponding conceptual model."

The interactions of levels between the modified Agile activities in Figure 21a) completely match the composition of the management transaction in Figure 2, since it is required that interaction must contain 4 obligatory attributes: flow A of state attributes, flow V of controls, identificator of process P and management function F. In this way, the modified Agile model is verified as it fully matches the causal knowledge model composition.

The validation determines whether a behavioural model is sufficiently accurate for Agile product owner needs in the context of tracing the content and integrity of themes, initiatives, epics, and user stories and their link to strategic business objectives.

Automatic validation of content and integrity of the EAS development project (state of project components themes, initiatives, epics, and user stories) is based on the normalised internal structure of the project components according to the MT framework. The status of the ongoing EAS development project can be evaluated automatically by comparing the existing specification

of the EAS project activities (existing in the project management system database) with the content composition of the corresponding activities required by the method when each activity is defined as MT(F, P, A, V). Both Table 17 and Table 18 serve as validation rules for the evaluation of the description of Agile activities and interactions in EAS project management. This provides a means for automatic validation.

As an example, the fragment of the database record with the identified gaps of vertical interactions between epic E01 and user stories S01 and S02 is presented in Table 19. Therefore, the modified Jira tool matches the modified Agile hierarchy in Figure 21 a).

Table 19. Fragment of the database record of vertical interaction evaluation

| **Theme** | **Initiative** | **Epic** | | | | | **User story** | **Validation** |
|---|---|---|---|---|---|---|---|---|
| | | **(ID)** | F | P | A | V | **(ID)** | |
| N | N | E01 Technical tasks (785) | N | Y | Y | Y | S01 Technical architecture review (800) | W11 |
| N | N | E01 Technical tasks (785) | Y | Y | Y | Y | S02 Segregate audit log and transaction log (903) | W15 |

Source: [103]

Here "Y" stands for existing information required by MT definition, "N" – for non-existing. Validation is automatically provided using the suggested method.

The results are also checked by the experts. The type of mismatching is identified using rules in Table 11. Advanced Jira tool automatically evaluates the information and presents a relevant error type. The model presented in this thesis was applied to 4 EAS case study projects, described in [134] and significant improvement in EAS project deliveries was observed.

## 2.5. Development Process State Evaluation

The semantics of vertical interaction (management control) and horizontal interaction (coordination) impact depends on the content of information flow transferred between the Agile activities.

The complete content of interaction between Agile activities must match the definition of MT = (F, P, A, V), meaning all the attributes: F, P, A, V must be specified and transferred.

Management control type (vertical interaction) in Table 18 is determined by the content of the Agile activity record in the Agile project management database and is reflected by the error message. For example, in epic E01 and user story S01 interaction (see Table 20) management control type is W11 as the P, A, V are specified, but the F is not specified. In the case of the interaction of epic E01 and user story S02, the management control type is W15 as all the attributes F, P, A, V are specified.

Based on the formal classification of vertical interaction (management control), see Table 18 and horizontal interaction (coordination) see Table 17 in section 2.3, it is possible to introduce the content evaluation indicator of EAS project complexity at the specific point in time, under assumptions:

Assumption 1: the upper level MT(F, P, A, V) sends a single flow V (impact, control message) to lower level MT*(F*, P*, A*, V*), representing the impact of MT to MT* (see Figure 24).

Under Assumption 1, the number of types of vertical interactions {W1, W2, ..., W15} (with different content) variants between Agile activities on different adjacent levels is 15 as listed in Table 18. Table 18 specifies the vertical interaction types – management control – between any adjacent Agile activities in Agile hierarchy. Every interaction type {W1, W2, ..., W15} (with different content) means a different information content transferred between the Agile activities. The vertical interaction types are the same for every level of Agile activities. Table 18 shows a specific example of vertical interaction content variants for theme impact on the initiative. Thus, there are four pairs of different level Agile activities and for all these pairs the Table 18 management control variants are applicable.

When combining variants in this way, the number of possible vertical interactions in Agile hierarchy with different semantics is 15 x 4 = 60.

Under Assumption 1, the number of horizontal interaction types on the same Agile hierarchy level is 15 as listed in Table 17. Table 17 specifies the horizontal interaction types – coordination – between any adjacent Agile activities in the same level of Agile hierarchy. Every interaction type {C1, C2, ..., C15} (with different content) means a different information content transferred between Agile activities. Table 17 shows a specific example of horizontal interaction content variants between initiatives. Coordination types in Table 17 are applicable for the same Agile activity type on all levels of Agile hierarchy (themes to themes, initiatives to initiatives, epics to epics, user stories to user stories), see Figure 33.

Basing themselves on the horizontal and vertical specifications described above, we evaluate the complexity of a typical EAS project.

We suggest annotating that the EAS project composition is as follows:

- Th – theme;
- I – initiative;
- E – epic;
- U – user story.

We suggest making some assumptions e.g. number 1 which are based on practical experience to illustrate the EAS project complexity calculation:

- assumption 1.1.1: one initiative I is only associated with only one theme Th;
- assumption 1.1.2: one epic E is only associated with one initiative I;
- assumption 1.1.3: one user story U is only associated with one epic E.

For a typical EAS development project, the average number of different Agile activity types are as follows:

- 1 theme (Th) which corresponds to some strategic objective;
- 1 initiative (I) per theme as the initiative is a more detailed specification of a theme and practice shows that usually one initiative is related to one theme in 1 EAS development project;
- 10 epics (E) per initiative;
- 24 user stories (U) per epic.

Several characteristic cases of vertical interfaces are relevant in Agile practice among the Agile activities:

1. On the level of themes:
   a) One theme Th is only associated with one initiative I;
   b) One theme Th is associated with many initiatives I;
   c) One theme Th is related to one management function (F);
   d) One theme Th is related to several management functions {F};
2. On the level of initiatives:
   e) One initiative I is only associated with one theme Th;
   f) One initiative I is associated with many themes Th;
   g) One initiative I is related to one management function (F);
   h) One initiative I is related to several management functions {F};
   i) One initiative I is related to one epic E;
3. On the level of epics:
   j) One epic E is only related to one initiative I;
   k) One epic E is only related to one management function (F);
   l) One epic E is related to several management functions {F};
4. On the level of user stories:
   m) One user story U is only related to one epic E;
   n) One user story U is only related to one management function (F);

Based on these characteristic cases, the EAS project complexity indicator calculations with examples are presented below. For a typical EAS development project, where there is only one management function r the number of vertical interactions (control messages between theme, initiative, epics and user stories) with different semantics is calculated (6):

$$Q1(v) = MT(Th, I) \times 15 + MT(I, E) \times 15 + MT(E, U) \times 15 \qquad (6)$$

The user story is considered to be the lowest level Agile hierarchy activity in this case.

Given the example of a typical EAS development project structure, the possible vertical interaction content variants expressed as complexity indicator Q1(v), where v stands for "vertical" are calculated as follows (7):

$$Q1(v) = 1 \text{ x } 15 + 10 \text{ x } 15 + 10 \text{ x } 240 \text{ x } 15 = 36165 \qquad (7)$$

That is:

- Number of relationships between theme and initiative interactions is 15, as there is only one theme–initiative relationship in the example;
- Number of relationships between initiative and epic interactions is 150, as there are 10 initiative–epic relationships in the example;
- Number of relationships between epics and user stories interactions is 3600, as there are 10 epics and 24 user stories relationships (each epic is related to 24 user stories) in the example.

To evaluate the complexity under each Agile hierarchy level between vertically adjacent activities (theme – initiative, initiative – epic, epic- user story), the alternative calculation could be done using the general formula (8):

$$Q1(v) = \{q1.1(Th, I);\ q1.2MT(I, E); q1.3MT(E, U)\} \qquad (8)$$

This illustrates the complexity of different Agile activity types (q1.1 = themes - initiatives, q1.2 – initiatives – epics, q1.3 – epics – user stories). This shows the possibilities to apply the generic formula (6) to evaluate the complexity of EAS project based on not only the whole Agile activity hierarchy but also adjacent Agile activity hierarchy levels (theme – initiative, initiative – epic, epic – user story) and shows that the biggest part of

complexity due to relatively larger size of coordination interactions is performed on the epic to user story level as follows (9):

$$Q1(v) = \{q.1.1 = 15; q1.2 = 150; q1.3 = 36000\} = 36165 \quad (9)$$

For a typical EAS development project, the number of horizontal interactions (coordination messages between theme and other themes, initiative and other initiatives, epic and other epics, user story and other user stories) with different semantics is calculated (10):

$$Q1(h) = MT(Th, Th') \times 15 + MT(I, I') \times 15 + \\ + MT(E, E') \times 15 + MT(U, U') \times 15 \quad (10)$$

User story is considered to be the lowest level Agile hierarchy activity in this case. Given the example of a typical EAS development project structure, the possible horizontal interaction content variants are calculated as follows (11):

$$Q1(h) = 10 \text{ x } 15 + 10 \text{ x } 24 \text{ x } 15 = 3750 \quad (11)$$

That is:

- Number of relationships between epics is 150, as there are 10 epics and each epic interacts with one another.
- Number of relationships between user stories is 360, as there are 24 user stories per epic (240 in total) and each user story interacts with one another only within that epic.

In this case, there is only one theme and one initiative, therefore, there are no horizontal interactions at these Agile hierarchy levels.

We suggest making some assumptions e.g. number 2 which are based on practical experience to illustrate the EAS project complexity calculation:

- assumption 1.2.1: one initiative I is only associated with one theme Th;
- assumption 1.2.2: one epic E is only associated with one initiative I;
- assumption 1.2.3: one user story U is associated with all epics E.

For a typical EAS development project, the average number of different activity types are as follows:

- 1 theme (Th) which corresponds to some strategic objective;

- 1 initiative (I) per theme as the initiative is a more detailed specification of a theme and practice shows that usually one initiative is related to one theme in 1 EAS development project;
- 10 epics (E) per initiative;
- 24 user stories (U) per epic.

For a typical EAS development project, the number of vertical interactions (control messages between theme, initiative, epics and user stories) for example number 2 is as follows (12):

$$Q1(v) = 1 \times 15 + 10 \times 15 + 10 \times 240 \times 15 = 36165 \quad (12)$$

That is:

- Number of relationships between theme and initiative interactions is 15, as there is only one theme–initiative relationship in the example;
- Number of relationships between initiative and epic interactions is 150, as there are 10 initiative–epic relationships in the example;
- Number of relationships between epics and user stories interactions is 36000, as there are 10 epics and 24 user stories relationships (240) in the example and all epics and user stories interact with one another.

In case one theme is related to several management functions, one theme can be related to several initiatives. We illustrate the EAS project complexity calculations based on the following assumptions which are based on practical experience:

- Assumption 1.3.1: one initiative I is only associated with one theme Th;
- Assumption 1.3.2: one epic E is associated with all initiatives I;
- Assumption 1.3.3: one user story U is associated with all epics E.

In practice, there has been a complex project where one theme was divided into 3 initiatives. The number of different activity types are as follows:

- 1 theme (Th) which corresponds to some strategic objective;
- 3 initiatives (I) per theme as the initiative is a more detailed specification of a theme and practice shows that usually one initiative is related to one theme in 1 EAS development project;
- 10 epics (E) per initiative (total of 30 epics);
- 24 user stories (U) per epic (total of 720 user stories).

In case one theme is related to several management functions, the number of vertical interactions (control messages between theme, initiative, epics and

user stories) with different semantics using formula (6) is calculated as follows (13):

$$Q1(v) = 3 \text{ x } 15 + 30 \text{ x } 15 + 10 \text{ x } 720 \text{ x } 15 = 108495 \quad (13)$$

That is:

- Number of relationships between theme and initiative interactions is 45, as there are 3 theme–initiative relationships in the example;
- Number of relationships between initiative and epic interactions is 450, as there are 30 initiative–epic relationships in the example;
- Number of relationships between epics and user stories interactions is 108000, as there are 30 epics and 24 user stories for each epic relationship (7200) in the example.

Now we examine horizontal interaction specification (coordination).

Assumption 2. Each internal element (F, P, A, V) of a coordinating MT (F, P, A, V) can send coordination effects to each element (F*, P*, A*, V*) of another MT* (F*, P*, A*, V*).

Assumption 2 applies to both horizontal and vertical interactions in the modified Agile hierarchy. Types of horizontal coordination are presented in Table 17.

Under Assumption 2, when one single element of MT – F, P, A, or V can send an impact to any element of other MT*(F*, P*, A*, V*), several combinations of (F, P, A, V) and (F*, P*, A*, V*) interactions are formed.

The number of horizontal and vertical interactions is calculated as combinations without repetition as in formula (14)

$$C_n^k = \frac{\mathrm{n!}}{\mathrm{k!\,(n-k)!}} \quad (14)$$

In the investigated case:

One MT element can send an impact C to one of the other MT* elements (F*, P*, A*, V*), number of variants of the content (listed in Table 17) is as follows (15):

$$C_4^1 = \frac{4!}{1!\,(4-1)!} + \frac{4!}{2!\,(4-2)!} + \frac{4!}{3!\,(4-3)!} + 1 = 15 \quad (15)$$

Thus, one MT element can send an impact C to any combination of other MT elements (F*, P*, A*, V*), and the number of combinations without repetition is 15.

In general, the total number of variants of the content in Table 17 (flow with different semantics) is 15 x 15 = 225 when two Agile activities are coordinated with each other (e.g. Figure 30). There are four levels in Agile hierarchy (theme, initiative, epic, user story). For a typical EAS development project, where there is only one management function r the number of horizontal interactions (coordination messages between theme, initiative, epics and user stories) with different semantics is calculated (16):

$$Q2(v) = (MT(Th, Th') \times 225) + (MT(I, I') \times 225) + \\ + (MT(E, E') \times 225) + (MT(U, U') \times 225) \quad (16)$$

For a typical EAS development project, the average number of different activity types are as follows:

- 1 theme (Th) which corresponds to some strategic objective;
- 1 initiative (I) per theme as the initiative is a more detailed specification of a theme and practice shows that usually one initiative is related to one theme in 1 EAS development project;
- 10 epics (E) per initiative;
- 24 user stories (U) per epic.

On this basis, the project indicator of complexity Q2 for a typical EAS development project in case of horizontal interactions (when each internal element (F, P, A,V) of a coordinating MT (F, P, A, V) can send coordination effects to each element of another MT* (F*,P*, A*, V*)) and there are 10 epics and 24 user stories under Assumption 2, can be calculated as follows (17):

$$Q2(h) = 10 \times 225 + 10 \times 24 \times 225 = 56250 \quad (17)$$

That is:

- Number of relationships between epics is 2250 as there are 10 epics, and each epic possibly sends 225 flows of different semantics.
- Number of relationships between user stories is 54000 as there are 24 user stories under each of 10 epics, and each user story possibly sends 225 flows of different semantics to each other.

In this case, there is only one theme and initiative and they do not have any horizontal interactions.

In the case of vertical interactions under Assumption 2, the project indicator of complexity Q2 can be calculated using formula (18) where for

every interaction of the Agile activities (MT(Th, I), MT (I, E), MT (E, U) ) there are 225 links in total.

$$Q2(v) = (MT(Th, I) \times 225) + (MT(I, E) \times 225) + + (MT(E, U) \times 225) \quad (18)$$

Thus, the EAS project vertical interactions complexity indicator can be calculated as follows (19):

$$Q2(v) = 1 \text{ x } 225 + 10 \text{ x } 225 + 10 \text{ x } 240 \text{ x } 225 = 542475 \quad (19)$$

To evaluate the complexity under each Agile hierarchy level between all activities in an EAS project vertically of all combinations (adjacent activities (theme – initiative, initiative – epic, epic- user story) and on the adjacent activities on the same Agile hierarchy level (theme – theme, initiative – initiative, epic – epic, user story – user story), the alternative calculation could be done using a general formula (20):

$$Q2(v) = \{q2.1(Th, I);\ q2.2MT(I, E);\ q2.3MT(E, U)\} \quad (20)$$

This illustrates the complexity of different Agile activity types (q1.1 = themes - initiatives, q1.2 – initiatives – epics, q1.3 – epics – user stories). This shows the possibilities to apply the generic formula (6) to evaluate the complexity of the EAS project based on not only the whole Agile activity hierarchy but also adjacent Agile activity hierarchy levels (theme – initiative, initiative – epic, epic – user story) and shows that the biggest part of complexity due to the relatively larger size of coordination interactions is performed on the epic to user story level.

Summarizing, the qualitative indicators Q1 and Q2 and their presented average and normalized values can help to evaluate the complexity and effort of communication and coordination in EAS project management in terms of required financing and other resources. Formulas (6) and (10) allow us to evaluate the complexity of the project of vertical and horizontal interactions and formulas (8) and (19) evaluate the relative complexity for adjacent Agile hierarchy levels for the whole EAS project. This data provides insights to EAS project experts (project managers, product owners, senior stakeholders) on the coordination effort required to manage the EAS project to ensure project alignment to strategic business objectives.

The detailed EAS project complexity indicators Q1 and Q2 calculations are presented in Table 20 and Table 21 as follows:

Table 20. Calculations of complexity indicators Q1 and Q2 for vertical interactions

| Formula | Theme | Initiative | Epic | User story | Interactions | | | Q1(v) average | Q2(v) average | Q1(v) normalized | Q2(v) normalized |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | Th-I | I-E | E-U | | | | |
| Q1(v) | th | i | e | u | th x i x 15 | i x e x 15 | e x u x 15 | - | - | Q1(v) / Q1(v) average | Q2(v) / Q2(v) average |
| Q2(v) | th | i | e | u | th x i x 225 | i x e x 225 | e x u x 225 | - | - | - | - |
| Typical project | 1 | 1 | 10 | 240 | 1 x 15<br>1 x 225 | 1 x 10 x 15<br>1 x 10 x 225 | 10 x 240 x 15<br>10 x 240 x 225 | 36165 | 542475 | 1 | 1 |
| Example project 1 | 1 | 1 | 20 | 480 | 1 x 15<br>1 x 225 | 1 x 20 x 15<br>1 x 20 x 225 | 20 x 480 x 15<br>20 x 480 x 225 | 144315 | 2164725 | 3,99 | 3,99 |
| Example project 2 | 1 | 2 | 40 | 960 | 1 x 15<br>1 x 225 | 2 x 20 x 15<br>2 x 20 x 225 | 40 x 960 x 15<br>40 x 960 x 225 | 576630 | 8649450 | 15,94 | 15,94 |

Source: [50]

Table 21. Calculations of complexity indicators Q1 and Q2 for horizontal interactions

| Formula | Theme | Initiative | Epic | User story | Interactions | | | Q1(h) average | Q2(h) average | Q1(h) normalized | Q2(h) normalized |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | Th-I | I-E | E-U | | | | |
| Q1(h) | th | i | e | u | th x 15 | i x 10 x 15 | e x 24 x 15 | - | - | Q1(h) / Q1(h) average | Q2(h) / Q2(h) average |
| Q2(h) | th | i | e | u | th x 225 | i x 10 x 225 | e x 24 x 225 | - | - | - | - |
| Typical project | 1 | 1 | 10 | 240 | - | 1 x 10 x 15<br>1 x 10 x 225 | 10 x 24 x 15<br>10 x 24 x 225 | 3750 | 56250 | 1 | 1 |
| Example project 1 | 1 | 1 | 20 | 480 | - | 1 x 20 x 15<br>1 x 20 x 225 | 20 x 48 x 15<br>20 x 48 x 225 | 14700 | 220500 | 3,92 | 3,92 |
| Example project 2 | 1 | 2 | 40 | 960 | - | 2 x 40 x 15<br>2 x 40 x 225 | 40 x 96 x 15<br>40 x 96 x 225 | 58230 | 873450 | 15,53 | 15,53 |

Source: [50]

The defined normalized and average complexity indicators Q1 and Q2 depend on the complexity of the projects that the organization is running. The experiment results demonstrated that a scale of 0 to 20 is typical of big organizations (running multiple projects simultaneously), however, there was a project that had a complexity indicator of over 2000. For smaller organizations, the average complexity indicators would be smaller.

## 2.6. Second Part Conclusions

The modelling of Agile activities showed the complex nature of EAS project management. The structure of MT proved to be a useful method to specify Agile activities content. A modified Agile activities hierarchy is developed considering the interaction between two adjacent Agile levels as a complex process with a feedback loop. The conceptual model of this interaction is defined as the management transaction (MT). MT is considered a self-managed activity, which specifies the causal link interaction between two adjacent levels in Agile hierarchy. Another feature of the modified Agile hierarchy is that the different types of control functions have been identified. Real-life examples provided of the vertical and horizontal interactions between themes, initiatives, epics, and user stories which are based on work experience with the Jira tool.

Vertical interaction of two Agile activities is considered a complex process with a feedback loop as well as horizontal interaction at the same level. The content variants for horizontal interactions (coordination) and vertical interactions (management control) are specified when each variant corresponds to a different combination of elements of the management transaction structure.

The content of interactions includes obligatory elements defined in MT structure: enterprise process (P), input flow A (process P state attributes), output flow V (process P controls) and management function (F). MT is related to some enterprise goal (G).

The taxonomy of coordination meta-types and coordination types is presented. The taxonomy is based on the conceptual structure of MT. It is used to evaluate the state of the project activities (themes, initiatives, epics, and user stories), and the content of causal interactions between them.

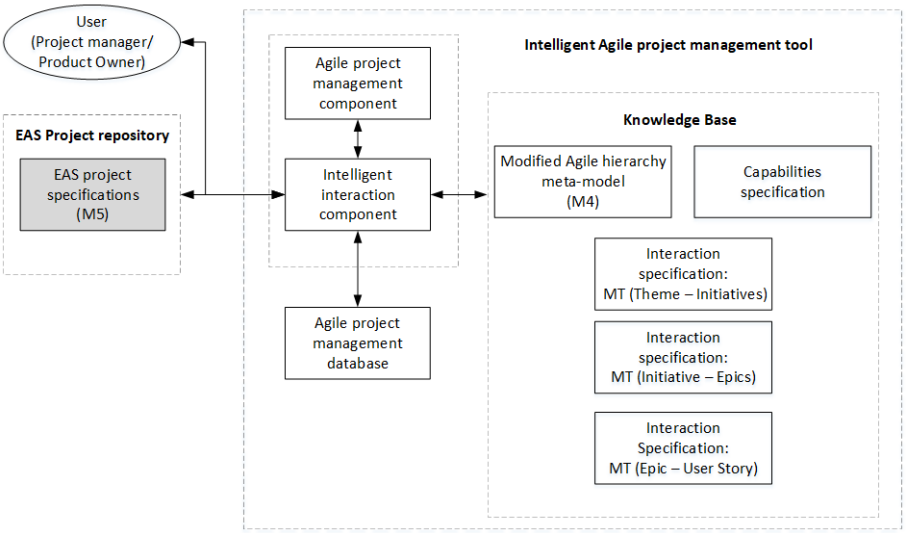
The content variants for horizontal and vertical causal interactions were listed and are used to evaluate the state of activities and their interactions in the EAS projects. The modified Agile hierarchy model verification and validation issues are argued.

The developed content variants show that there are over 3700 interaction content combinations with different semantics. The semantics of vertical and horizontal interaction impact depends on the content of information transferred between the Agile activities. The number of combinations increases with every new user story, epic, initiative, or theme introduced to the EAS project. The vast amount of interaction content combinations prove the complexity of enterprise management in the Agile environment. Therefore, the project management tools should include the proposed additional attributes that would allow monitoring project state more thoroughly.

EAS project complexity indicators were introduced that can help to evaluate the complexity and effort of communication and coordination in EAS project management in terms of required financing and other resources. The presented average and normalized complexity indicators help to evaluate the complexity of various EAS development projects and compare them.

# 3. ENHANCED AGILE MANAGEMENT TOOL ARCHITECTURE

To develop EAS intelligently and ensure that every task in the EAS project contributes to strategic business objectives using predefined knowledge is a must. By following the approach of a modified Agile hierarchy, the aim is to move away from the grey box approach to the white box approach of modelling domain knowledge (by presenting specifications to capture domain knowledge via the MT framework). The architecture of the enhanced Agile project management tool is presented in Figure 39.




Source: [106]

Figure 39. Enhanced Agile project management tool architecture

"User" in Figure 39 represents the managers for the Agile EAS project. They are interacting with the enhanced Agile project management tool (i.e. Jira or any other that would be enhanced by the AI component).

The "Intelligent Agile project management tool" is represented by nodes M6 and M7 in the conceptual model of EAS development management (Figure 14). The enhanced Agile project management tool includes components as follows:

- "Agile project management component" means the regular features that Agile management tools have, i.e., product backlog, sprint backlog, various attributes, reports, etc.

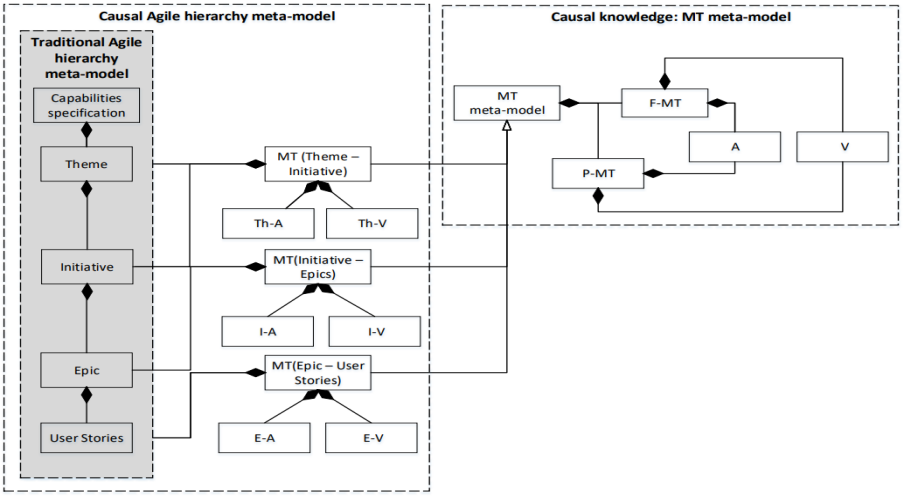
- “Intelligent interaction component” is the enhancement that is suggested to be added to the regular Agile project management tools. It uses real-time information from the Agile project management database (the information about TIES elements, like task descriptions, etc.). The „Intelligent interaction component“ interacts with the knowledge base (node M7 in Figure 14).

  “EAS project specifications” under the “EAS project repository” element represent node M2 from Figure 14, meaning it contains the observation-based business domain models (in BPMN, DMN, etc) and EAS project specifications (in UML, SysML, etc.) that are used for the development of EAS.

### 3.1. Agile Development Knowledge Base

“Knowledge base” (KB) is a repository of the causal knowledge frameworks (meta-models), used for validation of the EAS development solutions: modified Agile hierarchy meta-model, capabilities specifications, MT-based specifications of vertical interactions in Agile hierarchy and is represented by node M7 in Figure 14.

To realize the causal KB in the virtual environment, a conceptual knowledge base model (UML class model) was developed, which is shown in Figure 40.




Source: [106]

Figure 40. Conceptual model of the causal knowledge base (UML Class model).

The knowledge base consists of three interrelated parts: typical Agile hierarchy meta-model, causal knowledge component that stores the MT specification (meta-model) and meta-models of the causal interactions in the modified Agile hierarchy, namely meta-models of MT (theme, initiatives), MT (initiative, epics) and MT (epic, user stories).

### 3.2. Agile Project Management Data Structures

The causal Agile management repository specification (Figure 41) is developed following the conceptual model of the causal knowledge repository in Figure 40. The causal Agile management repository in Figure 41 consists of two parts: an enhanced project management database (Figure 41b) and a project state assessment knowledge base (Figure 41a).

The enhanced project management database (prototype is the Jira database) is supplemented with new DB tables CAPABILITY, BUSINESS_OWNER and new attributes required by the conceptual model in Figure 40. The table CAPABILITY contains identificator (ID_capability) and a description of the strategic goal item. To ease visualization, the CAPABILITY table is recreated twice. The table Agile_activity contains new attributes that ensure data storage of causal interactions: ID_MT – identificator of management transaction, ID_CAPABILITY – identificator of capability (i.e. strategic goal item), BUSINESS_OWNERS_ID – identificator of top-level manager. The table Sprint is not modified.

EAS "Project state assessment knowledge base" in Figure 41a is the implementation of the component "Causal Agile hierarchy meta-model" in Figure 45. The causal knowledge storage structure includes tables MT, A_FLOW_ATTRIBUTES and V_FLOW_ATTRIBUTES with functional dependencies between them. This data structure ensures the storage of causal interaction content as defined in the conceptual model of causal knowledge repository (Figure 45):

- Table MT ensures storage of MT = (F, P, A, V): identificator ID_MT, identificator of higher-level activity ID_activity (F) (F – management function) and identificator of lower-level activity ID_activity(P) (P – process);
- Table A_FLOW_ATTRIBUTES ensure the storage of MT = (F, P, A, V) flow A (information of process P state);
- Table V_FLOW_ATTRIBUTES ensure the storage of MT = (F, P, A, V) flow V (controls of process P state).

It is important to note, that "Project state assessment knowledge base" is linked to capability description in table CAPABILITY. This allowed tracing

EAS solution activities (theme, initiative, epics, user stories) against strategic objectives.

According to the causal modelling method, project state data on interactions between levels of Agile hierarchy are normalized by checking their compliance with the causal knowledge unit definition MT = (F, P, A, V). The storage of the received restructuring result is ensured by the specifications of tables MT, A_FLOW_ATTRIBUTES and V_FLOW_ATTRIBUTES, it can be seen in Figure 41a.

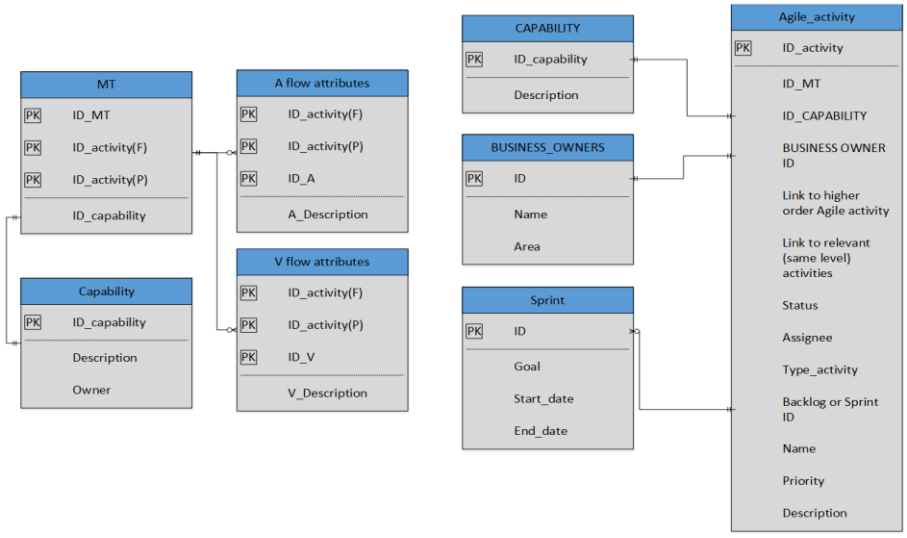
Thus, the specifications of the causal agile management knowledge repository comply with the definition of the conceptual model of the causal knowledge repository.

Figure 41b includes the implementation of the “Strategic capabilities” component of the “Causal knowledge repository” from Figure 43.

In summary, the EAS "Project Status Assessment Knowledge Bases" and "Improved Project Management Databases" specifications (Figure 29) meet the constraints defined by the conceptual models (Figure 39 and Figure 40).

Figure 41 shows how each causal Agile hierarchy interaction (MT(theme, initiative), MT(initiative, epic), MT(epic, user story)) should be defined and saves the actual state of the current EAS project state assessment against strategic enterprise objectives.

Records of the Agile activities in the project management database are associated with a causal model of Agile hierarchy and interactions through an additional set of attributes (Figure 41 (b), additional fields are capitalized).




Source: [106]

a) b)

Figure 41. Project management DB specification (knowledge base – a; an enhanced Agile project database – b)

It was chosen to present the fields with longer names for the sake of clarity. Furthermore, such usually encountered service information as date created, update, and user, that created or updated the record are not provided in each of the tables for the sake of simplicity.

Table 22 contains the records for EAS project requirements based on database structures in Figure 41. It indicates that there is missing theme and initiative information and epic information is missing link to theme (F).

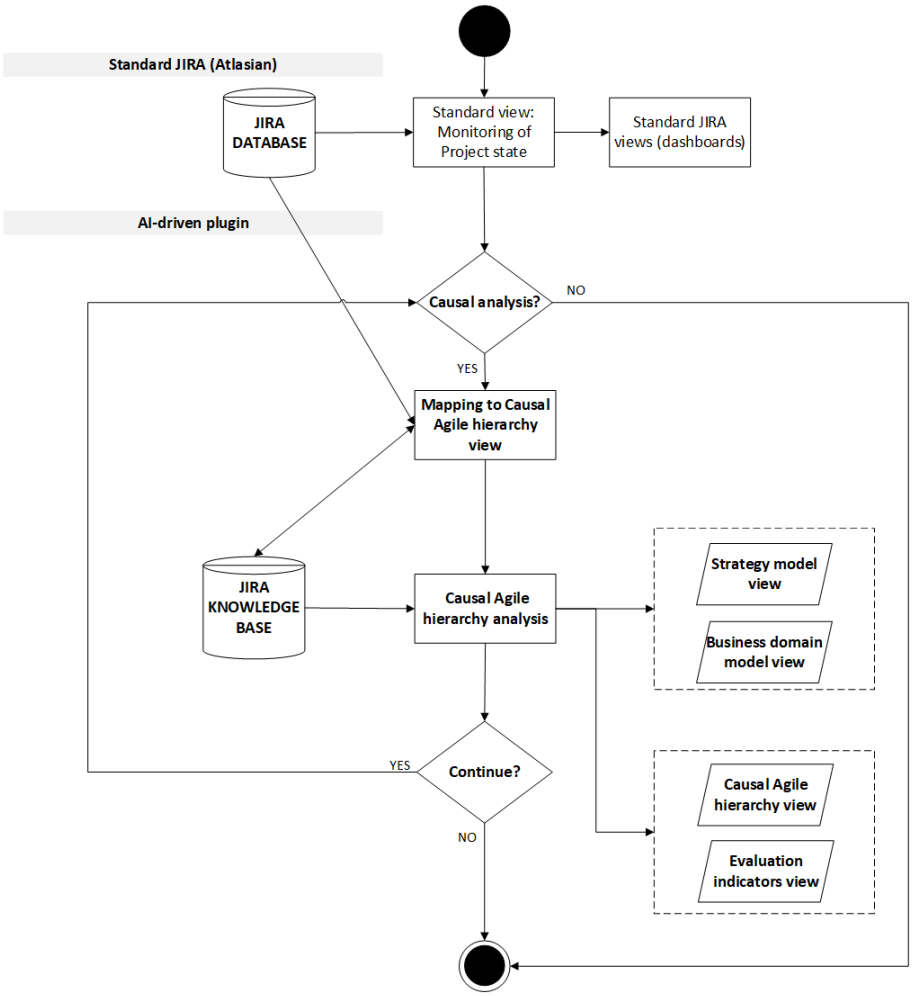
Table 22. Content of the database record of the enhanced JIRA tool

| **Theme** | **Initiative** | **Epic** | | | | | **User story** | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **ID** | **F** | **P** | **A** | **V** | **ID** | **F** | **P** | **A** | **V** |
| N | N | E01 Technical tasks (785) | N | Y | Y | Y | S01 Technical architecture review (800) | Y | Y | Y | Y |
| N | N | E01 Technical tasks (785) | Y | Y | Y | Y | S02 Segregate audit log and transaction log (803) | Y | Y | Y | Y |

Source: [106]

### 3.3. Enhanced User Interface

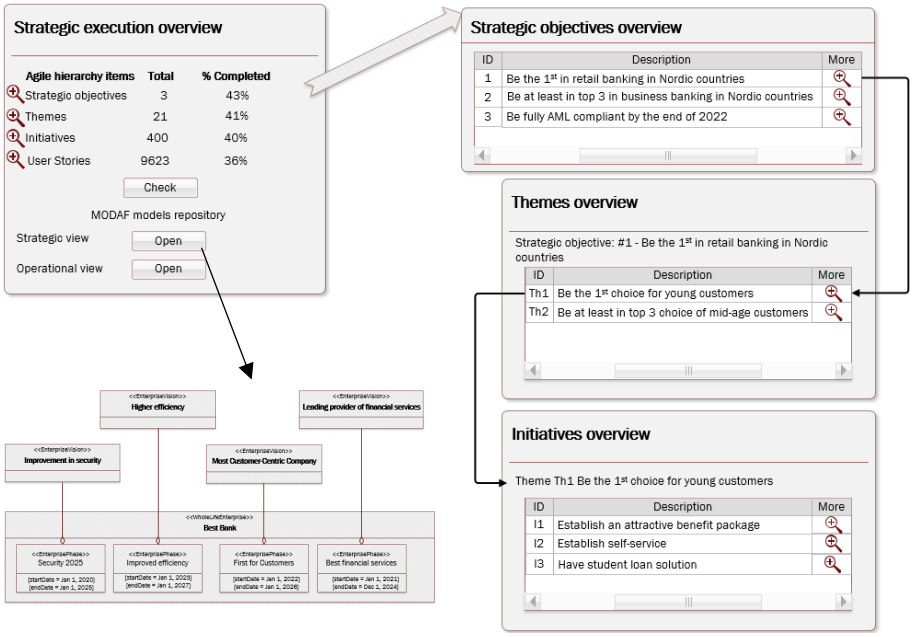
From the user perspective, the enhanced user interface contains the causal analysis option, which includes the aforementioned Jira knowledge base and causal Agile hierarchy analysis tools: a strategy model view and business domain model view that would represent the relevant knowledge, also, the causal Agile hierarchy view that contains MT specification and the evaluation indicators view. The enhanced Jira capabilities diagram is presented in Figure 46.




Source: Created by the author

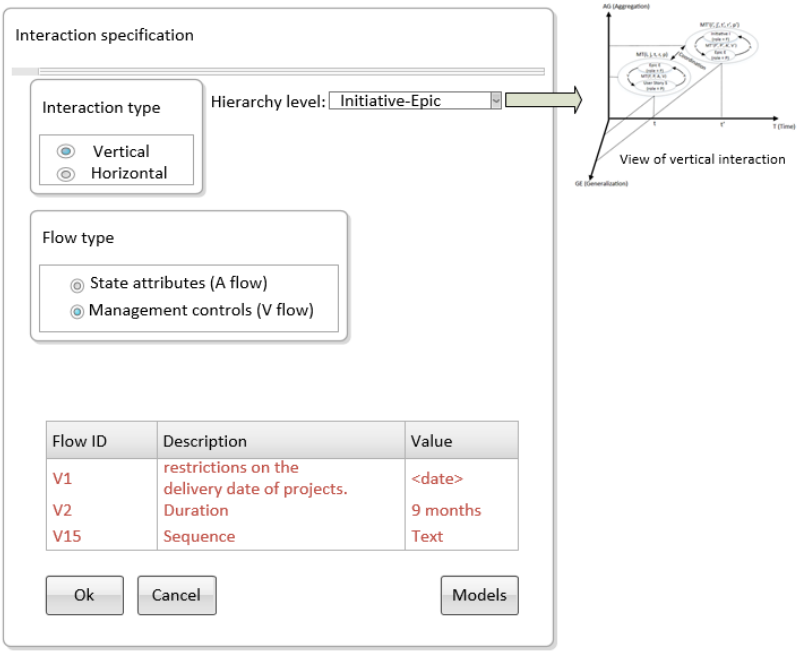
Figure 42. Enhanced Jira tool capabilities process diagram

The prototype of an intelligent interaction component is presented in Figures 43 to 46. This view is compiled using the content of the project management database record of the enhanced JIRA tool (Table 20). Table 20 contains the records for EAS project requirements based on database structures in Figure 41. It indicates that there is missing theme and initiative information and epic information is missing link to theme (F).



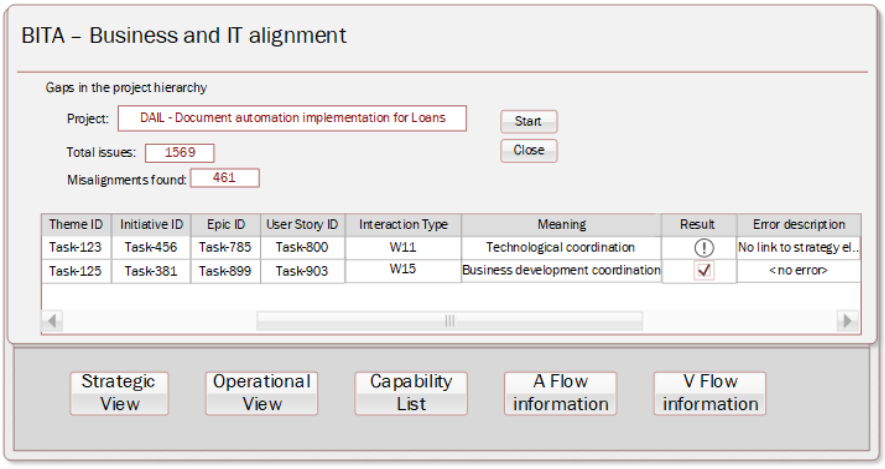

Source: Created by the author

Figure 43. Enhanced Jira interface dashboard for a manager



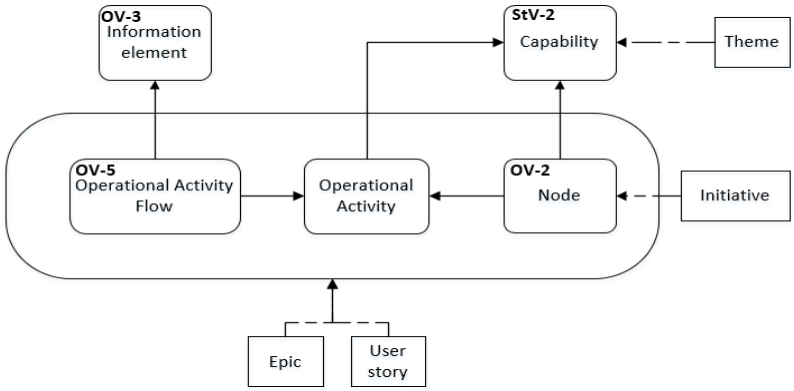

Source: Created by the author

Figure 44. Jira interaction specification input fields




Source: Created by the author

Figure 45. Prototype of the enhanced JIRA tool report for the product owner




Source: Created by the author

Figure 46. Strategic and operational view metamodel

Below is an example of the semantic information that the Jira tool would store (Jira database records content) and that is used to reflect the gaps of the project state against the modified Agile hierarchy meta-model:

- theme Th1(i+1, j, t, p, r) represents a long-term objective, i.e. “in 2 years to reduce operational cost by X amount”;
- one of the linked initiatives I1(i, j, t, p, r) - “Stop using system Y, and transfer the features of the system to new system Z”.;
- one of the linked epics E1(i-1, j, t, p, r) to the initiative: “in 3 months - transfer system Y function N to system Z”;
- one of the linked to the epic user stories U1(i-2, j, t, p, r): “in 2 weeks to transfer system Y function N component K to system Z”.

### 3.4. Third Part Conclusions

Current Agile software project management tools like Jira do not provide guidelines on how to define correct links between Agile activities on the different levels of the Agile management hierarchy. Given Jira would be able to capture the deep causal knowledge as suggested in this method, it would significantly increase the capabilities of monitoring the integrity of EAS project activities. This would ensure the alignment of each user story, epic, initiative, or theme to the strategic objectives and thus improve the success of business strategy execution.

Based on the results of the enhanced EAS project management tools report, the experts could provide the missing links and content of the TIES hierarchy activities. The report could also indicate that some of the EAS development project activities do not provide value to strategic business objectives thus they should be cancelled.

The presented conceptual model of causal EAS development management integrates causal knowledge into EAS design. Incorporating the causal models into the EAS development process and project management system enables the capability for validation of project content (i.e. specifications of user stories, epics, initiatives) using causal knowledge repository, including the alignment of strategic goals and EAS development solutions.

The basic component required for the intelligent Agile project management tool is the causal knowledge repository. A causal knowledge repository specification is proposed (based on the Jira tool). The repository consists of components as follows (required to store and verify causal knowledge): a traditional agile hierarchy related to capability specifications, and a causal agile hierarchy metamodel, including the specification of cross-level interactions. It also contains an Agile project management database of specific EAS project status that includes traditional Agile activity attributes and cause-based attributes.

This solution was verified by showing that the knowledge repository is built on a management transaction definition (causal unit of knowledge) that normalizes the internal structure of agile activities and interactions.

All this together allows for the creation of an intelligent project management system, which allows to evaluate the content of EAS development tasks (i.e. user stories, epics, initiatives) and align them with the requirements of strategic goals. The evaluation process should include procedures as follows:

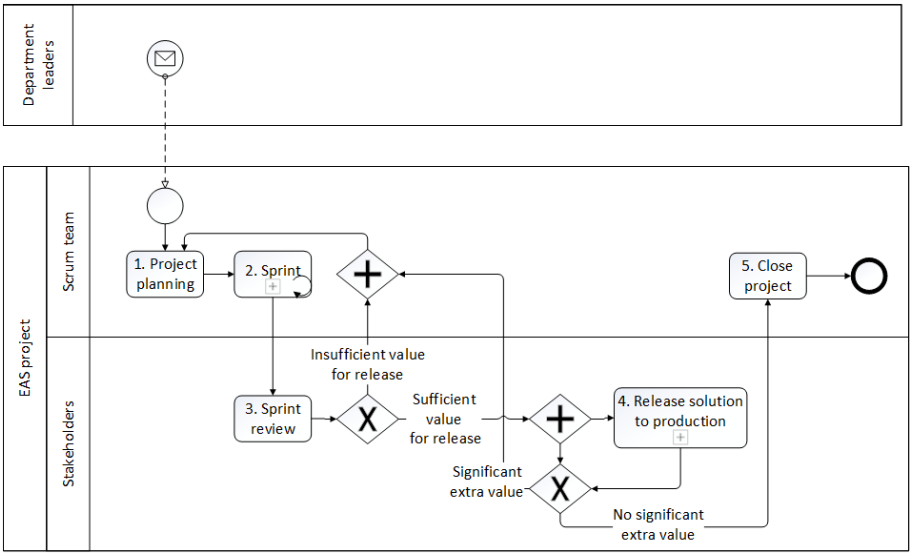
a) Tracing of enterprise strategic goals integrity with Agile hierarchy activities (theme, initiative, epic, user story) using the ID_capability attribute and information from KB tables AGILE_ACTIVITY and CAPABILITY (Figure 41);
b) Data compliance verification of the interaction content between Agile hierarchy layers (theme/initiative, initiative/epic, epic/user story) to the causal structure defined as MT framework, using the information from KB tables MT, A_FLOW_ATTRIBUTES and V_FLOW_ATTRIBUTES (Figure 41a).

The originality of this approach is the causal modelling solutions which ensure more successful project deliveries, supported by the enhanced Agile project management tool Jira having application development process evaluation capability. Project state information is transformed mapping to causal knowledge structure defined as MT. This makes it possible to identify the gaps of EAS as-is project data (stored in traditional Jira tool) against the required causal knowledge structure.

# 4. EXPERIMENTAL EVALUATION

The problem was researched using the inductive method. Delivering IT projects a pattern was observed between similar enterprise application software projects which tend to have requirements late in the project delivery lifecycle based on organizational strategy implementation (e.g. enterprise objective “Become first in customer experience”) which is translated into several priorities or capabilities (e.g. “Interactive sense of digital experience”). Based on the observations, a method was developed to identify the misalignment of information between information captured from organizational strategy to enterprise architecture framework and the information the development team has that works on executing the projects to utilize organizational capabilities and execute business strategy. A case study was done to evaluate the effectiveness of the method.

Delivering projects in a project delivery model based on the Scaled Agile Framework (SAF’e) [21] model, observations were made on four enterprise application software projects. The project specifics are that they are delivered in an Agile environment using Scrum framework on the team and enterprise levels – Scaled Agile Framework (SAF’e). The input for project requirements is gathered from four different Nordic countries working on one core process for each project. There are minor deviations in the detailed process of each country; the enterprise application software project must also address these deviations in the final version of the solution. Also, from an enterprise perspective, the dependencies between other teams working in the same area on their project deliveries must be addressed and aligned as part of SAF’e. The high-level project execution process for such projects is presented in Figure 47:




Source: Created by the author

Figure 47. High-level EAS project execution process

When the project is presented to the Scrum team, a series of activities occur:

1. Under "1 – Project planning" a series of requirements elicitation activities occur: as-is and to-be process mapping, user story writing, refinement, wireframing, etc. In this stage, there is heavy involvement both from the Scrum team and from the subject matter experts (SMEs). From this stage initial project requirements in the format of user stories (tasks) are developed.
2. Under "2 – Sprint" a typical Scrum process is performed – software development, testing, requirement clarification and future requirements analysis. At this stage "Change requests" are presented and "Bugs" are discovered.
3. Activity "3 – Sprint review" is specifically excluded from sprint activity as it involves stakeholders and for clarity, it is based on stakeholders' (managers, product owners) decisions on whether the developed software is released to end-users or additional development iteration is performed. At this stage "Change requests" or "Bugs" might also appear.
4. The "4 – Release solution to production" is an activity where depending on enterprise internal processes the software that was newly developed by the Scrum team is merged into the existing code-base. In rare cases at this stage, additional bugs might appear. After the

software is put into production the additional requirements might appear in the form of “Change requests”.

5. “Close project” activity is performed when the stakeholders determine that there is no significant value in conducting new iteration(s) of development. During this activity typically project documentation with technical details and user manuals are finalized and the project is closed depending on the internal processes of the enterprise. The left-over requirements (epics, user stories/tasks, bugs, etc.) are either kept for “version 2” of the project or archived for any reference.

The requirements in the format of themes, initiatives, epics and user stories, change requests and bugs were analysed from a project lifecycle of 8 to 60 months:

“Requirements” here are the requirements that were developed during activity “1 – Project planning” as in Figure 47. They are of the TIES hierarchy levels theme, initiative, epic or user story.

The results are displayed in Table 23 below.

Table 23. Enterprise application software projects initial requirements distribution

| **Parameter** | **Project #1** | **Project #2** | **Project #3** | **Project #4** |
|---|---|---|---|---|
| Requirements out of which: | 138 | 224 | 236 | 2539 |
| Strategic objective | Not defined | Not defined | Not defined | Not defined |
| -themes | 0 (not defined) | 0 (not defined) | 0 (not defined) | 4 |
| -initiatives | 1 | 2 | 2 | 10 |
| -epics | 9 | 15 | 17 | 314 |
| -user stories | 128 | 207 | 217 | 2211 |
| Change requests | 273 | 173 | 36 | 1341 |
| Bugs | 135 | 252 | 304 | 824 |
| Project duration | 8 months | 12 months | 10 months | 60 months |

Source: Created by the author

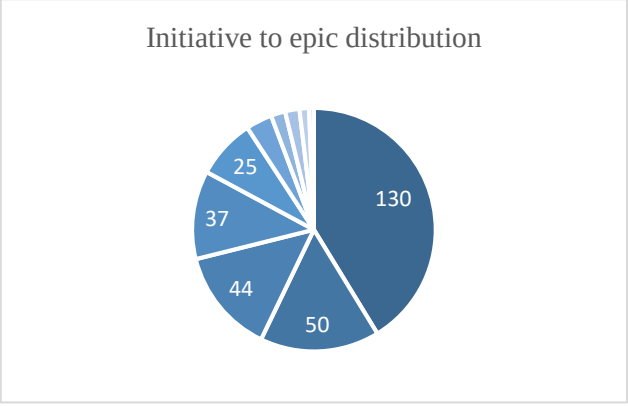
We examine Project 4 in more detail. In the case of project #4 the TIES composition goes as follows:

Table 24. EAS project theme to initiative composition example

| Theme | Initiative |
|---|---|
| Th-253<br>UAM Workflows | 1. I-241 – User and role management for administrators.<br>2. I-841 – Role assignment to existing users.<br>3. I-451 – Basic information provisioning. |
| Th-6377<br>UAM and SSO | 1. I-9426 Basic information provisioning implementation.<br>2. I-3996 UAM solution implementation.<br>3. I-4996 UAM implementation with security schema. |
| Th-4586<br>System improvements | 1. I-7476 Integration with legacy system. |
| Th-4286<br>Enterprise UAM implementation | 1. I-241 – User and role management for administrators' implementation.<br>2. I-3462 – Component integration PoC.<br>3. I-4462 – Role assignment to existing users' implementation. |

Source: Created by the author.

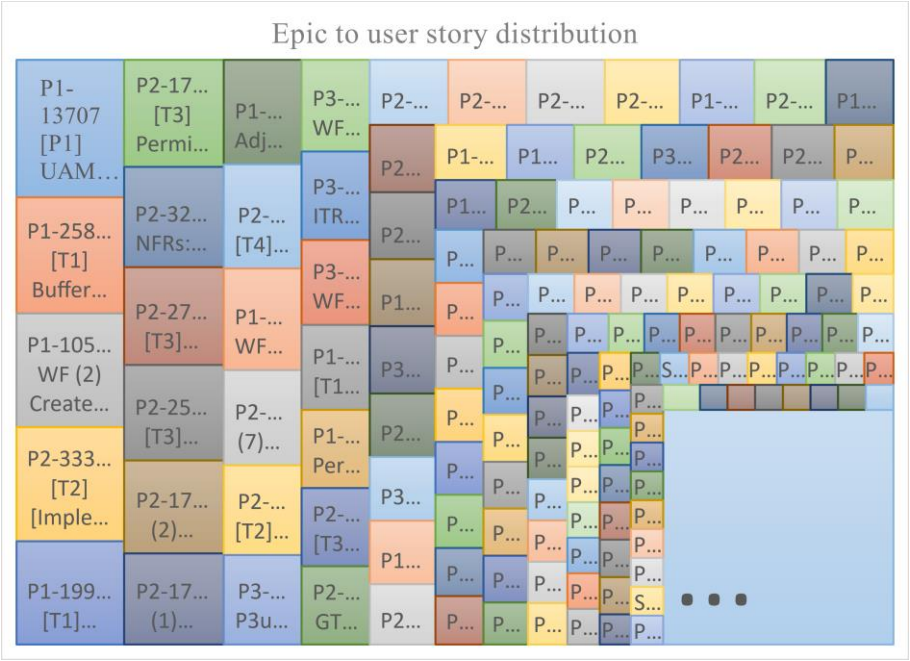
This situation matches assumptions 1.3.1 – 1.3.3 as described in Section 2.5. Further analysing the project composition on the initiatives and epics level the distribution of epics under each of the initiatives mentioned in Table 24 is presented in Figure 48.




Source: Created by the author

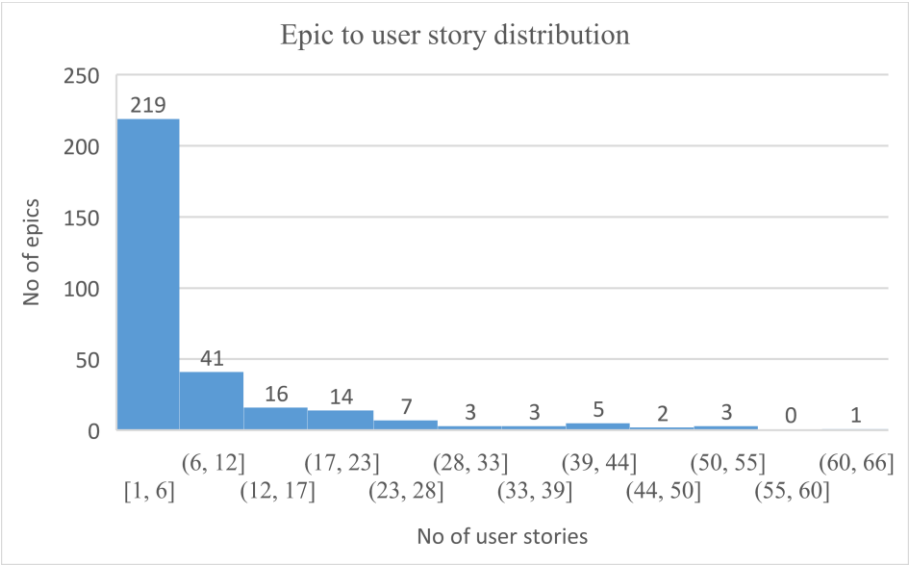
Figure 48. Epics distribution under initiatives

On the epics and user stories level the distribution of user stories under each epic is presented in Figures 49 and 50.




Source: Created by the author

Figure 49. User stories distribution under epics treemap chart



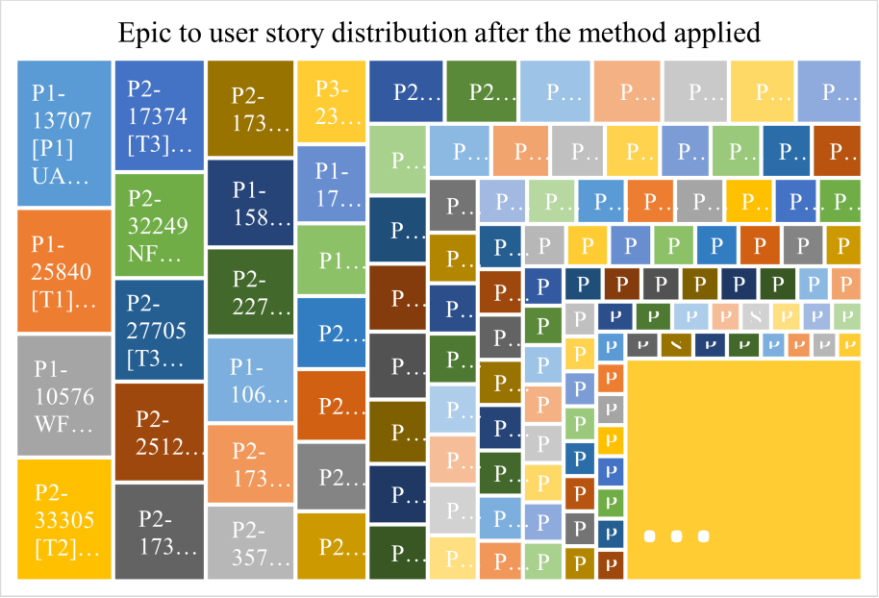

Source: Created by the author

Figure 50. User stories distribution under epics histogram

It could be noticed that the median of user stories under a single epic here is 3, which indicates disproportional distribution, placing 56% of all user

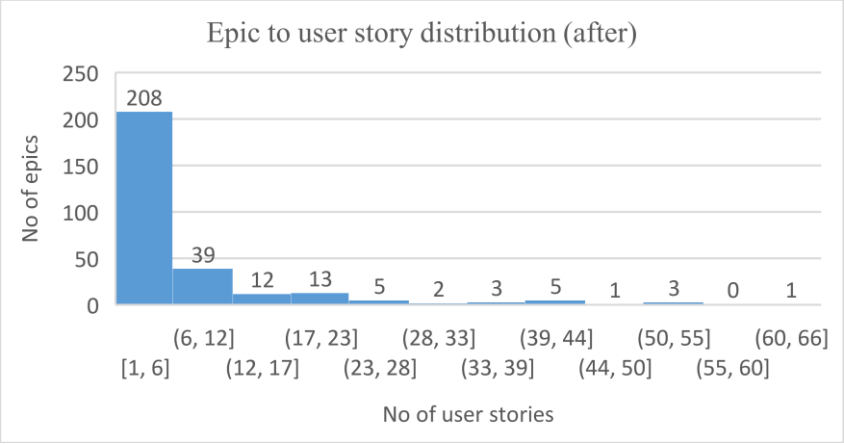
stories under epics that contain three or fewer user stories as the average number of user stories under an epic is 7. Furthermore, there are 17 epics where the number of user stories is 30 or more.

With the method applied, the distribution of epics to user stories looks like this (Figures 51, 52), indicating a reduction in epics with 3 or fewer user stories:




Source: Created by the author

Figure 51. User stories distribution under epics treemap chart (after the method applied)




Source: Created by the author

Figure 52. Epic to user story distribution histogram (after the method applied)

Table 18 in Subsection 2.3.5 contains the informational flow specification between Agile hierarchy activities in Table 24. Using Table 18, the informational flows specification completion indicates the situation of the information completeness of Agile hierarchy activities using RAG indicator, providing information about critical situations in the Agile activity hierarchy specification (Table 25) that require attention from EAS project managers to ensure that missing data is provided. E.g. if only epic is created that does not contain any user stories (as in flow W1) it would indicate a critical situation where attention is needed immediately (to ensure epic E contains at least one link to the user story U). In the case of W11, the epic and user story is present and also the state attributes are transferred between the user story and the epic, only management control information is missing, which is still considered a satisfactory situation.

Table 25. Interaction specification of RAG status for EAS project managers

| **MT(I, E): management control W** | **Epic E: MT(E,U) = (F*, P*, A* V*)** | | | | **RAG status** |
|---|---|---|---|---|---|
| | **F*** | **P*** | **A*** | **V*** | |
| W1 | X | – | – | – | R |
| W2 | – | X | – | – | R |
| W3 | – | – | X | – | R |
| W4 | – | – | – | X | R |
| W5 | X | X | – | – | A |
| W6 | – | X | X | – | R |
| W7 | – | – | X | X | R |
| W8 | – | X | – | X | R |
| W9 | X | – | – | X | R |
| W10 | X | – | X | – | R |
| W11 | X | X | X | – | G |
| W12 | – | X | X | X | A |
| W13 | X | – | X | X | R |
| W14 | X | X | – | X | G |
| W15 | X | X | X | X | G |

Source: Created by the author

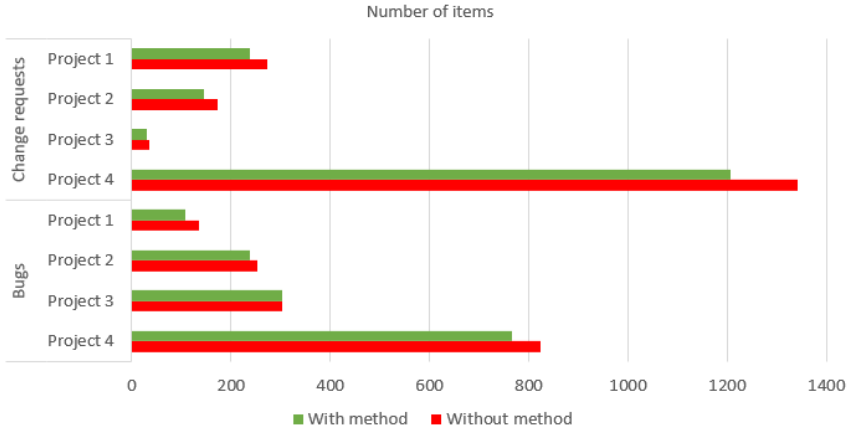
When using the suggested method to minimize the information gaps in business strategy execution and EAS development project delivery the data was analysed and the findings are displayed in Table 26. Change request here means – changes to user story content. Typically, this occurs when there is a misalignment between the user story delivery result (how the system works) and business needs, and a change request is created to adjust the system behaviour to a desired state. Bugs here are invalid system behaviour resulting in a system error.

Table 26. Enterprise application software projects requirements distribution when using the suggested method

| Parameter | Project #1 | Project #2 | Project #3 | Project #4 |
|---|---|---|---|---|
| Requirements out of which: | 121 | 181 | 216 | 2244 |
| Strategic objective | Become 1st choice for personal banking in the Nordics | Become 2nd by market share in business banking in the Nordics | Become 2nd by market share in business banking in the Nordics | Ensure state of art self-service experience for customers |
| -themes | 1 | 1 | 1 | 4 |
| -initiatives | 1 | 2 | 2 | 10 |
| -epics | 8 | 13 | 15 | 292 |
| -user stories | 111 | 165 | 198 | 1938 |
| Change requests | 238 | 146 | 36 | 1208 |
| Bugs | 107 | 238 | 304 | 765 |
| Project duration | 8 months | 11 months | 10 months | 54 months |

Source: Created by the author

The set of requirements in the TIES hierarchy did change as there were fewer epics and user stories developed to complete the EAS projects. Also, in change request and bugs categories, the differences are quite significant as in the first project the number of bugs was reduced by more than 20 % and in project 3 the number of change requests was reduced by more than 16%. The comparison results are displayed in Figure 53 and Table 27.




Source: Created by the author

Figure 53. Experimental evaluation results

Table 27. Enterprise application software projects requirement distribution comparison

| Parameter, % | Project #1 | Project #2 | Project #3 | Project #4 |
|---|---|---|---|---|
| Requirements | -12,3 | -19,2 | -8,5 | -11,6 |
| Change requests | -12,8 | -15,6 | -16,7 | -9,9 |
| Bugs | -20,7 | -5,6 | 0 | -7,2 |
| Project duration | 0 months | -1 months | 0 months | -6 months |

Source: Created by the author

The savings observed should be evaluated based on their required development effort and based on the average hourly cost of the developer, the cost savings can be calculated.

Furthermore, given the normalized (with the method applied) EAS project requirements distribution, the project complexity indicators as defined in Section 2.5 can be calculated as follows:

Table 28. Calculations of enterprise application software projects complexity

| Parameter, % | Project #1 | Project #2 | Project #3 | Project #4 |
|---|---|---|---|---|
| -themes | 1 | 1 | 1 | 4 |
| -initiatives | 1 | 2 | 2 | 10 |
| -epics | 8 | 13 | 15 | 292 |
| -user stories | 111 | 165 | 198 | 1938 |
| **Complexity indicators** | | | | |
| Q1 (v) | 0,37 | 0,9 | 1,24 | 234,84 |
| Q2 (v) | 0,37 | 0,9 | 1,24 | 234,84 |
| Q1 (h) | 3,58 | 8,63 | 11,95 | 2264,79 |
| Q2 (h) | 3,58 | 8,63 | 11,95 | 2264,79 |

As the complexity indicators calculations revealed, the complexity dramatically increases the more there are epics and user stories in EAS project hierarchy. This directly shows the project coordination resources (project managers, product owners) required to manage a project of such complexity.

## 4.1. Fourth Part Conclusions

The results of the experimental evaluation revealed that using the modified Agile project management method in EAS project development reduces the amount of coordination effort needed to ensure EAS development activities' alignment with strategic business objectives. Furthermore, the volume of TIES hierarchy elements is reduced by 13% on average. Also, the complexity indicators show the demand for EAS coordination – the organization can determine what coordination effort will be required to help ensure timely EAS project delivery, and the experimental evaluation revealed that the complexity indicator range is from 0,1 to over 2200 which is reflected in the amount of Agile activities and time spent delivering an EAS project. With such knowledge, the organization can make timely decisions about whether to continue a project that takes so much development (and coordination) resources to deliver.

# CONCLUSIONS

1. Agile software development project management methods and corresponding Agile tools based on those methods (Scrum, Kanban, etc.) that are used in practice have significant shortcomings in ensuring that the EAS project results (the software developed) are effective and aligned with the current business strategy being implemented, such as:
   1.1. a typical Agile process is very little formalized (by design) and this is one of the reasons why Agile tools are limited to capturing and storing the little structured information about the Agile activities.
   1.2. different enterprise management function types that are involved in EAS development (e.g. business management and software development management) are not identified and specified in Agile methods for EAS development;
   1.3. the project execution control, alignment to strategic business objectives and Agile activities coordination rely solely on experts working on a project which is a complicated and error-prone process.
   1.4. in large complex projects it is time-consuming and requires additional manual work to understand the progress of overall EAS development project execution as project leaders need to analyse long lists of records with many attributes and this is done regularly throughout the EAS project lifecycle.
2. Deficiency of the current Agile tools:
   2.1. current Agile project management tools do not contain automated project state analytics capability, because the current Agile methods are not sufficiently formalized (by design):
   a) no formally defined (not standardized) internal structure specifications of the Agile activities;
   b) formally undefined (non-standardized) vertical and horizontal specifications of mutual interactions (informational content) of Agile activities;
   c) the current Agile methods do not evaluate enterprise strategies (they are not modelled, and therefore not integrated with the Agile process)
   2.2. Despite high customization possibilities, Agile tools do not have a template for specifying Agile activities descriptions, i.e. to standardize the specifications of activities, this limits the possibility of automatic analysis of the project status; in addition Agile tools refer only to the fact of the relationship between different levels of

Agile hierarchy (theme-initiative, initiative-epic, epic-user story). Such unstructured information makes it difficult to algorithmize the analysis and evaluation of the status of the EAS project, especially the analysis of compliance with the business strategy because the content of interaction between the Agile activities is not described.

2.3. in the current Agile tools environment, there is no business domain model (knowledge model) that conveys mandatory business process relationships and attributes from business strategy to developers working on EAS project, resulting in not containing sufficient detailization of linking EAS development solutions to business strategy execution (from the SAM perspective) and Agile project management activities.

3. A causal modelling approach to the EAS project management in an Agile environment was used to rethink the typical Agile activities hierarchy (themes, initiatives, epics, and user stories). It was found that:
   3.1. the strategic business objectives and different enterprise management function types are not identified and specified in EAS development projects, therefore, the quality of the project execution and alignment to strategic business objectives suffers;
   3.2. there is a causal link between any two adjacent levels of activities of Agile hierarchy when data and information are transferred between activities;
   3.3. interaction of two activities from adjacent levels of Agile hierarchy is a complex process with a feedback loop (there is circular causality) where the mutual interaction between Agile activities is controlled by an expert (manager, project manager, product owner or developer).
4. A causal model of Agile project management was developed using a management transaction (MT) concept and capability concept from the enterprise architecture framework MODAF. The main features of the developed modified Agile method are as follows:
   4.1. Each activity in the modified Agile hierarchy is considered MT. The content of MT includes obligatory elements defined in MT structure: enterprise process (P), input flow A (process P state attributes), output flow V (process P controls) and management function (F). MT is related to a particular enterprise goal (G).
   4.2. Each interaction between two adjacent Agile activities is considered as a complex process with a feedback loop. The conceptual model of this interaction is defined as the MT. MT is considered a self-

managed activity, which specifies the causal link interaction between two adjacent levels in Agile hierarchy.

4.3. The different types of management control and coordination functions have been identified.

4.4. The unification of the structure of Agile activity interactions in the modified Agile hierarchy is a basis for formal analysis of the content of information transferred and the definition of the types of vertical interactions and horizontal interactions.

5. MODAF enterprise architecture framework was used to model and analyse business objectives, consistently identifying the strategically required target functional characteristics, which are called „capabilities“. This allows strategical business objectives integration with the Agile activity hierarchy when identified capabilities are mapped to top-level Agile activities “themes”.
6. The taxonomy of coordination types and management control types in the modified Agile hierarchy is presented (see Tables 15 and 16) and illustrated with real-life examples. The vertical and horizontal interaction specifications between themes, initiatives, epics, and user stories are provided which are based on work experience with the Jira tool.
7. The architecture of the project management tool (based on the modified Agile process model) is specified, including the specification of the knowledge base. Mapping of typical JIRA tool content given the enhanced Jira knowledge base (Figure 45 shows the way of normalization of the project content – descriptions of the Agile activities theme, initiative, epic, user story) provides background for evaluation of the state of the project.
8. The content variants for horizontal and vertical causal interactions were listed and are used to evaluate the state of activities and their interactions in the EAS projects:

8.1. The developed content variants determine that there are over 3700 interaction content combinations with different semantics. The semantics of vertical and horizontal interaction impact depends on the content of information transferred between the Agile activities. The number of combinations increases with every new user story, epic, initiative, or theme introduced to the EAS project. The vast amount of interaction content combinations prove the complexity of enterprise management in the Agile environment. Therefore, the project management tools should include the proposed additional attributes that would allow monitoring project state more thoroughly.

8.2. EAS project complexity indicators were introduced that can help to evaluate the complexity and the required effort of communication and coordination in EAS project management in terms of financing and other resources. The presented average and normalized complexity indicators help to evaluate the complexity of various EAS development projects and compare them.

9. Based on the results of the enhanced EAS project management tools report, the experts working on EAS projects (program managers, project managers, product managers, product owners or developers) could provide the missing links and content of the TIES hierarchy activities. The report could also indicate that some of the EAS development project activities do not provide added value to strategic business objectives thus they should be cancelled.
10. As the results of the experimental evaluation show, the proposed modified causal Agile method to ensure business and IT alignment in an Agile environment to link the requirements of EAS projects to strategic goals of the enterprise, provides significant improvement to EAS project deliveries by reducing the number of requirements, change requests and bugs throughout the EAS project development lifecycle by almost 13%. IT solutions are improved once missing information is added to project specifications, that enable the link of those EAS system requirements to strategic goals of the enterprise.
11. Despite significant advantages, the limitation of applying this method is the requirement to have the organization at a particular process capability maturity level with the business processes defined, i.e. at least at level 3, according to the CMMI process evaluation model [120].

# BIBLIOGRAPHY AND REFERENCES


1. Lee, E.A.: The Past, Present and Future of Cyber-Physical Systems: A Focus on Models, Sensors, 15(3) (2015), 4837-4869.
2. Jarrahi, M.H.: Artificial Intelligence and the Future of Work: Human-AI Symbiosis in Organizational Decision Making Business Horizons 61(4) (2018), 577-586.
3. Tang, X., Li, X., Ding, Y., Song, M., Bu, Y.: The pace of artificial intelligence innovations: Speed, talent, and trial-and-error. Journal of Informetrics, 14(4) (2020), 101094.
4. Conant, R.C, Ashby, R.W.: Every good regulator of a system must be a model of that system. International journal of systems science 1 (1970), 89-97.
5. Francis, B.A., Wonham, W.M.: The internal model principle of control theory Automatica 12(5) (1976), 457-465.
6. Zack, M.H.: If Managing Knowledge is the Solution, then What's the Problem? In: Yogesh Malhotra (eds.) Knowledge Management and Business Model Innovation, Idea Group Publishing (2001). DOI: 10.4018/978-1-878289-98-8.ch002.
7. Grieves, M.: Digital twin: Manufacturing Excellence Through Virtual Factory Replication. Florida Institute of Technology (2014).
8. Whiteley, A., Pollack, J., Matous, P.: The Origins of Agile and Iterative Methods. Journal of Modern Project Management, 8(3), 20-29. 2021. DOI: 10.19255/JMPM02502
9. Schwaber, K.: SCRUM Development Process. Object-Oriented Programming, Systems, Languages & Applications (OOPSLA), USA, 1995.
10. Schwaber, K., Sutherland, J.: The Scrum Guide. https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf#zoom=100 2020.
11. Stapleton, J.: DSDM: Dynamic Systems Development Method: The Method in Practice, 1st edn. Addison-Wesley, 1997.
12. Cockburn, A.: Surviving Object-Oriented Projects, 1st edn. Addison-Wesley, USA (1998)
13. Cockburn, A.: Crystal Clear: A Human-Powered Methodology for Small Teams, 1st edn. Addison-Wesley Professional, USA (2004)
14. Adaptive Software Development: A Collaborative Approach to Managing Complex Systems, 1st Edition, Dorset House, USA (1999)
15. Beck, K., 1999. Extreme Programming Explained: Embrace Change. Upper Saddle River, NJ 07458: Addison-Wesley Professional.
16. Ohno, T.: Toyota Production System: Beyond Large-Scale Production 1st Edition, USA Productivity Press, 1988
17. Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., W., Cunningham, Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R., Mellor, S., Schwaber, K., Sutherland, J., Thomas,

D.: Manifesto for Agile Software Development, http://agilemanifesto.org/, 2001. Last accessed 2023/06/24.
18. Portman, H.: Scaling agile in organizations - Guide for project managers and agile leaders. 1st edn. Van Haren Publishing BV, The Netherlands (2017) (in Dutch)
19. Larman, C., Vodde, B.: Large-Scale Scrum– More with LeSS Addison-Wesley Professional 2016
20. Leffingwell, D.: Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series), Addison-Wesley Professional, 2010
21. SAFe - scaled agile framework in practice, https://www.slideshare.net/Intland/safescaled-agile-framework-in-practice, last accessed 2020/02/29.
22. Poppendieck, M., Poppendieck, T.: Lean Software Development: An Agile Toolkit: An Agile Toolkit. Addison-Wesley Professional, USA (2003)
23. SCRUMstudy: Guide to the Scrum Body of Knowledge (SBoK Guide) 4th edn. ISBN: 978-0-9899252-0-4 USA (2022)
24. Kniberg, H., Ivarsson, A. Scaling agile@ Spotify. https://blog.crisp.se/wpcontent/uploads/2012/11/SpotifyScaling.pdf, last accessed 2023/07/02.
25. The OMG: Business Process Model And Notation (BPMN) Version 2.0 http://www.omg.org/spec/BPMN/2.0/, last accessed 2023/06/24.
26. The OMG: https://www.omg.org/spec/UML, last accessed 2023/06/24.
27. Gotel, O.C.Z., Finkelstein, C.W.: An analysis of the requirements traceability problem. In: Proceedings of IEEE International Conference on Requirements Engineering, pp. 94-101. IEEE, USA (1994), doi: 10.1109/ICRE.1994.292398.
28. The OMG: Semantics of Business vocabulary and Rules https://www.omg.org/spec/SBVR/, last accessed 2023/06/24.
29. Van der Aalst, W.M.P., Barros, A.P., ter Hofstede, A.H.M., Kiepuszewski, B.: Advanced workflow patterns. In Proc. of the 5th IFCIS Int. Conference on Cooperative Information Systems, Eilat, Israel, September 2000. Springer Verlag.
30. Medina-Mora R,Winograd T, Flores R, Flores F: The action workflow approach to workflow management technology. In: CSCW 92 Proceedings, pp 281–288 (1992)
31. Maruyama, M.: The Second Cybernetics: Deviation-Amplifying Mutual Causal Processes. American Scientist, 51(2), 164–179. (1963). DOI: http://www.jstor.org/stable/27838689 (1963).
32. Deming, W.E.: The New Economics. MIT Press, USA (1993).
33. Puustjärvi, J., Tirri, H., Veijalainen, J.: Managing overlapping transactional workflows. In: Constantopoulos, P., Mylopoulos, J., Vassiliou, Y. (eds) Advanced Information Systems Engineering. CAiSE

1996. Lecture Notes in Computer Science, vol 1080. Springer, Berlin, Heidelberg (1996). DOI: 10.1007/3-540-61292-0_19
34. Gudas, S.: Foundations of the Information Systems' Engineering Theory. Vilnius University, Vilnius (2012) (in Lithuanian)
35. Standish group: Chaos report 2015, https://www.standishgroup.com/sample_research_files/CHAOSReport2015-Final.pdf, last accessed 2021/06/10.
36. KPMG, AIPM, IPMA: The future of Project management: global outlook 2019 https://www.ipma.world/assets/PM-Survey-FullReport-2019-FINAL.pdf, last accessed 2021/07/12.
37. Pikkarainen M., Haikara J., Salo O., Abrahamsson P., Still J.: The impact of agile practices on communication in software development. Empirical Software Engineering 13(3), 303-337 (2008).
38. Barra C. L., Crawford B., Soto R., Misra S., Monfroy E.: Agile Software Development: It Is about Knowledge Management and Creativity. In: Murgante B. et al. (eds) ICCSA 2013 LNCS, vol. 7973, pp. 98-113. Springer, Berlin, Heidelberg (2013).
39. Crnogaj, K., Tominc, P., Rožman, M.: A Conceptual Model of Developing an Agile Work Environment. Sustainability 2022, 14, 14807. DOI: https://doi.org/10.3390/su142214807.
40. Van Eck, P., Blanken, H., Wieringa, R.: Project GRAAL: towards operational architecture alignment. International Journal of Cooperative Information Systems, 13(3), 235-255 (2004).
41. Chen, H.M., Kazman, R., Garg, A.: Managing misalignments between business and IT architectures: a BITAM approach. Journal of Science of Computer Programming, 57(1), 5-26, (2005).
42. Morkevičius, A., Gudas, S., Šilingas, D.: SBISAF: a service-oriented business and information systems alignment method. Informatica, 24(2), pp. 231-251, (2013).
43. British Ministry of Defence: MOD Architecture Framework (MODAF), https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/36757/20100602MODAFDownload12004.pdf, last accessed 2021/07/15.
44. The OMG: The TOGAF Standard, Version 9.2, https://www.opengroup.org/togaf, last accessed 2021/08/20.
45. ArchiMate 3.2 Specification, https://pubs.opengroup.org/architecture/archimate3-doc/, last accessed 2023/06/21.
46. Architecture Capability Team Consultation, Command & Control Board https://www.nato.int/nato_static_fl2014/assets/pdf/pdf_2018_08/20180801_180801-ac322-d_2018_0002_naf_final.pdf, last accessed 2023/05/21.
47. The OMG: https://www.omg.org/uaf/ , last accessed 2023/05/21.

48. Leading Agile, D. Prior.: Agile Planning With Ties With Tom Churchwell., https://www.leadingagile.com/podcast/agile- planning-ties-tom-churchwell/, last accessed 2021/06/20
49. Gudas, S., Lopata, A.: Towards internal modelling of the information systems application domain. Informatica 27, 1–29 (2016). DOI: 10.15388/Informatica.2016.74.
50. Gudas, S., Noreika, K: Causal Interactions in Agile Application Development. Mathematics. Basel: MDPI AG. eISSN 2227-7390. 2022, vol. 10, no. 9, art. no. 1497, p. 1-22. DOI: 10.3390/math10091497.
51. Deng, J.: Control problems of Grey Systems, Systems and Control Letters 1 (1982), 288-294.
52. Chan, J.W.K., Tong, T.K.L: Multi-criteria material selections and end-of-life product strategy: Grey relational analysis approach. Materials and Design 28 (2007), 1539-1546.
53. Javanmardi, E., Liu, S.: Exploring Grey Systems Theory-Based Methods and Applications in Analyzing Socio-Economic Systems Sustainability 11(15) (4192) (2019).
54. Gudas, S.,Valatavičius, A.: Extending model-driven development process with causal modeling approach. In: Dzemyda, G., Bernatavičienė, J., Kacprzyk, J. (eds.) Data science: new issues, challenges and applications, SCI, vol. 869, pp. 279–296. Springer Nature Switzerland (2020). DOI: 10.1007/978-3-030-39250-5_7.
55. Wright, S.: Correlation and causation. Journal of Agricultural Research, 20, 557-585 (1921)
56. Schurz, G., Gebharter, A.: Causality as a theoretical concept: explanatory warrant and empirical content of the theory of causal nets. Synthese, 193(4), 1073–1106. http://www.jstor.org/stable/24704835 (2016).
57. Bunge, M.: Causality and Modern Science, 3rd edn. Dover Publications, USA (2011).
58. Eckhard D., Falkenberg, E. D., Hesse, W., Lindgreen, P., Nilsson, B. E., Oei, J. L. H., Rolland, C., Stamper, R. K., Van Assche, F. J. M., Verrijn-Stuart, A. A., Voss, K.: A Framework of Information System Concepts The Frisco Report (1998)
59. Noreika, K.: Improving Enterprise Application Software Development Management With MODAF. In: Joint proceedings of the BIR 2021 workshops and doctoral consortium co-located with 20th international conference on perspectives in business informatics research (BIR 2021), pp. 141–152 (2021).
60. Alkhafaji, A., Nelson, R.A.: Strategic Management Formulation, Implementation, and Control in a Dynamic Environment. 1st edn. Routledge London, United Kingdom (2003).
61. Ulrich, D.: Organizational Capability as a Competitive Advantage: Human Resource Professionals as Strategic Partners. Human Resource Planning, 10, 4, 169–184. (1987)

62. Van De Ven, A. H., Delbecq, A.L., Koenig Jr., R.: Determinants of Coordination Modes within Organizations. American Sociological Review, vol. 41, no. 2, pp. 322–338, (1976).
63. Taxen, L., Riedl, R. Understanding Coordination in the Information Systems Domain. Journal of Information Technology Theory and Application (JITTA). 17. 5-40. (2016).
64. Onday, O.: Classical Organization Theory: From Generic Management of Socrates to Bureaucracy of Weber International Journal of Business and Management Review. Vol.4, No.1, pp. 87–105, (2016)
65. Henderson, J.C., Venkatraman, N.: Strategic alignment: A model for organization transformation via information technology" Working Paper 3223–90. Massachusetts Institute of Technology, 1990, 458 p. ISBN 9781245057264
66. Gerow, J.E., Thatcher, J.B., Grover, V.: Six types of IT-business strategic alignment: an investigation of the constructs and their measurement, European Journal of Information Systems, 24(5), pp. 465–491, DOI: 10.1057/ejis.2014.6 (2015)
67. The OMG: SOA, https://www.omg.org/technology/readingroom/SOA.htm, last accessed 2021/08/30
68. The OMG: Business Modeling Category - Specifications Associated, https://www.omg.org/spec/category/business-modeling/About-business-modeling/, last accessed 2021/07/06.
69. The OMG: Decision Model and Notation, https://www.omg.org/spec/DMN/, last accessed 2023/06/25.
70. Dahalin, M.Z., Razak, A.R., Ibrahim, H., Yusop, I.N., Kasiran, K.M.: An enterprise architecture methodology for business–it alignment: adopter and developer perspectives. In Soliman S.K. (eds.) Communications of the IBIMA, vol. 2011 (2011)
71. Zachman J. A.: A Framework for Information Systems Architecture. IBM Systems Journal, Volume 26, Number 3 (1987).
72. Zhang M., Chen H., Luo A.: A Systematic Review of Business-IT Alignment Research With Enterprise Architecture in IEEE Access, vol. 6, pp. 18933-18944 (2018).
73. Kim, H., Oussena, S., Essien, J., Komisarczuk, P.: Towards Event-Driven Enterprise Architecture. Uncovering Essential Software Artifacts through Business Process Archeology (Advances in Business Information Systems and Analytics), IGI Global, USA, 2014
74. Noreika, K.: Business Capabilities Utilization Enhancement Using Archimate for EAS Projects Delivery in an Agile Environment. In: Matulevičius, R., Robal, T., Haav, H-M., Maigre, R., Petlenkov, E. (eds.) Joint Proceedings of Baltic DB&IS 2020 Conference Forum and Doctoral Consortium co-located with the 14th International Baltic Conference on Data-bases and Information Systems (BalticDB&IS

2020) CEUR Workshop proceedings, vol. 2620, pp. 49–56, Estonia (2020).
75. Perez-Castillo, R., Ruiz-Gonzalez, F., Genero, M., Piattini, M.: A systematic mapping study on enterprise architecture mining. Enterprise Information Systems, 13(5), 675-718 (2019) DOI: 10.1080/17517575.2019.1590859
76. Gudas, S.: Causal Modelling in Enterprise Architecture Frameworks. Informatica, pp. 1–35, doi: 10.15388/21-INFOR446 (2021).
77. Scott, B: Second-order cybernetics as cognitive methodology. Systems Research 13(3), pp: 393–406 (1996)
78. Porter, M.E.: Competitive advantage. 1st edn. The Free Press, USA (1985)
79. Harmon, P.: Business Process Change, 2nd edn. Morgan Kaufmann, USA (2007).
80. Rummler, G.A., Brache, A.P.: Improving Performance. 1st edn. Jossey-Bass, USA (1990).
81. Rouse, M.: Transaction, https://www.techopedia.com/definition/16455/transaction-databases, last accessed 2023/06/22
82. Customer Driven Solutions Limited, Enterprise as a System of Systems, https://eaasos.info/Content/EaaSoS/EaaSoS_Intro.htm, last accessed 2023/06/22
83. Digital.ai Software, Inc.: 14th Annual State of Agile Report, https://stateofagile.com/#ufh- i-615706098-14th-annual-state-of-agile-report/702749, last accessed 2021/03/27
84. Venema, M.: Your Guide to 6 Scaled Agile Frameworks, https://www.nimblework.com/blog/scaled-agile-frameworks/, last accessed 2023/08/12
85. Cohn, M.: User Stories Applied: for Agile Software Development. 1st edn. Addison Wesley, USA (2004).
86. Agile Alliance: Agile Glossary, https://www.agilealliance.org/glossary/epic, last accessed 2021/07/10.
87. Atlassian: Epics, stories, themes, and initiatives, https://www.atlassian.com/agile/project- management/epics-stories-themes, last accessed 2021/05/17.
88. McDonald, K.: Beyond Requirements: Analysis with an Agile Mindset (Agile Software Development Series). 1st edn. Addison-Wesley Professional, USA (2015)
89. Noreika, K., Gudas, S.: Modelling the Alignment Between Agile Application Development and Business Strategies. In: Joint proceedings of the BIR 2021 workshops and doctoral consortium co-located with 20th international conference on perspectives in business informatics research (BIR 2021), pp. 59-73 (2021).
90. Idalko: The Beginner's Guide to Jira Align (2023)https://www.idalko.com/jira-align/. Last accessed 2023/09/09.

91. Microsoft Learn: About default processes and process templates: https://learn.microsoft.com/en-us/azure/devops/boards/work-items/guidance/choose-process?view=azure-devops&tabs=basic-process. Last accessed 2023/09/09.
92. The OMG: MDA - The architecture of choice for a changing world, https://www.omg.org/mda/, last accessed 2022/02/06
93. Kleppe, A., Warmer, J., Bast, W.: MDA explained: the model driven architecture: practice and promise. Addison Wesley, USA (2003).
94. Kulkarni, V., Reddy, S.: Model-driven development of enterprise applications. In: Nunes, N.J., Selic, B., da Silva, A.R., Alvarez, A.T. (eds.) UML Modeling Languages and Applications: <<UML>> 2004 Satellite Activities, LNCS, vol. 3297, pp. 118–128. Springer-Verlag Berlin, Heidelberg (2004).
95. Barat, S., Kulkarni, V.: Developing configurable extensible code generators for model-driven development approach. In: SEKE 2010, pp. 577–582. USA (2010).
96. Ambler, S.: Agile modeling: effective practices for eXtreme programming and the unified process. 1st edn. Wiley, USA (2002).
97. Ambler, S.: The Object Primer. 3rd edn. Cambridge University Press, USA (2004).
98. Kulkarni, V., Barat, S., Ramteerthkar, U.: Early experience with Agile methodology in a model-driven approach. In: Whittle, J., Clark, T., Kühne, T. (eds.) MODELS 2011: Model Driven Engineering Languages and Systems, LNCS, vol. 6981, pp. 578–590. Springer-Verlag GmbH Berlin Heidelberg, New Zealand (2011). DOI: 10.1007/978-3-642-24485-8.
99. Miller, L.: Case study of customer input for a successful product. In: ADC'05, pp. 225–234. IEEE, USA (2005). DOI: 10.1109/ADC.2005.16.
100. Knapp, J.: Sprint: How to solve big problems and test new ideas in just five days. 1st edn. Simon & Schuster, USA (2016).
101. 11 Lessons: managing design in eleven global brands, https://www.designcouncil.org.uk/sites/default/files/asset/document/ElevenLessons_Design_Council%20%282%29.pdf, last accessed 2022/02/20.
102. Matinnejad, R.: Agile model driven development: an intelligent compromise. In: SERA, pp. 197–202. IEEE, USA (2011).
103. Kirby, Jr. J.: Model-driven Agile development of reactive multi-agent systems. In COMPSAC'06, pp. 297–302. IEEE, USA (2006).
104. Guta, G., Schreiner, W., Draheim, D.: A Lightweight MDSD process applied in small projects. In: 35th Euromicro Conference on Software Engineering and Advanced Applications, pp. 255–258. IEEE, USA (2009). DOI: 10.1109/SEAA.2009.63.
105. Zhang, Y., Patel, S.: Agile Model-driven development in practice. IEEE Software 28(2), 84-91 (2011).

106. Noreika, K., Gudas, S.: Causal Knowledge Modelling for Agile Development of Enterprise Application Systems. Informatica, 2023, Vol. 34, No. 1, 121–146.
107. The Enterprise Architect: building an Agile enterprise, http://www.theenterprisearchitect.eu/blog/2008/02/18/mda-and-model-transformation/, last accessed 2022/03/10.
108. Alfraihi, H., Lano, K.: The integration of agile development and model driven development - a systematic literature review. In: 5th International Conference on Model-Driven Engineering and Software, pp. 451–458 (2017). DOI: 10.5220/0006207004510458.
109. Lano, K., Alfraihi, H., Yassipour Tehrani, S., Haughton, H: Improving the application of agile modelbased development: experiences from case studies. In: The Tenth International Conference on Software Engineering Advances, pp. 213–219. IARIA, Spain (2015).
110. Grigera, J., Rivero, J.M., Robles Luna, E., Giacosa, F., Rossi G.: From requirements to web applications in an Agile model-driven approach. In: Brambilla M., Tokuda T., Tolksdorf R. (eds) ICWE 2012: Web Engineering, LNCS, vol. 7387, pp. 200–214. Springer, Berlin, Heidelberg, Germany (2012). DOI: 10.1007/978-3-642-31753-8_15.
111. Nakicenovic, M. B. (2012). An agile driven architecture modernization to a model-driven development solution. International Journal on Advances in Software 5(3, 4), 308–322 (2012).
112. Robles Luna E., Grigera J., Rossi G.: Bridging Test and Model-Driven approaches in web engineering. In: Gaedke M., Grossniklaus M., Díaz O. (eds) ICWE 2009: Web Engineering, LNCS, vol. 5648, pp. 136–150. Springer, Berlin, Heidelberg, Spain (2009).
113. Rivero, J. M., Luna, E. R., Grigera, J., and Rossi, G.: Improving user involvement through a model-driven requirements approach. In: MoDRE 2013 International Workshop, pp. 20–29. IEEE, Brasil (2013).
114. Rivero, J. M., Grigera, J., Rossi, G., Luna, E. R., Montero, F., Gaedke, M.: Mockup-driven development: providing agile support for model-driven web engineering. Information and Software Technology 56(6), 670–687 (2014).
115. Cáceres, P., Díaz, F.J., Marcos, E.: Integrating an agile process in a model driven architecture. In: INFORMATIK 2004 – Informatik verbindet, Band 1, Beiträge der 34. Jahrestagung der Gesellschaft für Informatik e.V. (GI), Ulm, 20.-24, pp. 265–270 (2004).
116. Varajão, J., Lourenço, J., João, G.: Models and methods for information systems project success evaluation – A review and directions for research. Heliyon. vol 8. DOI: 10.1016/j.heliyon.2022.e11977. (2022).
117. Bokovec, K., Damij, T. Rajkovic, T. Rajkovic, V.: Evaluating ERP Projects with Global Efficiency Factors. In: Respício, A., Frédéric, A., Phillips-Wren, G.,Teixeira, C., Telhada, J. (eds.) Frontiers in Artificial Intelligence and Applications, vol. 212: Bridging the Socio-technical Gap

in Decision Support Systems, pp 395-406. IOS Press Ebooks (2010). DOI: 10.3233/978-1-60750-576-1-395
118. Komi-Sirviö, S.: Development and Evaluation of Software Process Improvement Methods. PhD Thesis. VTT Technical Research Centre in Finland. Finland (2004)
119. Cheng, C.K., Permadi, R.B.: Towards an Evaluation Framework for Software Process Improvement A Model for Software Process Improvement Evaluation. Master Thesis School of Engineering at Blekinge Institute of Technology Sweden (2009)
120. Ahern, M. D., Clouse, A., Turner, R.: CMMI Distilled: A Practical Introduction to Integrated Process Improvement. Addison-Wesley Professional USA (2008)
121. International Organization for Standartization: ISO/IEC 15504-5:2012 Information technology — Process assessment — Part 5: An exemplar software life cycle process assessment model https://www.iso.org/standard/60555.htm. Last accessed 2023/09/06.
122. Nurdiani, I., Börstler, J., Fricker, S., Petersen, K., Chatzipetrou, P.: Understanding the order of agile practice introduction: Comparing agile maturity models and practitioners' experience. Journal of Systems and Software, vol. 156, pp. 1–20, 2019. DOI: 10.1016/j.jss.2019.05.035.
123. Paulk, M. C., Curtis, B., Chrissis, M. B., Weber, C. V.: Capability maturity model, version 1.1, IEEE Software, vol. 10, no. 4, pp. 18–27, 1993. doi: 10.1109/52.219617.
124. Reitz. F.: Evaluating and Automating a Scaled Agile Framework Maturity Model Master's thesis. KTH royal institute of technology, Stockholm (2021)
125. Turetken, O., Stojanov, I., Trienekens, J. J. M.: Assessing the adoption level of scaled agile development: a maturity model for scaled agile framework, Journal of Software: Evolution and Process, vol. 29, no. 6, p. e1796, 2017. DOI: 10.1002/smr.1796
126. Kaplan, R. S., Norton, D. P.: Using the balanced scorecard as a strategic management system. Harvard Business Review 74(1), 75–85 (1996).
127. McBride, S.: RICE: Simple prioritization for product managers https://www.intercom.com/blog/rice-simple-prioritization-for-product-managers/ . Last accessed 2023/09/09.
128. Timeular: RICE Prioritization Method: Everything You Need to Know https://timeular.com/blog/rice-method/ Last accessed 2023/09/11.
129. Fernandes, B.C.L., Gomes, J.V.: OKR Methodology: Case Study in Sebrae Meier. International Journal of Strategic Decision Sciences. vol. 14. No. 1 (2023). DOI: 10.4018/IJSDS.318341.
130. Weekdone: 2021 Step by Step Guide to OKRs. https://static.weekdone.com/wp-content/uploads/2021-Step-by-Step-Guide-to-OKRs-1.pdf. Last accessed 2023/09/11.
131. Ortu, M., Destefanis, G., Adams, B., Murgia, A., Marchesi, M. Tonelli, R.: The JIRA Repository Dataset: Understanding Social Aspects of

Software Development. In Proceedings of the 11th International Conference on Predictive Models and Data Analytics in Software Engineering (PROMISE '15). Association for Computing Machinery, New York, NY, USA, Article 1, 1–4. DOI: 10.1145/2810146.2810147
132. Krasniqi, R., Do, H.: A multi-model framework for semantically enhancing detection of quality-related bug report descriptions. Empir Software Eng 28, 42 (2023). DOI: 10.1007/s10664-022-10280-w
133. Diamantopoulos, T., Nastos, D-N. Symeonidis, A.: Semantically-enriched Jira Issue Tracking Data. IEEE/ACM 20th International Conference on Mining Software Repositories (MSR), Melbourne, Australia, pp. 218-222 (2023). DOI: 10.1109/MSR59073.2023.00039.
134. Făgărăşan, C., Cristea, C., Cristea, M., Popa, O., Mihele, C. Pîslă, A.: Key performance indicators used to measure the adherence to the iterative software delivery model and policies. IOP Conference Series: Materials Science and Engineering, Volume 1256, Baile Felix SPA, Oradea, Romania 2022
135. Atlassian: Database schema https://dac-static.atlassian.com/server/jira/platform/attachments/jira_9.0_database_schema.pdf?_v=1.823.0. Last accessed 2023/09/11.
136. Vavpotič, D., Bala, S., Mendling, J., & Hovelja, T.: Software Process Evaluation from User Perceptions and Log Data. Journal of software: Evolution and Process, 34(4) (2022). DOI: doi.org/10.1002/smr.2438
137. Noreika, K.; Gudas, S.: Using Management Transaction Concept to Ensure Business and EAS Alignment in an Agile Environment. In Information and Software Technologies ICIST 2021. Communications in Computer and Information Science, 1st ed.; Lopata, A., Gudonienė, D., Butkienė, R. (eds.); Springer, Cham: Switzerland, 2021; Volume 1486, pp. 109–120.
138. Tekutov, J.: A model of domain knowledge content updating based on management information interactions. PhD Thesis, Vilnius (2013)
139. Malak, R.J. Jr. A Framework for Validating Reusable Behavioural Models in Engineering Design. Master thesis, Georgia Institute of Technology, USA, 2005.

# SUMMARY IN LITHUANIAN

## ĮVADAS

Šiuo metu daug įmonių remiasi taikomųjų programų sistemomis (TPS), kurios palaiko, organizuoja ir padeda valdyti procesus organizacijoje. Taikomųjų programų sistemos kuriamos naudojant skirtingus projektų valdymo metodus ir atitinkamus programų paketus (tokius kaip „JIRA“, „Azure DevOps“, „MS Project“ ir kitus). Projektų eigos stebėjimo ir kontrolės metodo parinkimas turi didelę įtaką taikomųjų programų sistemų vystymo projektų kokybei. Pritaikius tinkamą TPS kūrimo valdymo metodą, jau vykdant projektą galima įvertinti, kiek bus pasiekti verslo tikslai, įmonės strategijos kryptys, sąnaudos, kokybė ir ar programų sistemos vystymo projektas bus užbaigtas laiku.

Šiuo metu vyksta naujas gamybos sektoriaus ir kitų organizacinių veiklų (visų vertės grandinės etapų) skaitmeninimo etapas, vadinamas ketvirtąja pramonės revoliucija „Industry 4.0“ (4IR). Naujausių 4IR diegiamų informacinių technologijų esminis bruožas – programų sistemų intelektinimas, apimant žmogaus ir mašinos (kompiuterio, išmaniojo įrenginio) sąveiką, sistemų inžinerijos procesus, veiklos duomenų analitiką, gamybinės ir projektinės veiklos valdymą. Intelektinių sistemų kūrimas neatsiejamas nuo priežastingumo tyrimo konkrečioje veiklos srityje, surastų priežastinių sąveikų modelio, kuris vadinamas vidiniu modeliu, kūrimo [5].

Bet kurio tipo intelektinė sistema turi konkrečios probleminės srities priežastinių sąveikų modelį – vidinį modelį, kurio pagrindu ji gali daryti įtaką realaus pasaulio procesams. Formaliai intelektinių sistemų sudėtis yra apibrėžta Conanto ir kt. kaip „gerojo reguliuotojo“ teorema [4] ir remiasi vidinio modelio (VM) principu, kurį suformulavo Francis [5]. Vidinis modelis – dalykinės srities priežastinių žinių modelis, būtinas bet kurios valdymo (kontrolės) sistemos komponentas. Vidinį modelį galima apibūdinti kaip modelį, kuris specifikuoja aptiktas ir žinomas problemos srities priežastines priklausomybes. Tokių 4IR kartos sistemų kaip „skaitmeniniai dvyniai“ [7], kurios užtikrina tiek fizinių (laivai, lėktuvai ir kt.), tiek socialinių sistemų (miestai) valdymą, sudėtyje yra realybės srities priežastinių sąveikų modeliai – vidiniai modeliai.

Konkurencingas veiklos valdymas neįmanomas be taikomųjų programų sistemų (TPS), kurias naudoja personalas efektyviai apskaityti, analizuoti, palaikyti ir valdyti procesus organizacijoje. Šios disertacijos tyrimo sritis – taikomųjų programų sistemų (TPS) kūrimo proceso valdymas, siejant įmonės valdymo strategijas su taikomųjų programų sistemų inžinerija, kai naudojamas „Agile“ metodas TPS projektams valdyti. Tyrimo srities procesai

yra vertinami kaip sudėtinga sistema, turinti savivaldos galimybes, t. y. uždaro ciklo sąveiką tarp vidinių veiklų (valdymo transakcijų).

Tobulinant esamą „Agile“ metodą sukurtas modifikuotas „Agile“ metodas (priežastinio modeliavimo pagrindu) ir jo realizavimo projektas, kurio sudėtyje yra toks vidinis modelis.

Priežastinis modeliavimas skirtas atrasti priežastines procesų ir informacijos atributų priklausomybes įvairiose realaus pasaulio srityse. Priežastinį žinojimą Zackas apibrėžia kaip „priežastinių ryšių tarp veiksnių visumos aprašymą <...>, kuris suteikia organizacijoms priemonę, <...> kaip geriausiai pasiekti kokį nors tikslą“ [6]. Taigi pažangių sudėtingų (kompleksinių) sistemų kūrimo pagrindas ir būtina sąlyga yra priežastinis (vidinis) modeliavimas, kurio tikslas – atskleisti problemos srities priežastines priklausomybes, sužinoti konkrečiai sričiai būdingą priežastinį ryšį. Vidinis modeliavimas – paradigma, atitinkanti priežastinio modeliavimo metodą.

Priežastinio modeliavimo metodas padeda sukurti virtualų realaus pasaulio domeno žinių modelį (domeno priežastingumo modelį), kuris yra beveik identiškas realiai aplinkai, kad būtų galima imituoti tam tikrą procesą prieš skiriant dideles investicijas į tokio proceso įgyvendinimą realiame pasaulyje. Todėl aktualu apžvelgti tradicinius programų sistemų inžinerijos metodus, probleminės srities (domeno) modeliavimo kalbas, projektų valdymo metodus ir įrankius priežastinio modeliavimo paradigmos požiūriu.

„Agile“ požiūris atsirado kaip filosofija, būdas spręsti taikomųjų programų sistemų kūrimo projektų vykdymo trūkumus, kai sukurtos programų sistemų funkcijos niekada nenaudojamos, taip pat labai ilgų projekto trukmių problemą, dalijimosi žiniomis proceso spragas viso IT sistemos kūrimo proceso metu ir, galiausiai, spręsti nesutapimus tarp sukurtos taikomosios programinės sistemos ir verslo tikslų, kurių buvo siekiama kuriama programų sistema. „Agile“ požiūrio ir iteratyvių metodų ištakos siekia XX a. 3-iąjį dešimtmetį [8]. „Agile“ karkasai programų sistemoms kurti, kaip 1995 m. „Scrum“ [9, 10], 1997 m. DSDM [11], 1998 m. „Crystal“ [12, 13], 1999 m. ASD [14] ir XP [15] ir darbo valdymo sistema „Kanban“ [16] buvo „Agile“ požiūrio pirmtakai, padėję suformuluoti „Agile“ taikomųjų programų sistemų kūrimo manifestą“ [17], geriau žinomą tiesiog kaip „Agile“ manifestas“. Šį 2001 m. apibrėžė IT sistemų kūrėjų grupė, kuri norėjo spręsti pirmiau minėtus taikomųjų programų sistemų kūrimo projektų trūkumus. Tai pirmasis struktūruotas visuotinai pripažįstamas dokumentas, aprašantis „Agile“ požiūrį. Vienas iš pagrindinių „Agile“ požiūrio aspektų – iteratyvus ir inkrementinis požiūris į IT projektų vykdymą užtikrinant nuolatinį IT sistemos vystymą su reguliariu sistemos užsakovų grįžtamuoju ryšiu.

Kaip Portmanas aprašė knygoje „Agile“ plėtra organizacijose – vadovas projektų vadovams ir „Agile“ lyderiams“, „Agile“ turi daugiau kaip 15 skirtingų karkasų [18]. Populiariausi „Agile“ taikomųjų programų sistemų kūrimo karkasai darbui organizuoti ir valdyti komandos lygmenyje yra „Scrum“ [9, 10] ir „Kanban“ [16]. Visai organizacijai arba didelio masto projektams valdyti – „Large Scale Scrum“ (LeSS) [19], „Scaled Agile Framework SAF’e“ [20, 21] ir kiti karkasai, kurie remiasi „Lean“ taikomųjų programų sistemų kūrimo teorija [22]. „Scrum Body of Knowledge“ (SBoK) yra „išsamus „Scrum“ metodikų ir praktikų gidas“ [23], kurio 4-tajame leidime yra skyrių apie „Scrum“ plėtrą didelio masto organizaciniuose projektuose. „Agile“ karkasai taikomųjų programų sistemoms kurti naudojami vis dažniau, tačiau neretai yra vertinami kaip atsitiktiniai, o jų naudojimo sėkmę sudėtinga įvertinti tiesiogiai arba remiantis klasikinio projektų valdymo laiko, apimties ir kainos apribojimais.

Disertacijoje „Agile“ metodika, naudojama TPS projektams valdyti, apžvelgta priežastinio modeliavimo paradigmos požiūriu, nes tai yra pagrindas, kuriuo grindžiami efektyviausi projektų valdymo metodai (įskaitant tiek verslo valdymo, tiek taikomųjų programų sistemų inžinerijos valdymo veiklą). Konkretūs „Agile“ metodai, naudojami šioje disertacijoje, yra „Scrum“ komandos lygio veiklai valdyti ir „Scaled Agile Framework“ metodo atributai. Komandos sudėtis ir organizacinės struktūros remiasi „Spotify“ modeliu [20].

Disertacijoje pateikiamas modifikuotas priežastiniu modeliavimu pagrįstas „Agile“ projektų valdymo modelis, identifikuojantis projekto darbų, apibrėžtų kaip „Agile“ veiklos, hierarchinių sąveikų turinį (ir prasmę). Jis leidžia formuoti naujus projekto būsenos vertinimo rodiklius, kurių šiuo metu „Agile“ projektų valdymo priemonėse (pvz., „Atlassian Jira“) nėra.

Disertacijos autorius remiasi nuostata, kad taikomosios programų sistemos būsenos įvertinimas galimas tik tada, kai nurodytas „Agile“ veiklų sąveikų turinys, o tai įmanoma tik pasirinkus vidinio modeliavimo paradigmą (priežastinį modeliavimą).

Tradiciniai taikomųjų programų sistemų inžinerijos metodai, tokie kaip verslo procesų analizė naudojant BPMN [25], UML [26] ar kitas vizualinio modeliavimo notacijas (kalbas), skirti pavaizduoti verslo sritį iš išorinio modeliavimo perspektyvos. Reikalavimų atsekamumas [27] – tradicinis būdas užtikrinti verslo tikslų suderinimą su taikomųjų programų sistemų kūrimo reikalavimais. Tačiau šiuose metoduose ir tradicinio (krioklio tipo) bei „Agile“ projektų valdymo priemonėse, pvz., „Atlassian Jira“, nėra vidiniu modeliavimu grįsto priežastingumo.

Šioje disertacijoje pateikiamas naujas požiūris į taikomųjų programų sistemų kūrimo projektų valdymą „Agile“ būdu, paremtą priežastinio modeliavimo metodu, įskaitant organizacijos strateginių tikslų integraciją bei TPS projekto būsenos turinio įvertinimą. Pateikiami pagrindiniai programų sistemos kūrimo proceso tobulinimo rezultatų efektyvaus matavimo ir įvertinimo sąvokos ir procesai.

Didžiojoje šios disertacijos dalyje pateikiama TPS valdymo metodų, įskaitant „Agile“ projektų valdymo metodus („Scrum“, „SAF'e“, AMDD, ir t. t.), literatūros apžvalga priežastingumo aspektu. Populiariausi „Agile“ projektų valdymo įrankiai taip pat išanalizuoti priežastingumo aspektu. Įvertinta „Agile“ veiklų specifikacijos ir organizacijos strateginių tikslų sąveika. Literatūros apžvalga skirta nustatyti taikomosios programų sistemos kūrimo proceso stebėsenos bei progreso įvertinimo trūkumus ir surinkti reikalavimus programų sistemos kūrimo proceso matavimo ir turinio įvertinimo metodui, kurį siūlo autorius. Remiantis literatūros peržiūros rezultatais, suformuluotas „Agile“ programinės sistemos kūrimo matavimo ir įvertinimo modelis. Modelio tikslas – sudaryti pagrindą programų sistemos kūrimo procesui matuoti ir vertinti. Manoma, kad modelis pritaikomas daugeliui scenarijų, nes pateikiami nepriklausomi nuo konkrečių programų sistemos kūrimo iniciatyvų ir įrankių konceptai.

Probleminės srities (domeno) priežastingumo modeliavimas turi keletą skirtingų tikslumo lygių:

1. Priežasties ir pasekmės modeliavimas organizacinėse sistemose remiasi verslo taisyklėmis JEI <..> TAI <..> KITAIP <..> [28] ir yra naudojamas organizacijos valdymui modeliuoti.
2. Darbų sekos modeliai priežasties ir pasekmės būsenų aibę paverčia nuoseklia grandine, vientisa seka, kuri turi turėti gerai apibrėžtą pradžią ir pabaigą. Šie modeliai – nuoseklūs procesai, jų valdymas yra išorinis ir nėra modeliuojamas [29].
3. Modeliai, sudaryti iš grįžtamojo ryšio kilpų, kaip būtino komponento domeno priežastingumui aprašyti / specifikuoti, tokie kaip: veiksmų darbų sekos [30], priežastinės kilpos diagrama [31], „Deming PDCA“ ciklas [32], transakciniai darbų sekos modeliai [33]. Formalus tokių modelių pagrindas – kontrolės teorija. Pritaikant kontrolės teoriją organizacijos valdymui modeliuoti informacijos / žinių transformavimo atžvilgiu sukurtas valdymo transakcijos konstruktas [34], kuris naudojamas šioje disertacijoje modeliuoti Agile veiklų sąveikoms.

## Motyvacija

Taikomųjų programų sistemų (TPS) sprendimai moderniose organizacijose naudojami stebėti ir valdyti verslo procesams ir operacijoms: padėti organizacijoms išleisti savo produktus į rinką, bendradarbiauti su partneriais, pasiekti naujų strateginių tikslų, plėstis bei augti. TPS visada turi visiškai palaikyti verslo valdymo procesus, tokiu būdu užtikrindamos suderinamumą su organizacijos strateginiais tikslais, verslo operacijomis ir informacinės sistemos (IS) galimybėmis. Sykiu organizacija vykdo tolesnius TPS vystymo projektus, kuriuos reikia valdyti, o jų rezultatus privalu suderinti su esančiomis TPS bei organizacijos strategijomis.

Tačiau verslo valdymo poreikių ir TPS funkcionalumo suderinimo rezultatas yra dažniausiai siekiamas įvairių organizacijos valdymo hierarchijos lygių atstovų bendravimu ir bendradarbiavimu, o tai nėra atliekama sistemiškai ir nėra naudojami formalūs metodai užtikrinti verslo reikalavimų ir IS funkcionalumo suderinimą.

Tokia situacija priveda prie mažesnio galutinių vartotojų pasitenkinimo TPS projektų rezultatais ir blogesnių visos organizacijos veiklos rezultatų. „Standish“ grupė, lyderiaujanti projektų valdymo statistikos teikėja, dalijasi rezultatais nuo 2011 iki 2015 metų, kur sėkmingi projektai sudarė tik iki 31 % visų projektų [35]. Apibrėžimas „sėkmingas“ reiškia, kad projektas baigtas laiku, atitiko apimtį bei biudžeto apribojimus ir kad projekto suinteresuotieji asmenys buvo patenkinti projekto rezultatu. Projektų valdymo tyrimai, atlikti tokių organizacijų kaip KPMG, AIPM ir IPMA [36], parodė, kad tik iki 44 % TPS vystymo projektų įgyvendinti laikantis pradinio tikslo ir verslo ketinimų, tik iki 36 % atitiko biudžeto, o 30 % – laiko apribojimus.

Nors „Agile“ metodai vis labiau populiarėja [83], siekiant pagerinti sėkmingo IT projektų įgyvendinimo rodiklius, jų nepakanka. Ryšys tarp užduoties ar vartotojo istorijos TPS projekte ir įmonės strateginio tikslo nėra aiškiai apibrėžtas. Ši problema neišsprendžiama tradiciniu reikalavimų atsekamumo metodu [27], nes neapima verslo konteksto ar domeno priežastingumo. Tai rodo, kad verslo ir IT suderinamumo problema vis dar yra svarbi tyrimo sritis.

„Agile“ projektų valdymo įrankiai turi ribotas projekto eigos sekimo galimybes, o projektų valdymo įrankiai, kuriuose būtų vidinis domeno modelis, gali padėti sprendimų priėmimo procese ir automatiškai, be žmogaus įsikišimo, įvertinti projekto eigos rodiklius – t. y. atlikti TPS projekto sprendimų analizę, siekiant užtikrinti geresnį verslo ir IT suderinamumą.

Dabartiniai, toliau išvardyti, metodai reikalauja reikšmingų laiko investicijų verslo poreikių ir IT suderinamumui užtikrinti bei TPS projekto

reikalavimams įvertinti atsižvelgiant į organizacijų procesų modelius. Be to, jie nuodugniai neapibrėžia ryšio tarp strateginių verslų tikslų, organizacijos gebėjimų ir TPS reikalavimų. Tai tokie metodai kaip: verslo ir TPS suderinimo gairės dėl architektūros suderinamumo GRAAL [40], verslo ir IT suderinimo metodas BITAM [41], į paslaugas orientuotas verslo ir informacinių sistemų suderinimo metodas SBISAF [42] bei organizacijų informacinės architektūros karkasai: „Ministry of Defence Architecture Framework“ (MODAF), „Open Group Architecture Framework“ (TOGAF) [44], „ArchiMate“ [45], NAF [46], UAF [47].

Remiantis priežastiniu modeliavimu, verslo poreikių apibrėžimas TPS vystyti reikalauja modeliavimo metodų, kurie būtų orientuoti į specifinės dalykinės srities priežastingumo supratimą [34, 49, 50].

Tipinėje „Agile“ valdymo struktūroje apibrėžiamas tik pats ryšys tarp veiklų (projekto reikalavimų). Tačiau informacinių srautų tarp „Agile“ veiklų (temos, iniciatyvos, epikai, vartotojo istorijos) turinys nėra specifikuotas. Iš priežastinio modeliavimo perspektyvos informacinių transakcijų tarp „Agile“ hierarchijos veiklų sąveika privalo būti specifikuota.

Atliktas tyrimas leidžia tvirtinti, kad norint sukurti intelektinę aplinką TPS projektams, projekto būsenos stebėsenos ir įvertinimo galimybėms valdyti, reikalingas modifikuotas „Agile“ hierarchijos valdymo modelis, paremtas priežastinio modeliavimo metodika ir susietas su organizacijos strategijos modeliais.

Tyrimo objektas ir apimtis

Įmonės taikomųjų programų sistemų (TPS) projektavimo sprendimų suderinimas su įmonės strategija paremtais gebėjimais „Agile“ projektų valdymo aplinkoje.

Tyrimo problema

„Agile“ požiūris yra populiarus valdyti taikomųjų programų sistemų (TPS) kūrimo projektams. Tačiau „Agile“ metodai („Scrum“, „Kanban“, „SAFe“, „LeSS“) neapima verslo strategijos modeliavimo. Tai lemia IS funkcionalumo ir verslo poreikių neatitikimą. Šiame tyrime pagrindinis dėmesys skiriamas strateginių verslo tikslų ir sukurtų taikomųjų programų sistemų funkcionalumo neatitikimo problemai spręsti. „Agile“ projektų valdymo įrankiams trūksta organizacijos strateginių tikslų modeliavimo funkcionalumo, nes TPS projektų tikslams ir jiems įgyvendinti stinga suderinamumo su įmonės strateginiais tikslais. Taip pat dabartiniams „Agile“

projektų valdymo įrankiams trūksta TPS vystymo projekto veiklų turinio koordinavimo ir įvertinimo funkcionalumo.

Šiuo tyrimu siekiama permąstyti „Agile“ projektų valdymą naudojant priežastinio modeliavimo metodą. Priežastinis modeliavimas leidžia nustatyti priežastines sąveikas tarp skirtingų „Agile“ hierarchijos lygių veiklų atskleidžiant informacinį sąveikų turinį. Priežastinis modeliavimas remiasi formaliomis priežastinių sąveikų specifikacijomis ir sudaro prielaidas formalizuoti „Agile“ veiklų struktūros analizę ir tarpusavio transakcijas. Šioje disertacijoje verslo srities ir TPS projekto veiklų priežastinės sąveikos nurodomos kaip valdymo transakcijos VT. VT yra vientisa struktūra specifikuoti „Agile“ veikloms visuose „Agile“ hierarchijos lygiuose. Jos leido modifikuoti tradicinį „Agile“ projektų valdymo metodą ir sukurti priežastinį „Agile“ valdymo modelį. Priežastinis „Agile“ valdymo modelis padeda nuosekliai sekti TPS projekto turinio atitiktį organizacijos strateginiams tikslams.

Tyrimo tikslas

Sukurti modifikuotą „Agile“ projektų valdymo metodą ir projektų valdymo sistemos architektūrą naudojant priežastinio modeliavimo metodą, užtikrinantį TPS projekto sprendimų suderinimą su verslo strategija ir leidžiantį kompiuterizuoti TPS projekto specifikavimo būklės vertinimą atsižvelgiant į organizacijos strateginius tikslus.

Tyrimo uždaviniai

1. Ištirti „Agile“ projektų valdymo įrankius ir metodus TPS projektams valdyti, kurie įvertina verslo strategijos įgyvendinimą.
2. Ištirti galimybes integruoti priežastinio modeliavimo ir organizacijos informacinės architektūros karkasus organizacijos strateginiams tikslams kaip TPS funkcinių reikalavimų pagrindui nustatyti.
3. Sukurti modifikuotą „Agile“ projektų valdymo metodą, pagrįstą verslo srities priežastingumo modeliu, specifikuotu kaip valdymo transakcija, įskaitant verslo strategijos modeliavimą ir „Agile“ veiklų turinio koordinavimą.
4. Suderinti organizacijos strateginius verslo tikslus su aukščiausio lygio „Agile“ veiklomis ir identifikuoti funkcinius TPS projektų reikalavimus naudojant organizacijos informacinės architektūros karkasus strateginiams verslo tikslams dekomponuoti.

5. Sudaryti modifikuotos „Agile“ hierarchijos veiklų horizontalių ir vertikalių tarpusavio sąveikų taksonomiją, kuria remiantis būtų galima vertinti TPS projekto vykdymo būseną atsižvelgiant į organizacijos strateginius verslo tikslus ir naudojant kokybinius vertinimo kriterijus.
6. Suprojektuoti modifikuoto „Agile“ projektų valdymo metodo aplinkos architektūrą, sukurti modifikuotos „Agile“ projektų valdymo sistemos prototipą su TPS projektavimo sprendimų stebėsenos ir įvertinimo funkcionalumu ir atlikti eksperimentinį pasiūlyto metodo vertinimą.

Tyrimo metodika

Tyrimo tikslas ir metodika suformuluota apibendrinant plačią autoriaus TPS vystymo projektų valdymo patirtį ir daugiau kaip šešerių metų „Agile“ projektų valdymo metodo ir įrankių naudojimo įvairių sričių projektams įgyvendinti patirtį.

Atliekant tyrimą, dėmesys skirtas TPS vystymo sprendimų suderinamumui su organizacijos (verslo) strateginiais tikslais gerinti „Agile“ projektų valdymo požiūriu ir įrankiais apibrėžiant pagrindinius intelektinės projektų valdymo sistemos architektūros komponentus.

Tai pasiekta perkeliant realaus pasaulio probleminės srities (domeno) modeliavimą ir TPS vystymą į priežastinio modeliavimo paradigmą [34, 49, 50] ir taip įtraukiant verslo srities priežastines žinias į „Agile“ projektų valdymo procesą. Verslo srities sąveikos ir „Agile“ valdymo veiklų priklausomybės yra tiriamos naudojant priežastinio modeliavimo karkasą – valdymo transakcijos (VT) konstruktą.

Išorinis / vidinis modeliavimas atitinka „pilkosios sistemos“ modelį pagal „pilkųjų sistemų“ teoriją [51, 52, 53]. Metodologijos, naudojančios pilkosios dėžės požiūrį, yra skirtingo neapibrėžtumo (pilkumo) lygio ir kai kurie tokių metodologijų pavyzdžiai, kaip verslo taisyklės ar ontologijos, skirti pereiti nuo pilkosios link baltosios dėžės modeliavimo. Ribiniai atvejai, kurie paverčia pilkosios dėžės modelius į baltosios dėžės modelius, yra tie, kai modeliuojant naudojamos priežastinės žinios [54].

Priežastingumo samprata moksliškai tyrinėjama daugiau kaip šimtą metų [55]. Priežastingumas – svarbi šiuolaikinio mokslo ir pažangių technologijų sąvoka, padedanti atskleisti probleminės srities savybes, kurios nematomos išoriniam stebėtojui, ir sukurti atitinkamus modelius bei IT sistemas [57].

Taikant VT karkasą, atskleidžiama „Agile“ valdymo sąveikų struktūra ir informacijos turinys, ir taip sukuriamas pagrindas „Agile“ hierarchijos koordinavimo taksonomijai bei projekto būsenos įgyvendinimo įvertinimo indikatoriams.

Kitas tyrimo metodikos komponentas – organizacijos informacinės architektūros karkaso integravimas, siekiant nuosekliai dekomponuoti organizacijos strategiją į gebėjimų rinkinį [43, 46]. Gebėjimai specifikuoja su organizacijos strategija susijusių aukšto lygio veiklų reikalavimus ir procesų darbo sekas. Kitas žingsnis – identifikuoto gebėjimų rinkinio sulyginimas su priežastine „Agile“ hierarchija.

Konceptualus visapusis priežastinio „Agile“ valdymo požiūris remiasi FRISCO tetraedro modeliu [58].

Ginamieji teiginiai

Šiame tyrime ginami šie teiginiai:

1. „Agile“ hierarchijos praplėtimas įtraukiant įmonės strategijų modeliavimą sudaro prielaidas analizuoti vykdomo TPS projekto plėtros sprendimų suderinamumą su verslo valdymo poreikiais virtualioje aplinkoje.
2. Modifikuotas „Agile“ proceso modelis, pagrįstas vientisa (normalizuota) visų „Agile“ veiklų vidine struktūra (tema, iniciatyva, epžas, vartotojo istorija), kuri apibrėžiama kaip valdymo transakcija (VT), leidžia suvienodinti „Agile“ veiklų specifikacijų formas. Tai leidžia sukurti sistemą, skirtą TPS projekto turinio analizei, t. y. įvertinti TPS projekto specifikacijų būklę be žmogaus įsikišimo.
3. Modifikuotos „Agile“ proceso veiklos (tema, iniciatyva, epas, vartotojo istorija) turi vientisą vidinę struktūrą ir yra suderintos su įmonės strategijų ir veiklos procesų modeliais (apibrėžtais organizacijos informacinės architektūros modeliais), o tai suteikia galimybę specifikuoti intelektualaus projektų valdymo įrankio žinių bazę.

Pagrindinis indėlis ir naujumas

1. Naudojant priežastinio modeliavimo metodą, sukurtas modifikuotas „Agile“ procesas, apibrėžiantis TPS projekto valdymo veiklas kaip valdymo transakcijų VT = (F, P, A, V) hierarchiją.
2. Modifikuotas „Agile“ metodas papildytas struktūriniais veiklos modeliais – TPS projekto veiklos susietos su organizacijos veiklos strategijomis ir operatyviniais modeliais naudojant organizacijų informacinės architektūros karkasą MODAF, taip užtikrinant organizacijos valdymo strategijų ir TPS projekto sprendimų suderinimą bei automatizuotą projektų valdymo stebėseną, paremtą priežastinės srities žinių modeliais.

3. Modifikuoto „Agile“ proceso veiklos (tema, iniciatyva, epas, vartotojo istorija) apibūdinamos kaip valdymo transakcijos su tipine vidine struktūra VT = (F, P, A, V), o tai leido apibrėžti TPS projekto būsenas (koordinavimo tipus) ir kiekybinio įvertinimo parametrus.
4. Sukurti programiniai elementai (modeliai), reikalingi įdiegti žiniomis grįstą „Agile“ projektų valdymo komponentą, paremtą modifikuotu „Agile“ procesu:
   - vartotojo sąsajos išdėstymas – duomenų ataskaitų elementai;
   - žinių bazė (modelis) modifikuoto „Agile“ proceso elementų specifikacijai.
5. Pristatyti indikatoriai įvertinti santykinį TPS projekto sudėtingumą remiantis tipinio TPS projekto specifikacijomis. Indikatoriai leidžia įvertinti TPS projekto sudėtingumą pagal sąveikos specifikacijos turinį; pristatytos vidutinės bei normalizuotos jų vertės.

Praktinė reikšmė

Remiantis sukurta „Agile“ veiklų sąveikų klasifikacija apibrėžti kiekybiniai TPS projektų parametrai suderinamumui su verslo valdymo reikalavimais įvertinti.

Sukurtas „Agile“ projektų valdymo prototipas TPS projektų reikalavimų atitikčiai organizacijos verslo strateginiams tikslams stebėti.

Rezultatų aprobavimas

Tyrimų rezultatai pristatyti 5 tarptautinėse ir 5 nacionalinėse konferencijose. 3 straipsniai paskelbti recenzuojamuose konferencijos leidiniuose. 1 straipsnis publikuotas kaip knygų skyrius „Springer“ knygų serijoje, kuri priskiriama prie kitų recenzuojamų periodinių leidinių, tęstinių ar vienkartinių mokslo leidinių.

2 straipsniai paskelbti žurnaluose, nurodytuose “Clarivate Analytics Web of Science” duomenų bazės leidiniuose su citavimo indeksu.

Disertacijos apimtis ir struktūra

Ši disertacija suskirstyta į keturis pagrindinius skyrius, kurių pabaigoje pristatomos apibendrintos išvados ir literatūros sąrašas. Pirmajame skyriuje pateikiami su disertacijos tematika susiję darbai: verslo ir IT suderinimo metodai, „Agile“ metodai ir įrankiai, modeliais grįstas sistemų vystymas, priežastinis modeliavimas ir TPS projektų būsenos vertinimo metodai.

Antrajame skyriuje aprašomas priežastiniu modeliavimu pagrįstas „Agile“ valdymo metodas. Trečiajame skyriuje pristatoma patobulinto „Agile“ projektų valdymo įrankio architektūra. Ketvirtajame skyriuje pateikiamas programų sistemų kūrimo projektų eksperimentinio vertinimo tyrimas ir pristatomi rezultatai.

# 1. SUSIJĘ DARBAI

Daug kartų bandyta pagerinti verslo valdymo ir IT gebėjimų suderinamumą. Labiausiai pažymėtini bandymai pateikti šiame skyriuje.

## 1.1. Verslo ir IT suderinimo valdymo metodai

Strateginis valdymas – tai organizacijos ir jos aplinkos vertinimas siekiant ilgalaikių organizacijos prisitaikymo prie savo aplinkos tikslų per galimybių išnaudojimą ir grėsmių sumažinimą [60]. Organizacijos gebėjimas apibūdinamas kaip „įmonės gebėjimas valdyti žmones, norint įgauti konkurencinį pranašumą“.

Verslo valdymo ir IT galimybių derinimo metodai paprastai orientuoti į komunikaciją ir lyderystę. Jie neturi jokios formalios struktūros ir yra pagrįsti vadybiniu aspektu, rašytiniu ar žodiniu bendravimu bei „sveiku protu“; „norint suderinti verslą ir IT, nepakanka užtikrinti, kad informacija, apibrėžta kaip strateginiai įmonės tikslai, būtų sėkmingai perkelta į taikomųjų programų sistemų kūrimo užduotį“ [59].

Procesams modeliuoti naudojamas BPMN [68] gali būti vertinamas tik kaip išorinis (juodosios dėžės) modeliavimo metodas. Tačiau DMN [69] jau galima priskirti vidinei modeliavimo paradigmai, kuri leidžia nurodyti pilkosios (baltosios) dėžės modelius.

Organizacijų informacinės architektūros karkasai („Ministry of Defence Architecture Framework“ (MoDAF) [43], „The Open Group Architecture Framework“ (TOGAF) [44], „ArchiMate“ [45], NAF [46], UAF [47]) nesprendžia minėtos verslo ir IT suderinamumo problemos.

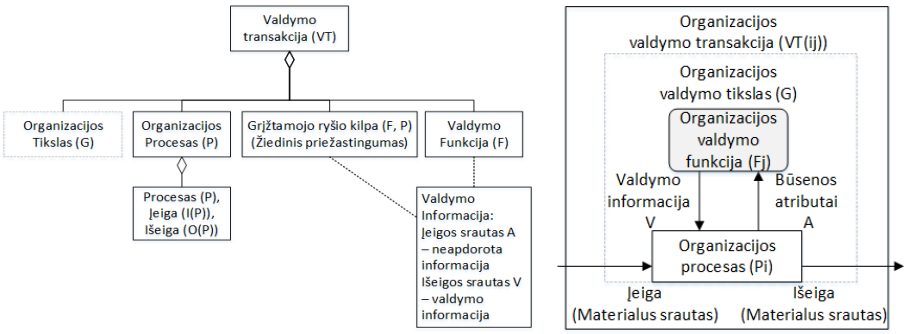
Akivaizdu, kad verslo procesų ir įmonės modeliavimo standartų paradigma keičiasi nuo išorinio modeliavimo į vidinį modeliavimą. Minėti metodai nėra pritaikyti naudoti populiariausiuose įmonių architektūros modeliavimo ar taikomųjų programų sistemų kūrimo valdymo įrankiuose, todėl darbas su labai dideliu informacijos kiekiu tenka žmogui (ekspertui). Tyrimo tikslas – sukurti efektyvų TPS projekto sprendimų derinimo su verslo strategija metodą ir įrankį, leidžiantį valdyti projekto eigą ir įvertinti TPS projekto būklę. Verslo analizės modelius dar reikia išversti į projekto reikalavimus arba pačius

reikalavimus koreguoti rankiniu būdu. Taip pat neatsižvelgiama į „Agile“ valdymo koncepcijas, todėl šioms problemoms spręsti reikia kitokio metodo.

## 1.2. „Agile“ veiklų priežastinis modeliavimas

Priežastingumas – svarbi sąvoka, žyminti būdingas domeno savybes (dėsningumus), paslėptas nuo išorinio stebėtojo. Vidinio modeliavimo paradigma – būtina sąlyga norint atrasti priežastinį ryšį įvairiose realaus pasaulio srityse ir sukurti pažangias (išmaniąsias, intelektualiąsias) visų tipų sistemas: fizines, įmonės, biologines sistemas ir kt. Vidinio modeliavimo paradigma reikalauja naujų modeliavimo metodų, leidžiančių atskleisti srities vidines (priežastines) priklausomybes. Atrastos srities priežastinės priklausomybės yra priežastinės žinios. Priežastinės žinios formaliai gali būti apibūdinamos kaip priežastinis modelis matematinėse išraiškose arba kaip konceptualus modelis (karkasas, metamodelis).

Šioje disertacijoje pateiktas tyrimas pagrįstas ankstesniais S. Gudo darbais [34, 49], kuriuose aprašyta įmonės valdymo priežastinio modeliavimo metodika. Pagrindinė „Agile“ veiklai modeliuoti skirta priežastinio modeliavimo koncepcija yra valdymo transakcija (VT) [34]. VT karkasas naudojamas kaip vientisas priežastinių žinių (giliųjų žinių) komponentas įmonės valdymo modeliui. VT yra svarbi tam tikram įmonės tikslui (G) ir fiksuoja žinias apie įmonės procesą (P), grįžtamojo ryšio kilpą (F, P), informacijos įvesties srautą (A), informacijos išvesties srautą (V) ir valdymo funkciją (F): VT = (F, P, A, V). Koncepcinė VT struktūra pateikiama 1 (a) paveiksle. Koncepcinis priežastinio ryšio modelis, pateikiamas 1 (b) paveiksle, atskleidžia vidinius VT žingsnius ir informacijos srautus, taip pat parodo grįžtamojo ryšio kilpą, kai informacija perduodama tarp F ir P.




(a) (b)

Šaltinis: [34]

1 pav. Valdymo transakcija: a) konceptuali struktūra, b) vidinis valdymo transakcijos modelis

„Agile“ taikomųjų programų sistemų kūrimo valdymas taip pat laikomas verslo valdymo metodų dalimi, nes jomis siekiama užtikrinti vertę klientams reguliariai tiekiant taikomąją programų sistemą. Taikomųjų programų sistemų kūrimo projektai yra skirti verslo strategijai įgyvendinti. Bėgant metams, „Agile“ metodai, tokie kaip „Scrum“, „Ekstremalus programavimas“ (XP), „ScrumBan“, kur „Scrum“ sujungiamas su „Kanban“ sistema, „Scaled Agile Framework“, „Spotify“ modelis arba hibridinės šių ir kitų metodų versijos, išpopuliarėjo dėl gebėjimo prisitaikyti prie pokyčių ir projekto išlaidų sumažinimo [83].

TPS projektai, valdomi naudojant „Agile“ metodus, turi skirtingus apibrėžimus, kurie atspindi užduočių hierarchiją TPS projekte, siekiant užtikrinti, kad kuriant taikomąją programinę sistemą būtų atsižvelgiama į verslo poreikius. „Agile“ programų sistemos projektų valdymo profesionalai ir tiekėjai naudoja įvairius hierarchinės struktūros elementų pavadinimus. Paprastai užduoties dydžiui apibrėžti naudojami keturi lygiai, o kiekvienas skirtingas lygis turi atskirą apibrėžimą. Hierarchijos lygiai nuo temų iki vartotojų istorijų taip pat žinomi kaip TIES, reiškiantys temas, iniciatyvas, epikus ir vartotojų istorijas [48, 84].

Daugelis projektų valdymo įrankių palaiko „Agile“ programų sistemų projektų valdymo procesą. „Atlassian Jira“ įrankis – vienas iš dažniausių [83]. Tačiau šiame ir kituose įrankiuose nenumatyti formalumai, kurie užtikrintų, kad ryšiai tarp skirtingų veiklų „Agile“ hierarchijos lygiuose būtų tinkamai nustatyti ir pagrįsti verslo kontekste. Tai reiškia, jog neužtikrinama, kad kiekviena vartotojo istorija, epas, iniciatyva ir tema būtų susieta su vienu iš strateginių verslo tikslų. Sąveikas nustato projektą vykdantys ekspertai.

Apskritai „Agile“ metodams ir valdymo įrankiams trūksta verslo konteksto apibrėžimo, todėl reikia kitokio požiūrio į esamą „Agile“ įrankių ir metodų struktūrą. Viena iš „Agile“ valdymo naudojant TIES metodą problemų – verslo strateginių tikslų nurodymo įgyvendinimo užtikrinimas, nes šio žingsnio stebėsenos nepalaiko „Agile“ valdymo įrankiai (kaip „Jira“). Strateginių tikslų įgyvendinimas priklauso nuo ekspertų, kurie analizuoja strateginių tikslų turinį ir nurodo aukščiausio lygio „Agile“ veiklas – temas.

Šios disertacijos kontekste išanalizuotas „Jira“ įrankis, siekiant nustatyti, kokios yra „Jira“ duomenų bazės sritys, skirtos TPS projektų projektavimo specifikacijoms ir projekto turiniui įvertinti pagal strateginius verslo tikslus. Nepaisant daugybės pritaikymo parinkčių, standartinėje „Jira“ sistemoje nėra konkrečių laukų, kuriuose būtų nurodytas vidinis domeno modelis, o turiniu pagrįsta TPS projekto specifikacija negali būti įvertinta. Be to, „Jira“

nenurodo jokios grįžtamojo ryšio specifikacijos, kuri yra privaloma priežastiniu ryšiu pagrįsto informacijos modeliavimo atveju.

### 1.4. Pirmosios dalies išvados

1. Šiuolaikiniuose „Agile“ metoduose stinga TPS projekto kūrimo veiklų sąveikos. Siekiant nustatyti realią TPS projekto būseną ir atitikimą strateginiams įmonės tikslams, reikia atlikti papildomą rankinį darbą.
2. Kyla poreikis sukurti modifikuotą „Agile“ valdymo metodą, kuris būtų pagrįstas priežastinio modeliavimo samprata, taip nustatant „Agile“ veiklų ir „Agile“ veiklų sąveikų turinį remiantis priežastinėmis žiniomis.

## 2. PRIEŽASTINGUMU PAGRĮSTAS „AGILE“ VALDYMO METODAS

„Agile“ projektų valdymo praktikos ir įrankiai nenustato struktūrizuotos ir formalios technikos ar modelių apibrėžti TIES hierarchijos elementų sąveikai. Verslo strategija išreiškiama ne formaliai, o dažniausiai tik žodžiu. Ji nėra nustatyta naudojant modeliavimo įrankius ir todėl negali būti tinkamai sujungta su aukščiausio lygio veikla „Agile“ hierarchijoje – tema.

MODAF karkase „gebėjimas“ – tai aukšto lygio organizacijos galimybių specifikacija [43]. Sulyginus verslo strategijos įgyvendinimo žingsnius su „Agile“ konceptais, sukuriama galimybė atlikti verslo strategijos transformaciją (1 lentelė) nuo strateginių tikslų iki konkrečių reikalavimų – vartotojo istorijų pavidalu.

1 lentelė. „Agile“ ir MODAF konceptų atitikmenys

| **„Agile“ konceptai** | **MODAF produktai ir konceptai** | |
|---|---|---|
| | **Elementai** | **Rodiniai** |
| Tema | Gebėjimas | StV-1<br>StV-2<br>StV-6 |
| Iniciatyva | Operatyvinis mazgas | StV-6 |
| Epikas | Operatyvinė veikla<br>Operatyvinės veiklos srautas | OV-2<br>OV-5 |
| Vartotojo istorija | Operatyvinė veikla<br>Operatyvinės veiklos vykdytojas<br>Operatyvinė rolė<br>Operatyvinės veiklos srautas | OV-5 |

Šaltinis: [106]

Iniciatyvos ir temos laikomos labai abstrakčiomis ir turi būti prilyginamos gebėjimo konceptui. MODAF karkaso gebėjimo koncepto pakanka apibrėžti neraiškią temų ir strateginių tikslų lygio informaciją. Atitikmenys pateikiamas 2-oje lentelėje.

2 lentelė. „Agile“ konceptų sulyginimas su VT ir MODAF konceptais

| **Statinis rodinys** | | **Dinaminis rodinys** |
|---|---|---|
| „Agile“ veiklų prilyginimas VT elementams | MODAF konceptų prilyginimas VT | VT elementai |
| Tema (T):<br>T := G; | StV konceptai:<br>Organizacijos fazė (EP),<br>Gebėjimas (C):<br>(EP, C) := (G) | Tikslas (G). |
| Iniciatyva (I):<br>I := VT (F, P, (A,V)); | OV konceptai:<br>Operatyvinis mazgas (N),<br>Operatyvinė veikla (O),<br>Operatyvinės veiklos srautas (AF):<br>(N, O, AF) := (F, P, (A,V)); | Tikslas (G).<br>Valdymo funkcija (F).<br>Procesas (P) – (P1, P2, ...,Pn).<br>Informaciniai srautai (A,V). |

Šaltinis: [106]

Atitikmenys 2-oje lentelėje parodo loginius skirtingų „Agile“ hierarchijos lygių ryšius.

„Agile“ veiklų hierarchija (temos, iniciatyvos, epikai, vartotojo istorijos) yra statinė ir neatvaizduoja sistemos valdymo dinamikos. Organizacijų informacinė architektūra taip pat statinė, tačiau ji detalizuoja „Agile“ veiklas. VT yra dinaminė struktūra, galinti atvaizduoti vidinę „Agile“ veiklų struktūrą ir sąveikas:

- VT atvaizduoja „Agile“ veiklų (temos, iniciatyvos, epikai, vartotojo istorijos) vidinę struktūrą, kai „Agile“ veiklos yra apibrėžtos kaip save valdančios veiklos;
- VT atvaizduoja vidinę gretimuose „Agile“ hierarchijos lygiuose esančių veiklų sąveiką vertikaliai (tema – iniciatyva, iniciatyva – epikas, epikas – vartotojo istorija), kai gretimai esančių „Agile“ hierarchijos lygių sąveikos yra laikomos save valdančiomis veiklomis. Šiuo atveju atskleidžiamas gretimai esančių „Agile“ hierarchijos lygių veiklų sąveikų turinys (grįžtamojo ryšio ciklas).
- VT atvaizduoja vidinę to paties „Agile“ hierarchijos lygio veiklų sąveikų struktūrą, jei to paties „Agile“ hierarchijos lygio veiklų sąveikos yra laikomos save valdančiomis. Tokiu atveju atskleidžiamas

to paties „Agile“ hierarchijos lygio veiklų sąveikų turinys (grįžtamojo ryšio ciklas).

Vertinant tipinę „Agile“ hierarchijos struktūrą ir remiantis priežastiniu modeliavimu, nustatyta:

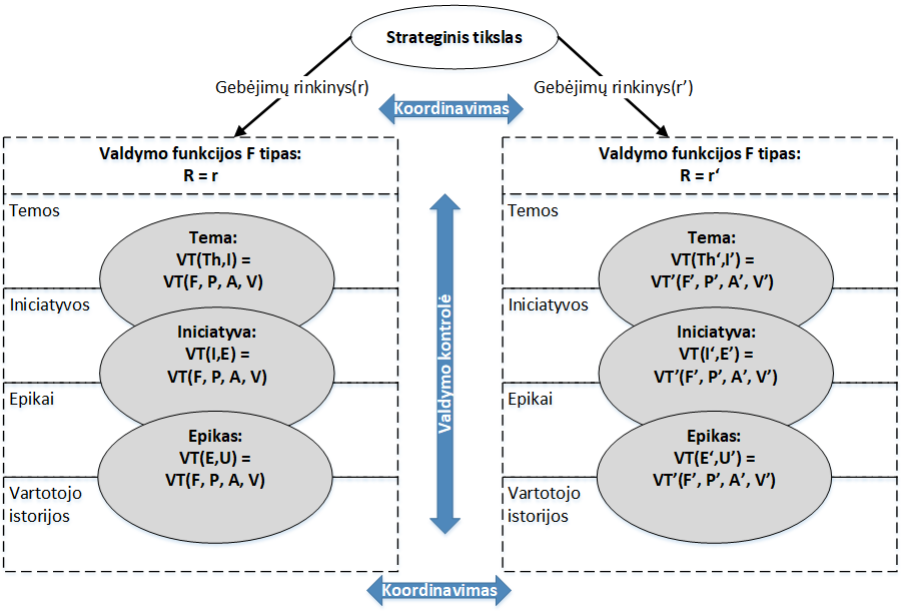
- strateginiai verslo tikslai ir skirtingų organizacijos valdymo funkcijų tipai (pvz., finansų, žmogiškųjų išteklių, taikomųjų programų sistemų kūrimo valdymas ir pan.) nėra identifikuojami ir specifikuojami TPS projektuose, todėl nukenčia projekto įgyvendinimo kokybė ir atitiktis organizacijos strateginiams tikslams;
- tarp bet kurių dviejų veiklų gretimuose lygiuose yra priežastinis ryšys, kai duomenys ir informacija perduodami tarp veiklų;
- sąveika tarp dviejų veiklų susijusiuose „Agile“ hierarchijos lygiuose yra sudėtingas procesas su grįžtamojo ryšio ciklu (žiedinis priežastingumas), kur bendrą „Agile“ veiklų sąveiką valdo ekspertas (vadovas, projektų vadovas, produkto savininkas arba programuotojas).

Apibendrinant, tipinis „Agile“ procesas yra labai mažai formalizuotas, nes yra taip sukurtas, o „Agile“ įrankių palaikymas užfiksuoti informacijai apie „Agile“veiklas yra ribotas. TPS vystymo projekto vykdymo kontrolė, suderinimas su strateginiais verslo tikslais ir „Agile“veiklų koordinavimas remiasi projekto ekspertų patirtimi, o tai yra sudėtingas ir klaidoms imlus procesas. Įvertinus šias realaus „Agile“ proceso charakteristikas ir remiantis priežastinio modeliavimo metodu, apibrėžiama modifikuota „Agile“ hierarchija.

Dėl šių priežasčių, remiantis priežastiniu modeliavimu, modifikuotoje Agile hierarchijoje turi būti įtraukta keletas dalykų:

- strateginių tikslų ir temų sąveikos specifikacija (T);
- identifikuoti skirtingi organizacijos valdymo funkcijų tipai, susiję su TPS vystymu (konkrečiai – taikomųjų programų sistemų vystymo valdymas bei verslo vystymo valdymas);
- dviejų gretimų „Agile“ hierarchijos lygių veiklų sąveika – sudėtingas procesas, apibrėžiamas kaip valdymo transakcija, į kurią įeina grįžtamojo ryšio ciklas tarp skirtingų hierarchijos lygio veiklų (apibrėžtas žiedinis priežastingumas). Informacinių srautų turinio specifikacija tarp temų, iniciatyvų, epikų, vartotojo istorijų yra normalizuota ir apima du poaibius – būsenos ir kontrolės atributus.

Įvertinus šiuos tipinio „Agile“ proceso trūkumus, apibrėžtas modifikuotas – priežastiniu modeliavimu grįstas – „Agile“ hierarchijos modelis, pateikiamas 2-ame paveiksle.




Šaltinis: sudaryta autoriaus pagal [50].

2 pav. Modifikuota „Agile“ veiklų hierarchija, kurioje veiklos apibrėžiamos kaip valdymo transakcijos (VT).

Svarbios toliau nurodytos modifikuotos (priežastinės) „Agile“ hierarchijos savybės:

1. Bet kuri „Agile“ veikla kiekviename „Agile“ hierarchijos lygyje q ∈ Q (tema, iniciatyva, epas, vartotojo istorija) priklauso kokiam nors valdymo funkcijos tipui (r ∈ R), kur R yra valdymo funkcijų tipų aibė. Organizacijos valdymo funkcijų tipai: finansų valdymas, žmogiškųjų išteklių valdymas, taikomųjų programų sistemų kūrimo valdymas ir pan.
2. Bet kuri „Agile“ veikla (temos, iniciatyvos, epai, vartotojo istorijos) kiekviename „Agile“ hierarchijos lygyje turi iš anksto nustatytą struktūrą, nes yra apibrėžta kaip valdymo transakcija VT = (F, P, A, V). Bet kuri „Agile“ veikla gali būti apibrėžta kaip F arba P, t. y. vaidinti F arba P vaidmenį valdymo transakcijoje remiantis „Agile“ veiklos lygiu modifikuotoje „Agile“ hierarchijoje.
3. Vertikalių bet kurių priežastinių „Agile“ veiklų VT(r) = (F(r), P, A, V) ir kokios nors kitos priežastinės „Agile“ veiklos VT′(r′)= (F′(r′), P′, A′, V′) sąveikų turinį galima atskleisti ir klasifikuoti analizuojant koordinavimo tipus W = (W1, W2, ..., Wn).
4. Horizontalių bet kurių „Agile“ veiklų VT(r) = (F(r), P, A, V) ir kokios nors kitos priežastinės „Agile“ veiklos VT′(r′)= (F′(r′), P′, A′, V′)

sąveikų turinį galima atskleisti ir klasifikuoti analizuojant koordinavimo tipus C = (C1, C2, ..., Cm).

5. „Agile“ hierarchijos lygyje q = „Temos“ kiekvienai valdymo funkcijai F, kuri yra r tipo:
   a) bet kuri tema (Th) yra apibrėžta kaip VTq(Th, I) = VTq(F(r), P, A, V), kur tema (Th) atlieka valdymo funkcijos vaidmenį (F), priežastingai susijusios iniciatyvos (I) atlieka proceso (P) vaidmenį, srautas A apibūdina P proceso būsenos atributų aibę (pvz., iniciatyvos būsenos duomenis), srautas V apibūdina proceso P kontrolių aibę (temos (Th) grįžtamąjį ryšį susijusioms iniciatyvoms (I).
   b) iniciatyva (I) apibūdina procesų aibę P = {P1, P2, ..., Pm}, kurie yra valdomi funkcijos F (vienos temos Th). Srautas A nustato būsenos atributų duomenų aibę A = {A1, A2, …, Am} su atitinkama susijusių procesų aibe P = {P1, P2, …, Pm}, kur A1 yra būsenos atributai procesui P1 (pvz., iniciatyvos I1 būsenos duomenys), A2 – būsenos atributai procesui P2 ir t. t.
   c) tema (Th atlieka F vaidmenį) atlieka kontrolių iniciatyvų aibę (aibė I = {I1, I2, ..., In}, kuri vaidina P vaidmenį) per grįžtamojo ryšio srautą V, siunčia kontrolės duomenis (nurodymus, reikalavimus), susijusius su atitinkamomis iniciatyvomis. Srautas V nustato kontrolių (grįžtamojo ryšio poveikių) aibę V = {V1, V2,…, Vn}, kur jos susietos su atitinkamais procesais P = {P1, P2,…, Pm}, o V1 yra valdymo kontrolės procesui P1 (poveikis iniciatyvai P1), V2 – valdymo kontrolės procesui P2 ir t. t.
   d) priežastinę sąveiką tarp temos ir iniciatyvos galima apibrėžti kaip valdymo transakcija formule (1), kur taip pat vertinami vertikalus sąveikos tipas (W) ir horizontalus sąveikos tipas (C):

$$VTq(r, (Th, I), C, W) = VTq\{(r, F(Th), P(I1), A1, V1, C, W), (r, F(Th), P(I2), A2, V2, C, W), ..., (r, F(Th), P(In), An, Vn, C, W)\} \quad (1)$$

kur:

- q – „Agile“ hierarchijos lygis;
- r – valdymo funkcijos tipas;
- VT(Th, I) – specifinė VT – tarp temos ir iniciatyvos (-ų);
- C – koordinavimo srauto tipas (horizontalus veiklų sąveikos tipas);
- W – valdymo kontrolės valdymo srauto tipas (vertikalus veiklų sąveikos tipas);

- F(Th) – veikla „tema“, atliekanti valdymo funkcijos vaidmenį F;
- {P(I1), ..., P(In)} – veiklų aibė, atliekanti „iniciatyvų“ procesų P vaidmenį;
- A – būsenos atributų A rinkinys = {A1, A2, ..., An};
- V – valdymo kontrolės V rinkinys = {V1, V2, ..., Vm}.

6. „Agile“ hierarchijos lygyje q = „Iniciatyvos“ kiekvienai valdymo funkcijai F, kuri yra r tipo,:
   a) bet kuri iniciatyva (I) yra apibrėžta kaip VTq(I, E) = VTq(F(r), P, A, V), kur iniciatyva (I) vaidina valdymo funkcijos vaidmenį (F), priežastingai susiję epai (E) vaidina proceso (P) vaidmenį, srautas A apibūdina P proceso būsenos atributų aibę, srautas V apibūdina proceso P kontrolių (iniciatyvos (I) grįžtamojo ryšio susijusiems epams (E) aibę.
   b) epas (E) apibūdina procesų aibę P = {P1, P2, ..., Pm}, kurie yra valdomi funkcijos F (vienos iniciatyvos I). Srautas A nustato būsenos atributų duomenų aibę A = {A1, A2, …,Am} su atitinkama susijusių procesų aibe P = {P1, P2, …, Pm}, kur A1 yra būsenos atributai procesui P1 (pvz., epo E1 būsenos duomenys), A2 – būsenos atributai procesui P2 ir t. t.
   c) iniciatyva (I atlieka F vaidmenį) vykdo kontrolių iniciatyvų (aibė E = {E1, E2, ..., En}, kuri atlieka P vaidmenį) aibę per grįžtamojo ryšio srautą V, siunčia kontrolės duomenis (nurodymus, reikalavimus), susijusius su atitinkamais epais. Srautas V nustato kontrolių (grįžtamojo ryšio poveikių) aibę V = {V1, V2, …, Vn}, susietų su atitinkamais procesais P = {P1, P2, …, Pm}, kur V1 yra proceso P1 valdymo kontrolės (poveikis epui E1), V2 – procesui P2 valdymo kontrolės ir t. t.
   d) priežastinę sąveiką tarp iniciatyvos ir epų galima apibrėžti kaip valdymo transakcija formule (2), kur taip pat vertinami vertikalus sąveikos tipas (W) ir horizontalus sąveikos tipas (C):

$$MTq(r, (I, E), C, W) = MTq\{(r, F(I), P(E1), A1, V1, C, W), (r, F(I), P(E2), A2, V2, C, W), ..., (r, F(I), P(En), An, Vn, C, W)\} \quad (2)$$

kur:

- q – „Agile“ hierarchijos lygis;
- r – valdymo funkcijos tipas;
- VT(I, E) – specifinė VT – tarp iniciatyvos ir epo (-ų);
- C – koordinavimo srauto tipas (horizontalus veiklų sąveikos tipas);

- W – valdymo kontrolės valdymo srauto tipas (vertikalus veiklų sąveikos tipas);
- F(Th) – veikla „iniciatyva“ valdymo funkcijos rolėje F;
- {P(E1), ..., P(En)} – veiklų „epai“ aibė, kuri vaidina procesų P vaidmenį;
- A – būsenos atributų A rinkinys = {A1, A2, ..., An};
- V – valdymo kontrolės V rinkinys = {V1, V2, ..., Vm}.

7. „Agile“ hierarchijos lygyje q = „Epai“ kiekvienai valdymo funkcijai F, kuri yra r tipo:
   a) bet kuris epas (E) yra apibrėžtas kaip VTq(I, E) = VTq(F(r), P, A, V), kur epas (E) vaidina valdymo funkcijos vaidmenį (F), priežastingai susijusios vartotojo istorijos (U) vaidina proceso (P) vaidmenį, srautas A apibūdina P proceso būsenos atributų aibę, srautas V apibūdina proceso P kontrolių (epo (E) grįžtamojo ryšio susijusioms vartotojo istorijoms (U) aibę.
   b) vartotojo istorija (U) apibūdina procesų aibę P = {P1, P2, ..., Pm}, kurie valdomi funkcijos F (vieno epo E). Srautas A nustato būsenos atributų duomenų aibę A = {A1, A2, …, Am} su atitinkama susijusių procesų aibe P = {P1, P2, …, Pm}, kur A1 yra būsenos atributai procesui P1 (pvz., vartotojo istorijos U1 būsenos duomenys), A2 – būsenos atributai procesui P2 ir t. t.
   c) epas (E vaidina F vaidmenį) vykdo vartotojo istorijų kontrolių (aibė U = {U1, U2, ..., Un} vaidina P vaidmenį) aibę per grįžtamojo ryšio srautą V, siunčia kontrolės duomenis (nurodymus, reikalavimus), susijusius su atitinkamomis vartotojo istorijomis. Srautas V nustato kontrolių (grįžtamojo ryšio poveikių) aibę V = {V1, V2, …, Vn}, susietų su atitinkamais procesais P = {P1, P2,…, Pm}, kur V1 yra proceso P1 (poveikis vartotojo istorijai U1) valdymo kontrolės, V2 – proceso P2 valdymo kontrolės ir t. t.
   d) priežastinę sąveiką tarp epo ir vartotojo istorijų galima apibrėžti kaip valdymo transakcija formule (3), kur taip pat vertinami vertikalus sąveikos tipas (W) ir horizontalus sąveikos tipas (C):

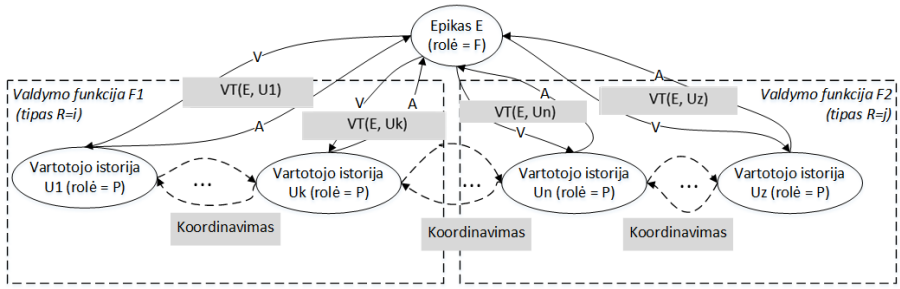
$$VTq(r, (E, U), C, W) = VTq\{(r, F(E), P(U1), A1, V1, C, W) (r, (F(E), P(U2), A2, V2, C, W), ..., (r, (F(E), P(Un), An, Vn, C, W)\} \quad (3)$$

kur:

- q – „Agile“ hierarchijos lygis;

- r – valdymo funkcijos tipas;
- VT(E, U) – specifinė VT – tarp epo ir vartotojo istorijų (-os);
- C – koordinavimo srauto tipas (horizontalus veiklų sąveikos tipas);
- W – valdymo kontrolės valdymo srauto tipas (vertikalus veiklų sąveikos tipas);
- F(Th) – veikla „epas“, vaidinanti valdymo funkcijos vaidmenį F;
- {P(E1), ..., P(En)} – veiklų „vartotojo istorijos“ aibė, vaidinanti procesų P vaidmenį;
- A – būsenos atributų A rinkinys = {A1, A2, ..., An};
- V – valdymo kontrolės V rinkinys = {V1, V2, ..., Vm}.

3-iame paveiksle pateikiama vertikalių sąveikų iliustracija modifikuotoje „Agile“ hierarchijoje. Epas (E) vaidina valdymo funkcijos F vaidmenį ir kontroliuoja vartotojo istorijų aibę (U = {U1, …, Uk, …, Un, …, Uz}). Visos vartotojo istorijos iš aibės {U} vaidina proceso P vaidmenį (t. y. visas vartotojo istorijas U kontroliuoja tas pats epas E). Kiekviena sąveika tarp epo E ir bet kurios iš susijusių vartotojo istorijų yra valdymo transakcija {MT(E, U1), ..., MT(E, Uk), ..., MT(E, Un), ..., MT(E, Uz)}. Vertikali sąveika modifikuotoje „Agile“ hierarchijoje, kur epas (E) vaidina F vaidmenį ir kontroliuoja aibę vartotojo istorijų (U = {U1, …, Uk, …, Un, …, Uz)) vadinama valdymo kontrole. Horizontali sąveika tarp vartotojo istorijų vadinama koordinavimu.




Šaltinis: sudaryta autoriaus.

3 pav. Modifikuota „Agile“ hierarchija: valdymo transakcijų pavyzdys tarp vieno epo (E), vaidinančio F vaidmenį, ir vartotojo istorijų (U) aibės, vaidinančios P vaidmenį.

8. „Agile“ hierarchijos lygyje q = „Vartotojo istorijos“ kiekvienai valdymo funkcijai F, kuri yra r tipo:
   a) bet kuri vartotojo istorija (U) yra apibrėžta kaip VTq(U, X) = VTq(F(r), P, A, V), kur vartotojo istorija (U) vaidina valdymo funkcijos vaidmenį (F), priežastingai susijusios veiklos (pvz., sub-

užduotys) (X) vaidina proceso (P) vaidmenį, srautas A apibūdina P proceso būsenos atributų aibę, srautas V apibūdina proceso P kontrolių (vartotojo istorijos (U) grįžtamojo ryšio susijusioms žemesnio lygio „Agile“ veikloms (X) aibę.

b) žemesnio lygio veikla (X) apibūdina procesų aibę P = {P1, P2, ..., Pm}, kuriuos valdo funkcija F (viena vartotojo istorija U). Srautas A nustato būsenos atributų duomenų aibę A = {A1, A2, …, Am} su atitinkama susijusių procesų aibe P = {P1, P2, …, Pm}, kur A1 yra būsenos atributai procesui P1, A2 – būsenos atributai procesui P2 ir t. t.
c) vartotojo istorija (U vaidina F vaidmenį) vykdo žemesnio lygio veiklų kontrolių aibę (aibė X = {X1, X2, ..., Xn} vaidina P vaidmenį) per grįžtamojo ryšio srautą V, siunčia kontrolės duomenis (nurodymus, reikalavimus), susijusius su atitinkamomis žemesnio lygio veiklomis. Srautas V nustato kontrolių (grįžtamojo ryšio poveikių) aibę V = {V1, V2,…, Vn}, susietų su atitinkamais procesais P = {P1, P2,…, Pm}, kur V1 yra proceso P1 valdymo kontrolės, V2 – proceso P2 valdymo kontrolės ir t. t.
d) priežastinę sąveiką tarp vartotojo istorijos ir žemesnio lygio „Agile“ veiklų galima apibrėžti kaip valdymo transakciją formule (4), kur taip pat vertinami vertikalus sąveikos tipas (W) ir horizontalus sąveikos tipas (C):

$$VTq(r, (U, X), C, W) = VTq\{(r, (F(U), P(X1), A1, V1, C, W), (r, (F(U), P(X2), A2, V2, C, W), ... (r, (F(U), P(Xn), An, Vn, C, W)\} \quad (4)$$

kur:

- q – „Agile“ hierarchijos lygis;
- r – valdymo funkcijos tipas;
- VT(U, X) – specifinė VT – tarp epo ir vartotojo istorijų (-os);
- C – koordinavimo srauto tipas (horizontalus veiklų sąveikos tipas);
- W – valdymo kontrolės valdymo srauto tipas (vertikalus veiklų sąveikos tipas);
- F(U) – veikla „vartotojo istorija“, atliekanti valdymo funkcijos vaidmenį F;
- {P(X1), ..., P(Xn)} – veiklų „vartotojo istorijos“ aibė, atliekanti procesų P vaidmenį;
- A – būsenos atributų A rinkinys = {A1, A2, ..., An};
- V – valdymo kontrolės V rinkinys = {V1, V2, ..., Vm}.

9. Bet kuri „Agile“ veikla bet kuriame „Agile“ hierarchijos lygyje laikoma baltąja dėžė tarpusavio (vertikaliose ir horizontaliose) sąveikų kontekste. Informacinis šių veiklų tarpusavio (koordinavimo) sąveikų turinys iš anksto apibrėžiamas vidine valdymo transakcijos VT (F, P, A, V) struktūra. Todėl sudaroma galimybė sukurti skirtingų „Agile“ veiklų sąveikų tipų (koordinavimo tipų) klasifikaciją.
10. Vertikalių ir horizontalių sąveikų tipų (koordinavimo tipų) klasifikacija sudaro sąlygas formaliai sąveikų turinio analizei.

## 2.1. Modelio verifikavimas ir validavimas

VT karkasas laikomas patvirtintu metodu fiksuoti probleminės srities priežastinėms žinioms [34, 49, 138]. Modelio verifikavimo apibrėžimas teigia, kad „modelio verifikavimas – procesas, kurio metu užtikrinama, kad kompiuterizuotas modelis pakankamai tiksliai atvaizduoja atitinkamą konceptualų modelį“ [139].

Modifikuotų „Agile“ veiklų sąveikos visiškai atitinka valdymo transakcijos – konceptualaus modelio – sudėtį, nes sąveikai specifikuoti būtini keturi privalomi VT karkaso atributai: srauto A būsenos atributai, srauto V kontrolės informacija, proceso P ir valdymo funkcijos F nurodymas. Toks modifikuotas „Agile“ modelis yra verfikuotas, nes visiškai atitinka priežastinių žinių modelio sudėtį.

Validavimas apibrėžia, ar elgesio modelis pakankamai tiksliai pagal „Agile“ produkto savininko poreikius užtikrina temų, iniciatyvų, epikų ir vartotojo istorijų ryšio vientisumą ir jų ryšį su strateginiais verslo tikslais.

Modelis pritaikytas 4-iuose TPS projektuose [134], rezultatus patikrino ekspertai. Gauti daug geresni TPS projektų vykdymo rezultatai.

## 2.2. Vystymo proceso būsenos įvertinimas

Vertikalių (valdymo kontrolė) ir horizontalių sąveikų (koordinavimas) poveikis priklauso nuo tarp „Agile“ veiklų perduodamo informacinių srautų turinio.

Visa perduodama „Agile“ veiklų ir jų sąveikų informacija turi atitikti VT = (F, P, A, V) struktūrą, o tai reiškia, kad visi atributai privalo būti specifikuoti ir perduodami.

Formaliai apibrėžta horizontalių ir vertikalių sąveikų klasifikacija (atitinkamai lentelės 17 ir 18 pagrindiniame tekste) sudaro prielaidas sukurti TPS projekto turinio sudėtingumo indikatorius.

1 prielaida: viena VT perduoda vieną srautą V kitai, žemesnio lygio VT* vertikaliai – vyksta valdymo kontrolė. Remiantis šia prielaida, tarp gretimų skirtingų „Agile“ veiklų modifikuotoje Agile hierarchijoje yra 15 perduodamų vertikalių sąveikų tipų (18 lentelė).

Remiantis šia prielaida, horizontalios sąveikos (koordinavimo) atveju perduodama taip pat iki 15 perduodamų horizontalių sąveikų tipų (17 lentelė).

Remdamiesi šiomis prielaidomis, panagrinėkime pavyzdį, kai:

- viena iniciatyva I yra susijusi tik su viena tema Th;
- vienas epikas E yra susijęs tik su viena iniciatyva I;
- viena vartotojo istorija U yra susijusi tik su vienu epik E.

Tipiniame TPS projekte vidutiniškai vykdoma „Agile“ veiklų:

- 1 tema, kuri atitinka kokį nors strateginį tikslą;
- 1 iniciatyva (I) temai;
- 10 epikų (E) vienai temai;
- 24 vartotojo istorijos (U) vienam epikui.

Skaičiuodami galimas vertikalias sąveikas minėtame pavyzdyje, gauname projekto sąveikų kiekį (5) – Q1(v):

$$Q1(v) = 1 \times 15 + 10 \times 15 + 10 \times 240 \times 15 = 36165 \quad (5)$$

Skaičiuodami galimas horizontalias sąveikas minėtame pavyzdyje, gauname projekto sąveikų kiekį (6) Q1(h):

$$Q1(h) = 10 \times 15 + 10 \times 24 \times 15 = 3750 \quad (6)$$

2 prielaida: kiekvienas vidinis valdymo transakcijos elementas (F, P, A, V) perduoda informacinį srautą kiekvienam kitam kitos VT* elementui (F*, P*, A*, V*) – t. y. iš viso galima perduoti 225 informacijos srauto elementus tarp dviejų „Agile“ veiklų. Remiantis minėtu pavydžiu ir 2 prielaida, sąveikų skaičius tarp „Agile“ veiklų pateikiamas taip (7):

$$Q2(h) = 10 \times 225 + 10 \times 24 \times 225 = 56250 \quad (7)$$

Vertikalių sąveikų skaičius gaunamas (8):

$$Q2(v) = 1 \times 225 + 10 \times 225 + 10 \times 240 \times 225 = 542475 \quad (8)$$

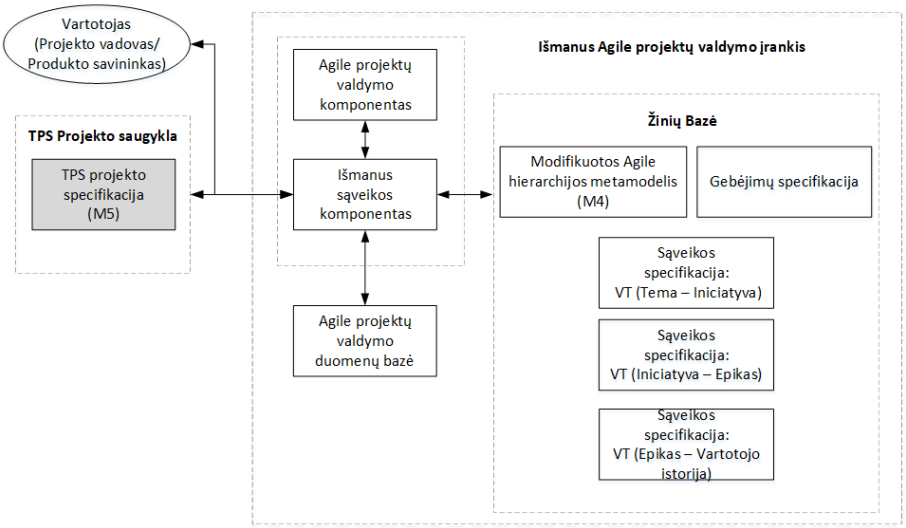
Laikant šiuos tipinio „Agile“ projekto skaičius normatyvu (atskaitos tašku), galima apskaičiuoti kitų projektų sudėtingumo indikatorius. Detalūs skaičiavimai pateikiami 20-toje ir 21-oje lentelėse.

### 2.3. Antrosios dalies išvados

1. „Agile“ veiklų modeliavimas priežastinio modeliavimu principu atskleidė sudėtingą TPS projektų valdymo prigimtį. VT struktūra pasiteisino kaip naudingas metodas apibrėžti „Agile“ veiklų turinį. Sukurta modifikuota „Agile“ veiklų hierarchija įvertinant sąveiką tarp dviejų gretimų „Agile“ lygių veiklų kaip grįžtamojo ryšio ciklą.
2. Pateikti TPS projekto sudėtingumo vertinimo kriterijai sudaro sąlygas vertinti TPS projekto veiklų būseną priežastinio „Agile“ modelio pagrindu ir atsižvelgiant į organizacijos strateginius tikslus.

## 3. PATOBULINTA „AGILE“ VALDYMO ĮRANKIO ARCHITEKTŪRA

Norint išmaniai plėtoti TPS ir užtikrinti, kad kiekviena TPS vystymo projekto užduotis prisidėtų prie strateginių verslo tikslų, būtina naudoti iš anksto nustatytas žinias. Vadovaujantis modifikuotos „Agile“ hierarchijos metodu, pereinama nuo pilkosios dėžės požiūrio prie baltosios dėžės požiūrio, modeliuojant domeno žinias (pateikiant specifikaciją domeno žinioms užfiksuoti per VT karkasą). Patobulinto „Agile“ projektų valdymo įrankio architektūra pateikiama 4-tame paveiksle:




Šaltinis: [106]

4 pav. Patobulinta „Agile“ projektų valdymo įrankio architektūra.

„Vartotojas“ 4 pav. reiškia „Agile“ TPS projekto vadovą. Jis sąveikauja su patobulintu „Agile“ projektų valdymo įrankiu (t. y. „Jira“ ar bet kuriuo kitu, kurį patobulintų DI komponentas). Patobulintas „Agile“ projektų valdymo įrankis apima šiuos komponentus:

- „Agile“ projektų valdymo komponentas“ – įprastos funkcijos, kurias apima „Agile“ projektų valdymo įrankiai, t. y. produkto užduočių sąrašas, sprinto užduočių sąrašas, įvairūs atributai, ataskaitos ir kt.
- „Išmanus sąveikos komponentas“ – patobulinimas, kurį siūloma pridėti prie įprastų „Agile“ projektų valdymo įrankių. Jis naudoja realaus laiko informaciją iš „Agile“ projektų valdymo duomenų bazės (informaciją apie TIES elementus, pvz., užduočių aprašymus ir kt.). „Pažangios sąveikos komponentas“ sąveikauja su žinių baze.
- TPS projekto specifikacijos, esančios elemente „TPS projekto saugykla“, apima stebėjimu pagrįstus verslo domeno modelius (BPMN, DMN ar kita notacija) ir TPS projekto specifikacijas (UML, SysML ar kita notacija), kurie naudojami TPS vystyti.
- „Žinių bazė“ (ŽB) – priežastinių žinių sistemų (metamodelių) saugykla, naudojama TPS kūrimo sprendimams validuoti: modifikuotas „Agile“ hierarchijos metamodelis, gebėjimų specifikacijos, VT pagrįstos vertikalios sąveikos specifikacijos „Agile“ sistemoje.

### 3.1. Trečiosios dalies išvados

1. Dabartiniai „Agile“ projektų valdymo įrankiai, kaip „Atlassian Jira“, nepateikia gairių, kaip nustatyti teisingus ryšius tarp skirtingų lygių „Agile“ valdymo hierarchijos veiklų.
2. Remiantis patobulintu TPS projektų valdymo įrankiu sugeneruotomis ataskaitomis, ekspertai galėtų pateikti trūkstamus „Agile“ veiklų ryšius ir turinį. Ataskaita taip pat parodytų, kurios TPS projekto veiklos nekuria vertės strateginiams verslo tikslams, todėl neturėtų būti atliekamos.

## 4. EKSPERIMENTINIS VERTINIMAS

Tyrimo problema ištirta indukciniu metodu. Vykdant IT projektus, tarp panašių taikomųjų programų sistemų projektų pastebėta pasikartojanti tendencija, kad vėlyvame projekto pristatymo ciklo etape jiems paprastai keliami reikalavimai, pagrįsti organizacijos strategijos įgyvendinimu (pvz., organizacijos tikslas: „Būti pirmaujančia įmone pagal kliento patirties vertinimą“), o tai paverčiama keliais prioritetais arba gebėjimais (pvz.,

„Interaktyvus skaitmeninės patirties pojūtis“). Sukurtas metodas, kuris leidžia nustatyti informacijos, gautos iš organizacijos strategijos per organizacijos informacinės architektūros karkasą, nesutapimą su informacija, kurią turi projektą vykdanti TPS kūrimo komanda, išnaudodama organizacinius gebėjimus ir vykdydama verslo strategiją. Remiantis sukurtu metodu, atlikta atvejų analizė, kad patikrinti metodo efektyvumą.

Atlikti 4-ių taikomųjų programų sistemų vystymo projektų stebėjimai. Projektai vykdomi taikant „Scrum“ karkasą komandų lygiu ir „Scaled Agile“ karkasą (SAF’e) [21] organizacijos lygiu. Informacija apie projekto reikalavimus surinkta iš 4 skirtingų šiaurės Europos šalių, vykdančių vieną kritinį organizacijos procesą kiekvienam projektui. Kiekvienos šalies detaliame procese yra nedidelių nukrypimų, o įmonės taikomosios programų sistemos projektas taip pat turi pašalinti šiuos nukrypimus galutinėje sprendimo versijoje – t. y. suderinti procesą tarp skirtingų šalių. Be to, iš įmonės perspektyvos, kaip „SAF’e“ dalis, turi būti sprendžiamos ir suderintos priklausomybės tarp kitų komandų, vykdančių projektą toje pačioje srityje.

Reikalavimai, pateikti temų, iniciatyvų, epų, vartotojų istorijų, pakeitimų užklausų ir klaidų formatu, buvo analizuojami projekto gyvavimo ciklu nuo 8 iki 60 mėnesių. Stebėjimų ir analizės rezultatai pateikti 3-ioje lentelėje.

3 lentelė. Pradinis taikomosios programų sistemos projektų reikalavimų pasiskirstymas

| **Parametras** | **Projektas Nr. 1** | **Projektas Nr. 2** | **Projektas Nr. 3** | **Projektas Nr. 4** |
|---|---|---|---|---|
| Reikalavimai, iš kurių | 138 | 224 | 236 | 2539 |
| Strateginis tikslas | Nenurodyta | Nenurodyta | Nenurodyta | Nenurodyta |
| -temos | 0 (nenurodyta) | 0 (nenurodyta) | 0 (nenurodyta) | 4 |
| -iniciatyvos | 1 | 2 | 2 | 10 |
| -epikai | 9 | 15 | 17 | 314 |
| -vartotojo istorijos | 128 | 207 | 217 | 2211 |
| Pakeitimų prašymai | 273 | 173 | 36 | 1341 |
| Klaidos | 135 | 252 | 304 | 824 |
| Projekto trukmė | 8 mėn. | 12 mėn. | 10 mėn. | 60 mėn. |

Šaltinis: sudaryta autoriaus

Taikant siūlomą metodą ir siekiant spręsti informacijos trūkumą verslo strategijos vykdymo ir taikomosios programų sistemos kūrimo projektų metu,

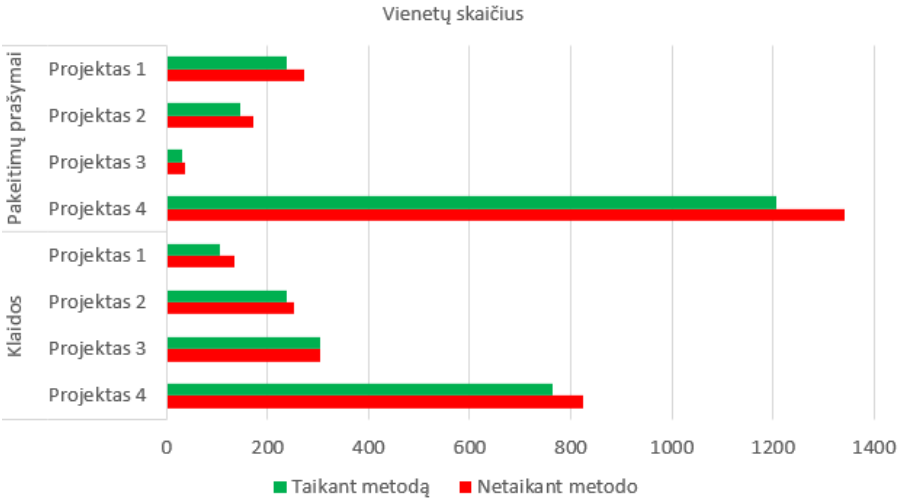
išanalizuoti projektų reikalavimai, pakeitimų prašymai ir klaidos, o išvados pateikiamos 4-oje lentelėje.

4 lentelė. Įmonės taikomosios programų sistemos projektų reikalavimų pasiskirstymas naudojant siūlomą metodą

| **Parametras** | **Projektas Nr. 1** | **Projektas Nr. 2** | **Projektas Nr. 3** | **Projektas Nr. 4** |
|---|---|---|---|---|
| Reikalavimai, iš kurių | 121 | 181 | 216 | 2244 |
| Strateginis tikslas | Tapti pirmuoju asmeninės bankininkystės pasirinkimu Skandinavijos šalyse | Tapti antruoju pagal verslo bankininkystės rinkos dalį Skandinavijos šalyse | Tapti antruoju pagal verslo bankininkystės rinkos dalį Skandinavijos šalyse | Užtikrinti šiuolaikišką savitarnos patirtį klientams |
| -temos | 1 | 1 | 1 | 4 |
| -iniciatyvos | 1 | 2 | 2 | 10 |
| -epikai | 8 | 13 | 15 | 292 |
| -vartotojo istorijos | 111 | 165 | 198 | 1938 |
| Pakeitimų prašymai | 238 | 146 | 36 | 1208 |
| Klaidos | 107 | 238 | 304 | 765 |
| Projekto trukmė | 8 mėn. | 11 mėn. | 10 mėn. | 54 mėn. |

Šaltinis: sudaryta autoriaus

Pradiniai TIES hierarchijos reikalavimų skaičiai pasikeitė, nes buvo vykdomas mažesnis skaičius epų ir vartotojo istorijų, kad būtų užbaigti TPS vystymo projektai. Taip pat pakeitimų prašymų ir klaidų kategorijų skirtumai buvo ganėtinai ryškūs, nes pirmame projekte klaidų skaičius sumažėjo daugiau kaip 20 %, o trečiame projekte pakeitimų prašymų skaičius sumažėjo daugiau kaip 16 %. Palyginimas pateiktas 5-tame paveiksle bei 5-toje lentelėje.




Šaltinis: sudaryta autoriaus

5 pav. Eksperimentinio vertinimo rezultatai

5 lentelė. Įmonės taikomosios programų sistemos projektų reikalavimų pasiskirstymo palyginimas

| **Parametras, %** | **Projektas Nr. 1** | **Projektas Nr. 2** | **Projektas Nr. 3** | **Projektas Nr. 4** |
|---|---|---|---|---|
| Reikalavimai | -12,3 | -19,2 | -8,5 | -11,6 |
| Pakeitimų prašymai | -12,8 | -15,6 | -16,7 | -9,9 |
| Klaidos | -20,7 | -5,6 | 0 | -7,2 |
| Projekto trukmė | 0 mėnesių | -1 mėnuo | 0 mėnesių | -6 mėnesiai |

Šaltinis: sudaryta autoriaus

Pastebėtas sutaupymas turėtų būti įvertintas atsižvelgiant į reikalaujamas plėtros pastangas, o sutaupytas išlaidas galima apskaičiuoti pagal vidutines vystytojo (programuotojo) valandines sąnaudas.

# IŠVADOS

1. „Agile“ programų sistemų vystymo projektų valdymo metodai ir susiję „Agile“ įrankiai turi reikšmingų trūkumų ir negali užtikrinti efektyvių TPS projektų rezultatų ir atitikties vykdomai verslo strategijai:
   1.1. tipinis „Agile“ procesas mažai formalizuotas, todėl „Agile“ įrankiais negalima užfiksuoti pakankamai formalizuotos informacijos apie „Agile“ veiklas;
   1.2. nėra nustatytos skirtingos veiklos valdymo funkcijos;
   1.3. TPS projekto vykdymo kontrolė, suderinimas su strateginiais verslo tikslais ir „Agile“ veiklų koordinavimas visiškai priklauso nuo projekto vykdymo ekspertų ir yra imlus klaidoms procesas;
   1.4. nėra formaliai apibrėžta vidinė „Agile“ veiklų struktūra;
   1.5. nėra formaliai standartizuota vertikalių ir horizontalių sąveikų specifikacija „Agile“ hierarchijoje;
   1.6. nepaisant plačių „Agile“ įrankių pritaikymo galimybių, „Agile“ projektų valdymo įrankiuose tarp priklausomų „Agile“ veiklų fiksuojamas tik pats ryšio faktas, tačiau nenurodomas ryšio turinys. Tai apsunkina TPS projekto būsenos analizės algoritmų kūrimą ir vertinimą, ypač suderinimą su verslo strategija.
2. Pritaikius priežastinio modeliavimo metodą TPS projektams valdyti „Agile“ aplinkoje išanalizuota tipinė „Agile“ veiklų hierarchija (temos, iniciatyvos, epai ir vartotojo istorijos). Nustatyta, kad:
   2.1. TPS projektuose nėra nustatomi strateginiai verslo tikslai ir skirtingos verslo valdymo funkcijos, todėl nukenčia projektų įgyvendinimo ir suderinimo su strateginiais verslo tikslais kokybė;
   2.2. tarp dviejų gretimai esančių „Agile“ hierarchijos lygių veiklų susidaro priežastinis ryšys, kai tarp veiklų perduodami duomenys ir informacija;
   2.3. dviejų gretimuose „Agile“ hierarchijos lygiuose esančių veiklų sąveika yra sudėtingas procesas su grįžtamojo ryšio ciklu (žiedinis priežastingumas), kai bendrą „Agile“ veiklų sąveiką valdo ekspertas (vadovas, projektų vadovas, produkto savininkas arba programuotojas).
3. Sukurtas priežastingumu grįstas „Agile“ projektų valdymo modelis (metodas) naudojant priežastinio modeliavimo valdymo transakcijos (VT) karkasą bei MODAF organizacijos architektūros karkaso gebėjimo sąvoką. Kiekviena „Agile“ veikla ir „Agile“ veiklų sąveika laikoma valdymo transakcija (VT). Taikant VT konceptą nustatyti skirtingi „Agile“ veiklų

valdymo ir koordinavimo tipai. Tai leido formalizuoti „Agile“ veiklų turinio analizę.

4. Koordinavimo ir valdymo tipai sudaro modifikuoto „Agile“ hierarchijos modelio taksonomiją (15 ir 16 lentelės pagrindiniame tekste).
5. Specifikuota žinių bazė, modifikuotu „Agile“ proceso modeliu paremta projektų valdymo įrankio architektūra. „Jira“ įrankio duomenų baze paremta žinių bazė sukuria pagrindą TPS projekto būsenai vertinti.
6. Apibrėžti horizontalaus koordinavimo ir vertikalaus valdymo sąveikų atvejai leidžia įvertinti „Agile“ veiklų būseną (atitikimą normalizuotam modifikuotam „Agile“ modeliui) ir apibrėžti TPS projektų sudėtingumo indikatorius.
7. Eksperimentinio vertinimo rezultatai parodė, kad modifikuotas priežastinis „Agile“ metodas, skirtas užtikrinti verslo ir IT suderinimą „Agile“ aplinkoje vertinant TIES „Agile“ hierarchijos veiklas, pakeitimų prašymus ir klaidas, duoda beveik 13 % geresnius TPS projektų vykdymo rezultatus. Pagerinamas verslo ir IT suderinamumas, identifikuojami „Agile“ hierarchijos veiklų ir įmonės strateginių tikslų neatitikimai, o trūkstamą informaciją gali užpildyti projektų ekspertai.
8. Nepaisant reikšmingų sukurto modifikuoto priežastiniu modeliavimu pagrįsto „Agile“ projektų valdymo metodo, esama tam tikrų apribojimų: organizacija turėtų būti bent 3 brandos lygio pagal CMMI procesų brandos vertinimo modelį.

**Karolis Noreika** 2015 m. baigė Vilniaus universiteto verslo informatikos studijų programą ir gavo informacijos sistemų magistro laipsnį. 2019 m. jis įstojo į informatikos inžinerijos doktorantūros studijas. Karolio tyrimo sritis apima IT projektų valdymą Agile aplinkoje, efektyvaus organizacijų valdymo metodus ir priemones. Doktorantas aktyviai dalyvauja Agile bendruomenės Lietuvoje veikloje, privačiai konsultuoja įmones bei skaito paskaitas įvairiomis temomis. Jis yra sertifikuotas produkto savininkas (CSPO), Scrum meistras (CSM) ir LeSS praktikas.

# LIST OF PUBLICATIONS

**Published papers on the topic of the dissertation:**

1. Gudas, S., Noreika, K: Causal Interactions in Agile Application Development. Mathematics. Basel: MDPI AG. eISSN 2227-7390. 2022, vol. 10, no. 9, art. no. 1497, p. [1-22]. DOI: 10.3390/math10091497. IF: 2.592 as of year 2021.
2. Noreika, K., Gudas, S.: Causal Knowledge Modelling for Agile Development of Enterprise Application Systems. Informatica. Vilnius University. eISSN 2227-7390. 2023, p. [1-26]. DOI: 10.15388/23-INFOR510. IF: 3.3 as of year 2023.

**Dissertation results presented at scientific conferences as conference proceedings:**

1. Gudas, S., Noreika, K.: Aligning Agile Application Software Development With the Archimate Framework. In: 11th International workshop on data analysis methods for software systems (DAMSS 2019), 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 Press, 2019. ISBN 9786090703243. eISBN 9786090703250.
2. Noreika, K.: Business Capabilities Utilization Enhancement Using Archimate for EAS Projects Delivery in an Agile Environment. In: Matulevičius, R., Robal, T., Haav, H-M., Maigre, R., Petlenkov, E. (eds.). In: Joint Proceedings of Baltic DB&IS 2020 Conference Forum and Doctoral Consortium co-located with the 14th International Baltic Conference on Data-bases and Information Systems (BalticDB&IS 2020) CEUR Workshop proceedings, vol. 2620, Estonia, pp. 49–56 (2020).
3. Noreika, K.: Improving enterprise application software development management with MODAF. In: Joint proceedings of the BIR 2021 workshops and doctoral consortium co-located with 20th international conference on perspectives in business informatics research (BIR 2021), Vienna, Austria, September 22-24, pp. 141–152 (2021).
4. Noreika, K., Gudas, S.: Modelling the Alignment Between Agile Application Development and Business Strategies. In: Joint proceedings of the BIR 2021 workshops and doctoral consortium co-located with 20th international conference on perspectives in business informatics research (BIR 2021), Vienna, Austria, September 22-24, pp. 59–73 (2021).

5. Gudas, S., Noreika, K.: Causal Interactions of the Agile Activities in Application Development Management. In: DAMSS: 12th Conference on data analysis methods for software systems. Druskininkai, Lithuania, December 2–4, 2021. Vilnius : Vilnius University Press, 2021. ISBN 9786090706732. eISBN 9786090706749.

**Dissertation results presented at scientific conferences as part of books:**

1. Noreika, K., Gudas, S: Using Management Transaction Concept to Ensure Business and EAS Alignment in an Agile Environment. In Information and Software Technologies ICIST 2021. Communications in Computer and Information Science, 1st ed.; Lopata, A., Gudonienė, D., Butkienė, R. (eds.); Springer, Cham: Switzerland, 2021; Volume 1486, pp. 109–120.

NOTES

Karolis Noreika

Application Software Development Process Using an Enhanced Agile Project Management Method

DOCTORAL DISSERTATION
Technological sciences,
Informatics Engineering (T 007)
Thesis Editor Zuzana Šiušaitė

Karolis Noreika

Taikomųjų programų kūrimo procesas naudojant modifikuotą Agile projektų valdymo metodą

DAKTARO DISERTACIJA
Technologijos mokslai
Informatikos inžinerija (T 007)
Santraukos redaktorė Jorūnė Rimeisytė-Nekrašienė

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