

VILNIUS GEDIMINAS TECHNICAL UNIVERSITY

Šarūnė SIELSKAITĖ

RESEARCH ON THE IMPACT OF
HUMAN FACTORS ON SOFTWARE
DEVELOPMENT PROCESSES

DOCTORAL DISSERTATION

TECHNOLOGICAL SCIENCES,
INFORMATICS ENGINEERING (T 007)

Vilnius, 2026

The doctoral dissertation was prepared at Vilnius Gediminas Technical University in 2020–2026.

Supervisor

Prof. Dr Diana KALIBATIENĖ (Vilnius Gediminas Technical University, Informatics Engineering – T 007).

The Dissertation Defence Council of the Scientific Field of Informatics Engineering of Vilnius Gediminas Technical University:

Chairman

Prof. Dr Nikolaj GORANIN (Vilnius Gediminas Technical University, Informatics Engineering – T 007).

Members:

Dr Habil. Piotr Lech ARTIEMJEW (University of Warmia and Mazury in Olsztyn, Poland, Informatics Engineering – T 007),

Prof. Dr Rimantas BUTLERIS (Kaunas University of Technology, Informatics Engineering – T 007),

Prof. Dr Habil. Gintautas DZEMYDA (Vilnius University, Informatics Engineering – T 007),

Prof. Dr Dmitrij ŠEŠOK (Vilnius Gediminas Technical University, Informatics Engineering – T 007).

The dissertation will be defended at the public meeting of the Dissertation Defense Council of the Scientific Field of Informatics Engineering in the *Aula Doctoralis* Meeting Hall of Vilnius Gediminas Technical University at **2 p.m. on 21 May 2026**.

Address: Saulėtekio al. 11, LT-10223 Vilnius, Lithuania.

Tel.: +370 5 274 4956; fax +370 5 270 0112; e-mail: doktor@vilniustech.lt

A notification on the intended defense of the dissertation was sent on 20 April 2026. A copy of the doctoral dissertation is available for review at the Vilnius Gediminas Technical University repository <https://etalpykla.vilniustech.lt> and at the Library of Vilnius Gediminas Technical University (Saulėtekio al. 14, LT-10223 Vilnius, Lithuania) and the Library of Kaunas University of Technology (K. Donelaičio g. 20, LT-44239 Kaunas, Lithuania).

Vilnius Gediminas Technical University book No. 2026-021-M

<https://doi.org/10.20334/2026-021-M>

© Vilnius Gediminas Technical University, 2026

© Šarūnė Sielskaitė, 2026

sarune.sielskaite@vilniustech.lt

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

Šarūnė SIELSKAITĖ

ŽMOGIŠKŲJŲ VEIKSNIŲ POVEIKIO
PROGRAMŲ SISTEMŲ KŪRIMO
PROCESAMS TYRIMAS

DAKTARO DISERTACIJA

TECHNOLOGIJOS MOKSLAI,
INFORMATIKOS INŽINERIJA (T 007)

Vilnius, 2026

Disertacija rengta 2020–2026 metais Vilniaus Gedimino technikos universitete.

Vadovas

prof. dr. Diana KALIBATIENĖ (Vilniaus Gedimino technikos universitetas, Informatikos inžinerija – T 007).

Vilniaus Gedimino technikos universiteto Informatikos inžinerijos mokslo krypties disertacijos gynimo taryba:

Pirmininkas

prof. dr. Nikolaj GORANIN (Vilniaus Gedimino technikos universitetas, Informatikos inžinerija – T 007).

Nariai:

habil. dr. Piotr Lech ARTIEMJEW (Varmijos ir Mazūrijos universitetas Olštynė, Lenkija, Informatikos inžinerija – T 007),

prof. dr. Rimantas BUTLERIS (Kauno technologijos universitetas, Informatikos inžinerija – T 007),

prof. habil. dr. Gintautas DZEMYDA (Vilniaus universitetas, Informatikos inžinerija – T 007),

prof. dr. Dmitrij ŠEŠOK (Vilniaus Gedimino technikos universitetas, Informatikos inžinerija – T 007).

Disertacija bus ginama viešame Informatikos inžinerijos mokslo krypties disertacijos gynimo tarybos posėdyje **2026 m. gegužės 21 d. 14 val.** Vilniaus Gedimino technikos universiteto *Aula Doctoralis* posėdžių salėje.

Adresas: Saulėtekio al. 11, LT-10223 Vilnius, Lietuva.

Tel.: (0 5) 274 4956; faksas (0 5) 270 0112; el. paštas doktor@vilniustech.lt

Pranešimai apie numatomą ginti disertaciją išsiųsti 2026 m. balandžio 20 d. Disertaciją galima peržiūrėti Vilniaus Gedimino technikos universiteto talpykloje <https://etalpykla.vilniustech.lt/> ir Vilniaus Gedimino technikos universiteto bibliotekoje (Saulėtekio al. 14, LT-10223 Vilnius, Lietuva) ir Kauno technologijos universiteto bibliotekoje (K. Donelaičio g. 20, LT-44239 Kaunas, Lietuva).

Abstract

The software development process (SDP) is a complex and multifaceted system of interrelated activities influenced by numerous variables, including the development methodologies and the human factor (HF), which play a central role. While methodologies, such as AGILE and WATERFALL, define structured approaches to software development, the actual trajectory, duration, and outcomes of the SDP are often significantly shaped by human-related factors. As a knowledge-intensive and dynamic process, SDP depends heavily on human expertise, collaboration, and decision-making. The human factor encompasses a wide range of behavioral, cognitive, and social dimensions, including individual skills, motivation, and team interactions, which introduce variability and uncertainty into the process. Due to its abstract nature and the difficulty of quantification, HF remains a subject of ongoing academic and practical research aimed at identifying the key human-centric determinants that influence the success of software development. Consequently, a deeper understanding and effective management of these factors are essential for achieving favorable project outcomes and advancing the maturity of software engineering practices.

This research introduces a novel approach to examining the influence of HF on the SDP, offering a comprehensive perspective through the lens of distinct software development methodologies. The proposed approach incorporates several innovative elements, including the application of fuzzification techniques to model HF uncertainties within SDP activities. By capturing the variability and unpredictability of human behavior, this approach allows for a more nuanced representation of HF impact. Additionally, the proposed approach uses a case-handling paradigm to model and simulate different SDP instances from real-world SDP scenarios, further enabling their dynamic and case-based analysis. To ensure the robustness and relevance of the findings, real HF-related data were collected from multiple IT organizations, providing a solid empirical foundation for the study.

The findings of this research reveal notable differences in how HF influences SDP performance across the WATERFALL and AGILE methodologies. These contrasting approaches result in varying degrees of HF impact on project timelines, quality, and risk levels. The dissertation's results not only contribute to a deeper understanding of HF's role in SDP but also provide valuable insights for researchers and practitioners involved in software development projects.

By elucidating the relationship between HF and different SDP methodologies, this research equips stakeholders with the knowledge needed to assess and mitigate software development risks.

Reziუმэ

Programinэs iрrangos kūrimo procesas yra sudэtinga tarpusavyje susijusių veiklų sistema, kurią lemia įvairūs kintamieji, ypač kūrimo metodologija ir žmogiškasis faktorius. Nors tokios metodologijos kaip AGILE ar WATERFALL modelis apibrэžia struktūruotus kūrimo principus, proceso eigą, trukmę ir rezultatus dažnai reikšmingai keičia žmogiškasis faktorius. Kaip dinamiškas ir žiniomis grįstas procesas, programinэs iрrangos kūrimas priklauso nuo žmonių kompetencijos, bendradarbiavimo ir sprendimų priэмimo. Žmogiškasis faktorius apima elgsenos, kognityvinius ir socialinius aspektus, įskaitant įgūdžius, motyvaciją ir komandos sąveikas, kurie įneša kintamumo ir neapibrэžtumo bei apsunkina proceso prognozavimą ir valdymą. Dėl sudэtingo apibrэžimo ir riboto išmatuojamumo žmogiškasis faktorius išlieka aktyvių tyrimų objektu, o jo supratimas ir valdymas yra būtinas siekiant sėkmingų projektų rezultatų ir programinэs inžinerijos brandos.

Šiame tyrime pristatomas naujas požiūris į žmogiškojo faktoriaus įtakos programinэs iрrangos kūrimo proceso analizę, leidžiantis kompleksiskai vertinti šią įtaką skirtingų programinэs iрrangos kūrimo paradigмų kontekste. Siūlomas metodas apima keletą inovatyvių elementų, įskaitant neraiškiųjų aibių logikos (angl. *fuzzy logic*) taikymą, siekiant modeliuoti žmogiškojo faktoriaus neapibrэžtumą programinэs iрrangos kūrimo proceso veiklose. Taip pavyksta tiksliau atspindėti žmogiškojo elgesio kintamumą ir nenuspėjamumą. Be to, taikomas atvejo analizės (angl. *case-handling*) principu pagrįstas modeliavimas ir simuliacija, leidžiantis tirti skirtingus programinэs iрrangos kūrimo proceso scenarijus, paremtais realiais duomenimis. Siekiant užtikrinti tyrimo patikimumą ir aktualumą, buvo surinkti empiriniai žmogiškojo faktoriaus duomenys iš kelių IT organizacijų.

Gauti rezultatai atskleidžia esminius skirtumus tarp AGILE ir WATERFALL paradigмų atžvilgiu to, kaip žmogiškasis faktorius veikia programinэs iрrangos kūrimo proceso efektyvumą. Šios paradigmos lemia skirtingą žmogiškojo faktoriaus įtakos mastą projekto trukmei, kokybei ir rizikos lygiui. Disertacijos išvados ne tik praplečia supratimą apie žmogiškojo faktoriaus vaidmenį programinэs iрrangos kūrimo procesui, bet ir suteikia vertingų įžvalgų tyrėjams bei praktikams, dirbantiems programinэs iрrangos kūrimo srityje.

Atskleisdami žmogiškojo faktoriaus ir skirtingų programinэs iрrangos kūrimo procesų paradigмų sąveiką, tyrimo rezultatai suteikia suinteresuotiesiems asmenims žinių, reikalingų programinэs iрrangos kūrimo rizikoms įvertinti ir mažinti.

Notations

Symbols

a_i, b_i – the coefficients of linear Takagi–Sugeno consequents (liet. *linijinių Takagi–Sugeno išvadų koeficientai*);

c_1, c_2, c_3, c_4 – adaptive parameters using trapezoidal MFs (liet. *adaptyvūs parametrai, naudojant trapezoidines priklausomybės funkcijas*);

R^2 – the coefficient of determination (liet. *determinacijos koeficientas*);

μ^{Trap} – trapezoidal membership function (liet. *trapezinė priklausomybės funkcija*);

μ^{Triang} – triangular membership function (liet. *trikampė priklausomybės funkcija*);

A – fuzzy set (liet. *neaiškioji aibė*);

$Task_i$ – all assignments that should be completed (liet. *visos užduotys, kurias reikia atlikti*);

H_i – actual time for task execution (liet. *faktinis užduoties atlikimo laikas*);

HF_i – human factor of a task executor (liet. *užduoties vykdytojo žmogiškasis veiksnys*);

SDP_i^j – software development process (liet. *programinės įrangos kūrimo procesas*);

\bar{x} – sample mean (liet. *imties vidurkis*);

μ – population mean (liet. *populiacijos vidurkis*);

s – sample standard deviation (liet. *imties standartinis nuokrypis*);

n – sample size (liet. *imties dydis*);

μ_{withHF} – sample with HF (liet. *imtis su žmogiškuoju veiksnium*);

$\mu_{\text{withoutHF}}$ – sample without HF (liet. *imtis be žmogiškojo veiksnio*).

Abbreviations

ABMS – Agent-Based Modeling and Simulation (liet. *agentais parentas modeliavimas ir simuliacija*);

AI – Artificial Intelligence (liet. *dirbtinis intelektas*)

ANFIS – Adaptive Neuro-Fuzzy Inference System (liet. *adaptivi neuro-neaiškioji išvadų sistema*);

ANN – Artificial Neural Networks (liet. *dirbtiniai neuroniniai tinklai*)

ATF – Automated Testing Frameworks (liet. *automatizuoto testavimo karkasai*);

BPMN – Business Process Modeling and Notation (liet. *verslo procesų modeliavimas ir notacija*);

CMMM – Case Management Modeling and Notation (liet. *atvejų valdymo modeliavimas ir notacija*);

DICE – Duration, Integrity, Commitment, Effort Framework (liet. *trukmės, integralumo, įsipareigojimo ir pastangų sistema*);

FIS – Fuzzy Inference System (liet. *neaiškioji išvadų sistema*);

HF – Human Factor (liet. *žmogiškasis veiksnys*);

IT – Information Technology (liet. *informacinės technologijos*);

MF – Membership Function (liet. *priklausomybės funkcija*);

MSE – Mean Squared Error (liet. *vidutinė kvadratinė paklaida*);

RMSE – Root Mean Squared Error (liet. *šakninė vidutinė kvadratinė paklaida*);

SDLC – Software Development Lifecycle (liet. *programinės įrangos kūrimo gyvavimo ciklas*);

SDM – Software Development Methodology (liet. *programinės įrangos kūrimo metodologija*);

SDP – Software Development Process (liet. *programinės įrangos kūrimo procesas*);

SSE – Squares Due to Error (liet. *paklaidų kvadratų suma*).

Contents

INTRODUCTION	1
Problem Formulation.....	1
Relevance of the Dissertation.....	1
Research Object.....	2
Aim of the Dissertation	2
Tasks of the Dissertation	2
Research Methodology.....	3
Scientific Novelty of the Dissertation	3
Practical Value of the Research Findings.....	3
Defended Statements.....	4
Approval of the Research Findings	5
Structure of the Dissertation.....	5
1. LITERATURE REVIEW ON THE SOFTWARE DEVELOPMENT PROCESS AND HUMAN FACTOR	7
1.1. Software Development Process Modeling Using Case-Based Modeling.....	8
1.1.1. Concept of a Software Development Process.....	8
1.1.2. Case-Based Modeling for the Software Development Process	9
1.2. Role of Human Factors in the Software Development Processes	10

1.2.1. Human Factor Concept.....	10
1.2.2. Significance of Human Factors in the Software Development Process.....	12
1.2.3. Influence and Relevance of Human Factors in Multiple Disciplines	20
1.2.4. Human Factor Modeling in the Software Development Process	23
1.2.5. Human Factor Challenges and Mitigation Strategies	25
1.3. Simulating the Software Development Process.....	28
1.3.1. Concept of Simulation of Complex Processes.....	28
1.3.2. Software Development Process Simulation.....	32
1.3.3. Fuzzy-Based Software Development Process Simulation	33
1.3.4. Applications of Fuzzy Sets in the Software Development Process	37
1.4. Conclusions of the First Chapter and Formulation of the Dissertation Tasks ...	39
2. FUZZY SET THEORY AND CASE-HANDLING BASED DYNAMIC SOFTWARE DEVELOPMENT PROCESS MODELING AND SIMULATION APPROACH	41
2.1. General Fuzzy and Case-Handling Based Dynamic Software Development Process Modeling and Simulation Approach Description	42
2.2. Collecting Real Data	43
2.3. Preprocessing Real Data.....	45
2.4. Developing the Software Development Process Models.....	47
2.4.1. Developing AGILE Software Development Process Model	47
2.4.2. Developing the WATERFALL Software Development Process Model	49
2.5. Human Factor Fuzzy Inference Using Adaptive Neuro-Fuzzy Inference System	51
2.6. Developing the Software Development Process Simulation Model.....	53
2.7. Performing Simulation Experiments	55
2.8. Systematizing and Analyzing Simulation Results.....	56
2.9. Conclusions of the Second Chapter.....	58
3. EXPERIMENTAL INVESTIGATION OF HUMAN FACTOR IMPACT ON SOFTWARE DEVELOPMENT PROCESS	59
3.1. Planning the Experiment	60
3.2. Experimental Investigation.....	61
3.2.1. Experiment Data.....	62
3.2.2. Modeling the Software Development Process with Case Management Model and Notation	66
3.2.3. Adaptive Neuro-Fuzzy Inference System in Software Development Process Simulation	70
3.2.4. Software Development Process Simulation Results	76
3.3. Statistical Analysis of Software Development Process Simulation Results	82
3.4. Conclusions of the Third Chapter.....	84

GENERAL CONCLUSIONS 87

REFERENCES 89

LIST OF SCIENTIFIC PUBLICATIONS BY THE AUTHOR ON THE TOPIC OF
THE DISSERTATION 99

SUMMARY IN LITHUANIAN..... 101

ANNEXES..... 115

 ANNEX A. Survey Questionnaire and Data Collection Template..... 116

Introduction

Problem Formulation

The software development process (SDP) is influenced by methodologies and the human factor (HF), which can significantly affect project outcomes despite structured approaches, such as AGILE and WATERFALL (Grenning, 2001; Andrei et al., 2019). Due to its complexity and uncertainty, HF remains difficult to define and measure and is widely studied (Dutra et al., 2021; Gunatilake et al., 2024). Its impact varies across SDPs, highlighting the need to analyze HF within different methodologies. Given the inherent uncertainty of HF and the limited use of real-world data in existing studies, this dissertation addresses the research question of how to fuzzify HF uncertainties and apply case-handling modeling techniques for SDP simulation using real-world HF data from IT organizations.

Relevance of the Dissertation

The relevance of this dissertation lies in addressing the underexplored impact of the human factor (HF) on the software development process (SDP). Although methodologies, such as AGILE and WATERFALL, provide structured approaches, human behavior introduces variability that significantly affects project

outcomes and contributes to persistent failure rates (Standish Group, 2021; Barros et al., 2024; Ockiya & Lock, 2023). Existing methodologies define roles, processes, and interactions but do not adequately capture how HF influences SDP performance (Andrei et al., 2019). Moreover, there remains a lack of comprehensive understanding of HF effects across different SDP frameworks (Andrei et al., 2019; Gunatilake et al., 2024).

Given that HF affects all stages of SDP through communication, decision-making, and adaptability, its variability introduces uncertainty and risks that are difficult to manage (Dutra et al., 2021). To address this gap, this dissertation proposes modeling HF uncertainties using fuzzy logic and simulating SDP performance with real-world data, providing valuable insights for both research and practice, particularly for IT organizations operating in dynamic environments.

Research Object

The research object is the influence of human factors on the performance of the software development process in AGILE and WATERFALL methodologies.

Aim of the Dissertation

The dissertation aims to enhance Software Development planning by extending its modeling and simulation process through the integration of HF.

Tasks of the Dissertation

The following tasks had to be solved to achieve the aim:

1. To perform a literature review on the SDP modeling and simulation with HF incorporation in it.
2. To develop an approach for predicting HF impact on SDP.
3. To gather data from real IT organizations regarding the impact of HF on SDPs, based on real-world IT project insights.
4. To implement the proposed approach and to investigate it experimentally for the suitability of predicting HF impact on SDP with real data.

Research Methodology

To examine the research object, a range of methods was applied. Information gathering, systematization, analysis, and comparison were used to explore and present insights from the fields of fuzzy inference and machine learning-based prediction. Logical reasoning, concept generalization, and conceptual modeling were employed to evaluate existing methods and develop a fuzzy inference and machine learning-based approach for predicting HF impact on the SDP. Conceptual modeling, implementation, experimentation, and statistical analysis were then used to assess the proposed approach's accuracy and feasibility. The experiment was carried out using real-world data.

Scientific Novelty of the Dissertation

The unique contribution and scientific novelty of the dissertation are as follows:

1. A proposed novel method for predicting HF impact on SDP integrates fuzzy logic with case-based simulation. The developed Case Management Model and Notation (CMMN) simulation model enables the simulation of processes with varying inputs and allows for comparison of the same SDP simulated under different process management paradigms, specifically AGILE and WATERFALL. This method offers a unique contribution by combining HF modeling with SDP simulations, facilitating more informed decision-making in software development.
2. Empirical SDP data were obtained from Lithuanian IT software development projects to ensure a more realistic, contextually grounded validation of the proposed method. By using real-world project information, this study helps bridge the divide between theoretical assumptions and the practical implementation of HF considerations within SDP.

These contributions aim to enrich the understanding of HF impacts across different development approaches, providing valuable insights for academic research and practical applications in software development.

Practical Value of the Research Findings

The practical value of this research lies in its potential to inform software development practices by providing actionable insights into how HF impacts different software development methodologies (SDMs). By examining the specific effects of HF within WATERFALL and AGILE SDPs, this study provides data-driven

recommendations that development teams can use to optimize their workflows, improve team efficiency, and reduce human-related risks:

1. **HF importance:** The HF variables revealed during the in-depth literature review inform SDP practitioners about variables that influence SDP more and should be considered most.
2. **Enhancing development model selection:** Project managers and development teams will be better equipped to choose between AGILE and WATERFALL models by considering the HF and its compatibility with team dynamics and project characteristics.
3. **Improving team performance:** By understanding how HF influences specific stages in the SDP, organizations can implement targeted training, resource allocation, and support measures to mitigate common HF-related issues, such as miscommunication, task-prioritization conflicts, and workload imbalances.
4. **Risk management:** Real-world data used in the dissertation allows companies to identify critical points in their processes where HF may pose risks, especially in large, complex projects. This insight will enable better forecasting of potential challenges, allowing for proactive measures to minimize disruptions.
5. **Supporting process optimization:** The comparative analysis offers guidelines on how adjusting task sequencing or incorporating flexibility into SDPs could help organizations maximize productivity and adapt to HF-driven variations in team performance.

By providing empirically grounded insights, this research supports software development teams in creating more resilient, efficient, and human-centered processes, ultimately contributing to higher-quality software products and enhanced project outcomes.

Defended Statements

The following statements, based on the results of the present investigation, may serve as the official hypotheses to be defended:

1. To obtain more accurate modeling and simulation of SDP. Not only should structural well-known characteristics be modeled and considered, but also an HF characteristic, which presents SDP uncertainties and can be modeled by applying the fuzzy set theory.
2. The proposed ANFIS-based HF modeling approach, using a real-world dataset and providing a robust framework for SDP modeling and simulation, enables accurate prediction of HF's impact on the

SDP. By integrating HF impact estimation with case-based simulation, the approach enables more precise prediction of software development outcomes, including project costs and timelines.

Approval of the Research Findings

The dissertation results were published in two scientific publications in refereed journals in the *Clarivate Analytics Web of Science* database. One publication was issued in the Q2 (2021) quartile (Sielskaitė & Kalibatienė, 2023) journal, and one publication was issued in the Q3 quartile (Sielskaitė & Kalibatienė, 2025) journal. Results were also disseminated in peer-reviewed conference proceedings, including Sielskaitė (2022) in the Baltic DB&IS 2022, Sielskaitė and Kalibatienė (2021) in ISD 2021, and Sielskaitė (2021) in the IEEE eStream conference.

Results were also presented at two scientific conferences in Lithuania and two abroad:

- 15th International Baltic Conference on Digital Business and Intelligent Systems (Baltic DB & IS) 2022, Riga, Latvia.
- Open Conference of Electrical, Electronic, and Information Sciences, eStream 2021, Vilnius, Lithuania.
- 12th conference on Data Analysis Methods for Software Systems (DAMSS) 2021, Druskininkai, Lithuania.
- 29th International Conference on Information Systems Development (ISD2021) 2021, Valencia, Spain.

Structure of the Dissertation

The dissertation consists of an introduction, three main chapters, general conclusions, references, a list of the author's publications on the topic of the dissertation, and a summary in Lithuanian. It includes 115 pages, 25 tables, 37 figures, and 176 references.

1

Literature Review on the Software Development Process and Human Factor

This chapter presents a literature review on the SDP and the HF, aiming to establish the theoretical foundation for the research. It examines key SDP methodologies, such as AGILE and WATERFALL, and analyzes how human-related aspects influence development processes and outcomes. Additionally, the chapter reviews existing studies on HF in software engineering, identifying research gaps and motivating the need for further investigation. The results of the First Chapter were disseminated in two scientific publications in peer-reviewed journals indexed in the Clarivate Analytics Web of Science database, including one Q2 quartile journal (Sielskaitė & Kalibatienė, 2023) and one Q3 quartile journal (Sielskaitė & Kalibatienė, 2025). In addition, the research findings were presented at four scientific conferences, two held in Lithuania and two international events, namely the International Baltic Conference on Digital Business and Intelligent Systems (2022), the Open Conference of Electrical, Electronic and Information Sciences eStream (2021), the Data Analysis Methods for Software Systems conference (2021), and the International Conference on Information Systems Development (2021).

1.1. Software Development Process Modeling Using Case-Based Modeling

1.1.1. Concept of a Software Development Process

The SDP is a structured approach to designing, building, testing, and maintaining software applications (Pressman & Maxim, 2020). The process begins with gathering and documenting user and business requirements, followed by designing the software architecture and selecting appropriate technologies. During implementation, developers write and integrate code while adhering to coding standards. Testing is then conducted to identify and resolve defects, ensuring software reliability and security. After successful deployment, the software enters the maintenance phase, where updates, optimizations, and bug fixes are handled. The SDP ensures that software products meet user expectations, remain scalable, and adapt to technological advancements, making it a fundamental part of modern software engineering.

The SDP consists of several activities that ensure the successful design, development, and deployment of a software system. These activities follow the Software development life cycle (SDLC) and are common through various development methodologies (like WATERFALL, AGILE, etc.) (Sommerville, 2016; Kaur & Sengupta, 2020). Main activities in SDP are requirements analysis, system design, implementation, testing, and maintenance (Sommerville, 2016).

WATERFALL and AGILE represent two fundamentally different approaches to software project management and development. The WATERFALL model follows a sequential and highly structured process in which project phases, such as requirements analysis, design, implementation, testing, and deployment, are executed in a predefined order, with limited flexibility for changes once a phase has been completed. In contrast, AGILE methodologies emphasize iterative development, continuous feedback, and adaptability, allowing requirements and solutions to evolve throughout the project lifecycle (Dikert et al., 2016; Thesing et al., 2021). AGILE approaches typically involve short development cycles, close collaboration among team members and stakeholders, and frequent delivery of incremental results. Despite these differences, both approaches aim to ensure effective project management and product quality, and in practice, they may share certain principles and practices or be combined in hybrid forms depending on project characteristics and organizational context (Palmquist et al., 2013).

SDP models define the structured approach to designing, developing, testing, and deploying software. These models help manage software projects by providing guidelines, workflows, and best practices. Different models can be chosen based on project requirements, complexity, flexibility needs, and costs.

1.1.2. Case-Based Modeling for the Software Development Process

Case-based modeling in software development is a structured approach that leverages past experiences, historical project data, and real-world scenarios to guide decision-making, improve efficiency, and optimize development workflows (Aamodt & Plaza, 1994; Van der Aalst et al., 2005). By analyzing previous software projects, including their challenges, solutions, and outcomes, this method enables teams to identify patterns, anticipate potential risks, and apply proven strategies to new projects. It is particularly valuable in managing uncertainties, allowing developers to reference past cases when making critical decisions, such as selecting the most suitable development methodology, estimating project timelines, or allocating resources effectively.

Since software development is inherently a process, various modeling approaches can be applied to structure and execute its components effectively. Some of the most widely recognized and utilized approaches include workflow-based modeling (Dumas et al., 2018; Li et al., 2024), case-handling (Van der Aalst & Kumar, 2012), rule-based modeling (Jensen & Kristensen, 2009), variability models (Czarnecki & Eisenecker, 2000), document-based approaches (Grossmann & Leymann, 2006), and policy-based (Kumar & Singh, 2016) modeling. Workflow-based modeling defines a structured sequence of tasks aimed at achieving a business objective, such as processing purchase orders or provisioning services, where tasks follow a predefined order with clear dependencies. In contrast, case-handling focuses on managing individual cases, allowing for greater flexibility by emphasizing the data and goals of each case rather than a fixed task sequence (Malone, 2011), which is especially useful in knowledge-driven domains, such as healthcare or legal services. Rule-based modeling governs system behavior through conditional rules that trigger specific actions when certain criteria are met, making it ideal for applications requiring complex business logic or automated decision-making. Variability models support the systematic management of differences and commonalities in software product lines, enabling the configuration of systems tailored to specific needs or contexts. Often used in administrative or compliance-heavy domains, document-based approaches treat documents as central entities within processes, modeling their creation, transformation, and flow between stakeholders. Lastly, policy-based modeling involves defining high-level policies that guide system behavior and decision-making, often used in domains like network management or access control, where behavior must align with organizational or regulatory constraints. Together, these approaches provide a diverse toolkit for capturing the dynamic, multifaceted nature of software processes. Workflows are commonly defined as a structured sequence of tasks designed to accomplish a business process (Dumas et al., 2018), such as processing purchase orders or provisioning services. A workflow is defined as a collection of tasks

organized to accomplish some business process (e.g., processing purchase orders over the phone or provisioning telephone service). Case-handling is useful for software development process modeling because it provides flexibility to manage complex, data-driven tasks where the sequence of activities may vary depending on the specific context and evolving state of each case.

Guo et al. (2025) demonstrated the growing relevance of case-based reasoning in modern software development by integrating it with large language models for functional test script generation. By combining case-based reasoning with large language model capabilities, the system enhances automation and adaptability, enabling more efficient handling of complex and evolving testing requirements. The study highlights that such hybrid approaches improve the reuse of prior knowledge while reducing manual effort in test development, illustrating how case-based modeling can support continuous learning and optimization within the SDP.

Case-based modeling, as one of these approaches, is especially useful in domains where historical data and previous successful solutions can inform current and future decisions (Aamodt & Plaza, 1994). This is particularly evident in expert systems (Jackson, 1999), AI-driven applications (Russell & Norvig, 2021), and business process automation (Davenport, 2013). By studying real-world cases, developers can recognize recurring patterns, refine system requirements based on validated methodologies, and enhance software quality through the reuse of proven design structures. Furthermore, integrating machine learning and artificial intelligence can enhance case-based modeling by automating case retrieval, adaptation, and continuous learning from new experiences (Ye et al., 2021). As a result, organizations that adopt case-based modeling can reduce errors, improve project predictability, and drive more efficient and informed decision-making throughout the software development lifecycle.

1.2. Role of Human Factors in the Software Development Processes

1.2.1. Human Factor Concept

The HF concept encompasses various psychological, cognitive, social, and organizational elements that influence human behavior and performance in different domains, including software development. HF is a multidisciplinary concept that originates from fields such as psychology, ergonomics, and human-computer interaction, aiming to understand how human abilities, limitations, and behaviors impact processes and outcomes (Wickens et al., 2004; Salvendy, 2012). In the

context of software development, HF refers to the characteristics, skills, and interactions of individuals and teams that contribute to the success or failure of projects (Carayon, 2006). It includes aspects such as experience, education, motivation, teamwork, communication, decision-making, and problem-solving capabilities. Researchers classify HF into different categories, such as cognitive factors (e.g., memory, attention, and reasoning), social factors (e.g., collaboration, leadership, and communication), and organizational factors (e.g., team structure, work environment, and resource management) (Carayon, 2006; Mahaju et al., 2023).

Gunatilake et al. (2024) analyzed the role of HF in software engineering by examining how human aspects influence interactions between software developers and end users. Their systematic literature review highlights that effective communication, collaboration, and mutual understanding are critical for successful software development outcomes. The authors emphasized that HFs, such as individual skills, experience, cognitive biases, and social dynamics, directly affect requirements elicitation, system usability, and user satisfaction. By addressing these HFs, organizations can improve stakeholder engagement, enhance requirement accuracy, and ultimately develop software systems that better meet user needs.

Kolus, Wells, and Neumann (2023) conceptualized HF as a multidimensional construct encompassing interactions among human capabilities, limitations, and the surrounding work system. In their study, HFs are defined not only in terms of individual attributes such as physical, cognitive, and psychological characteristics but also as elements shaped by task design, organizational structures, and environmental conditions. The authors emphasized that HF represents a systemic perspective, in which human performance emerges from the dynamic interplay between people, technology, and processes. This view highlights that variability in human behavior is inherent and must be accounted for in system design and analysis. Consequently, the HF concept is positioned as a critical component for understanding and managing system performance, particularly in complex socio-technical environments.

Understanding HF is essential for optimizing productivity, efficiency, and quality in software development, as it helps identify the key human-related challenges and opportunities within the process. By studying and managing HF effectively, organizations can create more cohesive teams, improve software quality, reduce human errors, and enhance overall project success.

1.2.2. Significance of Human Factors in the Software Development Process

HF plays a critical role in SDP, significantly influencing various aspects, such as productivity, quality, performance, and testing. Unlike purely technical components, the human element in software development is dynamic and multifaceted, encompassing personal attributes, skills, teamwork, communication, and decision-making processes. Researchers have extensively studied HF's impact on SDP, revealing that factors such as experience, education, cognitive abilities, social behavior, and teamwork directly affect software quality and overall project success. Due to the complexity and collaborative nature of software development, understanding and managing HF is essential to optimizing the SDP and achieving better outcomes (Wagner & Ruhe, 2018).

One of the primary ways for HF to affect SDP is through its influence on software development methodologies. Traditional and modern software development frameworks, such as WATERFALL and AGILE, rely heavily on human collaboration and expertise (Cockburn & Highsmith, 2001; Muhammad et al., 2023). However, AGILE methodologies emphasize human involvement, making HF one of the most significant contributors to project success. AGILE approaches, such as Scrum, rely on effective teamwork, communication, and decision-making, where factors like motivation, collaboration, and adaptability are crucial. Studies have shown that when teams exhibit strong cohesion, commitment, and engagement, software development projects tend to be more efficient, with fewer defects and faster delivery times. Conversely, a lack of coordination, poor communication, and insufficient experience can lead to project delays, cost overruns, and software failures (Grenning, 2001).

The HF also plays a vital role in determining software quality (Gueveyi et al., 2020). Researchers found that experience and education are key contributors to high-quality software products. Developers with more industry experience and higher levels of education tend to produce better-structured, more maintainable code, reducing technical debt and long-term project costs. Moreover, cognitive abilities and problem-solving skills directly impact a developer's ability to identify and resolve issues efficiently. In contrast, inexperienced or inadequately trained developers may struggle with complex tasks, increasing the likelihood of software bugs and security vulnerabilities. Additionally, the organizational and operational aspects of HF, such as team dynamics, leadership, and resource management, significantly influence software quality and project efficiency.

Another crucial area of research regarding HF in SDP is its impact on productivity (Gonçalves et al., 2017). Studies suggest that HFs, such as motivation, collaboration, and teamwork, have a direct correlation with software development productivity. Developers who are highly motivated and engaged in their work tend

to produce better results, whereas those who experience burnout or lack of motivation may struggle to meet deadlines and quality standards. Team collaboration is another critical factor in productivity: projects that foster open communication, knowledge sharing, and teamwork tend to be more successful than those with isolated or uncoordinated efforts. Additionally, researchers have explored how virtual team working, human resource management, and social interaction affect the efficiency of software development teams, particularly in remote or distributed work environments.

Performance is another key area influenced by HF in SDP. Since software development is a human-centered process (Pirzadeh, 2010), the effectiveness of a development team depends on individual and collective performance. Factors such as problem-solving capabilities, adaptability, communication skills, and teamwork all contribute to project success. Some researchers argue that different personality types may be better suited for specific software engineering roles, emphasizing the need to assess and allocate human resources effectively. For instance, developers with strong analytical and logical thinking skills may excel in backend development, while those with creative problem-solving abilities may be better suited for UI/UX design or front-end development.

Furthermore, HF has a significant impact on software testing, which is a crucial phase of SDP (Itkonen et al., 2012). Testing requires meticulous attention to detail, analytical skills, and strong communication between development and quality assurance teams. Studies have shown that human errors in testing processes can lead to critical defects that impact the final product. Effective testing teams rely on structured methodologies, teamwork, and experience to identify and resolve software issues efficiently. Additionally, communication between developers and testers is essential to ensure that bugs are correctly documented, understood, and addressed.

Despite the varied perspectives from which researchers analyze HF in SDP, there is a consensus that HF directly impacts software project outcomes. Studies suggest that managing HF effectively can lead to significant improvements in productivity, quality, and overall project success. To optimize the SDP, organizations should focus on fostering a positive work environment, promoting collaboration, investing in employee education and training, and implementing effective communication strategies. By understanding and addressing HF, companies can create a more efficient, innovative, and successful software development process.

Future research should continue to explore key HFs, such as motivation, collaboration, experience, education, communication, and commitment. Understanding how these factors interact with software development methodologies, team dynamics, and project management strategies will be essential for improving SDP outcomes. As software development continues to evolve with advancements in artificial intelligence, remote work, and automation, the role of HF will remain a

fundamental aspect of building high-quality, reliable, and innovative software solutions.

HF is essential in the SDP, affecting key areas like software quality, productivity, collaboration, and project management. Researchers, such as Ruiz and Salanitri (2019), emphasized the critical impact of human factors (HFs) on software project success, whereas Guveyi et al. (2020) classify HFs into personal, interpersonal, and organizational dimensions, highlighting aspects such as education, experience, and motivation. Other scholars have examined specific human-related variables, including commitment, collaboration, and team cohesion, demonstrating their direct influence on project costs and overall outcomes (Guveyi et al., 2020).

Table 1.1. Comparative table of the literature review

Reference	What is HF in the SDP?	How does HF influence the SDP?	Results and conclusion	The most influential social or human factors
1	2	3	4	5
Guveyi et al., 2020	At each stage of the SDP, human knowledge, intelligence, and experience affect software product quality.	Due to HF across the entire SDP, software quality depends on human behaviors.	Personal factors (experience and education) are the most important category of HFs.	Human knowledge, intelligence, and experience.
Machuca-Villegas et al., 2021	Social and human factors are of particular importance because they impact the results of software projects and are considered important elements affecting costs.	Personal aspects and human activities represent an opportunity to improve productivity.	The HF impact could be different depending on different SDP methodologies.	Commitment, collaboration, team cohesion, capabilities, and SDP experience.

Continued Table 1.1

1	2	3	4	5
Capretz, 2014	The software product depends on human activities, such as problem-solving capabilities, cognitive aspects, and social interaction.	HF is a make-or-break issue that affects most software projects.	Lack of study of the impact of HF on the SDP.	Problem-solving capabilities, cognitive aspects, and social interaction.
Pirzadeh, 2010	The SDP is a human-centered activity.	Humans play different roles in the SDP. This has an impact on process performance and success.	There is a lack of primary and secondary studies on the HFs related to the SDP, and the impact of HFs varies depending on the methodologies used in the SDP.	Teamwork, communication, virtual teamwork, and human resource management.
Amrit et al., 2014	Software development has been characterized in essence as a human activity where HF plays a critical role.	The growing importance of HFs in software development research is clearly evidenced by the ICSE 2014 conference entirely devoted to HFs ("Social Aspects of Software Engineering").	Main theories regarding the HFs in software development come from a field related to behavioral and social sciences.	N/A

Continued Table 1.1

1	2	3	4	5
Gonçalves et al., 2017)	In software factories, the HF represents the central investment capital to achieve productivity and quality.	The influence of HFs on the work environment is directly related to the quality of the product being developed and its productivity. Increase in efficiency and human effectiveness is the primary factor responsible for building the development of software products.	Few professionals express dissatisfaction with cognitive, operational, and organizational aspects, highlighting the need to observe, understand, and prioritize HF in a software factory.	Cognitive, operational, and organizational aspects.
Machuca-Villegas et al., 2021	Social and human factors play an important role in software engineering.	Social and human factors may affect the productivity of the software development team.	Created an instrument that proves that social and human factors exerted an influence on the productivity of the software development team.	Perceptions.
Dutra et al., 2021	The term "HF" represents a person's physical or cognitive features, or social behavior.	Different HF types can influence the SDP in different aspects (project team size, project tracking, delivery time, etc.).	The identified HFs and their influences can be considered most significant by IT organizations, researchers, and academics in SE practice.	A person's physical or cognitive features, social behavior, sentiments or attitudes, communication, motivation, and collaboration.

Continued Table 1.1

1	2	3	4	5
<p>Capretz & Ahmed, 2010</p>	<p>Personality types as HFs: extroversion, introversion, sensing, intuition, thinking, feeling, judging, and perceiving.</p>	<p>Different types of personalities could be a good fit for one or another SDP role, but may not fit other roles at all.</p>	<p>Software engineering benefits from a broad range of personality types, as no single type fits all tasks, and better SDP results emerge from diverse mental processes, outlooks, and values.</p>	<p>Extroversion, introversion, sensing, intuition, thinking, feeling, judging, and perceiving.</p>
<p>Barros et al., 2024</p>	<p>HF refers to team members' competence, psychological safety, team autonomy, and customer involvement, i.e., individual and interpersonal elements that impact collaboration and adaptability.</p>	<p>HF shapes team communication, coordination, and decision-making, directly affecting project outcomes.</p>	<p>Human-related factors are among the top success criteria in AGILE development, especially competence and active client participation.</p>	<p>Team competence, customer involvement, and psychological safety.</p>
<p>Gunatilake et al., 2024</p>	<p>HF includes empathy, emotional intelligence, communication clarity, mutual understanding, and interpersonal trust between developers and end-users.</p>	<p>HF affects the quality of requirements elicitation and overall user satisfaction through improved communication and understanding.</p>	<p>Trust, empathy, and emotional awareness are key to accurate requirements and successful user-centered software.</p>	<p>Empathy, communication clarity, and trust.</p>

End of Table 1.1

1	2	3	4	5
Muhammad et al., 2024	HF comprises openness to failure, psychological safety, team trust, risk perception, communication openness, and learning mindset.	HF either fosters or inhibits a team's ability to engage in experimentation and iterative learning.	A culture of psychological safety and trust enables more effective experimentation and process improvement.	Psychological safety, trust, and openness to change.
Mahaju et al., 2023	HF is defined as cognitive limitations (e.g., memory lapses, misunderstandings), stress, miscommunication, ambiguity, and work overload that affect engineers during requirements elicitation.	HF leads to early-stage errors in requirement specification, which propagate through the software lifecycle if unmanaged.	Addressing HF through better environments and processes reduces defects more effectively than focusing on individual performance.	Communication quality, stress level, and cognitive overload.

Capretz (2014) underscored the unpredictability of human behavior in software development, emphasizing the need to integrate human factors (HFs) into predictive models. Similarly, Pirzadeh (2010) examined HF in relation to development methodologies, particularly in requirements engineering, identifying developers as the key human role in the software development process (SDP). Amrit, Daneva, and Damian (2014) argued that HF research should be considered a subfield of empirical software engineering, highlighting methodological challenges in studying human-related variables.

Several studies focus on HF's role in specific SDP stages. For example, Gonçalves et al. (2017) analyzed HF's impact on software testing, identifying cognitive, operational, and organizational challenges that affect motivation and performance. Machuca-Villegas et al. (2021) proposed an instrument to measure HF perceptions in software teams, while acknowledging its limited practical application in improving productivity. Dutra et al. (2021) further categorized HFs

into different levels of influence within software teams, identifying over 100 relevant factors.

While each article emphasizes different HF dimensions, common threads emerge. The study on AGILE software development (AGILE Software Development Projects – Unveiling the Human-Related Critical Success Factors, 2024) underscores competence, psychological safety, and client involvement as critical success factors, demonstrating that human-centric conditions significantly enhance adaptability and performance. Similarly, the research on interactions between a developer and an end-user (The Impact of Human Aspects on the Interactions Between Software Developers and End-Users, 2024) emphasizes interpersonal elements, such as empathy, emotional intelligence, and communication clarity, all of which are shown to improve requirements quality and user satisfaction. In the context of continuous experimentation (Continuous Experimentation and Human Factors, 2023), HFs such as trust, openness to failure, and psychological safety are crucial in fostering a culture of innovation and iterative learning. Meanwhile, the article focused on error management in requirements engineering (Human Error Management in Requirements Engineering, 2023) draws attention to cognitive and emotional limitations, including stress and communication breakdowns, as major sources of early-stage defects. Collectively, these studies demonstrate that both socio-cognitive (e.g., trust, empathy) and organizational (e.g., team autonomy, process clarity) aspects of HF play a decisive role in SDP outcomes. Psychological safety and communication quality emerge as consistently influential across all contexts, suggesting that efforts to improve these factors may yield broad and sustained improvements in software development efficiency and quality.

Although numerous studies emphasize the importance of HF in the SDP, the literature also reveals several persistent challenges that significantly affect process outcomes. One of the main problems is the unpredictability of human behavior, which makes it difficult to accurately model and predict the impact of human-related variables on the SDP (Capretz, 2014). Additionally, many studies highlight the complexity of measuring and quantifying HF, since variables such as motivation, collaboration, communication, and psychological safety are often subjective and context-dependent (Amrit et al., 2014). Another major challenge is the variability of HF influence across different development methodologies, team structures, and organizational environments, meaning that the same factor may produce different effects depending on the project context (Machuca-Villegas et al., 2021). Research also indicates that inadequate communication, cognitive limitations, stress, and insufficient experience frequently lead to errors in early development stages, particularly during requirements engineering, which can propagate through later phases of the software lifecycle (Human Error Management in Requirements Engineering, 2023). Furthermore, studies point to difficulties in

maintaining effective collaboration, trust, and psychological safety within development teams, especially in distributed or remote environments, which can negatively impact productivity, knowledge sharing, and decision-making (AGILE Software Development Projects – Unveiling the Human-Related Critical Success Factors, 2024). Despite the recognized importance of HF, many organizations still lack systematic approaches for identifying, modeling, and managing these HF within the SDP. Therefore, one of the key research challenges is developing methods and models that can better capture the influence of HF and support more accurate prediction and improvement of SDP.

1.2.3. Influence and Relevance of Human Factors in Multiple Disciplines

Capretz and Ahmed (2010) explored personality types as HF, concluding that a diverse range of cognitive and behavioral traits is necessary for effective software engineering. Beyond software development, HF research has also been applied in generative design and aviation safety contexts. For example, Urquhart et al. (2022) highlighted the integration of ergonomics and human-machine interaction into algorithmic design workflows, while Demirel et al. (2024) proposed a human-centered generative design framework that embeds HF data from the start of the design process. In the aviation domain, studies, such as by Maurino (2000), emphasize that simply analyzing accident statistics is insufficient: instead, the human processes, organizational culture, and system-human interface must be accounted for to improve safety outcomes. More recently, Rodriguez (2024) examined maintenance operations and showed how HF elements like fatigue and communication breakdowns significantly impact error rates and safety in aviation. Collectively, this body of research underscores the deep and broad impact of HF, reinforcing the case for further investigation, enhanced management strategies, and the integration of human-centric approaches into software engineering methodologies to improve quality and efficiency.

HF and ergonomics play an increasingly critical role in the quality and safety of healthcare systems. Research has shown that integrating HF into system design, workflow organization, and digital transformation can significantly enhance both patient and staff outcomes. For instance, some authors (Sujan et al., 2021) emphasize the contribution of HF to safe and effective healthcare delivery, highlighting the importance of designing processes that account for human cognitive and physical capabilities. Similarly, other authors (Carayon et al., 2015) demonstrate that HF-based interventions in healthcare can improve procedural efficiency, reduce errors, and enhance overall system reliability. These findings underscore the necessity of incorporating human-centered approaches in healthcare to ensure safety, efficiency, and quality.

In industrial and manufacturing contexts, HF is critical for productivity, process quality, and worker safety. Systematic reviews have identified numerous empirical studies linking HF with the design of operational systems, including processes, equipment, and workspace layouts (Hashemi et al., 2018). Additionally, research by some authors (Kolus et al., 2023) demonstrates correlations between HF-related quality risks and musculoskeletal disorder risks, highlighting the interconnected nature of ergonomics, process quality, and worker well-being. These studies illustrate that managing HF in manufacturing not only enhances product quality but also contributes to employee safety and overall organizational performance.

Table 1.2. Key HF roles and objectives

Domain/Context	Key HF roles/objectives	Empirical examples/references	Impact/benefits
Healthcare	Integration into system design, workflow organization, and digital transformation; consideration of cognitive and physical capabilities.	Sujan et al. (2021); Carayon et al. (2015)	Improved patient and staff safety, enhanced efficiency, reduced errors, and increased system reliability.
Industrial/Manufacturing	Ensuring productivity, process quality, and worker safety; design of processes, equipment, and workspace.	Hashemi et. al. (2018); Kolus et al. (2023)	Enhanced product quality, reduced musculoskeletal disorder risk, and improved organizational performance.
Education/Technology/Enhanced Learning	Design of digital systems and learning environments; optimizing learner–technology interactions.	Smith (2007); Huang et al. (2023)	Improved learning outcomes, better usability, and more effective integration of technology in the educational process.

HF is also increasingly examined in the context of education and technology-enhanced learning environments. Studies have shown that human-centered approaches improve interactions between learners and digital systems, optimizing both usability and learning outcomes. For example, Smith (2007) explored how HF principles guide the design of educational technologies, while other authors (Huang et al., 2023) analyzed the role of human-centered AI in higher education,

emphasizing systematic incorporation of HF in technological interventions. These findings demonstrate that HF research extends beyond traditional industrial or healthcare settings and is essential for the effective design and implementation of educational systems and digital learning tools.

The key domains indicated in Table 1.2 include healthcare, industrial/manufacturing, and education/technology-enhanced learning. In healthcare, HF integration into system design and workflow organization enhances patient and staff safety, reduces errors, and improves overall system reliability. In industrial and manufacturing contexts, consideration of HF in processes, equipment, and workspace design supports productivity, ensures process quality, and mitigates occupational risks, such as musculoskeletal disorders. In education and digital learning environments, HF principles guide the design of technology-mediated interactions, optimizing usability and learning outcomes. Collectively, these examples underscore the critical importance of human-centered approaches across diverse domains, demonstrating that effective HF management improves system performance and contributes to well-being and efficiency.

Beyond demonstrating the importance of HF across different domains, research in fields such as healthcare, aviation, manufacturing, and education also provides valuable insights into how HF-related risks are modeled and mitigated. In these disciplines, various systematic approaches have been developed to analyze and manage the influence of human-related variables on complex systems. For example, in healthcare and aviation safety, human reliability analysis, ergonomic system design, and structured risk assessment frameworks are widely used to identify potential human errors and reduce their impact on operational outcomes. Similarly, manufacturing research emphasizes ergonomic modeling, workplace design optimization, and data-driven monitoring of worker performance to minimize fatigue, reduce errors, and improve productivity. In educational technology and human-computer interaction studies, user-centered design methodologies and cognitive modeling techniques are applied to better understand human behavior and improve system usability. These approaches demonstrate that HF can be systematically integrated into system design, evaluation, and decision-making processes. Many of these strategies could be adapted to software engineering, particularly in the context of software development process modeling and performance prediction. For instance, concepts such as human reliability assessment, human-centered system design, and data-driven modeling of human behavior could support the development of more accurate predictive models of the SDP and help mitigate the negative influence of HF on project outcomes. Therefore, the interdisciplinary experience of managing HF in other domains provides a valuable methodological foundation that can be adapted and extended within informatics engineering and software development research.

1.2.4. Human Factor Modeling in the Software Development Process

HF plays a crucial role in shaping the SDP, as the motivation, experience, and overall well-being of developers significantly impact productivity, code quality, and project success (Russo & Stol, 2020). A motivated development team tends to be more innovative, engaged, and efficient, leading to faster problem-solving and higher-quality software solutions. Factors such as job satisfaction, clear communication, team collaboration, and a positive work environment can enhance creativity and reduce burnout, which is a common issue in the software industry. Additionally, the experience level of employees directly affects the efficiency of the development process: senior developers with extensive knowledge can mentor junior team members, optimize workflows, and introduce best practices, whereas inexperienced developers may require more training and guidance, potentially slowing down progress. Cognitive load and stress levels also influence performance; excessive pressure, unrealistic deadlines, or inefficient task allocation can lead to fatigue, errors, and decreased productivity. Companies that prioritize HF by fostering a supportive work culture, providing continuous learning opportunities, and implementing effective team dynamics create an environment where developers can thrive. Ultimately, recognizing and addressing these HF ensures a smoother development process, leading to better software quality, faster delivery, and improved long-term success of software projects (Mohan & Banerjee, 2020).

Given the critical impact of HF on the SDP, modeling approaches have been developed to quantify and predict their influence on project outcomes. Several methods have been proposed, ranging from expert-based models, such as Cognitive Work Analysis and Human Reliability Analysis, to computational techniques like fuzzy logic and Adaptive Neuro-Fuzzy Inference Systems (ANFIS) (Amrit et al., 2014; Capretz, 2003; Jimoh et al., 2022). Expert-based models formalize the effects of cognitive, behavioral, and social factors on productivity, error rates, and overall efficiency, often relying on historical project data, observational studies, or surveys to quantify HF variables such as motivation, experience, teamwork, and communication efficiency.

Fuzzy logic and neuro-fuzzy approaches, including ANFIS, are particularly effective in addressing the uncertainty and subjectivity inherent in human behavior. These models combine expert knowledge with computational learning to predict HF impacts on software development outcomes, such as productivity, defect rates, and schedule deviations (Amrit et al., 2014). By integrating real-world data from software projects, these models allow project managers to anticipate potential risks and implement interventions before errors occur, bridging the gap between theoretical assumptions and practical applications.

Moreover, HF modeling is increasingly combined with case-based simulation frameworks that replicate project-specific scenarios. Such simulations enable teams

to evaluate “what-if” situations, optimize task allocations, and test the effectiveness of interventions designed to mitigate human-related risks (Capretz, 2003). By integrating HF models with simulation, organizations gain actionable insights into the complex interactions between human behavior and SDP, ultimately leading to improved efficiency, quality, and predictability of project outcomes. The most relevant methods on HF modeling in SDP are summarized in Table 1.3.

Table 1.3. Most relevant methods on HF modeling in the SDP

HF	Modeling/Analytical Approach	Impact on SDP	References
Motivation and Job Satisfaction	Fuzzy logic, ANFIS-based modeling, and survey-based behavioral analysis.	Increased productivity, reduced turnover, and higher software quality through enhanced engagement and creativity.	Mohan & Banerjee (2020); Amrit et al. (2014)
Experience and Skill Level	Cognitive Work Analysis and Bayesian inference models.	Improved code quality and reduced project delays through knowledge sharing and mentorship.	Capretz (2003); Jiang et al. (2016)
Team Collaboration and Communication	Social network analysis, agent-based simulation, and fuzzy modeling.	Enhanced coordination and problem-solving; minimized miscommunication and rework.	Jimoh et al. (2022); Amrit et al. (2014)
Cognitive Load and Stress	Neuro-fuzzy systems, workload modeling, and physiological monitoring.	Reduced human error and burnout; improved decision-making accuracy under pressure.	Tartarisco et al. (2015)
Organizational Environment and Leadership	System dynamics modeling and qualitative causal mapping.	Improved adaptability, reduced conflict, and better team cohesion and resilience.	Amrit et al. (2014); Mohan & Banerjee (2020)
Learning and Continuous Improvement	ANFIS-based learning models and reinforcement learning simulations.	Improved performance over time, faster adaptation to change, and enhanced process maturity.	Jimoh et al. (2022)

Overall, systematic incorporation of HF modeling in SDP is essential to manage uncertainty and optimize performance. By recognizing the role of motivation, experience, cognitive load, and team dynamics – and by applying computational and empirical modeling techniques – organizations can enhance project planning, reduce risks, and improve software quality and delivery timelines.

Although several approaches exist for modeling HF in the SDP, the choice of an appropriate modeling method depends on the nature of the analyzed variables and the characteristics of the available data. Expert-based models, such as Cognitive Work Analysis or Human Reliability Analysis, are useful for qualitative assessments but often rely heavily on expert judgment and structured observations, which may limit their ability to capture dynamic relationships between multiple factors. Statistical and probabilistic approaches, including Bayesian inference models, are effective when large and well-structured datasets are available; however, they may struggle to represent subjective or imprecise variables such as motivation, stress, or collaboration, which are inherently difficult to quantify precisely. In contrast, fuzzy logic-based approaches are particularly suitable for modeling human-related phenomena because they can handle uncertainty, vagueness, and linguistic assessments commonly associated with human behavior. Methods such as the ANFIS combine the interpretability of fuzzy rule-based systems with the learning capability of neural networks, enabling the model to capture complex nonlinear relationships between HF and the SDP. Therefore, fuzzy logic and ANFIS were selected in this research because they provide a flexible framework for integrating expert knowledge with empirical data while effectively addressing the ambiguity and subjectivity characteristic of HF variables in SDP modeling.

1.2.5. Human Factor Challenges and Mitigation Strategies

HF presents a multifaceted set of challenges in complex systems across various domains, including software development, healthcare, manufacturing, and education. These challenges arise from the inherent variability of human behavior, including cognitive limitations, emotional responses, and social interactions, which can significantly influence system performance, efficiency, and safety (Amrit et al., 2014; Carayon et al., 2015). Common HF-related issues include miscommunication, coordination failures, cognitive overload, fatigue, insufficient training, and inadequate adaptation to changing work environments (Kolus et al., 2022). The consequences of unaddressed HF issues can manifest as process inefficiencies, reduced product quality, increased error rates, and compromised safety outcomes. These challenges underscore the importance of systematically identifying and mitigating HF risks to improve organizational and project outcomes.

In software development, HF challenges often manifest as delays, budget overruns, and software defects due to human error or misalignment between team members' skills and project requirements. Developers' cognitive capacities, decision-making styles, and communication patterns directly influence project success (Capretz, 2003; Pirzadeh, 2010). To mitigate these risks, structured modeling approaches, such as ANFIS-based HF simulations or HF-informed process workflows, have been employed to predict potential bottlenecks and optimize team performance. Case-based simulation, scenario analysis, and workload assessment tools also enable managers to anticipate human-related errors and design interventions before issues escalate (Amrit et al., 2014).

In healthcare, HF challenges are particularly critical due to the high stakes involved. Errors stemming from fatigue, cognitive overload, or miscommunication can directly affect patient safety and treatment outcomes. Human-centered interventions focus on designing processes and technologies that accommodate human cognitive and physical limitations. For instance, the Systems Engineering Initiative for Patient Safety (SEIPS) model integrates HF principles into workflow and system design, promoting error prevention, operational efficiency, and staff well-being (Carayon et al., 2015). Additionally, training programs emphasizing teamwork, situational awareness, and adaptive decision-making have proven effective in reducing errors in complex clinical environments.

In manufacturing and industrial contexts, HF challenges include ergonomics issues, task complexity, and fatigue, which can negatively affect both productivity and quality. Systematic reviews demonstrate that over 70 empirical studies link HF with operational system design, including workstations, process flows, and safety measures (Kolus et al., 2018). Mitigation strategies in this domain include ergonomic assessments, standard operating procedures, automated support systems, team training, and fatigue management protocols (Kolus et al., 2023). By systematically addressing HF risks, organizations can reduce errors, improve workplace safety, and enhance product quality.

Education and technology-mediated learning environments also face HF-related challenges, including cognitive load, interface usability, and learner engagement. Human-centered design approaches in educational technology focus on adapting systems to learner capabilities, preferences, and limitations. For example, Stone (2008) highlighted how HF principles guide the development of educational technologies, while Huang, Liu, and Tili (2023) demonstrated the application of human-centered AI to optimize higher education tools for effective student interactions. These approaches show that HF considerations are not confined to physical or industrial systems but are equally relevant in digital and cognitive environments.

Table 1.4. HF challenges and typical mitigation strategies

Domain/Context	HF challenges	Typical mitigation strategies	References
Software development	Miscommunication, cognitive overload, fatigue, skill mismatches, and coordination failures.	ANFIS or HF-informed process modeling, scenario analysis, workload assessment, team training, and structured communication protocols.	Capretz (2003); Pirzadeh (2010); Amrit et al. (2014)
Healthcare	Fatigue, cognitive overload, miscommunication, workflow errors, and inadequate adaptation to system changes.	SEIPS model integration, human-centered process design, error prevention protocols, teamwork, and situational awareness training.	Carayon et al. (2015); Sujan et al. (2021)
Manufacturing / Industrial Production	Ergonomic risks, task complexity, fatigue, and quality-related errors.	Ergonomic assessments, standard operating procedures, fatigue management, automation support, and team training programs.	Kolus et al. 2018; Kolus et al., 2023
Education / Digital Learning	Cognitive overload, interface usability issues, engagement challenges, and accessibility issues.	Human-centered educational technology design, adaptive AI systems, usability testing, learner-centered instructional design, and feedback mechanisms.	Stone, 2008; Huang et al. (2023)

Overall, addressing HF challenges requires a combination of system design, process optimization, training, and continuous monitoring. Cross-domain evidence demonstrates that integrating HF principles can mitigate risks, enhance efficiency, improve quality, and reduce error rates. Across software development,

healthcare, manufacturing, and education, the adoption of human-centered strategies is essential to create resilient systems that accommodate human variability and optimize performance.

1.3. Simulating the Software Development Process

1.3.1. Concept of Simulation of Complex Processes

Simulation plays an important role in the analysis and improvement of business processes, as it enables organizations to evaluate potential changes and predict system performance without interfering with real operational environments. Traditional simulation models are often created manually, which can be time-consuming and prone to inaccuracies because they rely on subjective assumptions about process behavior.

Recent research has emphasized the increasing importance of more advanced and data-driven simulation approaches, particularly in the context of digital twins. According to Wooley et al. (2023), a key distinction between traditional simulation models and digital twins lies in the level of integration with real-world data and the ability to continuously update the model based on real-time information. While conventional simulations are often static and scenario-based, digital twins represent dynamic, continuously synchronized models that more accurately reflect the current state of the system. This capability significantly enhances the validity and applicability of simulation results, especially for decision-making in rapidly changing environments. Furthermore, Wooley et al. (2023) highlighted that simulations evolve toward digital twins when they incorporate features such as real-time data integration, bidirectional communication with the physical system, and adaptive model behavior. These characteristics enable more precise monitoring, prediction, and optimization of business processes. In contrast, traditional simulation approaches typically lack such feedback mechanisms, limiting their ability to capture emergent behaviors and system variability over time. By incorporating elements inspired by digital twin concepts, such as data-driven inputs, continuous validation, and adaptive modeling, simulation frameworks can be significantly enhanced. This solution improves the accuracy of process representation and allows for more robust scenario analysis and supports proactive decision-making. As a result, modern simulation approaches are increasingly positioned as a bridge between static modeling techniques and fully integrated digital twin systems, providing a more comprehensive understanding of complex business processes.

To address the mentioned limitations, Rozinat et al. (2009) proposed a data-driven approach in which simulation models are automatically discovered from

event logs using process mining techniques. Event logs contain detailed information about process execution, including activities, timestamps, resources, and case identifiers. By extracting multiple process perspectives, such as control flow, performance, data, and resource information, from these logs, it is possible to construct a comprehensive simulation model that more accurately reflects real process behavior. The authors demonstrate that these discovered models can be represented using Colored Petri Nets and applied to analyze process performance and evaluate alternative process designs through simulation experiments (Rozinat et al., 2009).

Simulation is an important technique used in business process management to analyze and evaluate process behavior under different conditions without affecting real organizational operations. Traditional business process simulation approaches typically rely on static process models where activities and their sequence are predefined before execution. However, such approaches are often insufficient in dynamic business environments where processes must adapt to changing conditions. Vasilecas, Kalibatiene, and Lavbič (2016) discussed that dynamic business processes require modeling approaches that allow activities and their sequence to change during process execution based on predefined rules and contextual information. In their proposed rule and context-based approach, process simulation considers both internal and external context factors and uses business rules to determine which activities should be executed at runtime. This enables the modeling of processes that evolve during execution and allows organizations to evaluate different scenarios in dynamic environments. The authors also propose a reference architecture and prototype simulation tool that supports rule- and context-based dynamic business process modeling and demonstrates its applicability through a case study (Vasilecas et al., 2016).

The concept of complex process simulation involves creating digital or mathematical models to replicate real-world systems, allowing for analysis, prediction, and optimization without direct real-world intervention. Complex processes, such as weather forecasting, traffic management, financial markets, and industrial operations, often involve numerous interdependent variables and dynamic interactions that are difficult to predict with traditional methods. Simulations provide a controlled environment where different scenarios can be tested, helping researchers and decision-makers understand system behavior, identify inefficiencies, and develop strategies for improvement. Advanced technologies such as artificial intelligence, machine learning, and high-performance computing have further enhanced simulation capabilities, enabling more accurate and detailed models. By reducing risks, lowering costs, and improving decision-making, the simulation of complex processes has become a crucial tool in fields ranging from engineering and medicine to economics

and climate science, driving innovation and efficiency in solving real-world challenges (Banks et al., 2010).

Table 1.5. Comparative analysis of simulation approaches

Aspect	Wooley et al. (2023)	Rozinat et al. (2009)	Vasilecas et al. (2016)	Banks et al. (2010)
1	2	3	4	5
Main focus	Distinction between traditional simulation and digital twins, including characteristics and evolution of simulation models toward real-time systems.	Automatic discovery of simulation models from event logs using process mining.	Rule and context-based dynamic business process modeling and simulation.	General concept of simulation for complex systems.
Type of process model	Hybrid simulation models evolving into digital twins with real-time synchronization and continuous updating.	Data-driven simulation models derived from real process execution data.	Dynamic process models where activities can change during execution.	Mathematical or digital models representing complex real-world systems.
Key data source	Real-time and historical data streams from physical systems, sensors, and operational environments.	Event logs containing activities, timestamps, resources, and case identifiers.	Business rules and contextual information (internal and external factors).	System variables, parameters, and interactions in complex environments.
Modeling approach	Data-driven, adaptive modeling with continuous feedback loops and system synchronization.	Extraction of multiple perspectives (control flow, performance, data, and resources).	Rule-based and context-aware modeling allowing runtime changes.	Computational modeling and scenario analysis.

End of Table 1.5

1	2	3	4	5
Representation/tools	Digital twin architectures, integration platforms, and data-driven simulation environments.	Colored Petri Nets for representing discovered models.	Reference architecture and prototype simulation tool.	Advanced technologies such as AI, machine learning, and high-performance computing.
Purpose of simulation	Provide real-time monitoring, prediction, and optimization through continuous alignment with real-world systems.	Analyze process performance and evaluate alternative process designs.	Evaluate dynamic scenarios and support adaptable business processes.	Analyze, predict, and optimize the behavior of complex systems.
Application area	Smart manufacturing, healthcare, logistics, and cyber-physical systems.	Business process analysis and improvement using process mining.	Dynamic business process management.	Broad fields such as engineering, finance, medicine, and climate science.

The comparison presented in Table 1.5 highlights different perspectives on the use of simulation for analyzing and improving processes. Rozinat et al. (2009) focused on a data-driven approach in which simulation models are automatically discovered from event logs using process mining techniques. This approach allows the creation of simulation models that accurately reflect real process behaviors by extracting information about control flow, performance, and resources from historical execution data. In contrast, Vasilecas, Kalibatiene, and Lavbič (2016) emphasized the need for dynamic business process modeling and simulation in environments where processes must adapt to changing conditions. Their rule- and context-based approach enables activities and their sequence to change during runtime based on business rules and contextual factors, thus supporting more flexible and adaptive process analysis. Meanwhile, Banks et al. (2010) provided a broader view of simulation as a method for modeling and analyzing complex systems, highlighting its role in testing scenarios, predicting system behaviors, and supporting decision-making across various domains. Extending these perspectives, Wooley et al. (2023) introduced a more recent viewpoint by examining the evolution of simulation models toward digital twins. Their work highlights that traditional simulation approaches, while valuable, are often limited by static assumptions and a lack of continuous integration with real-world data. In contrast, digital twin-oriented simulation models are characterized by real-time data synchronization, continuous updating, and bidirectional interaction with the physical system. This enables a more accurate and timely representation of system states,

as well as improved predictive and optimization capabilities. By incorporating these features, simulation moves beyond static scenario analysis toward dynamic, data-driven environments that support real-time decision-making and system monitoring.

Together, these studies demonstrate that simulation can be applied at different levels, from data-driven discovery of business process models to dynamic rule-based process adaptation, to the general modeling of complex systems, and further toward real-time, continuously evolving digital twin systems. This progression reflects a shift from static and retrospective analysis toward adaptive and predictive simulation paradigms, significantly enhancing the applicability of simulation in complex and rapidly changing environments.

1.3.2. Software Development Process Simulation

Simulation in the software development process is a powerful technique for modeling, analyzing, and optimizing various aspects of software engineering before actual implementation (Zhang, 2010; Ali & Unterkalmsteiner, 2023). It enables teams to evaluate system behavior, performance, and potential risks without costly or time-consuming real-world trials (Borshchev, 2013).

Some research (Robinson, 2006; Law, 2007; Montevechi et al., 2010; Software Solutions Studio, 2021) outlines the main categories of simulation models and emphasizes the importance of selecting the right type based on project goals and system characteristics. These include deterministic models, which produce predictable outcomes given specific inputs, and stochastic models, which incorporate randomness to reflect uncertainty in real-world processes. Another distinction is made between static models, which capture a snapshot of the system at a single point in time, and dynamic models, which represent how the system evolves over time.

By creating virtual environments that replicate real-world scenarios, simulations allow developers to predict system behavior, evaluate performance, and identify potential issues early in the development cycle. For example, load testing simulations can evaluate how an application handles high traffic volumes, while security simulations help detect vulnerabilities prior to deployment. For example, simulation tools like DICE Simulation allow developers to evaluate system performance and identify potential bottlenecks early in the design phase (Casale, Chandra & Tofan, 2022). Moreover, simulation techniques are increasingly used by AGILE teams to optimize internal workflows, support collaborative planning, and enhance data-driven decision-making. Advanced tools such as digital twins (Wooley et al., 2023), Monte Carlo simulations (Harrison, 2010), and automated testing frameworks (Winkler et al., 2010) further support these efforts by reducing development risks, minimizing costs, and increasing the overall quality of software products.

A digital twin simulation is a virtual model of a real-world object, system, or process that allows for monitoring, analyzing, and optimizing its performance. This simulation combines physical and digital data using sensors, AI, and data analytics to reflect and predict the state of the object and potential changes in real time (Wooley et al., 2023).

Monte Carlo simulation is a statistical method used for decision-making and problem-solving when there is a lot of uncertainty or random factors. This simulation is based on the generation of random numbers and is used in various fields to predict possible outcomes and assess risk (Harrison, 2010).

Automated Testing Frameworks (ATF) simulation is a method that uses automated testing tools and programs to verify how a software system or its components work by simulating various scenarios and verifying their response. It is a particularly useful tool when it is necessary to ensure that the system operates according to requirements, ensuring quality and performance without human intervention (Winkler et al., 2010).

By integrating simulation techniques into the software development lifecycle, teams can ensure more robust, scalable, and efficient applications while reducing the need for costly post-deployment fixes.

1.3.3. Fuzzy-Based Software Development Process Simulation

Fuzzy-based software development process simulation applies fuzzy logic principles to model the uncertainties and complexities inherent in software engineering (Kellner et al., 1999; Liu et al., 2023). Unlike traditional simulations that rely on precise, binary inputs, fuzzy-based simulations accommodate imprecise, vague, or uncertain data, making them well-suited to dynamic, evolving software development environments. These simulations help assess factors such as project risk, team performance, and resource allocation by considering variables like developer expertise, requirement stability, and productivity, which often have subjective or ambiguous values. By using fuzzy inference systems (Kalibatiené & Miliuskaitė, 2021), decision-makers can better evaluate project timelines, optimize resource distribution, and anticipate potential bottlenecks. Fuzzy inference systems have been successfully applied to project scheduling under uncertain resource availability; for instance, some authors (Bershtein et al., 2015) developed a fuzzy-based model to allocate multi-skilled resources in software projects, enabling decision-makers to better optimize resource distribution and anticipate bottlenecks. The following classical explanation of fuzzy sets is presented to understand the concept.

Definition 1: Let U be the universe of discourse (UoD) containing elements u . Then a set of ordered pairs (see Eq. (1.1)) (Zadeh, 1965; Zadeh, 1975)

$$A = \{u, \mu_A(u) | u \in U, \mu_A: U \rightarrow [0; 1]\} \quad (1.1)$$

is a fuzzy set A in U and, $\mu_A(u)$ is MF of u in A .

The value $\mu_A(u)$ represents the grade of membership of u in A and is interpreted as the membership degree to which u belongs to A . So, the closer the value $\mu_A(u)$ is to 1, the more u belongs to A .

Definition 2. The fuzzy number is a convex set on \mathbb{R} such that

1. There exists $u \in \mu_A(u) = 1$;
2. $\mu_A(u)$ is piecewise continuous;
3. $Supp(A) = [a, b]$, $a \leq b$, where neither a nor b is permitted to be infinite (Zadeh, 1975).

The literature offers different types of MF shapes, as presented in (Choi & Rhee, 2009). This research will apply four types of MF shapes (Zimmermann, 2011). Trapezoidal (1.2), Triangular (1.3), etc.

$$\mu^{Trapez}(u) = \begin{cases} 0, & \text{if } u < a_1, a_4 \leq u, \\ \frac{u - a_1}{a_2 - a_1}, & \text{if } a_1 \leq u < a_2, \\ 1, & \text{if } a_2 \leq u < a_3, \\ \frac{a_4 - u}{a_4 - a_3}, & \text{if } a_3 \leq u < a_4, \end{cases} \quad (1.2)$$

$$\mu^{Triang}(u) = \begin{cases} 0, & \text{if } u < a_1, a_3 \leq u, \\ \frac{u - a_1}{a_2 - a_1}, & \text{if } a_1 \leq u < a_2, \\ \frac{a_3 - u}{a_3 - a_2}, & \text{if } a_2 \leq u < a_3, \end{cases} \quad (1.3)$$

According to MF vagueness, if there exists a clear membership value based on MF $\mu_A(u)$ for every input data point, then the fuzzy set is of a type-1 and may be represented as in Equation (2). A can also be defined as presented in (1.4):

$$A = \int_{u \in U} \mu_A(u) / u, \quad (1.4)$$

where \int stands for union over all possible u . A fuzzy inference system (FIS) that uses fuzzy set theory to map inputs to outputs and consists of the following main components (Askari, 2017; Mamdani, 1974; Takagi & Sugeno, 1985; Miliuskaitė & Kalibatiene, 2020) (Fig. 1.1). Fuzzification is converting the data input into fuzzy linguistic variables by applying a particular fuzzification method. This method presents a way of determining the degree to which input variables belong to each of the appropriate fuzzy sets via the membership functions (MFs). Fuzzy Inference involves applying a set of fuzzy rules from a knowledge base to fuzzy inputs to generate a set of fuzzy outputs. Defuzzification translates the obtained

fuzzy output for the final user into an understandable crisp output, using a particular defuzzification method.

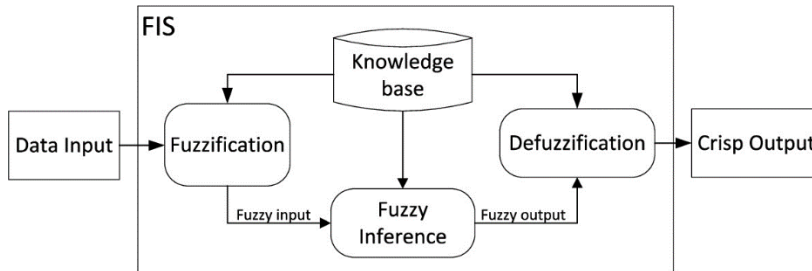


Fig. 1.1. Reference schema of FIS (Miliauskaitė & Kalibatiene, 2020)

To improve the performance of the FIS and to optimize rules in the Knowledge base in FIS, particular optimization methods, such as genetic algorithms, Particle Swarm Optimization, adaptive neuro-fuzzy inference, etc. (Kalibatiene & Miliauskaitė, 2021), can be used. For better performance and feasibility, ANFIS, which is a combination of a fuzzy inference system (FIS) and an adaptive neural network (Jang, 1993), is used for optimization.

Neuro-fuzzy systems are hybrid artificial intelligence techniques that combine the strengths of fuzzy logic with artificial neural networks. A review of the literature reveals that various neuro-fuzzy models have been proposed (Vieira et al., 2004), among which the Adaptive Neuro-Fuzzy Inference System (ANFIS), introduced by Jang (1993), is one of the most widely used. ANFIS integrates the learning capability and relational structure of neural networks with the decision-making mechanisms of fuzzy logic. Like neural networks, ANFIS is trained using a dataset to identify patterns and build an optimal model for a given problem. Once trained, the model is tested on previously unseen data to evaluate its generalization performance, where lower error rates indicate a better fit. A key limitation of traditional neural networks is their lack of interpretability, as the learned weight values are often opaque. ANFIS addresses this issue by incorporating a fuzzy inference system, which enhances the model's transparency and explainability. Due to this interpretable and adaptive structure, ANFIS has been effectively applied to a wide range of real-world problems (Kar, 2014).

According to Ighalo et al. (2021), Adaptive Neuro-Fuzzy Inference Systems (ANFIS) and Artificial Neural Networks (ANN) have been the most widely used artificial intelligence models for water quality monitoring and assessment systems, particularly those incorporating solar energy, over the past decade. The key advantages of applying AI methods in such monitoring and assessment applications include their speed and efficiency compared to traditional linear mathe-

mathematical or statistical techniques, cost-effectiveness, suitability for real-time monitoring (Yetilmezsoy et al., 2011; Ighalo et al., 2021; Demetillo et al., 2019), and improved prediction accuracy with lower error rates (Karaboga & Kaya, 2019).

ANFIS is used for fuzzy inference in this research because it combines the adaptability of neural networks with the interpretability of fuzzy logic, effectively modeling complex nonlinear relationships.

This adaptive neural network framework (Jang, 1993) uses fuzzy IF–THEN rules (e.g., Equation (1.5)) with tailored membership functions to generate specified input–output pairs:

$$R_i : \text{if } (x_1 \text{ is } A_1^{(i)}) \dots \text{and } \dots (x_n \text{ is } A_n^{(i)}) \text{ then } f_i = a_i^T \cdot x + b_i, \quad (1.5)$$

where $x \in \mathbb{R}^n$ is a vector of inputs characterized by an appropriate MF, and a_i, b_i are the coefficients of linear Takagi–Sugeno consequents. Training ANFIS involves determining parameters related to both the premise (input parameters) and the consequences (output) using an optimization algorithm.

The primary structure of ANFIS encompasses five layers.

In Layer 1, inputs (x_1, x_2) , $n = 2$) are fuzzified using trapezoidal MFs with adaptive parameters: c_1, c_2, c_3, c_4 . The trapezoidal function was chosen because of its flexible representation compared to triangular functions: it allows for a flat plateau in the middle, which can better capture certain types of fuzzy relationships (Kacker & Lawrence, 2007).

Layer 2 evaluates the rule strength (Eq. (1.6)):

$$w_i = \prod \mu_i(x). \quad (1.6)$$

Layer 3 normalizes the strengths of all rules (Eq. (1.7)):

$$\bar{w}_i = \frac{w_i}{\sum_i w_i}. \quad (1.7)$$

Layer 4 applies the rule R_i to obtain the output f_i (Jang, 1993)

Layer 5 computes the global model response:

$$f = \sum_i \bar{w}_i f_i. \quad (1.8)$$

After training is complete, ANFIS performance is determined by various statistical tests (Chicco et al., 2021). This study used the following statistical tests: coefficient of determination (R^2), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE).

The coefficient of determination (R^2) (Chicco et al., 2021) is used to analyze how differences in one variable can be explained by differences in another variable (Eq. (1.9)). It ranges in the interval $[0, 1]$.

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}, \text{ where } SS_{tot} = \sum_i (f_{train_i} - \overline{f_{train}})^2 \text{ and } SS_{res} = \sum_i (f_{train_i} - f_i)^2, \text{ where } SS_{res}, \quad (1.9)$$

denotes the sum of the squares of residuals, and SS_{tot} the total sum of squares.

The mean squared error (MSE) (Van der Aalst et al., 2005) (Eq. (1.10)) is used to measure the average of the squares of errors between the output f and the training f_{train} values.

$$MSE = \frac{1}{n \sum_{i=1}^n (f_i - f_{train_i})^2}. \quad (1.10)$$

The predicted result is precise when the value of MSE is closer to zero.

The root mean squared error (RMSE) is a metric commonly used for comparing values predicted by a model or estimating in relation to values observed. RMSE is defined as the square root of the average squared distance between the actual score and the predicted score (Chicco et al., 2021) (Eq. (1.11) (Sasaki et al., 2015)).

$$RMSE = \sqrt{\frac{\sum_i (f_i - f_{train_i})^2}{n}}. \quad (1.11)$$

RMSE is always non-negative and has a value of zero when the data fit perfectly. A lower RMSE is often preferable to a higher one (Sasaki et al., 2015).

The sum of squares is a statistical measure of deviation from the mean. The sum of squares due to error (SSE) measures the total deviation of the response values from the fit to the response values (Eq. (1.12) (Korkmaz, 2021)).

$$SSE = \sum_{i=1}^n (f_i - f_{train_i})^2. \quad (1.12)$$

The fit will be better for prediction if the SSE value is closer to 0, which means that the prediction model has a reduced random error component (Sasaki et al., 2015).

1.3.4. Applications of Fuzzy Sets in the Software Development Process

Fuzzy set theory, first introduced by Zadeh (1965), provides a mathematical framework for representing vagueness and partial truth, characteristics that frequently arise in software engineering activities. Unlike classical binary logic, fuzzy sets allow elements to belong to a set with varying degrees of membership, making them particularly suitable for domains where requirements, human judgments, and environmental conditions contain inherent uncertainty (Zadeh, 1965). This flexibility has motivated extensive research on integrating fuzzy logic into the Software Development Life Cycle (SDLC). One of the most prominent applications is software effort estimation, where models such as fuzzy analogy and

hybrid neuro-fuzzy systems have demonstrated improved accuracy over traditional estimation techniques, especially when project attributes are imprecise or linguistically defined (Idri et al., 2001; Idri et al., 2002).

In requirements engineering, fuzzy sets support the formalization of ambiguous stakeholder descriptions by enabling systematic handling of linguistic terms such as “high priority”, “moderate risk”, or “low stability”. This enhances traceability and reduces ambiguity during early development phases (Dubois & Prade, 1997). Testing activities also benefit from fuzzy inference systems, which can prioritize test cases or assess fault severity based on uncertain or qualitative indicators. Recent studies show that fuzzy-logic-driven test prioritization can reduce testing costs while preserving fault-detection capability (Karatayev et al., 2024). Additionally, fuzzy methods have been applied in software quality assessment, risk management, and maintenance planning, providing decision support when quantitative data are incomplete or noisy (Sommerville, 2016).

Johanyák and Zlatko (2025) provided a comprehensive overview of fuzzy logic’s application in software requirements engineering, emphasizing its effectiveness in handling the ambiguity and uncertainty inherent to early development phases. The authors highlight that stakeholder requirements are often expressed in vague, qualitative terms, which traditional binary logic struggles to formalize. By applying fuzzy set theory, these imprecise descriptions, such as priority levels, risk assessments, or user expectations, can be systematically represented and analyzed using degrees of membership. The study further demonstrates that fuzzy-based approaches improve requirements’ prioritization, consistency checking, and decision-making, ultimately enhancing the quality and reliability of software specifications. Overall, their work reinforces the role of fuzzy logic as a valuable tool for bridging the gap between human-centric requirement descriptions and formal system models.

Ouifak and Idri (2025) present a comprehensive systematic review of the use of fuzzy logic for enhancing the interpretability and explainability of machine learning models across various domains. The authors emphasize that modern machine learning techniques, particularly black-box models such as neural networks, often lack transparency, which limits their applicability in critical decision-making contexts. To address this issue, fuzzy logic is increasingly used to improve model interpretability through mechanisms such as fuzzy rule extraction and neuro-fuzzy systems, which combine learning capabilities with human-readable reasoning structures. The review confirms that integrating fuzzy logic into machine learning frameworks significantly enhances transparency, supports trust in AI systems, and facilitates more informed decision-making across domains.

Overall, incorporating fuzzy sets into software engineering processes enables more realistic and human-aligned modeling of uncertainty. By capturing grada-

tions of truth and supporting reasoning with imprecise information, fuzzy approaches enhance decision-making across requirements, estimation, testing, and maintenance. Their continued integration into machine-learning-based frameworks is expected to further improve predictive accuracy and robustness in complex software projects.

1.4. Conclusions of the First Chapter and Formulation of the Dissertation Tasks

1. HF plays a critical role in the software development process, influencing productivity, software quality, and overall team dynamics. Thus, HF must be systematically observed, deeply understood, and treated as a central focus in a software factory. It is not enough to simply acknowledge the presence of HFs: they must be actively managed and optimized by a dedicated management team that recognizes the value of intellectual capital. Skilled developers, testers, designers, and project managers bring unique cognitive abilities, experiences, and interpersonal skills that directly impact software development efficiency. However, inexperienced or unskilled developers, particularly beginners, can negatively affect project execution, leading to inefficiencies and delays.
2. When planning a project, it is essential to consider the composition of the development team and its members' varying skill levels, as these factors can significantly influence the project's outcome. Organizations must invest not only in training, well-being initiatives, and team-building strategies to foster a motivating and collaborative environment but also in talent assessment and strategic team assembly to ensure that human potential is leveraged effectively throughout the software development lifecycle.
3. Recognizing the significant impact of social and human factors, researchers have identified the need to develop and evaluate an instrument to measure perceptions of these factors within software development teams. Such an instrument could provide valuable insights into how HFs, such as communication, leadership, job satisfaction, cognitive workload, and emotional well-being, affect productivity and project outcomes. By systematically assessing these influences, development teams can implement targeted improvements to optimize team performance, reduce stress, and create a more sustainable and effective work environment.

4. Following the literature review, several key HFs influencing the SDP were identified, including motivation, collaboration, experience, education, communication, and commitment. For further research, three primary factors were selected: motivation, availability, and experience. These were chosen for their direct impact on individual and team performance, and their relevance in dynamic project environments. To model and analyze the influence of these factors, the ANFIS was used, case-handling modeling, and agent-based simulation. ANFIS is used to capture the nonlinear relationships between HF and project outcomes, leveraging its strength in combining learning capabilities with interpretability. Case-handling modeling allows for flexibility in process execution by adapting workflows to individual developer profiles and case-specific requirements, making it ideal for modeling knowledge-driven, human-centered tasks. Agent-based simulation is applied to represent the behavior and interactions of individual team members within the development environment, enabling the observation of emergent patterns and system-level impacts resulting from variations in HF. Together, these methods provide a comprehensive and nuanced approach to understanding and optimizing the role of HF in software development.

In conclusion, based on the reviewed literature, HF significantly affects software development outcomes, influencing productivity, software quality, and team cohesion. Organizations that prioritize human-centered management approaches can foster a more engaged and efficient workforce, ultimately leading to better software products.

Based on the analysis of scientific literature and the identified research gap concerning the impact of HF on the SDP, the main research directions of the dissertation are defined. To achieve the aim of the dissertation, the following research tasks are formulated:

1. To develop a fuzzy inference-based approach for predicting the impact of Human Factors on Software Development Projects.
2. To collect and analyze data from real IT organizations regarding the impact of Human Factors on Software Development Projects, based on empirical evidence from real-world IT projects.
3. To implement the proposed approach and experimentally investigate its suitability for predicting the impact of Human Factors on Software Development Projects using real data.

2

Fuzzy Set Theory and Case-Handling Based Dynamic Software Development Process Modeling and Simulation Approach

This chapter investigates the impact of HF on the SDP through simulation-based modeling using the AGILE and WATERFALL approaches. Simulation models were developed to represent the influence of HF based on real-world software development data. A fuzzy inference approach was incorporated by introducing uncertain parameters into the direct modeling phase to capture the variability and subjectivity of HF attributes within the SDP. ANFIS was constructed using historical data to derive predictive relationships between HF and process performance. To support the inference process, historical development data were collected and interpreted to train the ANFIS network. The results of this chapter were disseminated through two scientific publications in peer-reviewed journals indexed in the Clarivate Analytics Web of Science database, including one Q2 quartile journal (Sielskaitė & Kalibatienė, 2023) and one Q3 quartile journal (Sielskaitė & Kalibatienė, 2025).

2.1. General Fuzzy and Case-Handling Based Dynamic Software Development Process Modeling and Simulation Approach Description

This section describes the proposed novel fuzzy and case-handling based dynamic SDP modeling and simulation approach. It consists of the following main elements: (1) the SDP model; (2) initial data preprocessing; (3) ANFIS-based inference; (4) CMMN SDP modeling; and (5) the simulation of the SDP model.

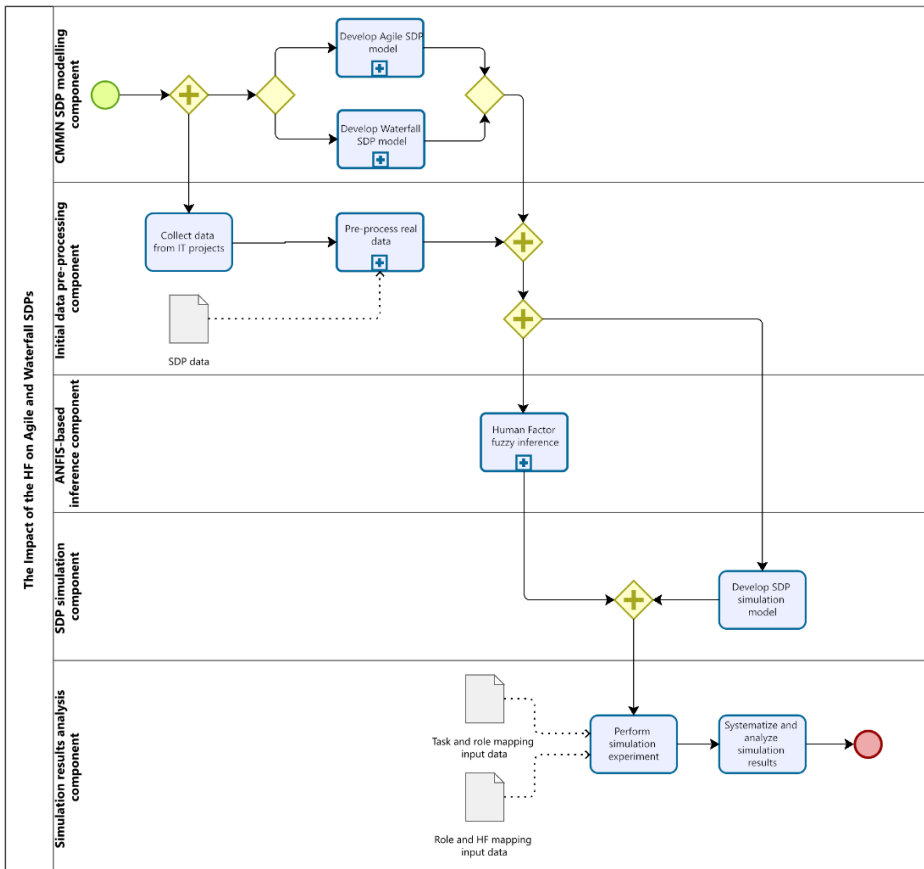


Fig. 2.1. BPMN depicting the impact of HF on AGILE and WATERFALL SDPs

Real data were first collected from software development projects, including task information, duration, responsible resources, and deviations from estimated

times. This data, incorporating HFs, was then loaded into ANFIS to perform fuzzification, rule evaluation, normalization, inference, and model response computation, followed by statistical tests. Concurrently, CMMN SDP models for AGILE and WATERFALL methodologies were created and transformed into CMMN SDP simulation models implemented in a prototype. Simulation was conducted using ANFIS-trained data and CMMN models, with input data organized into task-role and role-HF mapping tables, and results were then systematized and analyzed (Fig. 2.1).

2.2. Collecting Real Data

This study is supported by real-world data collected from multiple IT projects across various organizations. The survey was designed to collect empirical data on HF that may influence SDP, with a particular focus on parameters such as motivation, experience, and availability of employees involved in software development activities. The primary objective of the survey was to obtain indicative quantitative values of these factors that could be incorporated into the experimental simulation model. In addition to collecting information on HF, the survey also gathered data related to the estimated and actual execution times of tasks performed by employees. This information enabled identifying deviations between planned and actual task durations, which were subsequently used to derive quantitative indicators reflecting the potential influence of HF on task execution.

The survey target population consisted of professionals working in IT organizations and participating in SDP, including software developers, testers, and other technical team members. These respondents were selected because they directly contribute to the execution of software development tasks and therefore possess relevant insights into the human-related characteristics influencing process performance.

A purposive sampling strategy was applied in this study, where participants were selected based on their involvement in software development activities within IT organizations. This approach ensured that the collected data reflected realistic characteristics of software development teams. The survey data were then aggregated and anonymized before being used as input parameters for the simulation experiments.

To collect real-world data, a survey interviewing participants who know the most and have access to SDP data was chosen as the most suitable empirical strategy (Wohlin et al., 2012).

The methodology of performing the survey research is presented as follows:

Phase 1: Design and Sampling. In this phase, the survey research was planned, including survey objectives, target population, and sampling strategy.

Phase 2: Instrument Development. This phase involved the development of a survey questionnaire grounded in the data requirements of the study, which was then administered via email to selected IT organizations.

Phase 3: Data Collection. During this phase, the survey research was launched, and responses to the questionnaire were collected. Following data collection, the distribution of all variables included in the study was examined. Descriptive statistical analysis was conducted to assess skewness and kurtosis, and the results indicated that the values remained within acceptable ranges, thus satisfying the assumptions of normality required for subsequent statistical analyses.

Phase 4: Analysis. Here, the collected answers were preprocessed to develop a dataset necessary for further experiments of the whole research. The preprocessing steps are described in Section 2.3.

The primary data consisted of detailed records from three different organizations, specifying the duration of individual project tasks, performance evaluations of those tasks, and the identification of the specialists who completed them. In the next step, project managers or team leads were asked to evaluate each specialist based on selected HF, specifically, motivation, availability, and experience, using a standardized scale from 1 to 5. After gathering this information, the datasets were merged to create a comprehensive view, which allowed for the analysis of each task in terms of its estimated versus actual completion time, the quality assessment of the completed work, the identity of the specialist who performed it, and the corresponding HF scores assigned to that individual. This preprocessing stage enabled the preparation of a structured and enriched dataset suitable for subsequent modeling and analysis using ANFIS, case-handling, and agent-based simulation methods.

The collected data are stored in a publicly accessible spreadsheet available at the following link: https://docs.google.com/spreadsheets/d/1qhFvQQIh_gAXvd3_jIp7-dsII9mHjKR9/edit. Although the publicly presented dataset contains a relatively limited number of records, it is sufficient for the purposes of the experimental modeling conducted in this study. In simulation-based research, the primary objective is not statistical generalization from a large empirical dataset, but rather the analysis of process behavior under controlled experimental conditions. Therefore, the critical aspect of the data is not the absolute size of the initial dataset, but the distribution and representation of the parameters used in the model. These parameters serve as inputs for generating experimental scenarios within the simulation environment. Consequently, the limited number of records in the presented dataset does not negatively affect the validity of the experiments or the interpretation of the modeling results.

2.3. Preprocessing Real Data

Data preprocessing is a crucial step in data mining that involves transforming raw data into a clean and usable format. This process includes several stages: data cleaning, integration, transformation, and reduction. Data cleaning addresses missing values and inconsistencies, while integration combines data from different sources. Transformation processes data into appropriate formats, and reduction simplifies the dataset without losing essential information. These steps enhance data quality, making it more suitable for analysis and improving the performance of machine learning models (GeeksforGeeks, 2025).

Another step is data anonymization. Data anonymization is a data mining technique that ensures privacy by modifying or removing personally identifiable information from datasets. This process allows organizations to utilize valuable data for analysis while safeguarding individual identities. Common anonymization methods include data masking, which obscures sensitive details; generalization, which reduces data specificity; and perturbation, which introduces slight modifications to prevent re-identification (GeeksforGeeks, 2025)

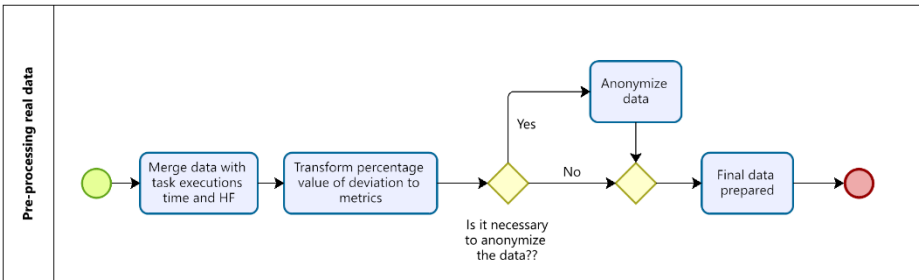


Fig. 2.2. Subprocess of preprocessing real data

Figure 2.2 illustrates the data preprocessing workflow applied prior to further analysis or simulation. The process begins with merging the collected data with task execution time and HF indicators. Next, the data are transformed by calculating percentage deviation metrics, which are used to normalize and prepare the values for subsequent analysis. After this transformation step, a decision point evaluates whether the dataset requires anonymization. If anonymization is necessary, the data are anonymized to ensure privacy and compliance with ethical requirements; otherwise, the process proceeds directly to the final stage. The workflow concludes with the preparation of the final dataset, which is then ready for use in subsequent analytical or simulation procedures.

As presented in Table 2.1, sensitive personal data was not collected in this research. The data set used in the experiment did not include any direct identifiers related to individuals, such as names, contact information, or other personal details that could allow the identification of specific employees. Nevertheless, after the survey was conducted and the data were collected, it was necessary to anonymize the names of the companies and projects from which the data were obtained. The companies, project names, and project activity names were anonymized by applying pseudonymization, replacing the original names with unique fictional identifiers.

Although the collected data did not contain direct personal identifiers, the decision to anonymize the names of organizations and projects was made to ensure a higher level of confidentiality and to prevent the potential identification of participating entities. In some cases, specific project names, technologies, or activity descriptions could indirectly reveal the identity of a company or allow the recognition of projects, especially within a limited professional or industry context. Therefore, anonymizing organizational and project-related information helps protect the confidentiality of participating organizations and reduces the risk of indirect identification.

Furthermore, the anonymization of company names was also implemented in accordance with ethical research practices and data protection principles. Participating organizations provided project-related data under the condition that their identities would not be disclosed in the published research. By replacing original company and project names with fictional identifiers, the study ensures that the data can be analyzed and presented while maintaining the privacy of the participating organizations and preserving the integrity of the research process.

Table 2.1. Example of anonymized collected data of tasks

Organization No.	Name of task	Actual	Estimated
Z	ZD11	22	28
Z	ZD12	18	23
Z	ZD13	20	25
Z	ZD14	20	24
Z	ZD15	18	39

Table 2.1 presents a comparison between the estimated and actual effort for five tasks (ZD11–ZD15). For most tasks, the actual values were lower than the estimated ones, indicating an overall tendency toward overestimation. The largest discrepancy between estimated and actual values is observed for Task ZD15, suggesting greater uncertainty or complexity in the estimation of this task.

Table 2.2. Example of anonymized collected data of employee HF parameters

Employee	Motivation	Experience	Availability
X1	3	3	3
X2	5	3	5
X3	3	3	4
X4	4	4	4
X5	5	5	3

Table 2.2 presents an example of anonymized HF data collected from employees working in IT organizations. The dataset includes three key HF parameters relevant to software development performance: motivation, experience, and availability. Each employee is represented by an anonymized identifier (in this example, X1–X5) to ensure confidentiality. The values assigned to each parameter represent the evaluated level of the corresponding factor.

2.4. Developing the Software Development Process Models

In this research, the two most popular software development methodologies (i.e., WATERFALL and AGILE) are used to investigate the impact of HF on the SDP. These two methodologies were chosen for this research as they are based on completely different principles, featuring a different sequence of activities, a different approach to changes in the environment and requirements, a different approach to the role of HF in the SDP, etc. AGILE is a flexible, iterative approach that prioritizes individuals, interactions, working solutions, and adaptation to change through iterative cycles, known as sprints, and is widely used in frameworks like Scrum, Kanban, and Extreme Programming (Cockburn, 2002). AGILE's benefits include enhanced customer satisfaction, faster delivery, and adaptability to change. In contrast, the WATERFALL model follows a linear, phase-based process, ideal for projects with fixed requirements, where each stage (requirements, design, implementation, testing, and maintenance) must be completed before moving further (Andrei et al., 2019). Though WATERFALL promotes discipline and predictability, it is less adaptable to change and less suited for complex projects.

2.4.1. Developing AGILE Software Development Process Model

This subsection presents the developed BPMN model for AGILE(Fig. 2.3).

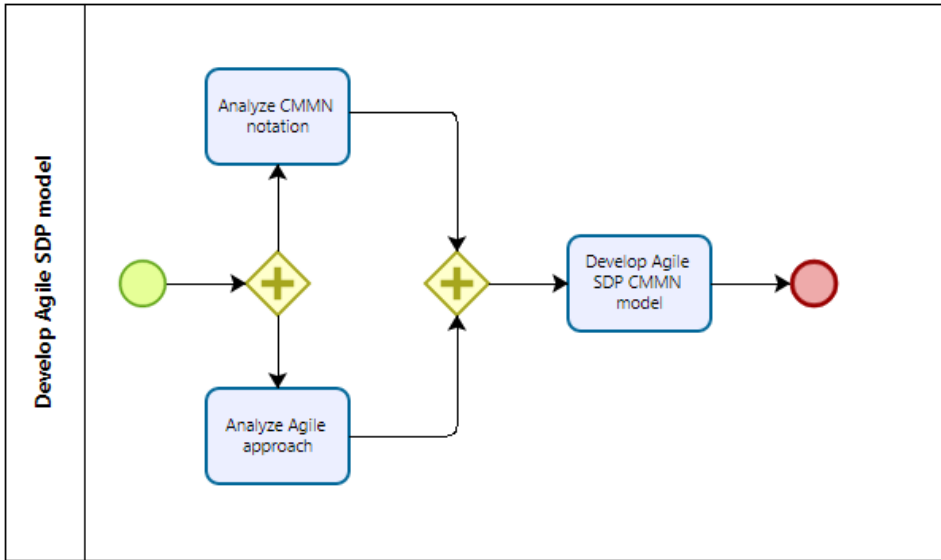


Fig. 2.3. Subprocess of developing AGILE SDP model

The developed model was structured according to the specific principles and practices of the AGILE methodology. It incorporates iterative development cycles, continuous feedback, and adaptive planning to realistically simulate the dynamic nature of AGILE-based software development. This allowed the model to better reflect human-centric factors such as team collaboration, flexibility, and responsiveness to change.

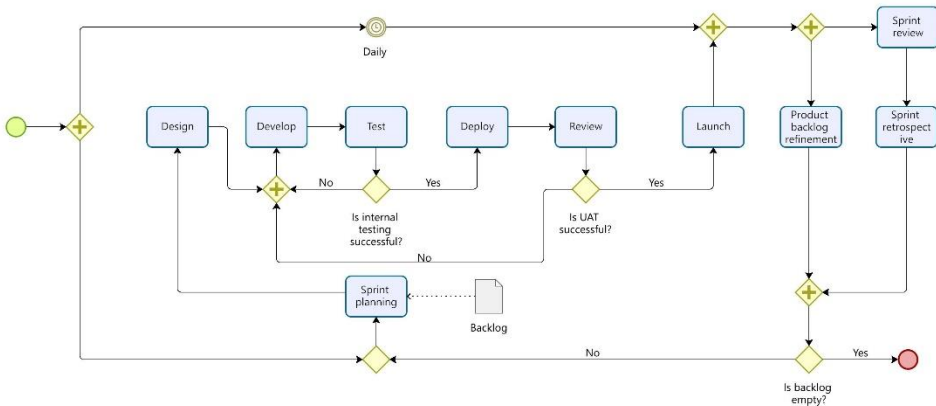


Fig. 2.4. BPMN depicting an AGILE SDP

Figure 2.4 presents a BPMN diagram of a classical AGILE SDP model, which was developed based on (Cockburn, 2002). Figure 5 illustrates the modeled AGILE software development process workflow, depicting the sequence of core activities from design to deployment, along with iterative feedback loops and decision points. Conditional branches represent testing outcomes, backlog management, and sprint-level evaluations. Meanwhile, recurring activities such as daily coordination and sprint review are integrated into the process flow. The model highlights the cyclic and adaptive nature of AGILE development, emphasizing continuous refinement, validation, and decision-driven progression.

2.4.2. Developing the WATERFALL Software Development Process Model

The model was also constructed based on the specific characteristics of the WATERFALL methodology.

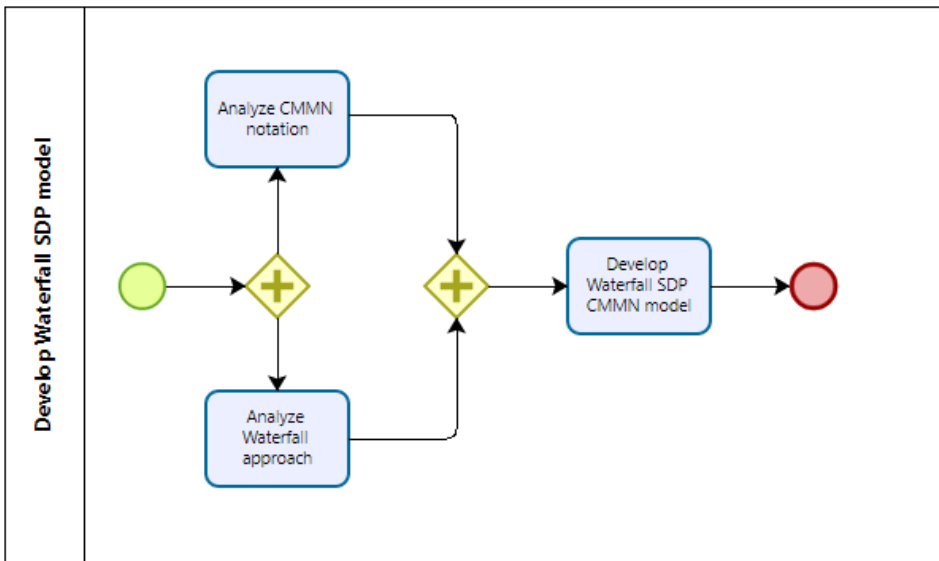


Fig. 2.5. Subprocess of developing WATERFALL SDP model

It follows a sequential development structure, with clearly defined and ordered phases such as requirements analysis, design, implementation, testing, and maintenance. This approach enabled the simulation to capture the linear and plan-driven nature of traditional software development, emphasizing documentation, upfront planning, and reduced flexibility.

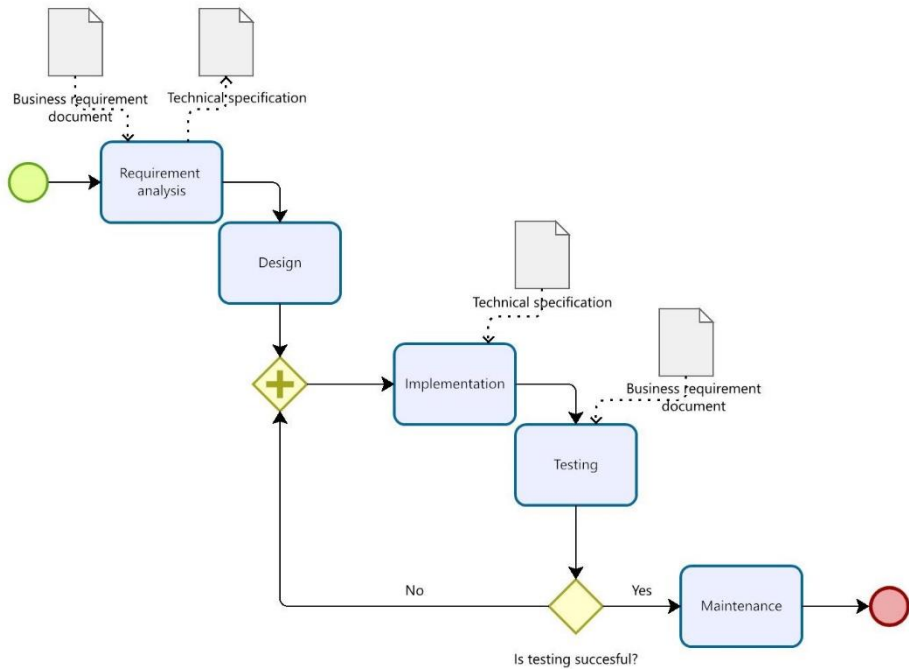


Fig. 2.6. BPMN depicting a WATERFALL SDP

Figure 2.6 presents a BPMN diagram of a classical WATERFALL SDP model, which was developed based on (Andrei et al., 2019). Figure 2.6 depicts the modeled WATERFALL software development process, illustrating a sequential flow from requirements analysis through design, implementation, and testing to maintenance. Decision points represent validation stages, where unsuccessful testing results in feedback loops to earlier phases, while successful validation leads to process completion. Associated artifacts, such as business requirement documents and technical specifications, are shown as inputs and outputs of the corresponding development stages, highlighting the structured and document-driven nature of the WATERFALL methodology.

The differences between WATERFALL and AGILE software development methodologies have been extensively analyzed in the literature. The primary distinction is that WATERFALL is a conventional, linear approach characterized by sequential phases: requirements gathering, design, implementation, testing, and maintenance, making it suitable for projects with stable requirements (Boehm, 1988). In contrast, AGILE values iterative development, emphasizing adaptability, collaboration, and customer involvement, with key features like iterative

cycles and frequent feedback that allow for responsiveness to changing requirements (Highsmith, 2002). These methodologies are often compared in terms of flexibility, with AGILE offering a more dynamic approach than WATERFALL's rigid structure. Furthermore, studies have examined their impacts on project success, team dynamics, and stakeholder satisfaction, enriching our understanding of their implications across various development contexts (Andrei et al., 2019).

The input of this step is formed of aspects of the AGILE and WATERFALL methodologies: main activities and data, sequence rules, and other relevant information. The output of this step is BPMN-based AGILE and WATERFALL SDP models.

The overall project can be depicted by the following equation:

$$SDP_i^j = \{(Task_i, H_i, Resource_i, HF_i) | i = 1, \dots, n, \quad (2.1)$$

where: $Task_i$ – all assignments that should be completed to move to another assignment or result; H_i – actual time for task execution; $Resource_i$ – competence that can perform a task; and HF_i – human factor of a task executor.

2.5. Human Factor Fuzzy Inference Using Adaptive Neuro-Fuzzy Inference System

HF fuzzy inference is used to model the complex, often imprecise relationships between qualitative human characteristics and their impact on software project outcomes. This study applied fuzzy logic to evaluate key HFs, such as motivation, availability, and experience, by translating subjective assessments into numerical values on a defined scale. These values serve as inputs to a fuzzy inference system, which uses a set of expert-defined rules to approximate how different combinations of HF levels influence task performance. This approach is particularly useful in software development contexts, where human behavior is not always easily quantified, and linear models may fail to capture the subtleties of human influence. By applying fuzzy inference, real-world decision-making processes can be simulated more accurately and assess how different HF profiles affect efficiency, task quality, and project timelines in a structured yet flexible manner.

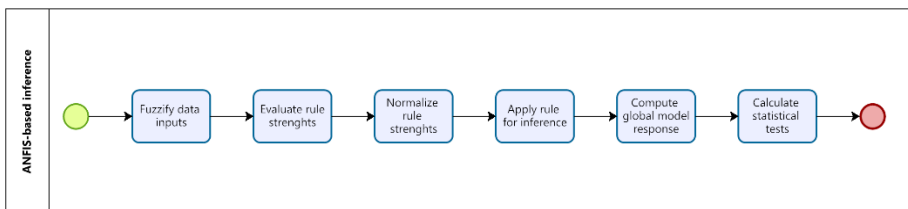


Fig. 2.7. Subprocess of ANFIS-based inference

Figure 2.7 depicts the sequential workflow of an ANFIS used to transform raw input data into statistically validated model outputs. The process begins with the fuzzification of input variables, in which numerical data are mapped to linguistic membership functions that represent degrees of belonging. This step allows the system to translate precise numerical inputs into fuzzy representations, making it possible to model uncertainty and vagueness inherent in HF variables. Each input value is evaluated against predefined membership functions, resulting in a set of fuzzy values that describe the degree to which the input belongs to specific linguistic categories.

The system evaluates the strength of each fuzzy rule by combining the membership values associated with the rule's antecedents. In this stage, logical operators, such as multiplication or minimum functions, are typically used to determine the activation level of each rule. The resulting rule strength reflects how strongly the given input data satisfies the conditions defined in the rule base. Since multiple rules may be activated simultaneously, the model must determine the relative contribution of each rule to the final output.

These rule strengths are subsequently normalized to ensure consistent weighting across the rule base. Normalization adjusts the calculated rule strengths so that their combined influence remains balanced and comparable. This step ensures that no single rule dominates the inference process unless it is strongly supported by the input data. By distributing the influence proportionally among the activated rules, the model maintains stability and interpretability.

The normalized strengths are then applied during the inference phase, where each rule contributes a partial output according to its corresponding consequent function. In the ANFIS framework, these consequent functions are typically represented as linear equations that map input variables to output values. Each rule, therefore, produces a rule-specific output that reflects the relationship between the inputs and the predicted system response.

Following the inference stage, the model computes the global response by aggregating the outputs of all activated rules. This aggregation process combines the weighted contributions of individual rules into a single numerical value representing the overall model prediction. The global response reflects the integrated behavior of the entire fuzzy rule base and represents the final output generated by the ANFIS model.

Finally, statistical tests are computed on the resulting outputs to assess model performance, reliability, and significance. These tests help evaluate the accuracy of the predictions, the consistency of the model behavior, and the robustness of the inferred relationships between input variables and outputs. By applying statistical validation, the reliability of the ANFIS model can be assessed, and its appli-

cability to real-world problems can be confirmed. Together, these stages constitute a structured ANFIS inference pipeline that supports interpretable, data-driven modeling in complex systems.

2.6. Developing the Software Development Process Simulation Model

The proposed CMMN model for SDP simulation was implemented in the prototype using the Python programming language (<https://www.python.org/>) with the Mesa plugin (<https://mesa.readthedocs.io/en/stable/>) (Fig. 2.8). Mesa is used to simulate the developed CMMN model, as it provides a flexible framework for building agent-based models and managing interactions between agents, the environment, and system states. Its modular architecture enables the definition of agent behaviors, scheduling mechanisms, and data-collection processes, which are essential for representing the dynamic nature of SDP.

For implementing the proposed CMMN model, Flowable (<https://flowable.com/>) was used as the process execution engine. Flowable supports the execution of CMMN models and allows the management of case-based processes, including tasks, stages, and event-driven activities. Integration between Mesa and Flowable was achieved through REST API communication. Specifically, requests are sent from Mesa to the Flowable REST API to retrieve input data, which defines a sequence of SDP activities, their duration, and constraints in .csv format. This integration ensures that the simulation model is grounded in a formally defined process structure while allowing dynamic execution and adaptation during simulation.

The output data are saved as an electronic spreadsheet (.csv) and are unique during each simulation run, enabling systematic analysis of different simulation scenarios, comparison of results, and evaluation of process performance under varying conditions. Additionally, the use of standardized data formats facilitates interoperability with external analysis tools and supports reproducibility of experimental results.

The choice of the Python language was based on several specific advantages. As a dynamic, high-level programming language, Python is widely used across multiple domains, including data science, artificial intelligence, and simulation modeling. From the perspective of developers, various language features, such as readability, extensive libraries, and rapid prototyping capabilities, significantly influence programming experience and reduce the complexity of tasks such as debugging and system integration (Peng et al., 2021). Python was also selected to implement the simulation model due to its flexibility in representing complex, nonlinear relationships required for HF modeling. Consequently, Python provides

a robust and scalable environment for developing simulation-based solutions that combine process modeling, agent-based approaches, and data-driven techniques (Foramitti et al., 2021).

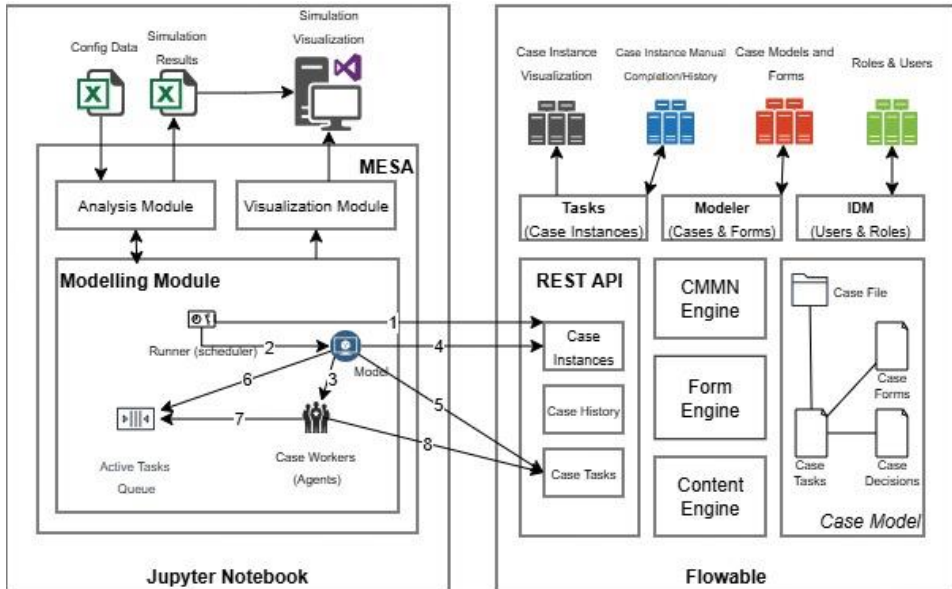


Fig. 2.8. Architecture of simulation model

The presented architecture integrates an agent-based simulation environment implemented in Jupyter Notebook using the MESA framework with the Flowable case management platform to enable dynamic interaction between simulated agents and workflow-driven processes. Within the Jupyter subsystem, the Modeling Module, comprising the simulation model, scheduler, and agent set, executes task-driven behaviors guided by an active task queue populated through interactions with Flowable. Analytical and visualization modules support the ingestion of configuration data, the production of simulation results, and the generation of graphical representations. Flowable provides complementary case management functionality, including REST-based interfaces for case instances, tasks, and histories, as well as execution engines for CMMN models, forms, and content. User-facing components facilitate case modeling, task monitoring, and identity management. Bidirectional communication between the simulation and Flowable enables case events, task assignments, and decision logic to influence agent behavior, while agent actions update case states in return. This integrated architecture supports the study of complex socio-technical processes by coupling rule-based workflow execution with adaptive agent-based simulation.

2.7. Performing Simulation Experiments

Agent-based modeling and simulation (ABMS) is a highly effective approach for replicating complex dynamic processes involving human actors. This methodology provides a distinct capability to model human behavior while incorporating influences from psychological factors, workflow sequences, and evolving conditions that impact business process outcomes. Commonly termed multi-agent modeling and simulation or multi-agent-based modeling and simulation, ABMS conceptualizes active components or decision-making entities as agents, whose behaviors are governed by the agent-based simulation paradigm within the framework of agent-based modeling. The agent-based simulation model consists of agents that operate according to predefined rules, pursuing specific objectives while interacting with their environment.

The fundamental principle of ABMS is the representation of emergent phenomena arising from multi-agent systems, making it particularly applicable to complex social systems, such as traffic detection. In this context, recent research highlights that ABMS is especially valuable in domains characterized by decentralized decision-making and high interaction complexity, such as the Vehicle Routing Problem. Laatabi et al. (2025) emphasized that agent-based approaches enable the modeling of autonomous entities (e.g., vehicles, customers, and dispatchers) that dynamically adapt to changing conditions, such as traffic fluctuations, time constraints, and stochastic events. This adaptability allows ABMS to capture real-world variability more effectively than traditional optimization techniques, which often rely on static assumptions. Furthermore, ABMS supports hybrid approaches, combining simulation with optimization or heuristic strategies, thereby enhancing solution quality and robustness in dynamic environments.

Constructing an agent-based model involves three critical phases: (1) identification and creation of agents, (2) definition of inter-agent relationships and environmental interactions, and (3) specification of the simulation environment. Laatabi et al. (2025) further noted that careful design of agent behaviors and interaction protocols is essential, as system-level outcomes emerge from local decision rules rather than centralized control. This bottom-up modeling paradigm enables the exploration of complex adaptive systems and facilitates experimentation with different coordination strategies among agents.

When simulating the SDP and depicting its dynamic execution through CMMN, the agent-based simulation approach emerges as the optimal choice, effectively representing the behavior of case executors and SDP roles across various implementation stages (Macal & North, 2010). In line with findings from the VRP domain, ABMS not only allows for realistic representation of individual

actor behavior but also supports scenario analysis, sensitivity testing, and evaluation of alternative process configurations under uncertainty, thereby strengthening its applicability in complex business process modeling.

The input data for this process consists of ANFIS-generated data, CMMN SDP models, and additional simulation input parameters. The ANFIS-generated data provides a data-driven foundation by capturing nonlinear relationships and adaptive patterns derived from historical or training datasets, thereby enhancing the realism and predictive capability of the simulation. Meanwhile, the CMMN SDP models define the structural and behavioral aspects of the case management process, including tasks, stages, milestones, and discretionary elements that influence process execution. These models serve as a blueprint for representing the dynamic and event-driven nature of the simulated system.

In addition, supplementary simulation input parameters, such as resource availability, execution time distributions, agent behavioral rules, and environmental constraints, are incorporated to further refine the simulation context and ensure alignment with real-world conditions. Together, these inputs enable the construction of a comprehensive and flexible simulation environment capable of capturing both deterministic and stochastic aspects of the process.

The output results include the developed simulation model and the corresponding simulation outcomes based on different CMMN SDP models. The simulation model itself represents an executable abstraction of the process, integrating agent behaviors, decision logic, and environmental interactions. The simulation outcomes provide detailed performance insights, such as process efficiency, resource utilization, task completion times, and system responsiveness under varying conditions. Furthermore, by comparing results across different CMMN SDP model configurations, it becomes possible to evaluate alternative process designs, identify bottlenecks, and assess the impact of structural or behavioral changes on overall system performance. This enables informed decision-making and supports process optimization in complex and dynamic environments.

2.8. Systematizing and Analyzing Simulation Results

Systematizing and analyzing simulation results is a critical step in validating findings and extracting meaningful insights. By conducting simulations with varying input parameters, a diverse set of outcome data was obtained, which was systematically examined to identify patterns, dependencies, and underlying relationships. This variability in input conditions enables a more comprehensive evaluation of system behavior under different scenarios, including typical and extreme cases, thereby increasing the robustness and generalizability of the results.

The analysis process involved applying statistical techniques, including the t-test (Afifah et al., 2022; Moore et al., 2016), to assess the significance of differences between datasets, ensuring the reliability of observed variations. In particular, the t-test allows for hypothesis testing by determining whether differences in sample means are statistically significant or likely due to random variation. This is especially important when comparing simulation outputs generated under different configurations or model assumptions.

The t-test is used to compare the sample mean to a known or hypothesized population mean:

$$t = (\bar{x} - \mu) / (s / \sqrt{n}), \quad (2.2)$$

where \bar{x} – sample mean, μ – population mean, s – sample standard deviation, and n – sample size.

By calculating the t-statistic and comparing it against critical values, it becomes possible to accept or reject the null hypothesis, thus providing a quantitative basis for evaluating differences between simulation scenarios. Statistical hypothesis testing provides a formal framework for decision-making, where sample data are used to determine whether observed differences are statistically significant or consistent with the null hypothesis (Rajić, 2026).

In addition to hypothesis testing, descriptive statistical measures, such as mean values, variance, and standard deviation, were utilized to summarize the distribution and dispersion of simulation outputs. These measures provide further insight into the stability and consistency of the modeled system. Moreover, sensitivity analysis was conducted to determine how changes in key input parameters influence the simulation results, thereby identifying the most impactful variables within the system.

Additionally, the comparison method was employed to evaluate trends across different simulation models, highlighting key similarities and deviations. This comparative analysis facilitated the identification of performance differences between alternative CMMN SDP model configurations and supported the detection of potential inefficiencies or structural limitations. Visualization techniques, such as graphs and charts, were also used to enhance interpretability and to clearly present trends and correlations within the data.

Through this structured and multi-layered analytical approach, actionable conclusions were derived, enhancing the accuracy, validity, and robustness of the simulation framework. Ultimately, this process supports informed decision-making by providing a reliable basis for evaluating alternative scenarios and optimizing system performance in complex and dynamic environments.

2.9. Conclusions of the Second Chapter

1. This chapter presents a fuzzy and case-based SDP modeling and simulation approach, enabling a comprehensive investigation of the impact of HF on SDP performance. This approach integrates real data preprocessing, CMMN-based modeling, and simulation modeling, creating a structured framework for evaluating various influences on software development outcomes.
2. One of the primary advantages of this approach is its utilization of real historical project data, which enhances the accuracy and relevance of results. By incorporating authentic project records, it was possible to predict HF-related impacts on SDP with a high degree of realism. Furthermore, the approach was systematically applied to two widely recognized SDP methodologies, “AGILE“ and WATERFALL, providing a comparative perspective on how task sequencing affects overall SDP performance. Through this comparative study, deeper insights into the role of HF in shaping process efficiency and adaptability were gained.
3. To rigorously validate the findings, a systematic analysis of simulation results was conducted, employing statistical methods such as the t-test to evaluate the significance of variations between datasets. Additionally, the comparison method was applied to examine trends across different simulation models, identifying consistent patterns and deviations that informed our conclusions. By leveraging these analytical techniques, a robust and well-supported assessment of the simulated HF effects on SDP methodologies was ensured.

In summary, this chapter established a structured approach for studying the impact of HF on SDP, reinforcing the importance of simulation-based assessments in software engineering research. The findings contribute to a deeper understanding of process efficiency, demonstrating how human-related variables interact with workflow dynamics across different development methodologies.

3

Experimental Investigation of Human Factor Impact on Software Development Process

This chapter presents an experimental investigation into the influence of HF on the SDP, combining simulation-based modeling with real historical data. The study uses a series of controlled simulation experiments, calibrated and validated using historical datasets, to analyze the relationship between HF and the resulting changes in SDP. Unlike traditional physical testing, this approach enables systematic exploration of parameter interactions and their short-term effects on system behavior. The experimental framework and numerical procedures were developed based on the methodology outlined in the Second Chapter. Results from both simulated scenarios and real-world data comparisons are presented to derive insights into the underlying dynamics and validate the observed effects. The results of the Third Chapter were disseminated through two scientific publications in peer-reviewed journals indexed in the Clarivate Analytics Web of Science database, including one Q2 quartile journal (Sielskaitė & Kalibatiėnė, 2023) and one Q3 quartile journal (Sielskaitė & Kalibatiėnė, 2025). In addition, the research findings were presented at four scientific conferences, two held in Lithuania and two international events, namely, the International Baltic Conference on Digital Business and Intelligent Systems (2022), the Open Conference of Electrical, Electronic and Information Sciences eStream (2021), the Data Analysis

Methods for Software Systems conference (2021), and the International Conference on Information Systems Development (2021).

3.1. Planning the Experiment

Careful planning of the experiment is essential to ensure that the research objectives are clearly defined, the variables are properly identified and controlled, and the collected data are suitable for further analysis. In the context of SDP analysis, experimental planning is particularly important because multiple technical and human factors may influence project performance and development outcomes.

This study's experimental plan includes several key stages that guide the preparation and execution of the research. First, relevant input parameters are selected based on both theoretical considerations and the availability of empirical data. These parameters primarily represent HF. By selecting these parameters, the experiment aims to analyze how different combinations of human-related characteristics influence the overall efficiency of SDP execution.

The experimental design also involves defining the output metrics used to evaluate the simulation results. In this research, the primary output indicator is the SDP, which reflects the level of deviation between the planned and the actual execution of project tasks. The SDP metric provides a quantitative measure for assessing project efficiency and identifying factors that may contribute to delays or deviations in the development process.

Another important component of the experimental plan is the design of simulation scenarios based on historical project data. Historical data provides realistic information about task execution times, employee performance, and project conditions observed in real software development environments. Using such data allows the simulation model to represent real-world situations more accurately and ensures that the experimental scenarios reflect practical development conditions rather than purely theoretical assumptions.

In addition, the experimental plan incorporates procedures for calibrating and validating the simulation model. Model calibration is performed to adjust the parameters of the simulation so that its behavior closely corresponds to the patterns observed in historical data. After calibration, the model is validated to ensure that it accurately represents the studied processes and can reliably reproduce the relationships between the selected input variables and the resulting performance indicators. Validation is an important step that increases the credibility of the simulation results and confirms that the model can be used for further analysis. Finally, once the model has been calibrated and validated, the influence of the selected HF on the SDP can be systematically analyzed. Different simulation scenarios are executed to evaluate how variations in HF affect the resulting performance metrics.

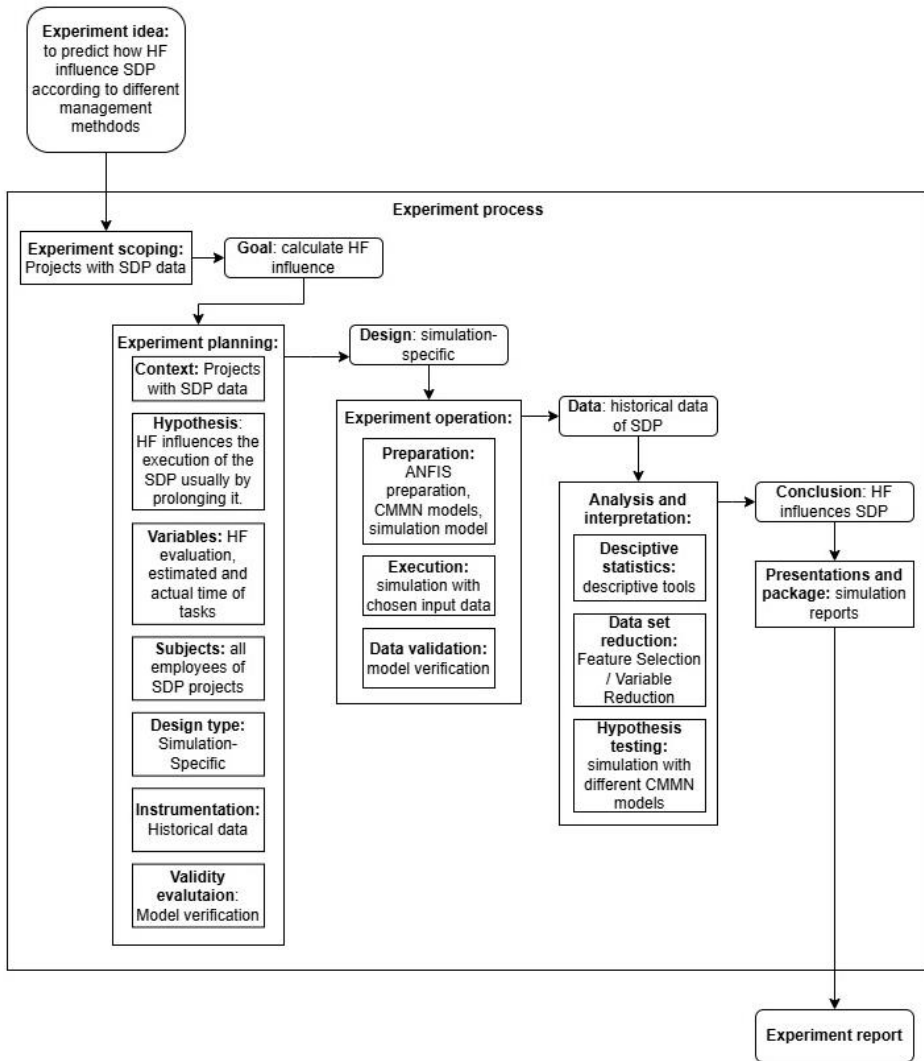


Fig. 3.1. Model of the experiment process

3.2. Experimental Investigation

In this study, data is required prior to conducting simulations to develop the simulation model itself. Therefore, when designing the experiment, it is also necessary to plan for the collection of initial data that will serve as input for the simulation model. Wohlin et al. (2012) stated that the experimental design framework

addresses only the phase in which the simulation model has already been developed. However, this study must take an additional step before the experiment can begin, namely, the implementation of the proposed model.

3.2.1. Experiment Data

Data for the experiment was collected from several IT organizations of different sizes and levels of complexity, ranging from small private companies to large international corporations. According to the classification proposed by Ayyagari et al. (2007), small organizations were defined as companies with up to 49 employees, medium organizations as those with 50–249 employees, and large organizations as companies with more than 250 employees. Organizations of varying sizes helped ensure that the collected dataset reflects a broader range of development environments and organizational structures, therefore improving the study's representativeness.

The analyzed data were obtained from software development projects related to the development and maintenance of information systems, web-based applications, and enterprise software solutions. The duration of the analyzed projects varied from several months to multiple years, depending on project scope and organizational context. Development teams typically consisted of software developers, testers, system analysts, and project managers, with team sizes ranging from small teams of three to five members to larger teams of more than ten participants. Such variation in project scale and team composition allowed the dataset to capture diverse development conditions and dynamics.

The main criteria for data collection included the estimated and actual task execution times for each employee (Table 3.1), which enabled the calculation of task performance deviations. In addition, expert evaluations were used to assess three key human factors that may influence project performance: motivation, experience, and availability (Table 3.2). These evaluations were provided by project managers or team leaders familiar with the employees' work performance and responsibilities. Based on the collected information, the average task deviation for each employee was calculated, providing a quantitative measure used in subsequent experimental analysis.

The data from projects of different types, durations, and team compositions increases the generalizability of the research results. It ensures that the experimental dataset reflects realistic software development environments.

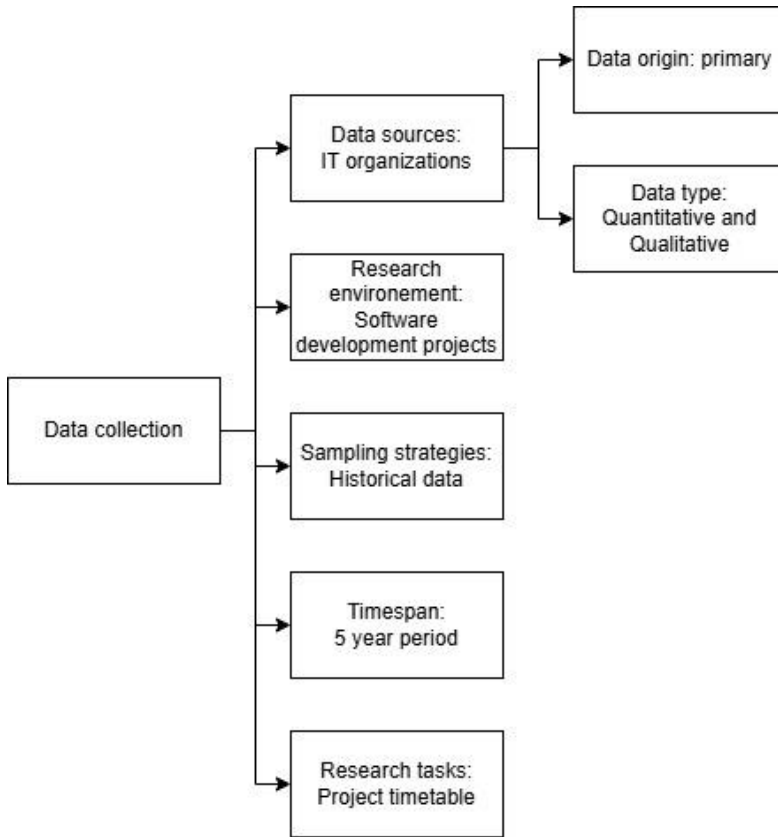


Fig. 3.2. Model of experiment data collection

Figure 3.2 illustrates the structure of the data used in the experiment and the main characteristics of the data collection process. The data were collected from IT organizations involved in software development projects, representing the research environment of the study. The dataset consists of primary data and includes both quantitative and qualitative information related to project activities and outcomes. Historical project data were used as the main sampling strategy, enabling the analysis of past project performance and development patterns. The collected data covers a five-year period, providing a sufficiently long timeframe to observe trends and variations in project execution. The primary research task associated with the dataset is the analysis of project timetables, which supports the evaluation of project planning and scheduling processes within software development projects.

Table 3.1. Snapshot of collected data

Task No.	Name of task	Actual (in hours)	Estimated (in hours)	Deviation
X1	Y1	182	258	71%
X2	Y2	5.5	10	55%
X3	Y3	6.5	6.5	100%
X4	Y4	95	136	70%
X5	Y5	10	14.5	69%
Average deviation of tasks for employee				95%

Deviation was calculated as the ratio of actual to estimated time (Table 3.1). If this ratio exceeds 100%, then actual time exceeded estimated time; on the contrary, if it is less than 100%, then actual time was lower than estimated time. Experts and stakeholders of every IT project under investigation were asked to evaluate their employees based on the HFs of motivation, experience, and availability by assigning a score from 1 to 5 as follows: Excellent (5), Good (4), Moderate (3), Low (2), or None (1). Experts were chosen using criteria that were the most suitable and related the person to the project team. The final column in Table 3.2 presents the average deviation of tasks for each employee from Table 3.1.

Table 3.2. Snapshot of summarized data for all employees

Employee	Motivation	Experience	Availability	Average deviation of tasks for employees
Z1	2	2	5	298%
Z2	2	4	5	125%
Z3	4	3	4	217%
Z4	3	5	1	98%
Z5	4	4	2	195%

The HFs that were analyzed were assessed to ensure that the experimental data did not correlate (Table 3.3). The correlation matrix illustrates the relationships among three key HFs: motivation, experience, and availability. As expected, motivation and experience show a moderately strong positive correlation ($r = 0.65$), indicating that individuals with higher motivation levels tend to have more experience, or vice versa. This suggests a potential reinforcing effect where experience may contribute to intrinsic motivation, or motivated individuals are more likely to accumulate experience over time. In contrast, availability is negatively correlated with motivation ($r = -0.12$) and experience ($r = -0.39$). While

the correlation with motivation is relatively weak and likely negligible, the moderate negative correlation with experience may imply that more experienced individuals tend to have less availability, possibly due to increased responsibilities or greater involvement in multiple projects. These findings highlight meaningful interdependencies among HF, which should be carefully considered in modeling their influence on SDP.

Table 3.3. Correlation of input data

	Motivation	Experience	Availability
Motivation	1	0.647	-0.121
Experience	0.647	1	-0.388
Availability	-0.121	-0.388	1

To process the obtained data using ANFIS, the percentage values of deviation were transformed into metrics from 1 to 5 (i.e., None (5), Low (4), Moderate (3), High (2), or Very high (1)) by applying the rules in Table 3.4. This transformation was necessary to standardize the input values and make them compatible with the fuzzy inference framework used in the ANFIS model. The metric scale represents different levels of deviation severity and allows the model to interpret numerical deviations in a structured and interpretable manner. In this scale, higher metric values indicate lower deviation levels and therefore better performance, while lower metric values represent higher deviation levels and greater inconsistency in task completion.

Table 3.4. Rules of splitting the data

Average deviation of tasks for employees	Metrics for ANFIS
($+\infty$; 200)	1
[200; 155)	2
[155; 125)	3
[125; 100)	4
[100; $-\infty$)	5

These rules define the intervals of average task deviation for employees and map them to the corresponding ANFIS metric values. The purpose of this discretization process is to reduce variability in raw numerical data while preserving the essential information required for model training and analysis. By grouping values into defined intervals, the dataset becomes more suitable for fuzzy rule generation

and improves the interpretability of the resulting model. The defined intervals represent different ranges of average deviation in employee task performance. Extremely large deviations fall into the lowest metric category, indicating very high deviation levels, whereas smaller deviations correspond to higher metric values. In other words, the mapping ensures that the ANFIS system can distinguish between different levels of deviation severity while maintaining a consistent representation of the data across all observations.

A final ANFIS data snapshot is presented in Table 3.5.

Table 3.5. Final data snapshot for ANFIS

Motivation	Experience	Availability	Deviation
2	2	5	1
2	4	5	3
4	3	4	1
4	5	1	4
4	4	2	2

The final data snapshot illustrates several sample records from the dataset, where each row represents an individual observation and each column corresponds to one of the HF variables used in the model. The input HF variables (motivation, experience, and availability) describe the characteristics of employees participating in software development tasks, while the deviation metric represents the categorized level of task completion deviation derived from the original percentage values.

3.2.2. Modeling the Software Development Process with Case Management Model and Notation

The development of the CMMN model for simulating the SDP aimed to represent all pertinent tasks and allocate them based on the respective process stages. To achieve this, the model was designed to reflect the logical structure and sequence of activities in the software development lifecycle. Each task in the model corresponds to a specific activity performed by project participants and is organized according to the stage of the development process in which it typically occurs. This structured representation allows the model to capture dependencies between tasks, decision points, and potential variations in process execution. Furthermore, CMMN enables modeling of flexible, dynamic processes, in which certain activities may be triggered depending on specific conditions or contextual factors.

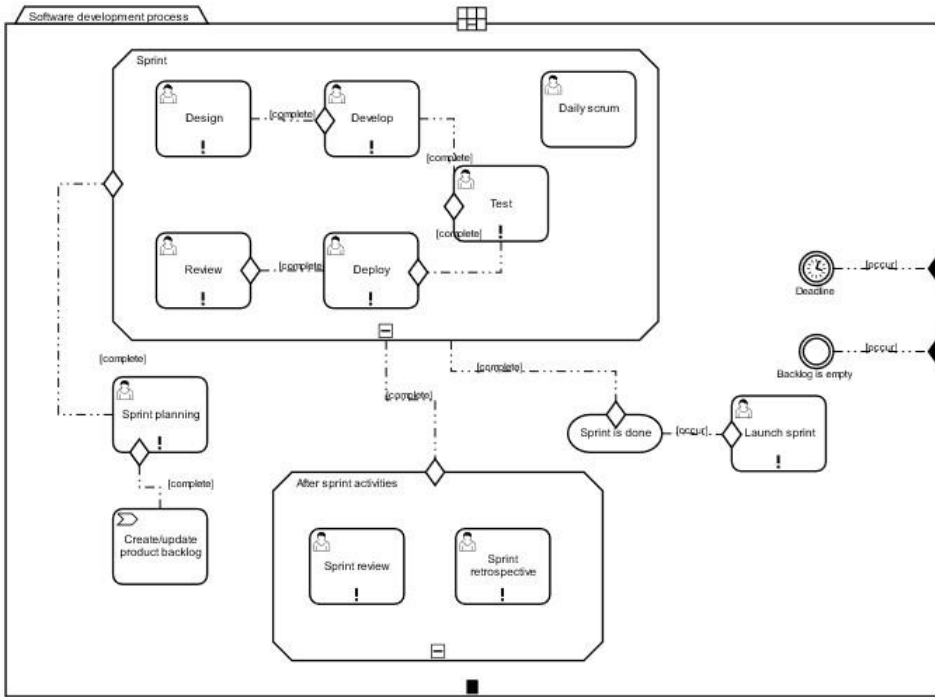


Fig. 3.3. Architecture of a simulation model for an AGILE SDP

Figure 3.3 presents the classical AGILE SDP model in CMMN, which is validated with Flowable (Fig. 3.4). The model captures the iterative and incremental nature of AGILE software development by representing key activities, stages, and decision points within a case-based structure. CMMN enables flexible process execution, allowing tasks to be triggered based on events and contextual conditions rather than predefined sequences, which aligns well with AGILE principles. The validation using Flowable ensures that the proposed model is not only conceptually correct but also executable within a real process engine. By deploying the model in Flowable, it becomes possible to verify the correctness of task flows, dependencies, and event-handling mechanisms, as well as to test different execution scenarios. This step confirms that the model accurately reflects CMMN and can be reliably used for further simulation and analysis.

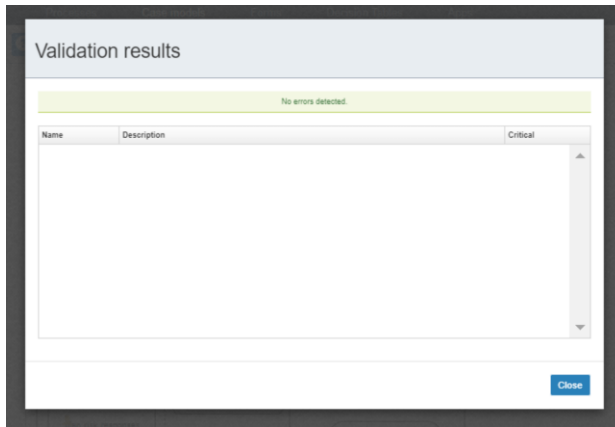


Fig. 3.4. Model validation results in Flowable

The main idea of this model was to represent the activities of the AGILE model and their dynamic iterative execution. The process begins with the preparation phase, where the product backlog is created or updated. This backlog contains the list of tasks, features, and requirements that need to be implemented during the development process. Following this step, sprint planning is conducted, during which the development team selects tasks from the product backlog and defines the objectives for the upcoming sprint.

Once the sprint is launched, the development activities take place within the sprint cycle. The sprint includes several key development tasks such as design, development, testing, review, and deployment. These activities are interconnected and may involve iterative feedback loops depending on the completion status of the tasks. Daily scrum meetings are conducted to monitor progress, identify obstacles, and ensure effective communication among team members.

After the completion of sprint tasks, the process proceeds to post-sprint activities. These activities include the sprint review and sprint retrospective. During the sprint review, the development team presents the completed work and evaluates whether the sprint goals were achieved. The sprint retrospective focuses on analyzing the development process, identifying potential improvements, and optimizing team performance for future iterations.

The model also includes several decision points that determine the continuation of the development cycle. For instance, if the sprint is considered completed successfully, a new sprint may be launched. Alternatively, if the backlog becomes empty or a project deadline is reached, the development process may be terminated. This iterative structure allows the SDP to adapt to changing requirements and continuously improve the delivered product through repeated development cycles.

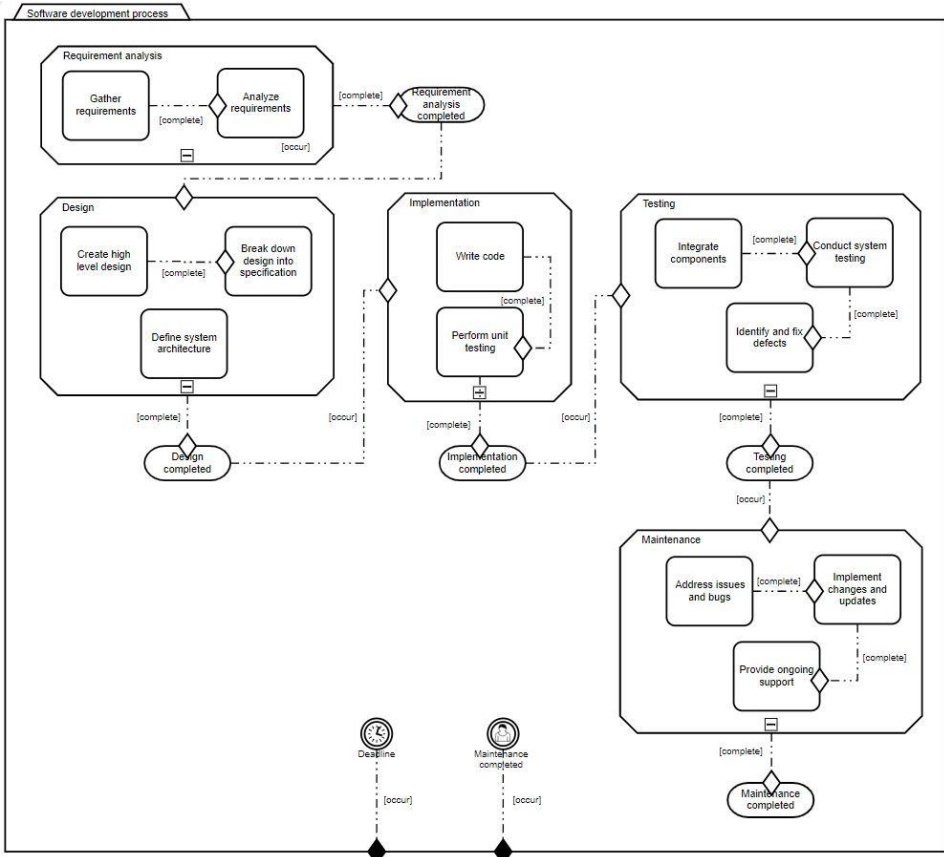


Fig. 3.5. Architecture of a simulation model for a WATERFALL SDP

Figure 3.5 presents the classical WATERFALL SDP model in CMMN, which is validated with Flowable. The main idea of this model was to represent the activities of the WATERFALL model and their sequential execution. The process begins with the requirement analysis phase, where the primary objective is to identify and understand the needs of stakeholders. During this stage, requirements are gathered and analyzed to determine the functional and non-functional characteristics of the system. The collected information forms the foundation for the subsequent development activities. Once the requirements have been thoroughly analyzed and validated, the process moves to the next stage.

The design phase focuses on translating the identified requirements into a structured system architecture. At this stage, developers create a high-level design that outlines the main components and their interactions. The system architecture

is defined to ensure that the software structure supports scalability, maintainability, and performance requirements. The design is then refined by breaking it down into detailed specifications that guide the implementation phase.

Following the design stage, the process proceeds to the implementation phase, where the actual development of the software system takes place. Developers write source code based on the previously defined specifications and system architecture. During this stage, unit testing is performed to verify that individual components of the system function correctly and meet the expected requirements. Successful completion of the implementation phase indicates that the core functionality of the system has been developed and verified at the component level.

After implementation, the process enters the testing phase, which aims to ensure the overall quality and reliability of the system. Components developed during the implementation phase are integrated, and system testing is conducted to verify that all parts of the system work together correctly. During testing, defects and inconsistencies may be identified and subsequently corrected. This iterative testing and debugging process helps improve the stability and performance of the software before release.

Finally, the development lifecycle transitions into the maintenance phase, which continues after the system deployment. In this phase, developers address issues and bugs that arise during system operation, implement updates and improvements, and provide ongoing technical support. Maintenance ensures that the software remains functional, secure, and aligned with evolving user requirements.

3.2.3. Adaptive Neuro-Fuzzy Inference System in Software Development Process Simulation

HF data were loaded into ANFIS, and FIS optimization was subsequently performed. To this end, a dataset was generated from real-world SDP data, and possible deviations from the estimated task execution time value were assigned to different combinations of HF parameters. The potential for deviation values was proportionately allocated in accordance with the ratio of estimated to actual time performance. Depending on these values, explanations were added to possible numerical deviations (Table 3.6).

The table defines the categorization of possible deviation values based on numeric intervals, providing a structured interpretation of deviations between estimated and actual time. Lower numeric values indicate very high to high positive deviations, where the actual time substantially exceeds the estimated time, reaching up to 210% above the initial estimates. As the numeric value increases, the level of deviation decreases, reflecting improved estimation accuracy. A numeric value of 5 represents no significant deviation, as the actual time is slightly below the estimated time, indicating a minor underestimation of approximately 10%.

Table 3.6. Explanations of possible numerical deviations

Numeric value	Possible deviation value	Explanation
[0; 1.8)	Very high	210% above estimated time
[1.8; 2.8)	High	74% above estimated time
[2.8; 3.8)	Moderate	39% above estimated time
[3.8; 5)	Low	14% above estimated time
5	None	10% below estimated time

Table 3.7. Hyperparameters for ANFIS

Hyperparameters	Description/Value
Fuzzy structure/FIS training data	Sugeno/genfis1
Generation of the FIS object	grid partition on the data
MF type (Input/Output)	Trimf/linear
Number of variables (inputs/outputs)	3/1
Optimization method	backpropagation
Maximum number of training epochs	200
Initial step size	0.001
AndMethod/OrMethod/ImpMethod/ AggMethod/DefuzzMethod	prod/probor/prod/sum/wtaver
Data for training/Data for testing	80/20
Number of rules	27

The ANFIS used in this study is a Sugeno-type model generated using the genfis1 function with grid partitioning applied to the training data. It employs triangular membership functions (trimf) for inputs and linear functions for outputs, with a total of three input variables and one output. The training process utilizes the backpropagation optimization method, with a maximum of 200 training epochs and an initial step size of 0.001. The fuzzy logic operators are configured as follows: product (prod) for the AND method, probabilistic OR (probor) for the OR method, product (prod) for the implication method, summation (sum) for the aggregation method, and weighted average (wtaver) for defuzzification. The dataset is divided into 80% for training and 20% for testing. Based on the input space partitioning, a total of 27 fuzzy rules were generated.

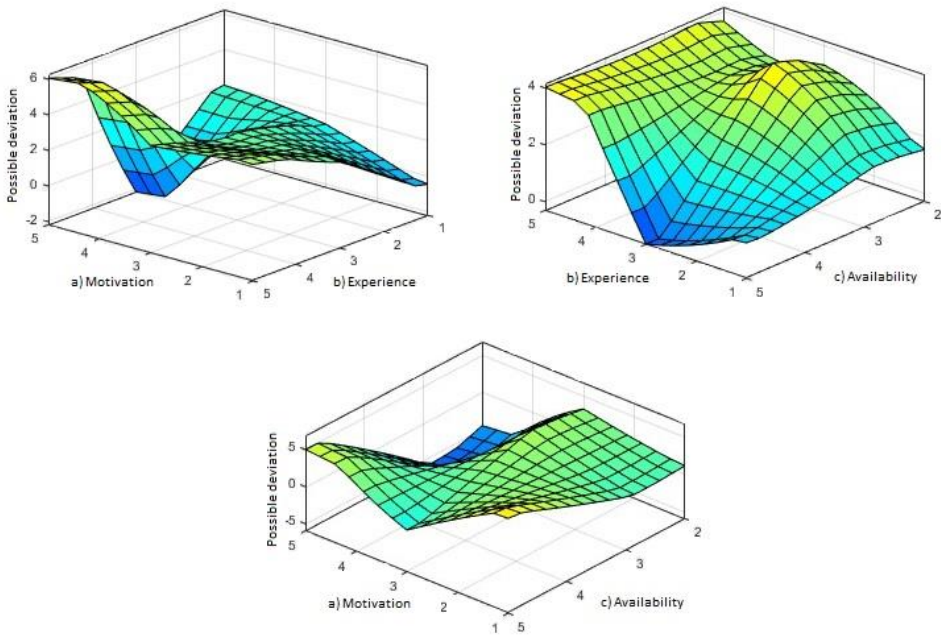


Fig. 3.6. Resulting surfaces of optimized HFs in the SDP – a prediction based on different inputs

After examining the generated schemas (Fig. 3.6), it becomes apparent that the impact of various HFs on the SDP is not uniform. Different HF influence the system in varying degrees, indicating that some variables have a stronger effect on the resulting deviations than others. Among these HFs, experience stands out as particularly influential, exhibiting a heightened sensitivity toward potential deviations. This observation suggests that individuals' levels of experience play a crucial role in determining both the likelihood and the magnitude of deviations within the system. As a result, higher experience levels are generally associated with lower deviation values and more stable project performance.

Conversely, while the motivation factor is also identified as a significant determinant, its effect appears to be comparatively less pronounced. The generated schemas suggest that the influence of motivation alone is not as strong as that of experience when predicting deviations. This implies that although individuals' motivational states may contribute to variations in performance, they are less decisive in determining deviation levels than the accumulated knowledge and skills represented by experience. Overall, the analysis highlights the importance of considering multiple HFs simultaneously, while emphasizing that experience remains one of the most critical contributors to reducing potential deviations in SDP.

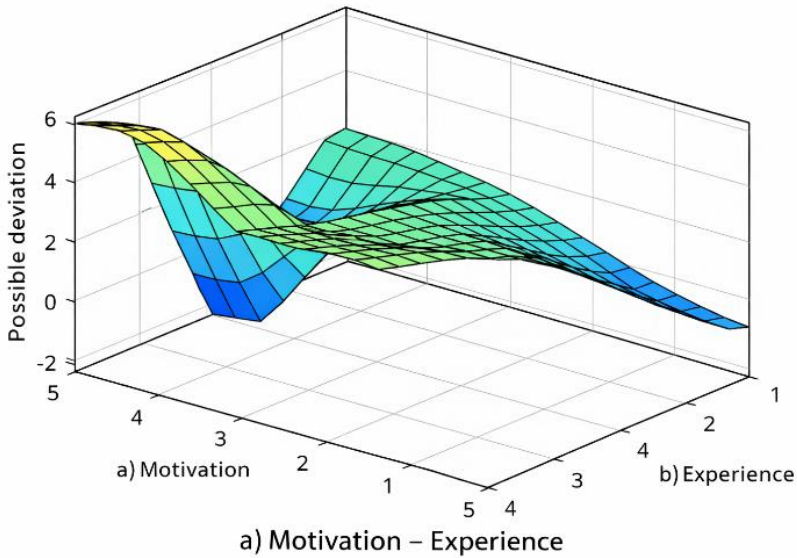


Fig. 3.7. Resulting surfaces of optimized HFs in the SDP (motivation and experience)

Figure 3.7 illustrates the interaction between motivation and experience and their combined effect on possible deviation. The results indicate that higher levels of motivation combined with greater experience are associated with lower deviation values, suggesting more accurate task time estimation and improved task performance. Conversely, low motivation together with limited experience leads to higher deviation values, indicating a greater risk of estimation inaccuracies and potential delays in task completion. The surface demonstrates a nonlinear relationship between the analyzed variables, implying that experience moderates the influence of motivation on deviation. Employees with higher experience levels tend to maintain more stable deviation values even when motivation varies, whereas less experienced employees are more sensitive to changes in motivation. This suggests that experience not only enhances individual capability but also acts as a stabilizing factor in performance under varying motivational conditions. Consequently, improving both motivation and experience levels may be critical for reducing uncertainty in task estimation and ensuring more consistent project outcomes.

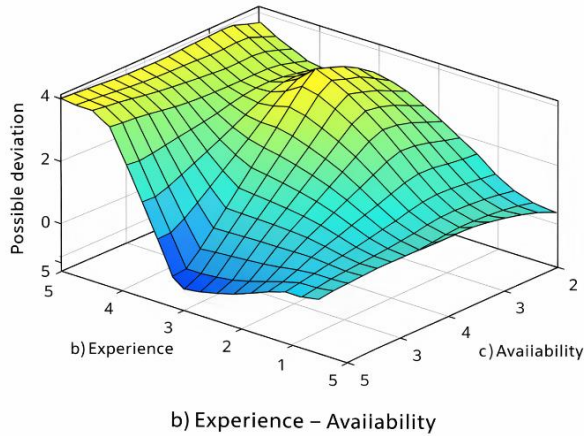


Fig. 3.8. Resulting surfaces of optimized HF's in the SDP (experience and availability)

Figure 3.8 shows how experience and availability jointly influence possible deviation. Higher experience levels generally correspond to reduced deviation; however, this effect is strengthened when availability is also high. Low availability combined with moderate experience results in increased deviation, highlighting the importance of resource availability in accurate time estimation. Overall, the surface suggests that experience alone is insufficient without adequate availability.

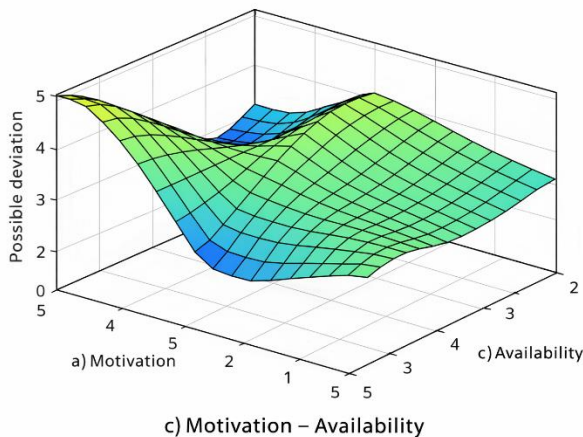


Fig. 3.9. Resulting surfaces of optimized HF's in the SDP (motivation and availability)

Figure 3.9 presents the combined effect of motivation and availability on possible deviation. Higher motivation and greater availability are associated with lower deviation values, indicating improved estimation accuracy. In contrast, low motivation, particularly when coupled with limited availability, leads to increased deviation. The surface reveals that availability plays a stabilizing role, reducing the negative impact of lower motivation on estimation outcomes.

Understanding these nuanced relationships between HFs and deviations is essential for devising effective strategies to mitigate risks and enhance system reliability. Within this investigation, the Fuzzy Inference System (FIS) employed three input parameters, each characterized by three fuzzy sets. This configuration resulted in a comprehensive rule set comprising 27 rules, forming the basis for constructing the ANFIS model. Fuzzy rules are presented in Table 3.8.

Table 3.8. Fuzzy rules

IF HF1	AND HF2	AND HF3	THEN PD (possible deviation)
Low	Low	Low	Very high
Low	Low	Moderate	Very high
Low	Low	Good	High
Low	Moderate	Low	Very high
Low	Moderate	Moderate	High
Low	Moderate	Good	Moderate
Low	Good	Low	High
Low	Good	Moderate	Moderate
Low	Good	Good	Low
Moderate	Low	Low	Very high
Moderate	Low	Moderate	High
Moderate	Low	Good	Moderate
Moderate	Moderate	Low	High
Moderate	Moderate	Moderate	Moderate
Moderate	Moderate	Good	Low
Moderate	Good	Low	Moderate
Moderate	Good	Moderate	Low
Moderate	Good	Good	Low
Good	Low	Low	High
Good	Low	Moderate	Moderate
Good	Low	Good	Low
Good	Moderate	Low	Moderate

End of Table 3.8

IF HF1	AND HF2	AND HF3	THEN PD (possible deviation)
Good	Moderate	Moderate	Low
Good	Moderate	Good	None
Good	Good	Low	Low
Good	Good	Moderate	None
Good	Good	Good	None

It is crucial to highlight that the exponential expansion of the partition size within the input space correlates directly with a substantial increase in the number of influential rules. This expansion significantly enhances the speed of system learning and application.

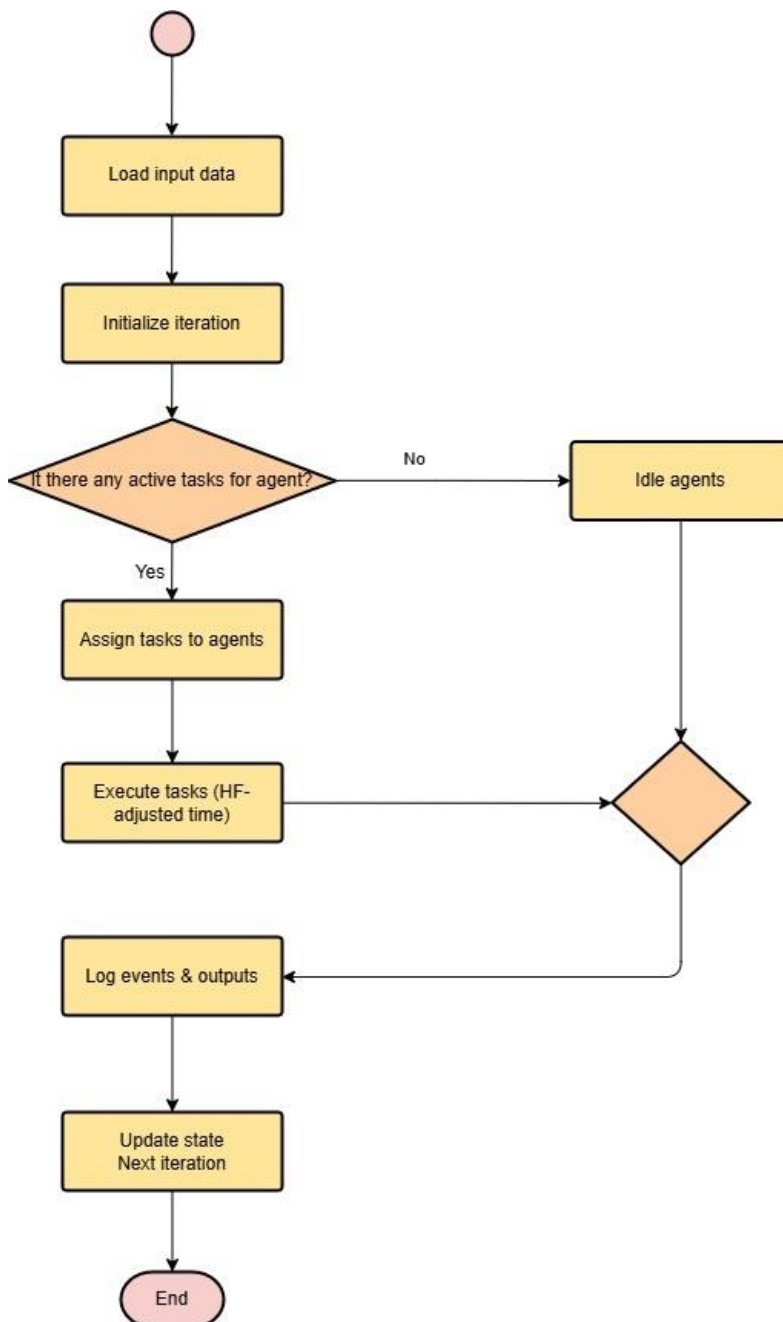
3.2.4. Software Development Process Simulation Results

The simulation is performed as described in the model (Fig. 3.10). The model defines the sequence of activities, decision points, and interactions between different process elements that occur during the execution of the simulated SDP. This allows the simulation to reflect the iterative and adaptive nature of a real-world SDP.

During the simulation, the predefined input parameters, including the selected HF and task-related variables, are applied to reproduce realistic SDP execution conditions. These parameters influence agent behavior, task execution time, and decision-making processes, thereby introducing variability and uncertainty into the simulation environment.

Furthermore, the simulation is executed iteratively across multiple scenarios, allowing the systematic variation of input parameters and enabling comparative analysis. This approach supports the identification of patterns, trends, and sensitivities within the system, particularly in relation to the impact of HF on task performance and process outcomes. The generated output data, such as task durations, deviations from planned values, and overall process efficiency, provide a quantitative basis for evaluating different configurations of the SDP.

As a result, the simulation enables the analysis of how different parameter values influence the overall performance of the development process. It also supports scenario testing, what-if analysis, and the assessment of potential improvements, thereby providing valuable insights for optimizing process design and enhancing decision-making in software project management.

**Fig. 3.10.** Simulation model

In each iteration, agents are assigned to the relevant SDP tasks that are currently available to them. If no active tasks are identified for a particular agent during the iteration, a message is displayed, and the agent remains idle until the next iteration. Throughout the simulation, the event log entries provide a detailed record for each iteration. This log allows the following to be monitored: the number and nature of tasks identified for a specific agent in each iteration, along with their assigned roles; the initiation of tasks; and the identification of agents that did not encounter any active tasks in the iteration. The simulation's output data is stored in electronic spreadsheet format and includes information such as the number of iterations, creation time, priority, task details, assigned agent, estimated time, actual time, and deviation. The quantity of stored output data is directly influenced by the input data. The output data from the simulation reveals the number of iterations, specifies which agent performed each task, indicates when the task commenced, provides the estimated time for the task, and records the actual time taken to complete the task. Deviation shows how the HF of each agent impacts task execution time.

First, it was necessary to determine whether HF has an influence on the SDP. Therefore, separate simulations were conducted using identical input parameters, both with and without HF data, to enable a direct comparison of the results.

Table 3.9. Comparison of the results obtained with and without HF

Task	Agent	Estima- ted	Actual Time for Task Exe- cution without HF	Actual Time for Task Exe- cution with HF
create product backlog	Agent 2	2	1	2.0
sprint planning	Agent 4	2	1	2.4
review	Agent 5	1	1	1.2
design	Agent 1	4	1	4.8
develop	Agent 5	10	7	12
test	Agent 3	3	1	2.4
daily scrum	Agent 5	1	1	3.59
deploy	Agent 5	2	5	1.2
sprint retrospective	Agent 4	2	1	1.2
launch sprint	Agent 5	1	5	2.4
sprint review	Agent 4	2	2	2.4

These results demonstrate that actual task execution times may vary depending on HF. For instance, as shown in Table 11, Agent 5 requires more time than initially expected to complete the development task due to the HF influence. In contrast, the deployment task is completed significantly faster, despite being performed by the same agent with identical HF values. Overall, these findings indicate that the prediction and computation of the HF enable a more accurate simulation of the SDP, as task durations reflect systematic changes in the HF rather than random variation.

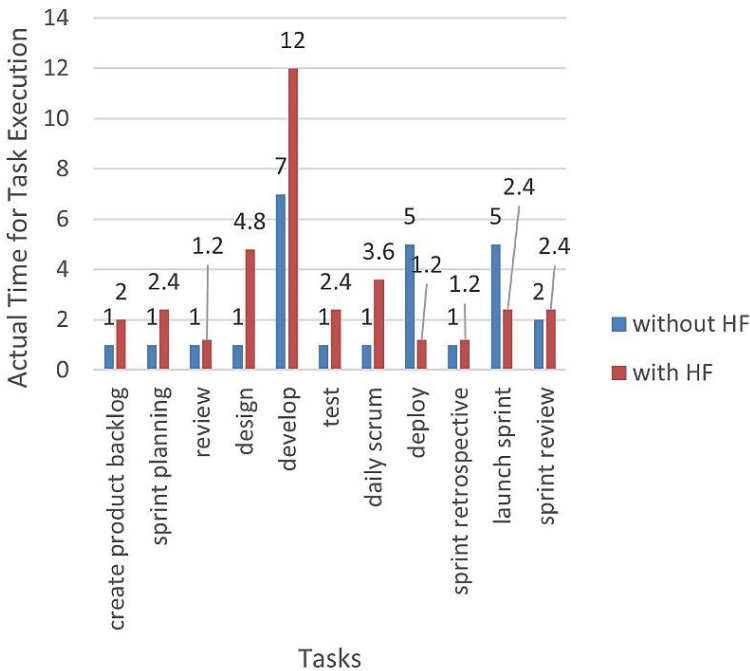


Fig. 3.11. Comparison of simulation results in a diagram

Figure 3.11 presents the actual execution time of selected tasks within an AGILE development process under two simulation scenarios: without HF and with HF included. The horizontal axis represents the individual AGILE process tasks, while the vertical axis denotes the actual time required for task execution.

The results show that the inclusion of HF leads to notable variations in task execution times compared to the baseline scenario without HF. Tasks such as development, design, and daily coordination activities exhibit increased execution times when HF are considered, indicating the sensitivity of these activities to human-related influences.

In contrast, certain tasks, including deployment and launch sprint, are completed more efficiently in the HF-based scenario, despite being performed by the same agents. This suggests that HF may positively affect performance for specific AGILE tasks.

Overall, Figure 3.11 demonstrates that incorporating HF into the simulation provides a more realistic and differentiated representation of task execution times within the SDP. Task durations reflect systematic changes driven by HF rather than random variation, thereby improving the accuracy of the SDP simulation.

Different simulations were conducted with identical HF calculations and historical project input data according to the AGILE and WATERFALL SDP models.

Table 3.10. Results of AGILE simulation

Task	Agent	Estimated	Actual	Deviation
Create product backlog	Agent 2	2	4.2	2.1
Launch sprint	Agent 5	1	1.74	1.74
Sprint planning	Agent 4	2	2.28	1.14
Deploy	Agent 5	2	3.48	1.74
Design	Agent 1	4	6.96	1.74
Develop	Agent 5	10	17.4	1.74
Review	Agent 5	1	1.74	1.74
Test	Agent 3	3	2.78	0.93
Daily scrum	Agent 5	1	1.74	1.74
Sprint retrospective	Agent 4	2	2.28	1.14
Launch sprint	Agent 5	1	1.74	1.74
Sprint review	Agent 42	2	2.28	1.14
Amount: 31				Amount: 48.62
Deviation				157%

Table 3.10 presents the results of the AGILE simulation, comparing estimated and actual effort across individual AGILE tasks performed by different agents. The table shows that, for most tasks, the actual time exceeds the estimated time, indicating a systematic underestimation during the planning phase. Tasks such as Develop, Design, and Create product backlog exhibit particularly large discrepancies between estimated and actual values, reflecting higher execution complexity and uncertainty.

The cumulative estimated effort amounts to 31 units, whereas the total actual effort reaches 48.62 units, resulting in an overall deviation of 157%. This significant deviation suggests that agent-related factors and task-specific characteristics

have a considerable impact on estimation accuracy within the simulated AGILE environment. Overall, the results highlight the importance of incorporating dynamic agent behavior and task variability when assessing effort estimation in AGILE project simulations.

Table 3.11. Results of the WATERFALL simulation

Task	Agent	Estimated	Actual	Deviation
Gather requirements	Agent 5	5	6.95	1.39
Analyze requirements	Agent 1	20	34.8	1.74
Define system architecture	Agent 2	15	31.5	2.1
Create high-level design	Agent 2	10	21	2.1
Break down design into specification	Agent 2	10	21	2.1
Perform unit testing	Agent 4	3	3.42	1.14
Write code	Agent 4	35	39.9	1.14
Integrate components	Agent 4	10	11.4	1.14
Conduct system testing	Agent3	10	9	0.9
Identify and fix defects	Agent 3	10	9	0.9
Provide ongoing support	Agent 4	30	34.2	1.14
Address issues and bugs	Agent 1	5	8.7	1.74
Implement changes and updates	Agent 4	20	22.8	1.14
Amount: 183			Amount: 253.67	
Deviation			139%	

Table 3.11 presents the results of the WATERFALL simulation, comparing estimated and actual effort for sequential development tasks executed by different agents. The results indicate that actual effort exceeds the estimated values for most tasks, particularly in early and design-intensive phases such as Analyze requirements and Define system architecture. These phases show substantial discrepancies, suggesting that upfront planning activities in the WATERFALL model are especially prone to underestimation.

The total estimated effort amounts to 183 units, while the cumulative actual effort reaches 253.67 units, resulting in an overall deviation of 139%. Although the deviation remains high, it is lower than that observed in the AGILE simulation, indicating comparatively more stable estimation performance within the WATERFALL approach. Overall, the results emphasize the impact of task complexity

and agent involvement on effort estimation accuracy in traditional, plan-driven development environments.

As simulations were performed with the same HF input data, the results of deviation from estimated to actual time according to the selected methodology show the impact of the HF. Average deviation was calculated by summing the estimated and actual times for all tasks and dividing this number by the sum of the estimated times of all tasks to produce a deviation indicator. The results show that the AGILE SDP is more affected by the HF because it has a higher deviation indicator on average, with the same resources. Repeated iterations could affect actual execution times and be more strongly influenced by HF.

3.3. Statistical Analysis of Software Development Process Simulation Results

A statistical analysis was conducted to compare estimation accuracy between two project management approaches: AGILE and WATERFALL. The hypothesis and results of calculations are presented in Table 3.12.

Table 3.12. Applying t-test

<i>Input</i>	The four samples obtained simulating AGILE with and without HF, and WATERFALL with and without HF.
H_0	$\mu_{\text{withHF}} = \mu_{\text{withoutHF}}$,
H_1	$\mu_{\text{withHF}} \neq \mu_{\text{withoutH}}$
H_2	$\mu_{\text{withHF}} > \mu_{\text{withoutH}}$
<i>Conclusions</i>	Based on the paired-samples t-test performed on the provided AGILE sample ($t(10) = 1.14$, $p = 0.28$), the author failed to reject H_0 at $\alpha = 0.05$, i.e., the data do not show a statistically significant difference in mean task execution time when HF are included versus excluded. However, the empirical pattern in the simulations reveals task-specific and systematic changes when HF are modeled (some tasks lengthen, others shorten). These patterns support the substantive claim that HF affects the dynamics and realism of SDP simulations even if the global mean difference was not statistically significant in the present (small) sample.

A paired-samples t-test was conducted to evaluate whether the inclusion of HF led to statistically significant differences in task execution times within the AGILE development process. The analysis compared the actual execution times

of the same set of tasks simulated with and without HF. The results indicate that the difference in task execution times between the two conditions was not statistically significant, $t(10) = 1.14$, $p = 0.28$. Although tasks simulated with HF generally exhibited higher variability and, in some cases, longer execution times, the overall effect of HF on task duration did not reach statistical significance.

One-sample t-tests were conducted to assess whether the deviation values for AGILE and WATERFALL simulations significantly differ from zero. The results indicate that both methodologies exhibit statistically significant deviations from zero, confirming systematic differences between estimated and actual task durations. The AGILE simulation shows a mean deviation of 3.22 (SD = 1.50), while the WATERFALL simulation demonstrates a slightly higher mean deviation of 3.39 (SD = 1.88). These findings suggest that estimation inaccuracies are present in both development approaches within the simulated environment.

Table 3.13. One-sample t-test results and confidence level for deviation values (AGILE and WATERFALL)

Attributes	AGILE Values	WATERFALL Values
Sample size (N)	12	13
Mean deviation	3.22	3.39
Standard deviation	1.50	1.88
Test value	0	0
Degrees of freedom (DF)	11	12
t Stat	7.44	6.50
P($T \leq t$) two-tail	< 0.001	< 0.001
t critical two-tail ($\alpha = 0.05$)	2.201	2.179
Confidence level	0.95	0.95
95% confidence interval	[2.27; 4.16]	[2.26; 4.53]
Effect size (Cohen's d)	2.15	1.80

The results summarized in Table 3.13 demonstrate a statistically significant deviation from the baseline scenario for both AGILE and WATERFALL development processes. In both cases, the null hypothesis was rejected ($p < 0.001$), indicating that the observed deviations are not attributable to random variation. The large effect sizes (Cohen's $d = 2.15$ for AGILE and 1.80 for WATERFALL) further confirm a strong practical impact. These findings indicate that the HF

inclusion introduces systematic and substantial changes in process execution, affecting both development methodologies in a comparable manner.

The average relative deviation for AGILE tasks was 55.22% (standard deviation = 36.28), compared to 43.62% (standard deviation = 45.85) for WATERFALL tasks. While WATERFALL tasks exhibited larger absolute deviations, the relative estimation accuracy of both methodologies was broadly comparable. Notably, the higher relative deviations observed in AGILE tasks suggest greater sensitivity to HF. This increased variability may reflect the adaptive and less prescriptive nature of AGILE processes, which allows human-related influences to more directly affect task execution. Consequently, project managers operating in AGILE environments should place greater emphasis on managing human factors, as these factors can substantially influence process predictability and overall project performance. In contrast, the more structured and sequential nature of the WATERFALL approach appears to mitigate, though not eliminate, the HF impact on execution variability.

3.4. Conclusions of the Third Chapter

1. The experiments were conducted following a predefined experimental plan and aimed at examining the influence of HF on SDP execution using simulation. The study focused on classical SDP models, namely AGILE and WATERFALL, acknowledging that real-world implementations may involve hybrid configurations. The experimental scope was therefore intentionally limited to classical models to ensure methodological consistency and comparability.
2. The experimental design was guided by the following hypotheses: (H_0) the inclusion of HF does not result in statistically significant differences in SDP task execution times compared to simulations without HF; and (H_1) the inclusion of HF introduces systematic deviations in SDP task execution behavior, affecting task execution times and increasing variability relative to baseline simulations. In addition, an exploratory hypothesis was formulated to assess methodological sensitivity: (H_2) AGILE SDP simulations exhibit greater sensitivity to HF than WATERFALL SDP simulations.
3. Initial experiments, previously reported in (Sielskaitė & Kalibatienė, 2023), examined AGILE SDP simulations using randomly generated HF values. These simulations demonstrate that variations in HF systematically affect task execution behavior, leading to deviations from initially estimated task durations. The results indicated that changes in the overall

HF indicator were associated with non-random deviations in execution time, providing preliminary support for H_1 .

4. Subsequent experiments incorporated empirically derived HF data, as reported in (Sielskaitė & Kalibatienė, 2025), to assess the task-level impact of individual human factors. The results showed that different HF dimensions influence SDP execution in distinct and task-specific ways, further supporting the hypothesis that HFs introduce structured variability into process execution rather than uniform shifts in average duration.
5. Additional statistical analyses were conducted to evaluate the magnitude and consistency of HF-induced deviations across process models. For AGILE SDP simulations, paired-samples t-tests comparing execution times with and without HFs did not reveal a statistically significant global difference in mean task duration, and therefore H_0 could not be rejected at the aggregate level. However, task-level variability increased when HFs were included, which is consistent with H_1 . In contrast, one-sample t-tests applied to deviation measures for both AGILE and WATERFALL simulations identified statistically significant deviations from baseline conditions, accompanied by large effect sizes, indicating that HFs introduce substantial and systematic changes in SDP execution.
6. Finally, a comparative analysis of relative deviation metrics revealed higher relative deviations and greater variability in AGILE simulations compared to WATERFALL simulations. This observation provides empirical support for the exploratory hypothesis H_2 , suggesting that AGILE processes are more sensitive to HF due to their adaptive and less prescriptive structure.

For future research, several directions are proposed. First, the developed approach should be validated using alternative modeling techniques to compare performance and generalizability. Second, the current dissertation focused primarily on individual-level HF; therefore, future work should aim to expand the scope by investigating and modeling team dynamics and collective behaviors. Third, the current model could be further extended to include additional parameters or structural enhancements to increase its accuracy and adaptability. Fourth, instead of the ANFIS framework, deep learning techniques could be explored as a more scalable and potentially more powerful alternative. Lastly, it is important to continue collecting and enriching the dataset to support broader analysis and improve the robustness of the developed models

General Conclusions

This section presents general conclusions of the dissertation regarding the formulated tasks as follows:

1. Based on the performed literature review on the SDP modeling and simulation with HF incorporation, the analyzed approaches and gaps found in integrating HF into SDP simulations emphasize the need for more accurate and flexible modeling techniques to reflect human influence in software development. Also, the literature analysis found that the most influential HFs on the SDP are as follows: communication, leadership, job satisfaction, cognitive workload, and emotional well-being.
2. The newly proposed fuzzy and case-based SDP modeling and simulation approach allows for the investigation of the HF impact on the SDP, since its main components are real data preprocessing and the development of both CMMN-based models and a simulation model. A fuzzy inference-based approach was developed to predict the impact of HF on the SDP. This approach was designed to effectively handle the inherent uncertainty and variability in human behavior by using fuzzy logic principles. The model incorporates the three key factors, motivation, experience, and availability, as fuzzy input variables, enabling nuanced predictions of their combined effects on SDP performance metrics. This approach offers an

- improved and adaptable tool for predicting HF impact compared to traditional deterministic models.
3. Data was collected from real IT organizations based on completed IT projects. The resulting datasets included both the estimated task durations and the actual time taken to complete those tasks, along with individual HF indicators for each team member. The primary dataset was compiled from three different IT organizations and included detailed records of individual project tasks, their execution duration, performance evaluations, and associated specialists. This dataset enables the empirical validation of assumptions regarding the impact of specific HF attributes, such as motivation, experience, and availability, on deviations in task execution time. Since the data were derived from real-world IT projects, they provide a realistic foundation for conducting meaningful simulations and experiments, enhancing both the practical relevance and credibility of the research findings.
 4. The proposed fuzzy inference-based approach was implemented as a working prototype and experimentally evaluated using both simulated and empirically grounded HF data. The conducted experiments demonstrate that the proposed approach is feasible and enables the systematic investigation of HF influence within both AGILE and WATERFALL SDP contexts. The experimental results show that the inclusion of HF leads to structured and non-random changes in task execution behavior across both methodologies. While a statistically significant global difference in mean task duration was not consistently observed for all cases ($t(10) = 1.14$, $p = 0.28$), deviation-based analyses revealed substantial and statistically significant departures from baseline scenarios. Specifically, the total effort deviation reached 157% in the AGILE model and 139% in the WATERFALL model. One-sample t-test results confirmed statistically significant deviations from baseline in both cases ($p < 0.001$), with large effect sizes (Cohen's $d = 2.15$ for AGILE and 1.80 for WATERFALL). The average relative deviation was higher for AGILE tasks (55.22%, $SD = 36.28$) compared to WATERFALL tasks (43.62%, $SD = 45.85$), indicating greater variability and sensitivity to HF in AGILE environments. These quantitative results demonstrate that HF has a measurable and substantial impact on task execution and estimation accuracy. Overall, the experimental findings demonstrate that the proposed model can reliably represent HF-related effects and provide practical support for project managers and development teams in anticipating and managing risks associated with human resource variability in SDP planning and execution.

References

- Aamodt, A., & Plaza, E. (1994). Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Communications*, 7(1), 39–59.
- Agile Alliance. (2001). *Manifesto for AGILE software development*. <https://cir.nii.ac.jp/crid/1574231875735983616>
- Ali, N. B., & Unterkalmsteiner, M. (2023). Use and evaluation of simulation for software process education: A case study. *arXiv*. <https://doi.org/10.48550/arXiv.2307.12484>
- Afifah, S., Mudzakir, A., & Nandiyanto, A. B. D. (2022). How to calculate paired sample t-test using SPSS software: From step-by-step processing for users to practical examples in the analysis of the effect of application anti-fire bamboo teaching materials on student learning outcomes. *Indonesian Journal of Teaching in Science*, 2(1), 81–92. <https://doi.org/10.17509/ijotis.v2i1.45895>
- Law, A. M. (2015). *Simulation modeling and analysis*.
- Amrit, C., Daneva, M., & Damian, D. (2014). Human factors in software development: On its underlying theories and the value of learning from related disciplines. A guest editorial introduction to the special issue. *Information and Software Technology*, 56(12), 1537–1542. <https://doi.org/10.1016/j.infsof.2014.07.006>
- Russell, S. J., & Norvig, P. (2022). *Artificial intelligence: A modern approach (4th US ed.)*. Pearson. <https://aima.cs.berkeley.edu/>

- Askari, S. (2017). A novel and fast MIMO fuzzy inference system based on a class of fuzzy clustering algorithms with interpretability and complexity analysis. *Expert Systems with Applications*, *84*, 301–322. <https://doi.org/10.1016/j.eswa.2017.04.045>
- Aurino, D. E. M. (2000). Human factors and aviation safety: What the industry has, what the industry needs. *Ergonomics*, *43*(7), 952–959. <https://doi.org/10.1080/001401300409134>
- Ayyagari, M., Beck, T., & Demirgüç-Kunt, A. (2007). Small and medium enterprises across the globe. *Small Business Economics*, *29*(4), 415–434. <https://doi.org/10.1007/s11187-006-9002-5>
- Banks, J., Carson, J. S., Nelson, B. L., & Nicol, D. M. (2010). *Discrete-event system simulation* (5th ed.). Scientific Research Publishing.
- Barros, L., Tam, C., & Varajão, J. (2024). AGILE software development projects – unveiling the human-related critical success factors. *Information and Software Technology*, *170*, 107432. <https://doi.org/10.1016/j.infsof.2024.107432>
- Beecham, S., Baddoo, N., Hall, T., Robinson, H., & Sharp, H. (2008). Motivation in software engineering: A systematic literature review. *Information and Software Technology*, *50*(9–10), 860–878. <https://doi.org/10.1016/j.infsof.2007.09.004>
- Bernardi, S., Gómez, A., Merseguer, J., Perez-Palacin, D., & Requeno, J. I. (2022). DICE simulation: A tool for software performance assessment at the design stage. *Automated Software Engineering*, *29*(1), 36. <https://doi.org/10.1007/s10515-022-00335-z>
- Bershtein, L., Knyazeva, M., & Rozenberg, I. (2015). Fuzzy resource-constrained project scheduling for GIS software development. In *Proceedings of the 2015 Conference of the International Fuzzy Systems Association and the European Society for Fuzzy Logic and Technology* (pp. 1542–1548). <https://doi.org/10.2991/ifsa-eusflat-15.2015.219>
- Boehm, B. (1986). A spiral model of software development and enhancement. *ACM SIGSOFT Software Engineering Notes*, *11*(4), 14–24. <https://doi.org/10.1145/12944.12948>
- Boehm, B., & Turner, R. (2003). *Balancing agility and discipline: A guide for the perplexed*. Addison-Wesley.
- Bogdan-Alexandru Andrei, Andrei-Cosmin Căsu-Pop, Sorin-Cătălin Gheorghe, & Costin-Anton Boiangiu. (2019). A study on using WATERFALL and AGILE methods in software project management. *Journal of Information Systems & Operations Management*. <http://www.rebe.rau.ro/RePEc/rau/jisomg/SU19/JISOM-SU19-A12.pdf>
- Borshchev, A. (2013). *The big book of simulation modeling: Multimethod modeling with AnyLogic 6*. AnyLogic North America.
- Capretz, L. F. (2003). Personality types in software engineering. *International Journal of Human-Computer Studies*, *58*(2), 207–214. [https://doi.org/10.1016/S1071-5819\(02\)00137-4](https://doi.org/10.1016/S1071-5819(02)00137-4)

- Carayon, P. (2006). Human factors of complex sociotechnical systems. *Applied Ergonomics*, 37(4), 525–535. <https://doi.org/10.1016/j.apergo.2006.04.011>
- Carayon, P., Schoofs Hundt, A., Karsh, B.-T., Gurses, A. P., Alvarado, C. J., Smith, M., & Flatley Brennan, P. (2006). Work system design for patient safety: The SEIPS model. *Quality in Health Care*, 15(Suppl 1), i50–i58. <https://doi.org/10.1136/qshc.2005.015842>
- Carlos Eduardo Rodriguez. (2024). Evaluating the impact of human factors on aircraft maintenance errors: A risk-based analysis framework for business aviation. *World Journal of Advanced Engineering Technology and Sciences*, 13(2), 764–777. <https://doi.org/10.30574/wjaets.2024.13.2.0647>
- Chicco, D., Warrens, M. J., & Jurman, G. (2021). Supplemental information 1: Enhanced versions of Table 3 and Table 4 with standard deviation reported for each average result. In *PeerJ Computer Science*, 7, 1–24. PeerJ Inc. <https://doi.org/10.7717/peerj-cs.623/supp-1>
- Choi, B.-I., & Rhee, F. C.-H. (2009). Interval type-2 fuzzy membership function generation methods for pattern recognition. *Information Sciences*, 179(13), 2102–2122. <https://doi.org/10.1016/j.ins.2008.04.009>
- Cockburn, A., & Highsmith, J. (2001). AGILE software development, the people factor. *Computer*, 34(11), 131–133. <https://doi.org/10.1109/2.963450>
- Cockburn, A. (2007). *AGILE software development: The cooperative game* (2nd ed.). Addison-Wesley.
- Czarnecki, K., & Eisenecker, U. W. (2000). *Generative programming: Methods, tools, and applications*. Addison-Wesley.
- Davenport, T. (1993). *Process innovation: Reengineering work through information technology*. Harvard Business School Press.
- Demirel, H. O., Goldstein, M. H., Li, X., & Sha, Z. (2024). Human-centered generative design framework: An early design framework to support concept creation and evaluation. *International Journal of Human-Computer Interaction*, 40(4), 933–944. <https://doi.org/10.1080/10447318.2023.2171489>
- Dikert, K., Paasivaara, M., & Lassenius, C. (2016). Challenges and success factors for large-scale AGILE transformations: A systematic literature review. *Journal of Systems and Software*, 119, 87–108. <https://doi.org/10.1016/j.jss.2016.06.013>
- Dingsøy, T., Nerur, S., Balijepally, V., & Moe, N. B. (2012). A decade of AGILE methodologies: Towards explaining AGILE software development. *Journal of Systems and Software*, 85(6), 1213–1221. <https://doi.org/10.1016/j.jss.2012.02.033>
- Dubois, D. J. (1980). *Fuzzy sets and systems: Theory and applications* (Vol. 144). Academic Press.
- Dumas, M., La Rosa, M., Mendling, J., & Reijers, H. A. (2018). *Fundamentals of business process management* (2nd ed.). Springer.

- Dutra, E., Diirr, B., & Santos, G. (2021). Human factors and their influence on software development teams: A tertiary study. In *Proceedings of the Brazilian Symposium on Software Engineering* (pp. 442–451). <https://doi.org/10.1145/3474624.3474625>
- Dybå, T., & Dingsøyr, T. (2008). Empirical studies of AGILE software development: A systematic review. *Information and Software Technology*, 50(9–10), 833–859. <https://doi.org/10.1016/j.infsof.2008.01.006>
- Fernando Capretz, L. (2014). Bringing the human factor to software engineering. *IEEE Software*, 31(2), 104–104. <https://doi.org/10.1109/MS.2014.30>
- Fiandini, M., Nandiyanto, A. B. D., al Husaeni, D. F., al Husaeni, D. N., & Mushiban, M. (2023). How to calculate statistics for significant difference test using SPSS: Understanding students' comprehension on the concept of steam engines as power plant. *Indonesian Journal of Science and Technology*, 9(1), 45–108. <https://doi.org/10.17509/ijost.v9i1.64035>
- Foramitti, J. (2021). AgentPy: A package for agent-based modeling in Python. *Journal of Open Source Software*, 6(62), 3065. <https://doi.org/10.21105/joss.03065>
- Ford, D., Smith, J., Guo, P. J., & Parnin, C. (2016). Paradise unplugged: Identifying barriers for female participation on Stack Overflow. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering* (pp. 846–857). <https://doi.org/10.1145/2950290.2950331>
- GeeksforGeeks. (2025). Data preprocessing in data mining. <https://www.geeksforgeeks.org/data-science/data-preprocessing-in-data-mining/>
- Goncalves, W. F., de Almeida, C. B., de Araujo, L. L., Ferraz, M. S., Xandu, R. B., & de Farias, I. (2017). The influence of human factors on the software testing process: The impact of these factors on the software testing process. In *2017 12th Iberian Conference on Information Systems and Technologies (CISTI)* (pp. 1–6). <https://doi.org/10.23919/CISTI.2017.7975873>
- Gunatilake, H., Grundy, J., Hoda, R., & Mueller, I. (2024). The impact of human aspects on the interactions between software developers and end-users in software engineering: A systematic literature review. *Information and Software Technology*, 173, 107489. <https://doi.org/10.1016/j.infsof.2024.107489>
- Guo, S., Liu, H., Chen, X., Xie, Y., Zhang, L., Han, T., ... Wang, J. (2025). Optimizing case-based reasoning system for functional test script generation with large language models. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Vol. 2* (pp. 4487–4498). <https://doi.org/10.1145/3711896.373725>
- Gurung, G., Shah, R., & Jaiswal, D. P. (2020). Software development life cycle models: A comparative study. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 6(4), 30–37. <https://doi.org/10.32628/CSEIT206410>
- Guveyi, E., Aktas, M. S., & Kalipsiz, O. (2020). Human factor on software quality: A systematic literature review. In *Proceedings of the Conference on Computational*

- Science and Its Applications* (Vol. 12252, pp. 918–930). Springer. https://doi.org/10.1007/978-3-030-58811-3_65
- Harrison, R. L., Granja, C., & Leroy, C. (2010). Introduction to Monte Carlo simulation. <https://doi.org/10.1063/1.3295638>
- Highsmith, J. (2002). *AGILE software development ecosystems*. Addison-Wesley.
- Idri, A., Abran, A., & Khoshgoftaar, T. M. (2001, August). Fuzzy analogy: A new approach for software cost estimation. In *International Workshop on Software Measurement* (pp. 28–29).
- Idri, A., Abran, A., & Khoshgoftaar, T. M. (2002, June). Estimating software project effort by analogy based on linguistic values. In *Proceedings of the Eighth IEEE Symposium on Software Metrics* (pp. 21–30). IEEE. <https://doi.org/10.1109/METRIC.2002.1011322>
- Itkonen, J., Mäntylä, M. V., & Lassenius, C. (2013). The role of the tester’s knowledge in exploratory software testing. *IEEE Transactions on Software Engineering*, 39(5), 707–724. <https://doi.org/10.1109/TSE.2012.55>
- Jang, J.-S. R. (1993). ANFIS: Adaptive-network-based fuzzy inference system. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(3), 665–685. <https://doi.org/10.1109/21.256541>
- Jensen, K., & Kristensen, L. M. (2009). Introduction to modelling and validation. In *Coloured Petri Nets* (pp. 1–12). Springer. https://doi.org/10.1007/b95112_1
- Jimoh, R. G., Olusanya, O. O., Awotunde, J. B., Imoize, A. L., & Lee, C.-C. (2022). Identification of risk factors using ANFIS-based security risk assessment model for SDLC phases. *Future Internet*, 14(11), 305. <https://doi.org/10.3390/fi14110305>
- Johanyák, Z. C., & Zlatko, C. (2025). A comprehensive survey on fuzzy logic applications in software requirements engineering. *GRADUS*, 12(2). <https://doi.org/10.47833/2025.2.CSC.008>
- Kacker, R. N., & Lawrence, J. F. (2007). Trapezoidal and triangular distributions for Type B evaluation of standard uncertainty. *Metrologia*. <https://doi.org/10.1088/0026-1394/44/2/003/META>
- Kalibatiene, D., & Miliuskaitė, J. (2021a). A dynamic fuzzification approach for interval type-2 membership function development: Case study for QoS planning. *Soft Computing*, 25(16), 11269–11287. <https://doi.org/10.1007/s00500-021-05899-8>
- Kalibatiene, D., & Miliuskaitė, J. (2021b). A hybrid systematic review approach on complexity issues in data-driven fuzzy inference systems development. *Informatika*, 32(1), 85–118. <https://doi.org/10.15388/21-INFOR444>
- Kar, A. K. (2014). Revisiting the supplier selection problem: An integrated approach for group decision support. *Expert Systems with Applications*, 41(6), 2762–2771. <https://doi.org/10.1016/j.eswa.2013.10.009>

- Karatayev, A., Ogorodova, A., & Shamoï, P. (2024, May). Fuzzy inference system for test case prioritization in software testing. In *2024 IEEE 4th International Conference on Smart Information Systems and Technologies (SIST)* (pp. 352–357). IEEE. <https://doi.org/10.1109/SIST61555.2024.10629262>
- Kellner, M. I., Madachy, R. J., & Raffo, D. M. (1999). Software process simulation modeling: Why? What? How? *Journal of Systems and Software*, *46*(2–3), 91–105.
- Kolus, A., Wells, R., & Neumann, P. (2018). Production quality and human factors engineering: A systematic review and theoretical framework. *Applied Ergonomics*, *73*, 55–89. <https://doi.org/10.1016/j.apergo.2018.05.010>
- Kolus, A., Wells, R. P., & Neumann, W. P. (2023). Examining the relationship between human factors related quality risk factors and work related musculoskeletal disorder risk factors in manufacturing. *Ergonomics*, *66*(7), 954–975. <https://doi.org/10.1080/00140139.2022.2119285>
- Korkmaz, M. (2021). A study over the general formula of regression sum of squares in multiple linear regression. *Numerical Methods for Partial Differential Equations*, *37*(1), 406–421. <https://doi.org/10.1002/num.22533>
- Laatabi, A., Gaudou, B., Hanachi, C., & Stolf, P. (2025). A survey on the use of agent-based modeling and simulation for the vehicle routing problem. *ACM Computing Surveys*, *58*(4), 1–34. <https://doi.org/10.1145/3769070>
- Law, A. M. (2007). *Simulation modeling and analysis* (4th ed.). McGraw-Hill.
- le Dinh, T., Le, T. D., Uwizeyemungu, S., & Pelletier, C. (2025). Human-centered artificial intelligence in higher education: A framework for systematic literature reviews. *Information*, *16*(3), 240. <https://doi.org/10.3390/info16030240>
- Li, Y., Zhang, H., Dong, L., Liu, B., & Yang, L. (2024, September). An experience report on modeling software process in industrial context: Challenges and solutions. In *Proceedings of the 2024 International Conference on Software and Systems Processes* (pp. 1–12). <https://doi.org/10.1145/3666015.3666024>
- Liu, B., Zhang, H., Dong, L., Wang, Z., & Li, S. (2024). Metrics for software process simulation modeling. *Journal of Software: Evolution and Process*, *36*(11), e2676. <https://doi.org/10.1002/smr.2676>
- Luiz Fernando Capretz, & Faheem Ahmed. (2010). Making sense of software development and personality types. *IT Professional*. https://www.eng.uwo.ca/electrical/faculty/capretz_l/docs/publications/IEEE-IT-Professional-v2.pdf
- Macal, C. M., & North, M. J. (2010). Tutorial on agent-based modelling and simulation. *Journal of Simulation*, *4*(3), 151–162. <https://doi.org/10.1057/jos.2010.3>
- Machuca-Villegas, L., Gasca-Hurtado, G. P., Morillo Puente, S., & Restrepo Tamayo, L. M. (2021). An instrument for measuring perception about social and human factors that influence software development productivity. *JUCS - Journal of Universal Computer Science*, *27*(2), 111–134. <https://doi.org/10.3897/jucs.65102>

- Machuca-Villegas, L., Gasca-Hurtado, G. P., & Muñoz, M. (2021). Measures related to social and human factors that influence productivity in software development teams. *International Journal of Information Systems and Project Management*, 9(3), 43–67. <https://doi.org/10.12821/ijispm090303>
- Mahaju, S., Carver, J. C., & Bradshaw, G. L. (2023). Human error management in requirements engineering: Should we fix the people, the processes, or the environment? *Information and Software Technology*, 160, 107223. <https://doi.org/10.1016/j.infsof.2023.107223>
- Malone, P. (2011). Review of *Mastering the unpredictable: How adaptive case management will revolutionize the way that knowledge workers get things done*, by K. D. Swenson. *Journal of Applied Management and Entrepreneurship*, 16(1), 134.
- Mamdani, E. H. (1974). Application of fuzzy algorithms for control of a simple dynamic plant. *Proceedings of the Institution of Electrical Engineers*, 121(12), 1585–1588. <https://doi.org/10.1049/piee.1974.0328>
- Meyer, A. N., Fritz, T., Murphy, G. C., & Zimmermann, T. (2014). Software developers' perceptions of productivity. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering* (pp. 19–29). <https://doi.org/10.1145/2635868.2635892>
- Miliauskaitė, J., & Kalibatiene, D. (2020). Complexity in data-driven fuzzy inference systems: Survey, classification and perspective. *Baltic Journal of Modern Computing*, 8(4), 572–596. <https://doi.org/10.22364/bjmc.2020.8.4.08>
- Montevecchi, J. A. B., Leal, F., de Pinho, A. F., Costa, R. F., de Oliveira, M. L. M., & Silva, A. L. F. (2010, December). Conceptual modeling in simulation projects by mean adapted IDEF: An application in a Brazilian tech company. In *Proceedings of the 2010 Winter Simulation Conference* (pp. 1624–1635). IEEE. <https://doi.org/10.1109/WSC.2010.5678908>
- Moore, D. S., McCabe, G. P., & Craig, B. A. (2016). *Introduction to the practice of statistics* (9th ed.). W. H. Freeman.
- Muhammad, A. P., Knauss, E., Bärghman, J., & Knauss, A. (2023, December). Continuous experimentation and human factors: An exploratory study. In *International Conference on Product-Focused Software Process Improvement* (pp. 511–526). Springer Nature Switzerland.
- Muhammad, A. P., Knauss, E., Bärghman, J., & Knauss, A. (2024). Continuous experimentation and human factors. In *Proceedings of the International Conference on Product-Focused Software Process Improvement 2023* (Vol. 14483, pp. 511–526). Springer Science and Business Media Deutschland GmbH. https://doi.org/10.1007/978-3-031-49266-2_35
- Ockiya, T. F., & Lock, R. (2023, November). A review of human factors in remote software project management: A progressive look at human based issues in remote software development environments. In *Proceedings of the 2023 12th International*

- Conference on Software and Information Engineering* (pp. 15–21). <https://dl.acm.org/doi/10.1145/3634848.3634858>
- Ouifak, H., & Idri, A. (2025). A comprehensive review of fuzzy logic based interpretability and explainability of machine learning techniques across domains. *Neurocomputing*, *647*, 130602.
- Palmquist, M. S., Lapham, M. A., Miller, S., Chick, T., & Ozkaya, I. (2013). *Parallel worlds: AGILE and WATERFALL differences and similarities* (No. CMU-SEI2013TN021).
- Peng, Y., Zhang, Y., & Hu, M. (2021). An empirical study for common language features used in Python projects. In *2021 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)* (pp. 24–35). <https://doi.org/10.1109/SANER50967.2021.00012>
- Pirzadeh, L. (2010). *Human factors in software development: A systematic literature review*. <https://odr.chalmers.se/bitstreams/db3f3124-1e37-408a-ac31-82f694de260a/download>
- Rajić, V. (2026). Statistical hypothesis testing: A comprehensive review of theory, methods, and applications. *Mathematics*, *14*(2). <https://doi.org/10.3390/math14020300>
- Robinson, S. (2006). Conceptual modeling for simulation: Issues and research requirements. In *Proceedings of the 2006 Winter Simulation Conference* (pp. 792–800). IEEE. <https://doi.org/10.1109/WSC.2006.323160>
- Rozinat, A., Mans, R. S., Song, M., & van der Aalst, W. M. (2009). Discovering simulation models. *Information Systems*, *34*(3), 305–327. <https://doi.org/10.1016/j.is.2008.09.002>
- Ruiz, M., & Salanitri, D. (2019). Understanding how and when human factors are used in the software process: A text-mining based literature review. In *Proceedings of the International Conference on Product-Focused Software Process Improvement* (Vol. 11915, pp. 694–708). Springer. https://doi.org/10.1007/978-3-030-35333-9_54
- Russo, D., & Stol, K. J. (2020). Gender differences in personality traits of software engineers. *IEEE Transactions on Software Engineering*, *48*(3), 819–834. <https://doi.org/10.1109/TSE.2020.3003413>
- Salvendy, G. (Ed.). (2012). *Handbook of human factors and ergonomics*. John Wiley & Sons.
- Sasaki, K., Sakamoto, S., Uchida, H., Shigeta, T., Matsunami, M., Kanazawa, H., Fukuda, A., Nakazawa, A., Sato, M., Ito, S., Horikawa, R., Yokoi, T., Azuma, N., & Kasahara, M. (2015). Two-step transplantation for primary hyperoxaluria: A winning strategy to prevent progression of systemic oxalosis in early onset renal insufficiency cases. *Pediatric Transplantation*, *19*(1), E1–E6. <https://doi.org/10.1111/petr.12376>
- Schwaber, K. (2004). *AGILE project management with Scrum*. Microsoft Press.

- Serrador, P., & Pinto, J. K. (2015). Does AGILE work? A quantitative analysis of AGILE project success. *International Journal of Project Management*, 33(5), 1040–1051. <https://doi.org/10.1016/j.ijproman.2015.01.006>
- Sommerville, I. (2016). *Software engineering* (10th ed.). Pearson.
- Smith, T. J. (2007). The ergonomics of learning: Educational design and learning performance. *Ergonomics*, 50(10), 1530–1546. <https://doi.org/10.1080/00140130701587608>
- Standish Group. (2021). *Standish Chaos Report 2021: Success through SAFE*. <https://www.successthroughsafe.com/blog-1/2021/11/13/standish-chaos-report-2021>
- Stone, N. J. (2008). Human factors and education: Evolution and contributions. *Human Factors*, 50(3), 534–539. <https://doi.org/10.1518/001872008X288466>
- Sujan, M., Pickup, L., Bowie, P., Hignett, S., Ives, F., Vosper, H., & Rashid, N. (2021). The contribution of human factors and ergonomics to the design and delivery of safe future healthcare. *Future Healthcare Journal*, 8(3), e574–e579. <https://doi.org/10.7861/fhj.2021-0112>
- Takagi, T., & Sugeno, M. (1985). Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man, and Cybernetics*.
- Tartarisco, G., Carbonaro, N., Tonacci, A., Bernava, G. M., Arnao, A., Crifaci, G., Cipresso, P., Riva, G., Gaggioli, A., de Rossi, D., Tognetti, A., & Pioggia, G. (2015). Neuro-fuzzy physiological computing to assess stress levels in virtual reality therapy. *Interacting with Computers*, 27(5), 521–533. <https://doi.org/10.1093/iwc/iwv010>
- Terry Ruas, & William Irvin Grosky. (2020). *Software engineering: A practitioner's approach* (9th ed.). https://www.researchgate.net/publication/365946272_Software_Engineering_A_Practitioner's_Approach_9_Th_Edition
- Thesing, T., Feldmann, C., & Burchardt, M. (2021). AGILE versus WATERFALL project management: Decision model for selecting the appropriate approach to a project. *Procedia Computer Science*, 181, 746–756. <https://doi.org/10.1016/j.procs.2021.01.227>
- Software Solutions Studio. (2021). *Types of simulation models: Choosing the right approach for a simulation project*. <https://softwaresim.com/blog/types-of-simulation-models-choosing-the-right-approach-for-your-simulation-project/>
- Urquhart, L., Wodehouse, A., Loudon, B., & Fingland, C. (2022). The application of generative algorithms in human-centered product development. *Applied Sciences*, 12(7), 3682. <https://doi.org/10.3390/app12073682>
- van der Aalst, W. M. P., Weske, M., & Grünbauer, D. (2005). Case handling: A new paradigm for business process support. *Data & Knowledge Engineering*, 53(2), 129–162. <https://doi.org/10.1016/j.datak.2004.07.003>

- Vasilecas, O., Kalibatiene, D., & Lavbič, D. (2016). Rule- and context-based dynamic business process modelling and simulation. *Journal of Systems and Software*, 122, 1–15. <https://doi.org/10.1016/j.jss.2016.08.048>
- Vieira, J., Dias, F. M., & Mota, A. (2004). Artificial neural networks and neuro-fuzzy systems for modelling and controlling real systems: A comparative study. *Engineering Applications of Artificial Intelligence*, 17(3), 265–273. <https://doi.org/10.1016/j.engappai.2004.03.001>
- Wagner, S., & Ruhe, M. (2018). A systematic review of productivity factors in software development. <https://doi.org/10.48550/arXiv.1801.06475>
- Wang, J., & Kumar, A. (2005). A framework for document-driven workflow systems. In *Lecture Notes in Computer Science* (Vol. 3649, pp. 285–301). Springer. https://doi.org/10.1007/11538394_19
- Wickens, C. D., Lee, J. D., Liu, Y., & Becker, S. E. G. (2004). *An introduction to human factors engineering*. Pearson.
- Winkler, D., Hametner, R., Östreicher, T., & Biffel, S. (2010). A framework for automated testing of automation systems. In *2010 IEEE 15th Conference on Emerging Technologies & Factory Automation (ETFA 2010)* (pp. 1–4). <https://doi.org/10.1109/ETFA.2010.5641264>
- Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., & Wesslén, A. (2012). *Experimentation in software engineering*. Springer. <https://doi.org/10.1007/978-3-642-29044-2>
- Wooley, A., Silva, D. F., & Bitencourt, J. (2023). When is a simulation a digital twin? A systematic literature review. *Manufacturing Letters*, 35, 940–951. <https://doi.org/10.1016/j.mfglet.2023.08.014>
- Ye, X., Zhao, Z., Leake, D., Wang, X., & Crandall, D. (2021). Applying the case difference heuristic to learn adaptations from deep network features. <https://doi.org/10.48550/arXiv.2107.07095>
- Zadeh, L. A. (1965). Fuzzy sets. *Information and Control*, 8(3), 338–353. [https://doi.org/10.1016/S0019-9958\(65\)90241-X](https://doi.org/10.1016/S0019-9958(65)90241-X)
- Zadeh, L. A. (1975). The concept of a linguistic variable and its application to approximate reasoning-III. *Information Sciences*, 9(1), 43–80. [https://doi.org/10.1016/0020-0255\(75\)90017-1](https://doi.org/10.1016/0020-0255(75)90017-1)
- Zhang, H., Kitchenham, B., & Pfahl, D. (2010). Software process simulation modeling: An extended systematic review. In *Proceedings of the International Conference on Software Process* (Vol. 6195, pp. 309–320). Springer. https://doi.org/10.1007/978-3-642-14347-2_27
- Zimmermann, H.-J. (2011). *Fuzzy set theory—and its applications* (4th ed.). Springer.

List of Scientific Publications by the Author on the Topic of the Dissertation

Papers in the Reviewed Scientific Journals

Sielskaitė, Š., & Kalibatiėnė, D. (2023). On fuzzy and case-based dynamic software development process modeling and simulation approach. *Applied Sciences*, 13(11), 6603. <https://doi.org/10.3390/app13116603>

Sielskaitė, Š., & Kalibatiėnė, D. (2025). The impact of human factors on software development processes applying the AGILE and WATERFALL methodologies: A case study using real data. *International Journal of Computers, Communication & Control*, 21(2), Article 6807. <https://doi.org/10.15837/ijccc.2025.4.6807>

Papers in Other Editions

Sielskaitė, Š. (2022) Comparison of business process modeling approaches. In *Proceedings of the Baltic DB&IS 2022 – The 15th International Baltic Conference on Databases and Information Systems*. <https://ceur-ws.org/Vol-3158/paper1.pdf>

Sielskaitė, Š. & Kalibatiėnė, D. (2021). On CMMN model for software system project simulation. In *Proceedings of the ISD 2021: 29th International Conference on Information Systems Development*. <https://doi.org/10.1109/eStream53087.2021.9431516>

Sielskaitė, Š. (2021). On CMMN model for software system project simulation. In *2021 IEEE Open Conference of Electrical, Electronic and Information Sciences (eStream)*. <https://doi.org/10.1109/eStream53087.2021.9431516>

Summary in Lithuanian

Įvadas

Problemos formulavimas

Programinės įrangos kūrimo procesą (SDP) veikia tiek metodologijos, tiek žmogiškasis faktorius, kuris, nepaisant struktūruotų požiūrių, tokių kaip AGILE ir WATERFALL, gali reikšmingai paveikti projektų rezultatus (Grenning, 2001; Andrei et al., 2019). Dėl savo sudėtingumo ir neapibrėžtumo žmogiškasis faktorius išlieka sunkiai apibrėžiamas ir kiekybiškai įvertinamas, todėl yra plačiai tiriamas (Dutra et al., 2021; Gunatilake et al., 2024). Jo poveikis skiriasi priklausomai nuo SDP, todėl svarbu analizuoti žmogiškąjį faktorių įvairių metodologijų kontekste. Atsižvelgiant į žmogiškojo faktoriaus būdingą neapibrėžtumą ir ribotą realių duomenų taikymą esamuose tyrimuose, ši disertacija sprendžia mokslinį klausimą, kaip formalizuoti žmogiškojo faktoriaus neapibrėžtumą taikant neapibrėžtųjų aibių (angl. *fuzzy*) metodus ir kaip pritaikyti atvejų valdymo modeliavimo technikas SDP simuliacijai, naudojant realius IT organizacijų žmogiškojo faktoriaus duomenis.

Darbo aktualumas

Šios disertacijos aktualumas grindžiamas nepakankamai ištirtu žmogiškojo faktoriaus poveikiu programinės įrangos kūrimo procesui (SDP). Nors tokios metodologijos kaip AGILE ir WATERFALL suteikia struktūruotus požiūrius, žmonių elgsena įneša kintamumo, kuris reikšmingai veikia projektų rezultatus ir prisideda prie nuolat išliekančių

nesėkmingų projektų rodiklių (Standish Group, 2021; Barros et al., 2024; Ockiya & Lock, 2023). Esamos metodologijos apibrėžia vaidmenis, procesus ir sąveikas, tačiau nepakankamai atskleidžia, kaip žmogiškasis faktorius daro įtaką SDP veikimui (Andrei et al., 2019). Be to, vis dar trūksta išsamaus supratimo apie žmogiškojo faktoriaus poveikį skirtinguose SDP modeliuose (Andrei et al., 2019; Gunatilake et al., 2024).

Atsižvelgiant į tai, kad žmogiškasis faktorius daro įtaką visiems SDP etapams per komunikaciją, sprendimų priėmimą ir gebėjimą prisitaikyti, jo kintamumas sukuria neapibrėžtumą ir rizikas, kurias sudėtinga valdyti (Dutra et al., 2021). Siekiant užpildyti šią spragą, disertacijoje siūloma modeliuoti žmogiškojo faktoriaus neapibrėžtumą taikant neapibrėžtųjų aibių logiką ir simuliuoti SDP veikimą naudojant realius duomenis, taip suteikiant vertingų įžvalgų tiek moksliniams tyrimams, tiek praktiniam taikymui, ypač IT organizacijoms, veikiančioms dinamiškoje aplinkoje.

Tyrimo objektas

Žmogiškųjų veiksmų poveikis programinės įrangos kūrimo proceso efektyvumui AGILE ir WATERFALL paradigmosse.

Darbo tikslas

Praplėsti programinės įrangos kūrimo planavimą, išplečiant jo modeliavimą ir simuliacijos procesą, integruojant žmogiškąjį faktorį.

Darbo uždaviniai

Darbo tikslui pasiekti sprendžiami šie uždaviniai:

1. Atlikti mokslinės literatūros apžvalgą, analizuojant programinės įrangos kūrimo proceso (SDP) modeliavimo ir simuliacijos tyrimus, integruojant žmogiškojo faktoriaus (HF) aspektus.
2. Pasiūlyti neraiškiųjų aibių teorija grindžiamą metodą, skirtą HF poveikio SDP prognozuoti.
3. Atlikti empirinių duomenų rinkimą iš realių IT organizacijų, siekiant nustatyti HF įtaką SDP, remiantis praktinių projektų įžvalgomis.
4. Sukurti ir įdiegti siūlomą metodą prototipo pavidalu bei atlikti eksperimentinį tyrimą, siekiant įvertinti siūlomo metodo tinkamumą prognozuojant HF poveikį SDP, naudojant realius duomenis.

Tyrimų metodika

Tyrimo objektui taikytas įvairių metodų kompleksas. Informacijos rinkimas, sisteminiams, analizė ir palyginimas taikyti siekiant išnagrinėti ir pristatyti įžvalgas, susijusias su neraiškiųjų aibių teorijos taikymu bei mašininio mokymosi metodais grindžiamu prognozavimu. Loginis samprotavimas, konceptų apibendrinimas bei koncepcinis modeliavimas

taikyti vertinant esamus metodus ir kuriant neraiškiųjų aibių teorija bei mašininio mokymosi pagrindu grindžiamą metodą HF poveikiui SDP prognozuoti.

Koncepcinis modeliavimas, projektavimas, įgyvendinimas, eksperimentavimas ir statistinė analizė naudoti siūlomo metodo tikslumui ir tinkamumui įvertinti. Eksperimentinis vertinimas buvo atliekamas naudojant realius duomenis, surinktus iš IT organizacijų.

Darbo mokslinis naujumas

Disertacijos originalus mokslinis indėlis ir mokslinis naujumas:

1. Pasiūlytas naujas metodas, skirtas HF SDP prognozuoti, integruojant neryškiąją logiką su atvejų grįsta simuliacija. Sukurtas atvejų valdymo modelio ir notacijos (angl. *Case Management Model and Notation*, CMMN) simuliacinis modelis sudaro galimybę simuliuoti procesus esant skirtingoms įvestims ir palyginti to paties programinės įrangos kūrimo projekto simuliacijos rezultatus pagal skirtingas procesų valdymo paradigmas, konkrečiai – AGILE ir WATERFALL. Šis metodas išsiskiria tuo, kad sujungia žmogaus veiksmų modeliavimą su programinės įrangos kūrimo projektų simuliacijomis, taip sudarydamas prielaidas labiau pagrįstam sprendimų priėmimui programinės įrangos kūrimo srityje.
2. Empiriniai programinės įrangos kūrimo projektų duomenys surinkti iš Lietuvos informacinių technologijų programinės įrangos kūrimo projektų, siekiant užtikrinti realistiškesnį ir kontekstualiai pagrįstą siūlomo metodo validavimą. Naudojant realių projektų duomenis, šis tyrimas prisideda prie atotrūkio tarp teorinių prielaidų ir praktinio žmogaus veiksmų taikymo programinės įrangos kūrimo projektuose mažinimo.

Šie moksliniai rezultatai prisideda prie gilesnio žmogaus veiksmų poveikio skirtingoms programinės įrangos kūrimo metodikoms supratimo ir suteikia vertingų išvalgų tiek akademiniais tyrimams, tiek praktiniam taikymui programinės įrangos kūrimo srityje.

Darbo rezultatų praktinė reikšmė

Tyrimas suteikia naujų išvalgų programinės įrangos kūrimo praktikai, išryškindamas žmogiškojo faktoriaus (angl. *Human Factor*, HF) poveikį įvairioms programinės įrangos kūrimo paradigmoms. Analizuojant HF poveikį tiek WATERFALL, tiek AGILE SDP, tyrimas pateikia duomenimis pagrįstas rekomendacijas, kurias kūrėjų komandos gali naudoti siekiant optimizuoti darbo eigą, didinti komandos efektyvumą ir mažinti žmogiškajam faktoriui priskirtas rizikas:

1. **HF svarba.** Išsami literatūros apžvalga identifikavo pagrindinius HF kintamuosius, leidžiančius SDP specialistams nustatyti, kurie veiksniai daro didžiausią įtaką procesui ir kuriems reikėtų skirti ypatingą dėmesį.
2. **Kūrimo modelio pasirinkimo tobulinimas.** Projektų vadovai ir kūrėjų komandos geriau įvertins, kada pasirinkti AGILE arba WATERFALL paradigmas, atsižvelgdami į HF ir jo suderinamumą su komandos dinamika bei projekto charakteristikomis.

3. **Komandos veiklos gerinimas.** Suprasdamos, kaip HF veikia konkrečius SDP etapus, organizacijos gali įgyvendinti tikslines mokymo programas, resursų paskirstymo strategijas ir palaikymo priemones, siekiant sumažinti dažniausiai pasitaikančias HF sukeltas problemas, tokias kaip netinkama komunikacija, užduočių prioritetų konfliktai ar darbo krūvio disbalansas.
4. **Rizikų valdymas.** Naudojant realių projektų duomenis, įmonės gali identifikuoti kritines proceso vietas, kuriose HF gali kelti riziką, ypač dideliuose ir sudėtinguose projektuose. Tai suteikia galimybę geriau prognozuoti galimus iššūkius ir imtis proaktyvių priemonių, kad būtų sumažintos trikdžių tikimybės.
5. **Proceso optimizavimo palaikymas.** Lyginamoji analizė pateikia gaires, kaip koreguojant užduočių seką ar įtraukiant lankstumą į SDP, organizacijos gali maksimaliai padidinti produktyvumą ir prisitaikyti prie HF įtakos komandos veiklos svyravimams.

Pateikdama empirinių pagrindų pagrįstas išvalgas, ši disertacija padeda programinės įrangos kūrimo komandoms kurti labiau atsparius, efektyvius ir žmogaus poreikius atitinkančius procesus, o tai galiausiai prisideda prie aukštesnės kokybės programinės įrangos produktų ir geresnių projektų rezultatų.

Ginamieji teiginiai

1. Siekiant tikslesnio programinės įrangos kūrimo procesų (SDP) modeliavimo ir simuliacijų, būtina modeliuoti ir vertinti ne tik struktūrines, plačiai žinomas projekto charakteristikas, bet ir žmogiškojo faktoriaus (HF) charakteristikas, leidžiančias atskleisti programinės įrangos kūrimo procesams būdingą neapibrėžtumą ir tinkamas modeliuoti taikant neryškiųjų aibių teoriją.
2. Siūlomas ANFIS pagrįstas HF modeliavimo metodas, taikomas naudojant realių projektų duomenų rinkinį, sudaro pagrindą tiksliam HF poveikio programinės įrangos kūrimo procesams prognozavimui. Integruojant HF poveikio vertinimą su atvejų grįsta simuliacija, šis metodas leidžia tiksliau prognozuoti SDP rezultatus, įskaitant projekto sąnaudas ir įgyvendinimo terminus.

Darbo rezultatų aprobavimas

Disertacijos rezultatai publikuoti dviejuose mokslo straipsniuose, įtrauktuose į *Clarivate Analytics Web of Science* duomenų bazę. Vienas straipsnis publikuotas Q2 kvartilio žurnale (Sielskaitė & Kalibatienė, 2023), o kitas – Q3 kvartilio žurnale (Sielskaitė & Kalibatienė, 2025). Rezultatai taip pat publikuoti recenzuojamuose konferencijų leidiniuose, įskaitant Sielskaitė (2022), Sielskaitė ir Kalibatienė (2021) bei Sielskaitė (2021).

Disertacijoje atliktų tyrimų rezultatai paskelbti 4 mokslinėse konferencijose Lietuvoje ir užsienyje:

- 15-oje tarptautinėje Baltijos konferencijoje apie skaitmeninį verslą ir intelektines sistemas (Baltic DB & IS), 2022 m., Rygoje, Latvijoje.
- eStream 2021: atviroje elektros, elektronikos ir informacinių mokslų konferencijoje, 2021 m., Vilniuje, Lietuvoje.

- DAMSS 2021: 12-oje konferencijoje „Data Analysis Methods for Software Systems“, 2021 m., Druskininkuose, Lietuvoje.
- ISD 2021: 29-oje tarptautinėje konferencijoje „Information Systems Development“, 2021 m., Valensijoje, Ispanijoje.

Disertacijos struktūra

Disertaciją sudaro įvadas, trys pagrindiniai skyriai, bendrosios išvados, literatūros sąrašas, disertacijos autoriaus publikacijų sąrašas ir santrauka lietuvių kalba. Disertaciją sudaro 115 puslapiai, 25 lentelės, 37 paveikslai ir 176 literatūros šaltiniai.

1. Programinės įrangos kūrimo proceso ir žmogiškojo faktoriaus literatūros apžvalga

Programinės įrangos kūrimo procesas yra kompleksinis, daugiasluoksnis veiklų rinkinys, apimantis reikalavimų analizę, projektavimą, programavimą, testavimą, diegimą bei priežiūrą. Šis procesas grindžiamas įvairiais metodais, kurie siekia užtikrinti tvarkingą, efektyvų ir prognozuojamą programinės įrangos sistemos sukūrimą. Tradiciniai modeliai, tokie kaip WATERFALL, pasižymi griežta fazių seka ir iš anksto apibrėžtais etapais, o šiuolaikinės metodikos, pavyzdžiui, AGILE, pabrėžia iteratyvumą, lankstumą ir glaudų suinteresuotųjų šalių įtraukimą. Nepaisant to, kad paradigmos suteikia struktūrą bei formalizuoja procesą, vis daugiau mokslinių tyrimų rodo, jog vien techniniai ar organizaciniai aspektai nėra pakankami – lemiamą reikšmę turi žmogiškasis faktorius.

Žmogiškasis faktorius literatūroje apibrėžiamas kaip psichologinių, kognityvinių, socialinių bei organizacinių elementų visuma, daranti tiesioginę įtaką projekto eigai, rezultatams ir produktyvumui. Tyrimai rodo, kad tokie aspektai kaip motyvacija, patirtis, bendradarbiavimas, komunikacija ir įsipareigojimas yra kritiniai kintamieji, galintys reikšmingai nulemti tiek proceso sėkmę, tiek nesėkmę. Pavyzdžiui, aukštos kvalifikacijos ir patyrę specialistai, gebantys efektyviai komunikuoti bei bendradarbiauti, dažniausiai sukuria kokybiškesnius sprendimus per trumpesnę laiką. Tuo tarpu prasta komunikacija, motyvacijos stoka ar per didelis kognityvinis krūvis gali sukelti projektų vėlavimus, klaidų daugėjimą ir net projekto žlugimą.

Įvairūs tyrėjai (Gueveyi et al., 2020; Dutra et al., 2021) pabrėžia, kad žmogiškasis faktorius daro įtaką visoms programinės įrangos kūrimo proceso fazėms: nuo reikalavimų išgryninimo iki testavimo ir priežiūros. Reikalavimų analizės etape itin svarbūs empatiniai gebėjimai, gebėjimas aiškiai formuluoti poreikius bei kurti pasitikėjimu grįstą ryšį tarp užsakovų ir kūrėjų. Projektavimo ir kūrimo etape reikšminga patirtis, kūrybiškumas bei problemų sprendimo įgūdžiai, o testavimo metu – kruopštumas, analitiniai gebėjimai ir aiški komunikacija tarp programuotojų bei kokybės užtikrinimo specialistų. Literatūros apžvalga atskleidžia, jog programinės įrangos kokybė tiesiogiai koreliuoja su žmogiškojo faktoriaus kintamaisiais, o organizacijos, kurios tinkamai valdo šiuos veiksnius, pasiekia aukštesnį projekto efektyvumą.

Pastaraisiais metais itin daug dėmesio skiriama AGILE sąveikai su žmogiškuoju faktoriumi. AGILE pagrįsta komandinio darbo, bendradarbiavimo ir nuolatinės komunikacijos principais, todėl žmogiškieji aspektai tampa vienu iš pagrindinių sėkmės kriterijų. Tyrimai rodo, kad tokie veiksniai kaip psichologinis saugumas, komandos autonomija, kliento įsitraukimas ir tarpusavio pasitikėjimas daro kritinę įtaką projekto rezultatams. WATERFALL paradigmoje žmogiškojo faktoriaus reikšmė pasireiškia kiek kitaip: čia svarbiausia tiksli reikalavimų dokumentacija, planavimo drausmė ir gebėjimas laikytis iš anksto numatytų procedūrų. Šis kontrastas parodo, jog žmogiškojo faktoriaus įtaka nėra vienalytė – ji priklauso nuo pasirinktų paradigmu.

Kitas svarbus literatūroje pabrėžiamas aspektas – žmogiškojo faktoriaus modeliavimas. Nors tradiciniai programinės įrangos kūrimo metodai retai suteikia priemones, kaip kiekybiškai vertinti žmogaus elgsenos kintamumą, mokslininkai siūlo pasitelkti pažangias analitines priemones, tokias kaip neraiškiųjų aibių teorija ar neuro-neraiškiųjų sistemų metodai. Šios technikos leidžia atspindėti neapibrėžtumą, kylantį dėl motyvacijos svyravimų, patirties stokos ar kitų subjektyvių veiksnių, taip sukuriant labiau realistiškus modelius. Literatūros analizė rodo, kad žmogiškojo faktoriaus integravimas į programinės įrangos kūrimo proceso modeliavimą suteikia galimybę iš anksto prognozuoti galimus rizikos taškus, optimizuoti resursų paskirstymą bei sumažinti klaidų tikimybę.

Apibendrinant galima teigti, jog literatūroje išryškinaamos kelios kartinės kryptys:

- Žmogiškasis faktorius yra neatsiejama programinės įrangos kūrimo proceso dalis – jis daro poveikį produktyvumui, kokybei, bendradarbiavimui ir galutiniam projekto rezultatui.
- Skirtingos paradigmos skirtingai pabrėžia žmogiškojo faktoriaus svarbą – AGILE orientuojasi į žmonių tarpusavio sąveikas, o WATERFALL labiau į planavimo ir dokumentavimo discipliną.
- Žmogiškojo faktoriaus modeliavimas yra būtinas – vien tik techniniai aspektai neleidžia pakankamai tiksliai prognozuoti projekto baigties, todėl būtina įtraukti neraiškiųjų aibių teorijos ir kitų pažangių metodų taikymą.
- Kritiniai žmogiškojo faktoriaus kintamieji – motyvacija, patirtis, prieinamumas, komunikacija ir įsipareigojimas – yra svarbiausi tolesnių tyrimų ir praktikos objektai.

Tokia literatūros apžvalga sudaro teorinę pagrindą vėlesniems tyrimams, kuriuose gilinama si žmogiškojo faktoriaus neapibrėžtumų modeliavimą ir eksperimentinį jų poveikio programinės įrangos kūrimo procesui įvertinimą.

2. Neraiškiųjų aibių teorijos ir atvejų valdymu grįstas dinaminis programinės įrangos kūrimo proceso modeliavimo ir simuliacijos metodas

Šiame skyriuje pristatytas tyrimo metodas, apimantis programinės įrangos kūrimo proceso modeliavimą ir simuliaciją, taikant neraiškiųjų aibių teoriją bei atvejų valdymo (angl.

Case Management) principus. Toks metodas leidžia įvertinti žmogiškojo faktoriaus neapibrėžtumą bei jo įtaką įvairių paradigimų (AGILE ir WATERFALL) pagrindu organizuojamiems projektams.

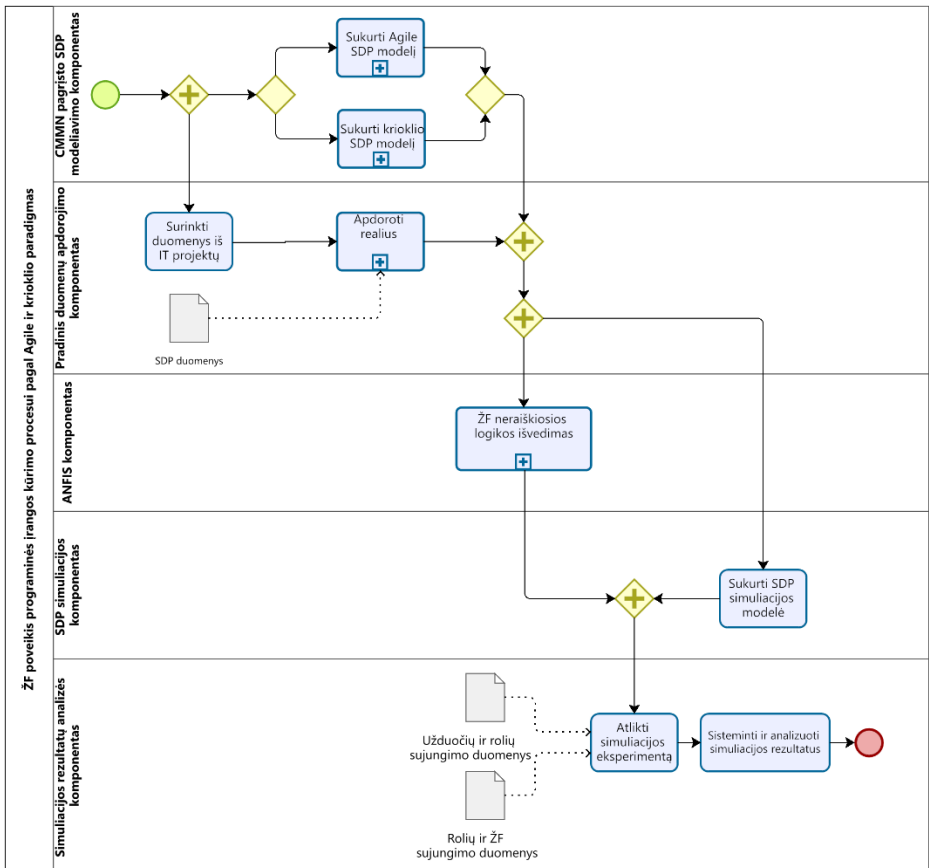
Programinės įrangos kūrimo procesas yra dinamiškas, neapibrėžtas ir stipriai priklausantis nuo žmonių įgūdžių bei elgsenos. Tradiciniai matematiniai ar statistiniai modeliai tik iš dalies gali atspindėti tokį kompleksiskumą, nes jie reikalauja tikslų ir deterministinių duomenų. Todėl pasirinkta neraiškiųjų aibių teorija, kuri leidžia formalizuoti neapibrėžtumą ir subjektyvumą. Taip galima išreikšti tokias savybes kaip „didelė motyvacija“, „vidutinė patirtis“ ar „mažas prieinamumas“, suteikiant joms skaitinę reikšmę intervale nuo 0 iki 1.

Neraiškiųjų aibių teorija suteikia galimybę apimti kiekybinius ir kokybinius aspektus, kurie dažnai yra neišmatuojami tradiciniais metodais. Tai leidžia sukurti modelius, galinčius atspindėti realaus darbo aplinkos sąlygas, kuriose sprendimai dažnai grindžiami ne tik objektyviais duomenimis, bet ir subjektyviomis nuostatomis bei patirtimi. Be to, tokie modeliai padeda analizuoti procesų jautrumą įvairiems žmogaus veiksniams, tokiems kaip motyvacija, komandinio bendradarbiavimo lygis ar žinių pritaikymo gebėjimas.

Taikant neraiškiųjų aibių logiką, galima ne tik vertinti atskirus procesų parametrus, bet ir prognozuoti galimas klaidas ar vėlavimus, atsirandančius dėl žmogiškojo faktoriaus. Taip modeliai tampa naudingesni sprendimų priėmimo kontekste, leidžiant projektų vadovams geriau planuoti išteklius, optimizuoti užduočių paskirstymą ir mažinti riziką. Galiausiai, neraiškiųjų aibių teorija suteikia lanksčią priemonę, leidžiančią nuosekliai integruoti įvairius subjektyvius įvertinimus į formalius procesų modelius, taip priartinant teorinius modelius prie realių projektų valdymo sąlygų.

Tyrimė taikytas ANFIS (angl. *Adaptive Neuro-Fuzzy Inference System*) metodas, jungiantis neuroninių tinklų mokymosi mechanizmus su neraiškiųjų aibių logika. ANFIS leidžia išmokti taisykles ir priklausomybes tarp įvesties (pvz., žmogiškojo faktoriaus rodiklių) bei išvesties (projekto eigos, užduočių trukmės, kokybės) duomenų. Tai užtikrina ne tik tikslesnes prognozes, bet ir interpretaciją – tyrėjas gali suprasti, kodėl vienas ar kitas žmogiškojo faktoriaus derinys lemia konkrečią projekto eigą.

Kitas kertinis metodo komponentas – CMMN (angl. *Case Management Model and Notation*), leidžiantis modeliuoti programinės įrangos kūrimo procesą kaip atvejų seką. Skirtingai nei BPMN (angl. *Business Process Model and Notation*), kuris orientuojasi į fiksuotas veiklų sekas, CMMN suteikia lankstumo valdyti procesą pagal kontekstą ir kintančius duomenis. Ši savybė ypač tinkama programinės įrangos kūrimui, kurio eiga dažnai priklauso nuo neprognozuojamų situacijų, pavyzdžiui, užsakovo poreikių pasikeitimų ar netikėtų techninių kliūčių.



S2.1 pav. BPMN diagrama, vaizduojanti žmogiškojo faktoriaus poveikį AGILE ir WATERFALL programinės įrangos kūrimo procesams

Tyrimo metu surinkti empiriniai duomenys iš kelių IT organizacijų, atstovaujančių skirtingo dydžio įmones (nuo mažų iki tarptautinių korporacijų). Pagrindiniai duomenų šaltiniai:

- užduočių atlikimo numatyta ir faktinė trukmė;
- užduočių vykdytojai ir jų kompetencijos;
- projektų vadovų arba komandų lyderių ekspertiniai vertinimai apie darbuotojų motyvaciją, patirtį ir prieinamumą (naudojant 1–5 skalę).

Gauti duomenys apdoroti keliais etapais:

- duomenų valymas – pašalintos klaidos, neatitikimai ir trūkstamos reikšmės;
- anonimizavimas – užtikrinta, kad individualūs darbuotojai negalėtų būti atpažinti;

- transformacija – ekspertiniai vertinimai buvo normalizuoti ir pritaikyti neraiškiųjų aibių sistemai.

Taip suformuota struktūruota duomenų bazė, tinkama ANFIS mokymui ir simuliacijoms.

Siekiant įvertinti žmogiškojo faktoriaus poveikį, sukurti du pagrindiniai programinės įrangos kūrimo proceso modeliai: AGILE ir WATERFALL.

AGILE modelis pasižymi iteratyvumu, glaudžiu bendradarbiavimu su klientu ir nuolatiniu prisitaikymu prie pokyčių. Modeliavimas buvo pagrįstas sprintų ciklais, komandinio darbo dinamika ir lankstumu, todėl itin tinkamas žmogiškojo faktoriaus kintamųjų (pvz., motyvacijos ar prieinamumo) pokyčių atspindėjimui.

WATERFALL modelis atitiko klasikinį linijinį kūrimo procesą, kuriame kiekviena fazė (reikalavimų analizė, projektavimas, kūrimas, testavimas, priežiūra) yra griežtai nuosekli. Šis modelis leido pavaizduoti situacijas, kai žmogiškojo faktoriaus pokyčiai daro mažiau tiesioginės įtakos eigos lankstumui, tačiau gali lemti klaidų kaupimąsi ar vėlavimus.

Abu modeliai realizuoti pasitelkiant BPMN ir CMMN schemas, kurios vėliau transformuotos į simuliacijos prototipą.

ANFIS naudotas tam, kad iš surinktų duomenų sudarytų ryšius tarp žmogiškojo faktoriaus (įvesties kintamųjų) ir užduočių eigos (išvesties rezultatu). Buvo pasirinkti trys pagrindiniai žmogiškieji faktoriai:

- motyvacija – lemia darbuotojo įsitraukimą ir pastangų lygį;
- patirtis – nusako gebėjimą atlikti užduotis kokybiškai ir per trumpesnę laiką;
- prieinamumas – apibrėžia darbuotojo galimybę būti pasiekiamam ir įsitraukti į užduotis laiku.

Kiekvienam kintamajam buvo priskirtos neraiškiosios reikšmės (pvz., „žema“, „vidutinė“, „aukšta“), kurios ANFIS pagalba konvertuotos į taisyklių sistemą. Taip buvo galima prognozuoti, kaip konkrečios žmogiškojo faktoriaus kombinacijos paveiks projekto eigą.

Simuliacijai pasirinkta agentų pagrindu grįsta modelių (ABMS) simuliacija, leidžianti modeliuoti ne tik procesą, bet ir atskirų dalyvių elgesį. Kiekvienas agentas atitiko projekto dalyvį (pvz., programuotoją, testuotoją, projektų vadovą) ir turėjo jam priskirtas savybes pagal surinktus žmogiškojo faktoriaus duomenis. Agentai sąveikavo tarpusavyje pagal CMMN logiką, todėl galutiniai rezultatai atspindėjo dinamišką komandinio darbo pobūdį.

Ekspperimentai atlikti keičiant įvesties parametrus:

- žmogiškojo faktoriaus lygius (aukšta vs. žema motyvacija, patirtis ir pan.);
- projekto valdymo paradigimą (AGILE, WATERFALL);
- užduočių seką bei resursų paskirstymą.

Rezultatai analizuoti taikant statistinius testus (pvz., t testą), siekiant įvertinti reikšmingus skirtumus tarp simuliacijų.

Šio tyrimo metu sukurtas metodas leidžia kompleksiskai vertinti žmogiškojo faktoriaus poveikį programinės įrangos kūrimo procesui:

1. Integruotas požiūris – sujungti ANFIS ir CMMN suteikia galimybę modeliuoti tiek neapibrėžtumą, tiek procesų lankstumą.
2. Realūs duomenys – metodas remiasi empiriniais duomenimis iš IT organizacijų, todėl rezultatai yra ne tik teoriniai, bet ir praktiškai pritaikomi.
3. Paradigmų palyginimas – simuliacija parodė, kad AGILE procesai yra jautresni žmogiškojo faktoriaus svyravimams, bet kartu turi didesnę adaptacijos potencialą, o WATERFALL – labiau pažeidžiamas dėl klaidų kaupimosi.
4. Universali taikymo galimybė – metodas gali būti pritaikomas įvairiems programinės įrangos kūrimo proceso scenarijams, suteikiant galimybę prognozuoti rizikas, optimizuoti resursų naudojimą ir gerinti projekto planavimą.

3. Žmogiškojo faktoriaus poveikio programinės įrangos kūrimo procesui eksperimentinis tyrimas

Trečiasis disertacijos skyrius skirtas eksperimentiniam tyrimui, kurio tikslas – empiriškai patikrinti ir patvirtinti sukurta žmogiškojo faktoriaus poveikio programinės įrangos kūrimo procesui modeliavimo ir simuliacijos metodą. Tyrimo mokslinė reikšmė grindžiama tuo, kad jis remiasi empiriniais duomenimis, gautais iš realių IT organizacijų, kas sudaro sąlygas lyginti teorinius modelius su praktiniais rezultatais bei užtikrina objektyvių įrodymų apie žmogiškojo faktoriaus įtaką projekto eigai ir rezultatams gavimą.

Eksperimento planavimas buvo grįstas klasikine eksperimentinio tyrimo logika: aiškiai apibrėžtos hipotezės, pasirinkti įvesties ir išvesties parametrai, nustatytas modelio validavimo būdas. Tyrimo planas apėmė šiuos etapus:

1. Įvesties parametru atranka – pasirinkti trys pagrindiniai žmogiškieji faktoriai: motyvacija, patirtis ir prieinamumas. Šie kintamieji atrinkti dėl jų tiesioginės įtakos individualiam darbuotojo produktyvumui ir komandinio darbo dinamikai.
2. Išvesties parametru nustatymas – projekto sėkmė buvo vertinama pagal kelis rodiklius: užduočių atlikimo laikas, nukrypimų nuo suplanuotos trukmės dydis, darbų kokybė bei bendras proceso efektyvumas.
3. Simuliacijos scenarijų sukūrimas – buvo parengti eksperimentiniai scenarijai, imituojuojantys tiek pagal AGILE, tiek pagal WATERFALL paradigmą valdomų projektų eigą, pritaikant įvairius žmogiškųjų faktoriaus lygių derinius (pvz., aukšta motyvacija, vidutinė patirtis, žemas prieinamumas).
4. Validavimo procedūra – simuliacijų rezultatai buvo lyginami su realiais projekto duomenimis, taip užtikrinant metodo patikimumą ir tikslumą.

Eksperimentui panaudoti duomenys iš skirtingo dydžio IT organizacijų – nuo mažų įmonių (iki 50 darbuotojų) iki stambių tarptautinių korporacijų. Tokia duomenų įvairovė užtikrino, kad rezultatai nebūtų šališki vienai konkrečiai organizacijos struktūrai ar darbo kultūrai.

Surinkti duomenys apėmė:

- numatytą ir faktinę užduočių atlikimo trukmę;
- nukrypimus procentine išraiška;

- ekspertinius vertinimus apie darbuotojų motyvaciją, patirtį ir prieinamumą (pagal standartizuotą 1–5 skalę).

Šie duomenys susisteminti į lenteles, kuriose kiekviena užduotis turėjo aiškiai priskirtą vykdytoją, faktinį atlikimo laiką bei atitinkamus žmogiškojo faktoriaus rodiklius. Tokia struktūra leido atlikti tikslias koreliacines ir statistines analizes.

Eksperimentas vykdytas dviem kryptimis:

1. Simuliacijos eksperimentai – pasitelktas sukurtas agentų pagrindu grįstas modelis (ABMS), kuriame kiekvienas agentas atitiko projekto dalyvį. Agentai sąveikavo tarpusavyje pagal CMMN logiką, o jų elgesį veikė ANFIS pagrindu išmoktos taisyklės. Tai leido imituoti tiek optimalius, tiek rizikingus žmogiškojo faktoriaus derinius.
2. Lyginamoji analizė su realiais duomenimis – simuliacijos rezultatai buvo tikrinami prieš faktinius duomenis. Taip buvo galima įvertinti modelio tikslumą, nustatyti, kuriose situacijose simuliacija atspindi realybę tiksliai, o kur reikia papildomų korekcijų.

Eksperimento metu nustatyta, kad didžiausi nukrypimai nuo suplanuotų terminų atsiranda tada, kai darbuotojų motyvacija yra žema, o prieinamumas ribotas. Patirtis, priešingai, turėjo tiesioginę įtaką darbo kokybei: patyrę specialistai atliko užduotis be klaidų, tuo tarpu mažiau patyrę – generavo daugiau defektų, kuriuos reikėjo taisyti.

Statistinė analizė parodė kelias reikšmingas tendencijas:

- AGILE paradigma yra jautresnė žmogiškojo faktoriaus svyravimams, tačiau kartu pasižymi didesniu adaptacijos potencialu. Pavyzdžiui, jei dalies komandos narių prieinamumas buvo ribotas, AGILE sprintų struktūra leido greičiau persikirstyti užduotis ir sumažinti poveikį galutiniam terminui.
- WATERFALL paradigmoje žemi žmogiškojo faktoriaus rodikliai turėjo kaupiamąjį efektą. Jei reikalavimų analizės ar projektavimo etapuose išsiveldavo klaidų dėl nepakankamos patirties ar komunikacijos stokos, šios klaidos pereidavo į vėlesnes fazes, sukeldamos papildomų išlaidų ir reikšmingai padidindamos projekto trukmę.
- Motyvacija pasirodė esanti vienas iš labiausiai kritinių veiksnių, darančių įtaką tiek AGILE, tiek WATERFALL projektams. Aukštos motyvacijos komandos pasiekė reikšmingai geresnius rezultatus, nepriklausomai nuo pasirinkto projektų valdymo būdo.
- Patirtis labiau veikė kokybinius rodiklius (klaidų skaičių, kodo struktūros tvarkingumą), o prieinamumas – kiekybinius (užduočių trukmę, nukrypimus nuo terminų).

Rezultatai patvirtinti taikant statistinius testus (pvz., t testą), kurie parodė, kad nustatyti skirtumai yra statistiškai reikšmingi.

Eksperimentinis tyrimas patvirtino, kad sukurtas metodas yra tinkamas prognozuoti žmogiškojo faktoriaus įtaką programinės įrangos kūrimo procesui.

Eksperimentai atlikti pagal iš anksto parengtą planą, siekiant simuliacijos būdu įvertinti žmogiškųjų faktorių įtaką programinės įrangos kūrimo proceso vykdymui. Tyrime

analizuoti klasikiniai programinės įrangos kūrimo proceso modeliai – AGILE ir WATERFALL. Nors praktikoje dažnai taikomi hibridiniai modeliai, tyrimas sąmoningai apribotas klasikinėmis jų formomis, siekiant užtikrinti metodologinį nuoseklumą ir rezultatų palyginamumą. Eksperimentinis tyrimas grįstas šiomis hipotezėmis: (H_0) žmogiškasis faktorius įtraukimas nesukelia statistiškai reikšmingų skirtumų programinės įrangos kūrimo proceso užduočių vykdymo trukmėse, palyginti su simuliacijomis be žmogiškojo faktoriaus; (H_1) žmogiškojo faktoriaus įtraukimas lemia sisteminius užduočių vykdymo nukrypimus ir didesnę trukmių variaciją; taip pat suformuluota tiriamoji hipotezė (H_2), teigianti, kad AGILE programinės įrangos kūrimo proceso simuliacijos yra jautresnės žmogiškojo faktoriaus poveikiui nei WATERFALL. Pirminiai eksperimentai (Sielskaitė & Kalibatienė, 2023), kuriuose AGILE programinės įrangos kūrimo proceso simuliacijose naudotos atsitiktinai generuotos žmogiškojo faktoriaus reikšmės, parodė, kad žmogiškojo faktoriaus variacijos sistemiskai veikia užduočių vykdymą ir sukelia nukrypimus nuo pradinių trukmės įverčių, taip suteikdamos preliminarią pagrindą hipotezei H_1 . Vėlesniuose tyrimuose (Sielskaitė & Kalibatienė, 2025), panaudojus empiriškai nustatytus žmogiškojo faktoriaus duomenis, nustatyta, kad skirtingos žmogiškojo faktoriaus variacijos daro nevienodą, užduočiai specifinį poveikį programinės įrangos kūrimo proceso vykdymui, o tai rodo, kad žmogiškasis faktorius lemia struktūruotą proceso variaciją, o ne vienodą vidutinės trukmės pokytį. Papildomos statistinės analizės parodė, kad AGILE simuliacijose porinių imčių t testas neatskleidė statistiškai reikšmingo bendro vidutinės užduočių trukmės skirtumo, todėl H_0 agreguotame lygmenyje nebuvo atmesta, tačiau įtraukus žmogiškojo faktoriaus padidėjo užduočių vykdymo variacija, kas atitinka H_1 . Tuo tarpu vienos imties t testai, taikyti AGILE ir WATERFALL simuliacijų nukrypimų rodikliams, atskleidė statistiškai reikšmingus nukrypimus nuo bazinių sąlygų ir didelius efekto dydžius, rodančius sisteminių žmogiškojo faktoriaus poveikį programinės įrangos kūrimo proceso vykdymui. Lyginamoji analizė taip pat parodė didesnius santykinius nukrypimus ir variaciją AGILE simuliacijose nei WATERFALL modelyje, kas empiriškai patvirtina tiriamąją hipotezę H_2 ir leidžia teigti, kad dėl adaptyvios ir mažiau formalizuotos struktūros AGILE procesai yra jautresni žmogiškųjų veiksnių poveikiui.

Taip eksperimentinė dalis ne tik validuoja metodą, bet ir pateikia praktines rekomendacijas, kurios gali būti naudingos tiek akademinėi bendruomenei, tiek IT organizacijoms, siekiančioms didinti savo projektų sėkmės rodiklius.

Bendrosios išvados

1. Atlikus literatūros analizę apie programinės įrangos kūrimo procesų (SDP) modeliavimą ir simuliaciją, įtraukiant žmogiškojo faktoriaus (HF) aspektus, gauti rezultatai parodė, kad nagrinėtuose požiūriuose egzistuoja spragų integruojant HF į SDP simuliacijas. Tai pabrėžia poreikį taikyti naujus modeliavimo metodus, galinčius atspindėti žmogiškojo faktoriaus įtaką programinės įrangos kūrimui. Literatūros analizė taip pat atskleidė, kad didžiausią įtaką SDP daro tokie HF aspektai kaip komunikacija, lyderystė, darbo pasitenkinimas, kognityvinė apkrova ir emocinė gerovė.
2. Naujas pasiūlytas neraiškiųjų aibių teorija ir atvejo analizės pagrindu sukurtas SDP modeliavimo bei simuliacijų metodas suteikia galimybę modeliuoti HF ir

tirti jo poveikį SDP. Pagrindinės metodo sudedamosios dalys yra realių duomenų parengimas, CMMN pagrindu sukurtų modelių ir simuliacijos modelio plėtra. Sukurtas neraiškiųjų aibių teoriją taikantis metodas leidžia prognozuoti HF poveikį SDP. Šis metodas skirtas veiksmingai spręsti su žmogaus elgesiu susijusį neapibrėžtumą ir kintamumą, remiantis neraiškiosios aibės teorijos principais. Modelyje integruoti trys pagrindiniai HF veiksniai – motyvacija, patirtis ir prieinamumas – kaip neraiškiosios įvesties kintamieji, leidžiantys generuoti diferencijuotas prognozes apie jų bendrą poveikį SDP veikimo rodikliams. Siūlomas pritaikomas ir tikslus HF poveikio prognozavimo įrankis, palyginti su tradiciniais deterministiniais modeliais.

3. Surinkus duomenis iš realių IT organizacijų, sudaryti duomenų rinkiniai apėmė tiek planuotą užduočių vykdymo trukmę, tiek faktinį jų atlikimo laiką, taip pat individualius žmogiškojo faktoriaus rodiklius kiekvienam komandos nariui. Pagrindinis duomenų rinkinys suformuotas iš trijų skirtingų IT organizacijų ir apėmė detalius projektinių užduočių įrašus, jų vykdymo trukmę, atlikimo kokybės vertinimus bei su užduotimis susijusius specialistus. Šis duomenų rinkinys sudaro prielaidas empiriškai validuoti prielaidas apie atskirų žmogiškojo faktoriaus atributų – tokių kaip motyvacija, patirtis ir prieinamumas – įtaką užduočių vykdymo laiko nuokrypiams. Duomenys gauti iš realių IT projektų, tad jie užtikrina realistišką pagrindą simuliacijų ir eksperimentų vykdymui bei didina tyrimo rezultatų praktinį aktualumą ir patikimumą.
4. Siūlomas neraiškiųjų aibių teorijos pagrindu sukurtas metodas įgyvendintas kaip veikiantis prototipas ir eksperimentiškai įvertintas, naudojant tiek simuluotus, tiek empiriškai pagrįstus žmogiškojo faktoriaus duomenis. Atlikti eksperimentai parodė, kad siūlomas metodas yra įgyvendinamas praktikoje ir leidžia sistemishkai tirti žmogiškojo faktoriaus įtaką tiek AGILE, tiek WATERFALL programinės įrangos kūrimo proceso kontekstuose. Eksperimentiniai rezultatai rodo, kad žmogiškojo faktoriaus įtraukimas lemia struktūrizuotus ir neatsitiktinius užduočių vykdymo elgsenos pokyčius abiejose paradigmos. Nors statistškai reikšmingas bendras vidutinės užduočių trukmės skirtumas ne visais atvejais buvo nustatytas ($t(10) = 1,14$, $p = 0,28$), nuokrypių analizė atskleidė reikšmingus ir statistškai patikimus nukrypimus nuo bazinio scenarijaus. Nustatyta, kad bendras darbo sąnaudų nuokrypis siekė 157 % AGILE modelyje ir 139 % WATERFALL modelyje. Vienos imties t testo rezultatai patvirtino statistškai reikšmingus nuokrypius abiem atvejais ($p < 0,001$), o efektų dydžiai buvo dideli (Cohen $d = 2,15$ AGILE ir $1,80$ WATERFALL). Vidutinis santykinis nuokrypis buvo didesnis AGILE užduotyse (55,22 %, $SD = 36,28$), palyginti su WATERFALL užduotimis (43,62 %, $SD = 45,85$), kas rodo didesnę kintamumą ir jautrumą žmogiškajam faktoriui AGILE aplinkoje. Šie kiekybiniai rezultatai patvirtina, kad žmogiškasis faktorius turi išmatuojamą ir reikšmingą įtaką užduočių vykdymui bei vertinimo tikslumui. Apibendrinant, eksperimentiniai rezultatai rodo, kad siūlomas modelis patikimai atspindi žmogiškojo faktoriaus poveikį ir gali būti praktiškai taikomas projektų vadovams bei kūrimo komandoms, siekiant prognozuoti ir valdyti rizikas, susijusias su žmogiškųjų išteklių kintamumu programinės įrangos kūrimo proceso planavimo ir vykdymo metu.

Annexes

Annex A. Survey Questionnaire and Data Collection Template

ANNEX A. Survey Questionnaire and Data Collection Template

Survey Information

Purpose of the survey:

The purpose of this survey is to collect anonymized data on human factors influencing software development tasks and the deviation between estimated and actual task execution time. The collected data is used exclusively for academic research and simulation modeling purposes.

Confidentiality:

All responses are anonymized. No personal or company-identifying information will be published.

To ensure clearer data collection and separation of variables, the information is collected in two tables: one describing task execution time data and the other describing human factor attributes of employees.

Please provide the estimated and actual execution time for each task performed in the software development process:

Task ID	Employee ID	Estimated Task Time (hours)	Actual Task Time (hours)
T1	X1		
T2	X2		
T3	X3		

Notes:

- Estimated time refers to the planned duration of the task before execution.
- Actual time refers to the real time spent completing the task.

Please evaluate the following human factor parameters for each employee using a scale from 1 to 5:

Employee ID	Motivation (1-5)	Experience (1-5)	Availability (1-5)
X1			
X2			
X3			

Notes:

Scale description: 1 – Very low, 2 – Low, 3 – Moderate, 4 – High, 5 – Very High.

Šarūnė SIELSKAITĖ

RESEARCH ON THE IMPACT OF HUMAN FACTORS ON SOFTWARE
DEVELOPMENT PROCESSES

Doctoral Dissertation

Technological Sciences,
Informatics Engineering (T 007)

ŽMOGIŠKŲJŲ VEIKSNIŲ POVEIKIO PROGRAMŲ SISTEMŲ KŪRIMO
PROCESAMS TYRIMAS

Daktaro disertacija

Technologijos mokslai,
Informatikos inžinerija (T 007)

Lietuvių kalbos redaktorė Deimantė Grigaitė

Anglų kalbos redaktorė Jūratė Griškėnaitė

2026 04 20. 10,8 sp. l. Tiražas 20 egz.
Leidinio el. versija <https://doi.org/10.20334/2026-021-M>
Vilniaus Gedimino technikos universitetas
Saulėtekio al. 11, 10223 Vilnius
Spausdino UAB „Ciklonas“,
Žirmūnų g. 68, 09124 Vilnius