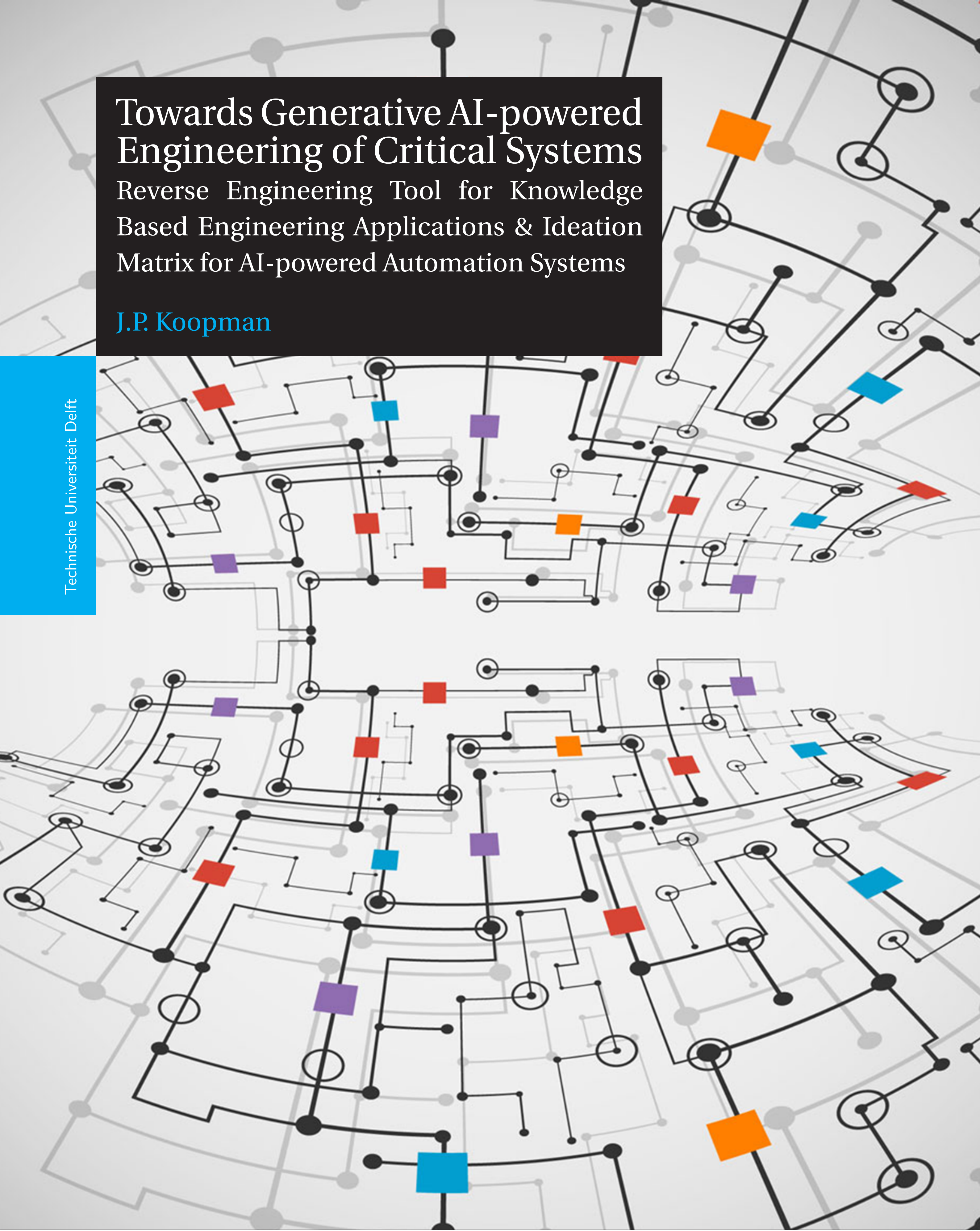
Towards Generative AI-powered Engineering of Critical Systems

Reverse Engineering Tool for Knowledge Based Engineering Applications & Ideation Matrix for AI-powered Automation Systems

J.P. Koopman



Technische Universiteit Delft

TU Delft Delft University of Technology

Challenge the future

Towards Generative AI-powered Engineering of Critical Systems

Reverse Engineering Tool for Knowledge Based Engineering Applications & Ideation Matrix for AI-powered Automation Systems

by

J.P. Koopman

in partial fulfillment of the requirements for the degrees of

MSc. Aerospace Engineering
&
MSc. Communication Design for Innovation

at the Delft University of Technology,

to be defended publicly on Friday March 7, 2025 at 14:00.

Student number:	4306430
Project duration:	September 1, 2023 – March 7, 2025
Aerospace Thesis Committee:	
Chair:	Prof. dr. ir. L.L.M. (Leo) Veldhuis
Responsible Supervisor:	Dr. ir. G. (Gianfranco) la Rocca
Examiner:	Drs. C. (Caroline) Wehrmann
Additional:	Dr. ing. A. (Alexander) Heidebrecht
Communication Design for Innovation Committee:	
1st supervisor, chair:	Drs. C. (Caroline) Wehrmann
2nd Supervisor:	Dr. ir. S. (Steven) Flipse
3rd Supervisor:	Dr. ir. G. (Gianfranco) la Rocca

Cover: *Technological background with circular circuit, by Harryarts (Creative Commons License)*
Style: *TU Delft Report Style, with modifications by Daan Zwaneveld*

TU Delft
Delft University of Technology

Preface

I would like to express my deepest gratitude to those who have supported me throughout my academic journey and the completion of this research. Balancing two master's theses has been both a challenge and a privilege, and I could not have done it without the guidance, encouragement, and patience of those around me.

First and foremost, I extend my sincere appreciation to my Science Communication supervisors. Caroline Wehrmann, your unwavering support, insightful feedback, and ability to help me navigate the complexities of science communication have been invaluable. Your expertise and encouragement made this research both stimulating and rewarding. I would also like to thank Steven Flipse for his help and suggestions, particularly in the first and final stages of my research.

For my aerospace engineering thesis, I would like to thank Gianfranco La Rocca for his guidance, critical questions and curious mindset towards my work. This combined thesis process has taught me perseverance, a healthy dose of realism, and made me push through big challenges. I also want to thank Darpan Bansal, who worked alongside me during the initial part of my Aerospace thesis, and offered help in many ways during the setup of the high-powered workstation that was critical to my Artificial Intelligence (AI)-related research.

A heartfelt thank you to Alexander Heidebrecht for providing much-needed reassurance that my research was not only valid but important for future AI specialists. Your insights and encouragement have been invaluable in helping me see the broader impact of my work.

To my family — Mom, Dad, Floor, and Huub — your unwavering support, love, patience and encouragement have been an anchor for me throughout this journey. Your support has meant more than I can express.

To my wonderful girlfriend, Josine, words cannot fully capture my gratitude. Your patience, willingness to engage in my research, and ability to provide an outsider's perspective helped bridge the gap between AI builders and users in a way I hadn't thought possible. Your belief in me, thoughtful insights, and of course, your hugs, have been a constant source of strength.

To Yvonne, my coach on both a career and personal level, thank you for helping me build on the skill set I need as a rounded engineer. Your guidance, support, and insights have been instrumental in shaping my professional growth, and I am incredibly grateful for your mentorship.

My roommates, Tijn and Yfke, deserve special mention for keeping me sane through laughter, evening games, and much-needed distractions. Even during the most stressful moments, you both found ways to lift my spirits, and for that, I am incredibly grateful.

To my friends, both engineers and beyond, thank you for your patience as I disappeared into my research. Your support, professional insights, and much-needed breaks from academia have been invaluable. Whether through board games, coffee-fuelled discussions, or simply listening, you all played a crucial role in getting me through this journey.

Finally, I want to extend my gratitude to all the participants who contributed to my research, whether by sharing their experiences or engaging in discussions that shaped my understanding of AI communication. Without your willingness to participate, this work would not have been possible. To everyone who has been part of this journey, directly or indirectly, thank you. I look forward to seeing you all at my graduation drinks and party.

Let's inspire each other to be the best versions of ourselves, while keeping in touch on a personal level and enjoying life to its fullest.

J.P. Koopman
Rotterdam, February 2025

Nomenclature

Acronyms

KBE Knowledge Based Engineering 1

MOKA Methodology and tools Oriented to Knowledge-based engineering Applications 170

MDO Multi-disciplinary Design Optimization 169

DEFAINE Design Exploration Framework based on AI for froNT-loaded engineering 8

SysML Systems Modeling Language 2

UML Unified Modeling Language 9

MBSE Model-Based Systems Engineering 1

FEM Finite Element Method 16

c2m code-to-model 11

INCOSE International Council on Systems Engineering 171

SRE Software Reverse Engineering 10

API Application Programming Interface 31

XMI XML Metadata Interchange 16

LLM Large Language Model 1

REProcess Reverse Engineer Process model 1

MDKBE Model-Driven Knowledge-Based Engineering 1

AI Artificial Intelligence i

BDD Block Definition Diagram 9

IBD Internal Block Diagram 9

AD Activity Diagram 9

RD Requirement Diagram 10

UAV Unmanned Aerial Vehicle 18

OMG Object Management Group 38

MSOSA Magic Systems of Systems Architect 38

GUI Graphical User Interface 13

CAD Computer Aided Design 139

GenAI Generative Artificial Intelligence 1

HAI Human-AI 195

ML Machine Learning 196

SMM Shared Mental Model 105

KPI Key Performance Indicator 129

Contents

1

EXECUTIVE SUMMARIES

Two executive summaries are presented, corresponding to both parts of this integrated thesis project.

PART I: AEROSPACE ENGINEERING THESIS

The rapid advancement of Generative Artificial Intelligence (GenAI) has introduced transformative possibilities in engineering disciplines, including software development, design automation, and Model-Based Systems Engineering (MBSE). In the aerospace industry, complex engineering workflows rely on Knowledge Based Engineering (KBE) applications, which often lack explicit workflow documentation. This lack of documentation hampers traceability, system verification, and reusability. The Model-Driven Knowledge-Based Engineering (MDKBE) approach has been proposed to address these challenges by integrating Knowledge Based Engineering with MBSE, enhancing automation and structuring computational workflows.

A major challenge within MDKBE is the ability to automatically extract structured process representations from existing KBE applications. Given the increasing role of GenAI and Large Language Model (LLM) models in engineering, this research explores their potential for automating reverse engineering processes to improve the accessibility and reuse of engineering knowledge.

OBJECTIVES

The objective of this research was to develop a novel reverse engineering method for KBE applications by leveraging dynamic analysis, graph transformation techniques, and LLMs. Specifically, the research aimed to:

- Design a system - called the Reverse Engineer Process model (REProcess) tool - to extract structured process models from KBE applications.
- Investigate the feasibility of using LLMs to abstract engineering workflows and generate effective human-interpretable diagrams.
- Implement a prototype of the REProcess tool, capable of automatically reverse engineering singular KBE workflows.

MAIN RESEARCH QUESTIONS

The research was guided by the following key questions:

1. What are the system specifications for the envisioned REProcess tool?
2. To what extent can state-of-the-art LLMs be used to reverse engineer a process model that captures the workflows of KBE applications?
3. How can a practically useful reverse engineering tool be built around the LLM-based reverse engineering method?

Conclusions

The research successfully developed and evaluated the REProcess prototype, demonstrating the feasibility of LLM-based reverse engineering for KBE applications. Key findings include:

- The proposed dynamic precedents analysis and graph transformation approach effectively extracted structured process representations from KBE applications.
- LLMs proved effective in generating interpretable abstractions of engineering workflows but exhibited limitations in dependency tracing.
- The REProcess tool provided structured, visual representations of process models, enhancing model traceability and reusability.

Despite these successes, challenges remain in optimizing computational performance, improving dependency tracing accuracy, and aligning Systems Modeling Language (SysML) representations with KBE execution behavior.

Key Recommendations

- **Enhance dependency tracing methods:** Future research should refine the precedents analysis algorithm to improve accuracy and completeness in capturing interdependencies within KBE applications.
- **Optimize computational efficiency:** Implementing smaller, fine-tuned LLMs or applying model quantization techniques can reduce computational overhead and improve performance.
- **Further development of the REProcess tool:** Integrate additional visualization techniques and interactive modeling capabilities to enhance usability for aerospace engineers.

This research highlights the transformative potential of LLM-assisted reverse engineering in aerospace engineering, offering a pathway toward enhanced automation, knowledge management, and engineering process optimization.

Part II: Communication Design for Innovation Thesis

The rapid advancement and increasing adoption of Generative Artificial Intelligence (GenAI) across industries have created new challenges for interdisciplinary collaboration between AI experts and domain experts. Effective communication between these groups is essential for the successful integration of GenAI into complex, high-stakes domains such as engineering. However, misaligned expectations, knowledge gaps, and differences in working culture often hinder collaboration. While existing communication strategies focus on either technical documentation or general interdisciplinary teamwork, they do not adequately address the unique challenges posed by GenAI projects. This thesis explores structured communication methods to bridge these gaps and facilitate better collaboration.

Objectives

The primary objective of this research was to design and evaluate a structured communication tool that improves interdisciplinary collaboration in GenAI projects. Specifically, the goal was to enable domain experts to articulate their needs more effectively and help AI experts communicate technical constraints in a way that aligns with domain requirements. To achieve this, the research aimed to:

- Identify key communication challenges in GenAI projects.
- Develop a structured approach to facilitate communication between AI and domain experts.
- Validate the effectiveness of the proposed approach through expert feedback and case studies.

Main Research Questions

1. What are the key collaboration and communication challenges between AI experts and domain experts in interdisciplinary AI projects, and how do these relate to each other?
2. How can a structured communication tool improve mutual understanding and alignment between these groups?

Conclusions

The research identified a **lack of shared knowledge** as the main barrier to effective interdisciplinary collaboration in GenAI projects. This knowledge gap manifests in two ways: *misaligned expectations* before project initiation and *miscommunications* during project execution. AI experts and domain experts often struggle to establish a shared mental model, leading to unrealistic expectations, inefficient workflows, and difficulty in integrating AI into domain-specific applications.

To address these issues, a structured communication tool based on a morphological chart was developed. This tool systematically maps AI capabilities to domain needs, providing a visual and interactive way for teams to explore design options. Expert validation sessions confirmed that this approach:

- Enhances expectation management by clarifying what GenAI can and cannot do.
- Encourages structured discussions that align technical feasibility with domain requirements.
- Reduces miscommunication by providing a shared reference framework.

Key Recommendations

- **Integrate structured communication tools early in project initiation:** Establishing a shared understanding at the outset significantly improves collaboration and reduces friction during development.
- **Encourage cross-training between AI and domain experts:** Providing AI literacy training for domain experts and domain-specific training for AI experts can enhance mutual understanding and improve the effectiveness of interdisciplinary teams.

This research underscores the importance of structured communication in GenAI projects and provides a concrete approach for improving interdisciplinary collaboration. Future work should explore the scalability of the proposed tool across different domains and further refine its usability across additional domains.

Part III: Discussion of Integrated Thesis Project

By designing and implementing a GenAI-powered engineering tool that demonstrates how selective utilization of GenAI technology can be used to implement new and improved work practices while mitigating inherent risks, and developing a communication tool that facilitates (technical) domain experts and AI experts to collaboratively conceptualize and develop other GenAI-powered systems to enhance existing work practices, this combined thesis provides a significant contribution to the adoption of GenAI technology for the engineering of complex, critical systems.

2

REPORT INTRODUCTION

This chapter briefly introduces the overarching context of this combined thesis project. Subsequently, it outlines how both parts of the combined thesis contribute to a single, overarching objective and describes the relation and synergy between both parts.

2.1. PROJECT CONTEXT

The public release of ChatGPT in November 2022 was a pivotal moment in the history of Generative Artificial Intelligence (GenAI). Since then, many businesses and organizations have been scrambling to adapt and adopt this new technology. At the same time, the extreme popularity of the technology has sparked huge additional investments causing developments to accelerate at an ever-increasing pace. Initially, many knowledge workers believed the technology was not mature enough to consider handing off parts of their work to AI, but the extremely rapid progress of the technology has meant this sentiment is changing and the question of adoption is starting to shift from ’if’ to ’how’.

This trend also applies to the world of (aerospace) engineering. While typically conservative, the vast amount of information associated with engineering processes - think of requirements specifications, concept renders, 3D models, source code, simulation data, test data, etc. - means that there is great potential for data driven approaches, and in particular GenAI, to assist humans in design and engineering processes. Notably, the keyword here is *assist*. There is a broad consensus under scientists and (governmental) organizations that GenAI should be used to assist humans, rather than fully replace them [1–4]. Two main reasons for this are the black-box nature of GenAI models, and its tendency to hallucinate[1] when prompted to perform a task that is too difficult to solve, instead of notifying the user or simply returning an error. These characteristics set GenAI apart from other, more traditional AI domains. An overview of key AI domains and their interrelations is presented in Figure 2.1.

When comparing the characteristics of the different types of AI described in Figure 14.3, a key trend when moving from the outermost layer towards the inner circles is that the mapping from input to output becomes increasingly more abstract and harder to trace. For an expert system, this mapping is completely clear since the rules and knowledge are explicitly programmed into it by humans. On the other hand, such a clear mapping is completely absent for GenAI models, which is why they considered a black-box. This, combined with the potential for hallucination, is very problematic from an engineering perspective and explains why many engineers are skeptical or even dismissive towards the technology as a whole. On the flip side, however, there is also a beneficial trends when moving inwards in Figure 2.1. Namely, the ability to perform complex tasks increases, as well as the ability to deal with large amounts of variable and unstructured data.

2.1.1. COMPLEX AND CRITICAL SYSTEMS

A context where the adoption of novel technologies like GenAI is typically very slow is the engineering of complex and critical systems. Here, the term *complex* refers to systems that consist of many interacting

[1] Hallucination is the phenomenon where a GenAI system generates output that is nonsensical and inaccurate but presents it as if it were true or factual, therefore making it easy to overlook at first glance.

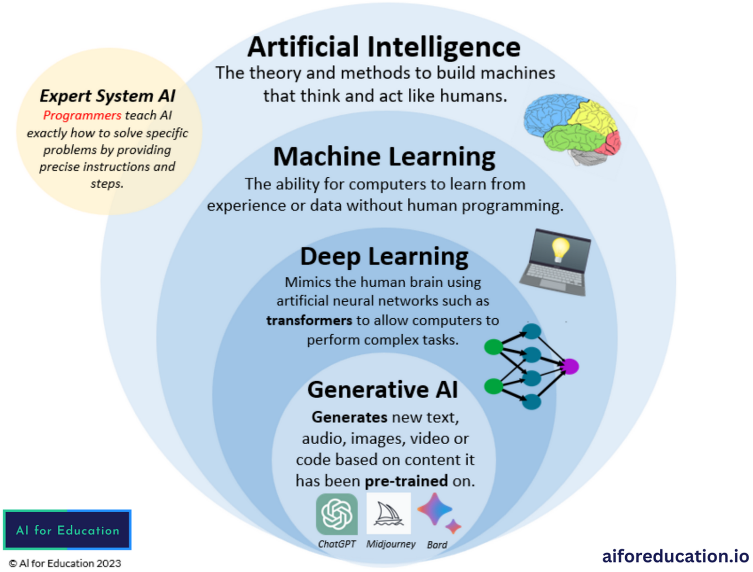


Figure 2.1: A comparative visualization of AI, Machine Learning, Deep Learning, and Generative AI. Obtained from [5].

components. These different components often correspond to different (engineering) domains like the engines and the wings of an aircraft, for example. The term *critical* refers to the potential for severe consequences due to failures of the system. For instance, personal injury, loss of human life, or severe legal/financial repercussions. Besides aircraft, some notable examples of complex and critical systems are nuclear reactors, payment processing networks, and emergency telecommunications.

Within these domains, the design of every component is evaluated on aspects like safety, reliability, quality and other key performance metrics. Because of the emphasis on safety and reliability, these domains typically follow a highly conservative approach toward the adoption of novel technologies. However, as mentioned before, GenAI also has a lot of potential due to its ability to perform complex tasks and deal effectively with large amounts of variable and unstructured data. Based on this combination of characteristics, the integration of GenAI technology within the engineering process of complex and critical systems is characterized as a *high-risk, high-reward* area of application.

2.1.2. EXPLAINABLE AI

An approach that can be adopted to harness the potential of GenAI while mitigating its inherent weaknesses and risks is the concept of Explainable AI. In fact, as per the European Act, it is even mandatory to do so when developing high-risk GenAI applications [1]. The essence of this approach is that the human users remain responsible for the overall output. To achieve this, the GenAI system should provide transparency and ensure traceability of its internal GenAI processes so that humans can be involved in the process. This can be done in several ways, ranging from direct control where humans direct the AI during every step of the way, to more distant roles where humans monitor the process (e.g. via a dashboard) or simply have to approve the final output. The most suitable approach will vary on a case by case basis. Through careful selection and implementation, such Explainable AI approaches can ensure the trustworthiness of the output and mitigate the previously discussed risks and downsides of GenAI.

Explainable AI is closely related to Trustworthy AI and Reliable AI. However, these related concepts typically refer to an even broader scope by also including ethical and societal concerns such as fairness, bias, and social norms [1]. While these aspects are highly important, this thesis research will focus specifically on the development of transparent and traceable (Gen)AI systems. Therefore, the term Explainable AI will be used throughout the rest of this report.

Given that the widespread diffusion of GenAI technology itself has only happened very recently, awareness about more detailed concepts like Explainable AI is still largely absent under professionals. Furthermore, although the concept of Explainable AI has been described at a high level by other authors and organizations, a key question that remains is how to practically implement it within GenAI-powered engineering systems.

2.2. COMBINED THESIS OBJECTIVE AND SETUP

The overarching objective of this thesis is to investigate and contribute to the realization of GenAI's potential to support engineering processes - in particular engineering of complex, critical systems. To this end, two separate lines of research were pursued which correspond to the two Masters degrees for which this combined thesis project was carried out.

The first research line constitutes the **aerospace engineering** part of this combined thesis. Here, a GenAI-powered software tool was designed and a prototype was implemented that can be used in a novel engineering approach based on KBE and MBSE. This approach and the developed software are further introduced in chapter 3. Aside from the practical benefits provided by the developed tool itself, it also serves as a real-world demonstration of how Explainable AI principles can be applied to integrate GenAI technology into engineering workflows. This part of the combined thesis was completed first.

The second research line corresponds to the **science communication** part of this combined thesis. From the perspective of this second part, the previous development of the GenAI-powered software tool served as an in-depth case study. By going through this development exercise, a lot of valuable knowledge and experience was gained about GenAI technology, as well as about the development process. This knowledge and experience was put to use during the science communication part of this thesis to investigate other GenAI projects where similar applications are developed. The focus of this investigation was on the interdisciplinary collaboration that typically characterizes GenAI development projects. By combining the knowledge gained during the aerospace part with insights obtained from interviews and literature, a communication tool was designed that facilitates the collaborative development of novel GenAI-powered engineering systems. The science communication part of the thesis is introduced further in chapter 10.

In the third and final part of this combined thesis, a number of high-level findings and conclusions are presented that were determined based on the integration of both parts of the project. These are presented in chapter 18.

PART I

Aerospace Thesis

3

Aerospace thesis introduction

A key trend in the field of aerospace engineering is that work is becoming more complex and multidisciplinary. Meanwhile, global competition and ambitious sustainability goals are pushing Tier 1 and Tier 2 suppliers to reduce recurring development costs and lead times. In response to these challenges, the aerospace sector has been working on the development and adoption of new technologies and engineering approaches. A prime example of this is the recently concluded Design Exploration Framework based on AI for froNT-loaded engineering (DEFAINE) project. One of the main outcomes of the DEFAINE project was the proposal and development of a novel methodology for the development of Knowledge Based Engineering (KBE) applications based on Model-Based Systems Engineering (MBSE) [6].

Additional background information for readers who are less familiar with KBE or MBSE is provided in Appendix A.

3.1. Model-Driven Knowledge Based Engineering

The DEFAINE project used rather convoluted phrases to refer to their novel methodology such as "the MBSE based methodology for KBE application development". To improve clarity and conciseness, this thesis adopts the term MDKBE approach to refer to the same methodology.

Beyond the persistent goals of reducing recurring development costs and lead times, the purpose of the MDKBE approach is to address several key challenges related to KBE applications: 1) their development is often case-based and unstructured, **lacking a standardized development methodology**; 2) they tend to become **black-box applications** due to having large code bases, as well as software development being prioritized over writing and updating documentation; and (3) they suffer from **limited reusability and project-to-project knowledge transfer**, as high-level design knowledge is often not captured, and there is no standard exchange format for transferring knowledge between different KBE tools [7, 8].

To address these issues, the MDKBE approach was developed as a framework for systematically capturing, developing, and maintaining KBE applications. A high-level visual representation of the MDKBE approach is provided in Figure 3.1. This figure focuses on presenting the four main steps (highlighted in bold text) of DEFAINE's novel methodology for developing KBE applications. A detailed description of each of these steps is provided below.

3.1.1. Step 1: Knowledge model

The first step of the MDKBE approach consists of domain experts and KBE developers collaboratively creating the so-called **knowledge model** using SysML and the CATIA Magic modeling platform. This knowledge model consists of a *requirements package*, containing the stakeholder needs and requirements for the KBE application; the *process package*, specifying the behavior of the KBE application (i.e. the engineering workflows embedded in the source code); and the *product package* which describes the KBE application's (hierarchical) software structure [10]. The corresponding requirements-product-process ontology, showing the relations between model elements from each of these packages, is presented in Figure 3.2.

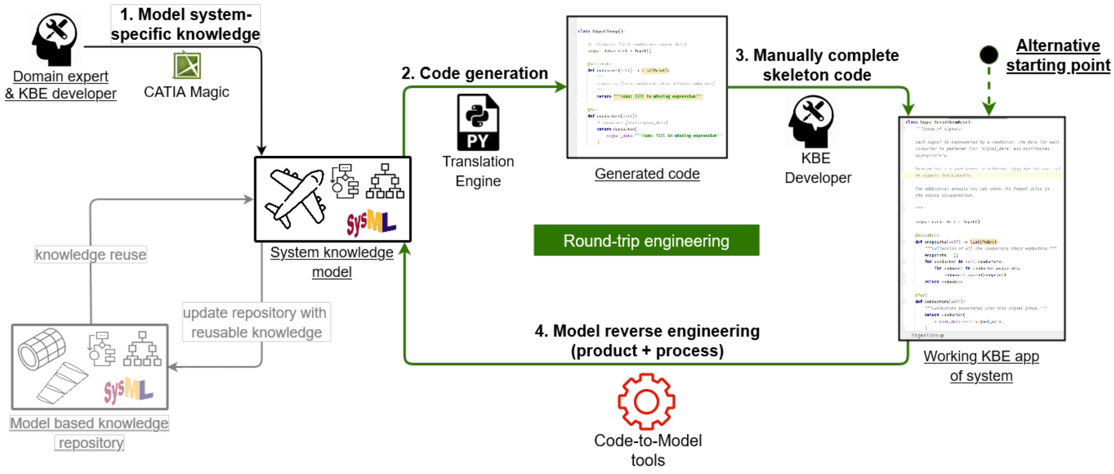


Figure 3.1: The envisioned Model-Driven Knowledge-Based Engineering approach, with the four main steps for the development of a new KBE application highlighted in bold text Adapted from [9].

SYSML-FOR-KBE

To define the knowledge model, the MDKBE approach envisioned in the DEFAINE project uses a subset of SysML diagrams, consisting of the Package, Requirement, Activity, Block Definition, and Internal Block diagrams [8]. The specific SysML diagrams and methods used in the MDKBE approach to capture the product, process and requirements packages of the KBE application knowledge model are outlined below. If available, their respective Unified Modeling Language (UML) counterparts are also mentioned as a reference for readers with a software background. For a more detailed specification of the knowledge model, including individual model elements and their mapping to specific aspects of the ParaPy KBE system, the reader is referred to [9, 10]. Background information about the notation and usage of the mentioned SysML diagrams and methods can be found in Friedenthal *et al.* [11] (Ch.3) as well as online documentation[1].

- **The product model** is contained in the product package of the KBE application knowledge model. It is defined using structure diagrams from SysML, specifically Block Definition Diagrams (BDDs) and Internal Block Diagrams (IBDs). The UML counterparts of these are Class and Component diagrams, respectively. The product model captures the hierarchical structure of the KBE application's software components (Python classes, in case of ParaPy). The model elements used to represent the individual software components in the BDDs and IBDs are called *blocks*.

  Using an analogous concept from the MBSE domain, the product model corresponds to the physical architecture[2]. This is also referred to as the *structural architecture* in this report, because it is considered to be more intuitive and better suited to the software-centric project context.

- **The process model** is contained in the process package of the KBE application knowledge model. It is defined using behavior diagrams from SysML, specifically Activity Diagrams (ADs). These diagrams also featured in UML so they have a direct counterpart. The process model captures the internal processes of the KBE application, i.e. the engineering workflows embedded in the source code. The model elements used to represent individual process steps in ADs are called *actions*. The term *activity* refers to the collective sequence of actions defined in an activity diagram.

  Again using some analogous concepts from the MBSE domain, the product model corresponds to the functional[3] and behavioral[4] architectures.

- **The allocation matrix** is a SysML concept that can be used to link the product and process models. This process, called *functional allocation*, is facilitated by the allocation matrix. It consists of specifying which

[1] Recommended source: https://sysml.org
[2] https://sebokwiki.org/wiki/Behavioral_Architecture_(glossary)
[3] https://sebokwiki.org/wiki/Functional_Architecture_(glossary)
[4] https://sebokwiki.org/wiki/Behavioral_Architecture_(glossary)

individual processing steps (i.e. actions) are performed by which structural software components (i.e. blocks). This process is depicted by the arrow between the process and product nodes in Figure 3.2.

- **The requirements model** is contained in the requirements package of the KBE application knowledge model. It is defined by means of Requirement Diagram (RD) diagrams, for which no UML equivalent exist. The requirements model captures the stakeholder needs and requirements for the KBE application.
- **The satisfaction matrix** is a SysML concept that can be used to link the requirements model to the process and the product model. This process, called *requirements allocation*, is facilitated by the satisfaction matrix. It consists of specifying the specific model element(s) that have to satisfy each individual requirement. As depicted by arrows originating from the requirements node in Figure 3.2, both process model elements (activities or actions) and product model elements (blocks) can be linked to the requirements.

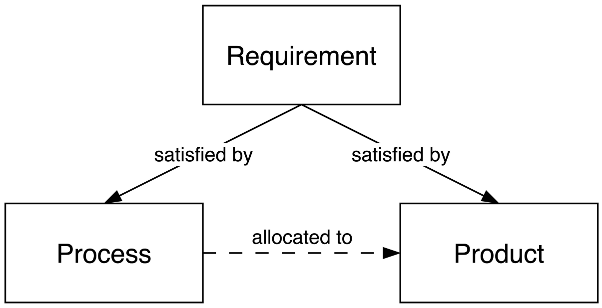


Figure 3.2: Relations between requirements, product features and process steps. Obtained from [10].

HIGH-LEVEL KNOWLEDGE MODEL

Theoretically, the knowledge model could be used to capture all relevant information regarding a particular KBE application. However, adding a lot of highly detailed information into the model is cumbersome and less efficient than writing it directly into the source code of the KBE application. Therefore, the goal during this first step of the MDKBE approach is to create a high-level knowledge model that specifies the main requirements, engineering processes and structural software components. Fine-grained details follow in later steps.

3.1.2. STEP 2: AUTOMATIC CODE GENERATION

The second step involves using the *Translation Engine* developed by Fernandes [12] to automatically generate the skeleton code of the KBE app. This results in a Python project filled with packages, modules, etc., where each module contains boilerplate code for standardized syntax elements like class and input parameters declarations, along with doc strings. The information used for this automated code generation is taken from the product model. A small section of KBE application code generated automatically by the Translation Engine is shown on the left-hand side of Figure 3.3.

3.1.3. STEP 3: MANUAL CODE COMPLETION

The third step of the MDKBE approach is performed by KBE developers. They take the auto-generated skeleton code as a starting point and manually complete the code to turn it into a fully functional KBE application. This manual code completion consists of adding domain-specific knowledge and operational logic into the source code app, similar to "traditional" KBE development. An example of completed KBE code is presented on the right-hand side of Figure 3.3. The manually added knowledge and rules are highlighted in green.

3.1.4. STEP 4: MODEL REVERSE ENGINEERING

A consequence of the manual code completion step is that the KBE application source code now contains more information than its knowledge model. Moreover, KBE developers could also have made significant structural changes to the code. As a result, the knowledge model created during step 1 has become outdated. Considering that an important function of the knowledge model is to act as documentation, this is not desirable. Therefore, the fourth and final step of the MDKBE approach consists of generating an updated knowledge model. For the product and process models, this generation is performed automatically through the use of Software Reverse Engineering (SRE) methods that extract model information directly from the updated KBE application. As

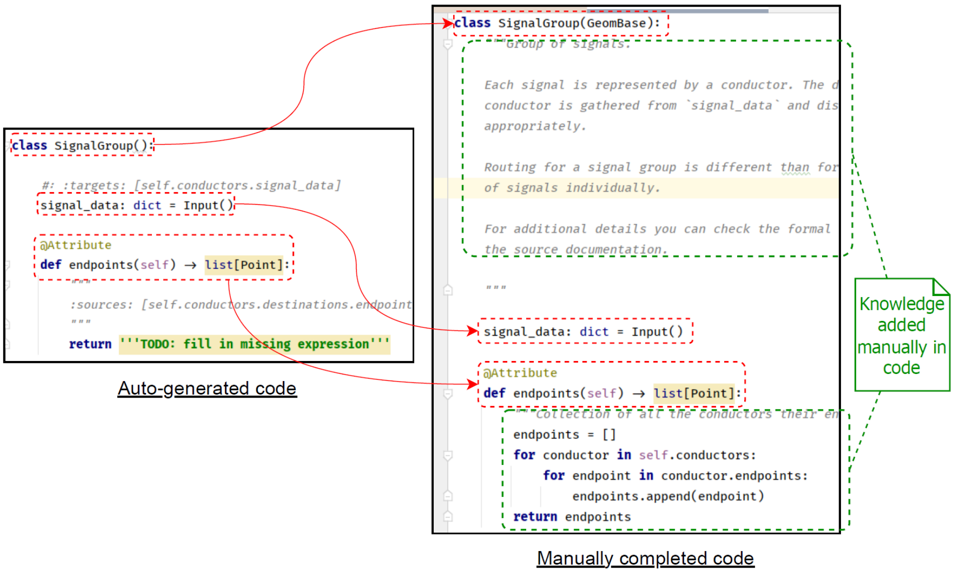


Figure 3.3: A small section of auto-generated KBE application code using the Translation Engine (MDKBE step 2) is shown on the left. On the right, the same section is shown after the KBE application code has been manually completed (MDKBE step 3) by a KBE developer. The added knowledge and rules are highlighted in green. Obtained from [10].

such, these tools are referred to as code-to-model (c2m) tools. For the requirements model, such automated extraction is not possible. If applicable, updating the requirements still has to be done manually.

ALTERNATIVE STARTING POINT

The alternative starting point on the right-hand side of Figure 3.1 is used to indicate that any working KBE application can be used as the input for c2m tools. Thus, even existing KBE applications which completely lack a knowledge model can still be reverse engineered in order to generated their corresponding product and process models. This is a key benefit from the reverse engineering tools as it greatly simplifies to adoption of the MDKBE approach for existing KBE applications.

3.1.5. ROUND-TRIP ENGINEERING

Round-trip engineering, or round-tripping for short, refers to the process of seamlessly exchanging data between different tools, models, or systems while preserving consistency across iterations. As denoted in Figure 3.1, in case of the MDKBE approach the term is used to refer to the closed loop formed by the combination of forward engineering (steps 2 and 3) and reverse engineering (step 4). Using this closed loop information flow, a working process is envisioned where KBE developers work with both the model and the code side-by-side, seamlessly switching between them. Whenever they make a change on one side, the other representation is immediately updated accordingly. Once all software components of the MDKBE approach are fully implemented, this round-tripping approach can be realized.

3.1.6. MODEL BASED KNOWLEDGE REPOSITORY

A final component of the MDKBE approach presented in Figure 3.1 is the model based knowledge repository. The purpose of this repository is to facilitate the transfer of project knowledge and reuse of existing KBE application code. Essentially, it is a database containing the knowledge models from many different KBE applications. The defining characteristic of this database is that it stores the individual knowledge model elements that define (high-level) features of existing KBE applications along with their corresponding segments of source code. This facilitates re-use of these existing features during the subsequent development of other KBE application using the MDKBE approach. Namely, instead of defining the entire knowledge model from scratch during the first modeling step, these existing elements can be imported from the model based knowledge repository. During subsequent automatic code generation, the source code attached to existing knowledge model elements can be used to generate more than just skeleton code for these parts of the KBE application, reducing the amount of manual code completion required during step 3.

3.1.7. MDKBE IMPLEMENTATION ACHIEVED IN DEFAINE

The DEFAINE project successfully developed the methodology and software needed to perform the first three steps of the MDKBE approach. Additionally, the software needed to perform the fourth step was partially implemented. A c2m tool was developed to perform the reverse engineering of the *product* model. However, a c2m tool for the *process* model was not developed within DEFAINE. Correspondingly, the objective of this thesis project is to take the next step towards implementing the MDKBE approach by making a significant contribution to the development of this c2m tool, which will be referred to as the REProcess tool from hereon. To provide a foundation for the discussion of the REProcess tool development, an overview of the field of SRE is presented in the next section.

3.2. SOFTWARE REVERSE ENGINEERING

As defined by Chikofsky and Cross [13], "*reverse engineering can be viewed as a process of analyzing a system to: identify the system's components and their interrelationships, and create representations of the system in another form or at a higher level of abstraction*". A high-level overview of key SRE concepts and their interrelations is presented in Figure 3.4. From the concepts and interrelations presented in this figure, those in the green and dark red boxes (the middle and bottom groups) are most relevant for this thesis work. The relevant concepts used in the SRE overview from Canfora *et al.* [14] are briefly outlined below.

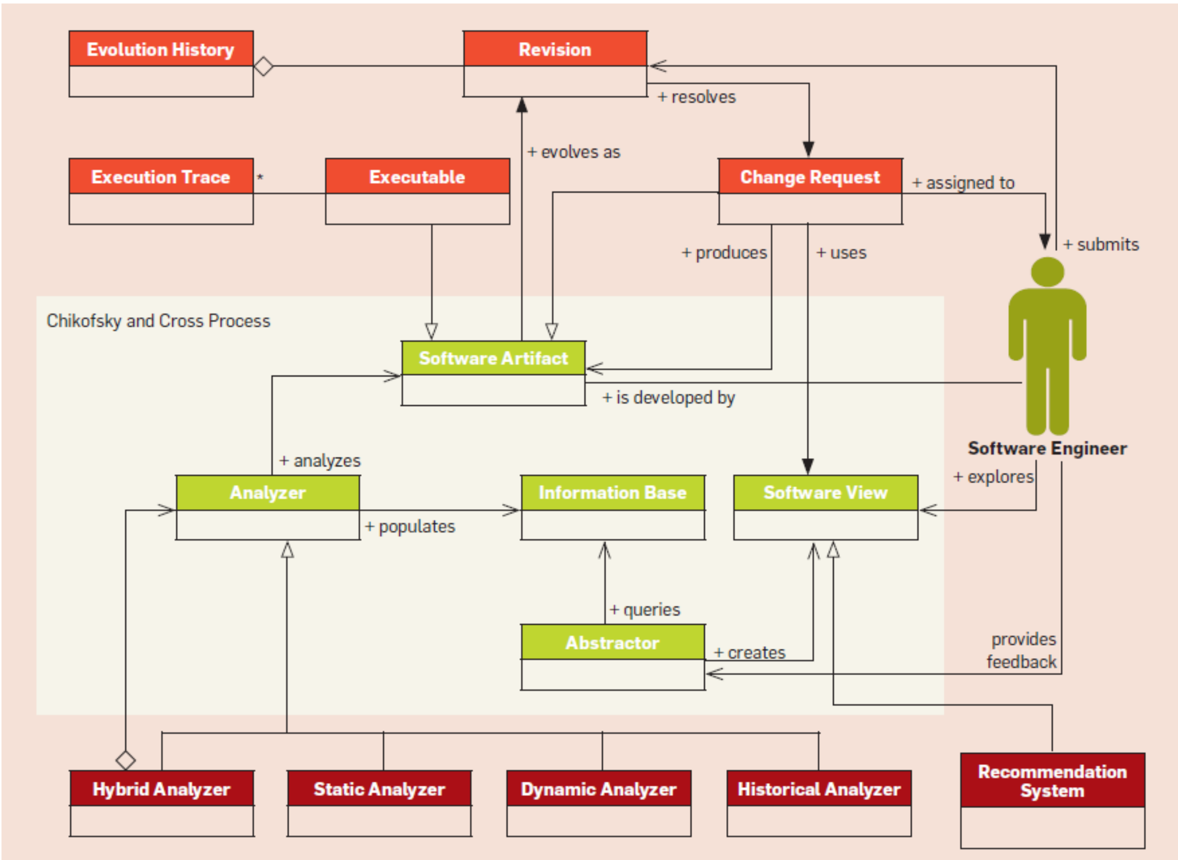


Figure 3.4: UML Class diagram of the key concepts of software reverse engineering and their interactions. Obtained from [14].

- **Software Artifact:** This is an umbrella term for the various things used as input for the software reverse engineering process. The most prominent example is the source code itself, but also included are executables (i.e. compiled source code), execution traces, project documentation (including e.g. model views drafted during initial software design phases), test cases, and change requests.
- **Analyzer:** The analyzer's job constitutes processing the software artifacts in order to extract and generate data that can be used reliably for subsequent abstraction steps. The research field corresponding to this specific part of the reverse engineering process is called program analysis. As depicted in Figure 3.4,

several main types of program analysis exist. From these, static and dynamic analysis methods are most relevant for this thesis project.

- *Static analysis* methods operate by analyzing the source code itself. The word "static" refers to the fact that the software does not have to run in order to perform the analysis. Instead, static analyzers operate by parsing the source code in order to extract data such as semantic information (e.g. comments and parameter names), code structure and dependencies.
- *Dynamic analysis* methods operate by running the source code and then tracking what happens during execution. As such, they inherently require analyzed code to be executable, which may be a limiting factor depending on context. Some examples of dynamic analysis includes measuring runtime performance and generating process logs.
- *Hybrid analysis* methods are a combination of static and dynamic analysis methods.
- *Historical analyzers* capture information about the evolution of a software product. Version control systems, like Git[5], are a prime example.

- **Information Base:** A data container storing output generated by the analysis methods. Having such a database is particularly relevant when multiple analysis methods are used in parallel to provide information for the subsequent abstraction step. In the case of the REProcess tool, it also stores the internal representation of the process model. Importantly, this internal representation is not in SysML format. A SysML model is only generated when the process model is exported as the final output.
- **Abstractor:** The task of the abstractor is to construct higher level representations of the analyzed software applications. As indicated in Figure 3.4, these higher-level representations typically are *software views*. A wide variety of abstraction methods exist. Main categories of abstraction methods include filtering, clustering, pattern detection, formal concept analysis, and heuristic algorithms [15].
- **Software Views:** The generation of software views typically is the end goal of SRE. These software views can take many forms and represent many different aspects of the targeted software system. They can range from highly formalized visualizations of the structural and functional software architecture, such as UML and SysML diagrams, to less formal and more data-driven representations, like correlation matrices depicting the relations between an application's software components [16], layered architecture graphs [17], or polymetric views [18].

  Note: specifically in case of the REProcess tool, it would be more precise to substitute *software view* by the word *model* in Figure 3.4 because the former may suggest that the objective is to generate a single SysML activity diagram while the actual goal is to generate a more elaborate SysML model capturing the entire set of engineering workflows embedded in the KBE application. The distinction between the overarching process model and individual activity diagrams is further discussed in chapter 4.

3.2.1. EXISTING REVERSE ENGINEERING METHODS

In a conducted literature review of existing reverse engineering methods, the state-of-the-art was established and potentially useful tools and methods that could be adopted for development of the REProcess tool are identified. Based on this review, a gap in existing work is defined as the combination of *automation* and *high-level* abstraction. To illustrate this, two examples of existing SRE methods are discussed in more detail.

SEMANTIC CLUSTERING

The semantic clustering approach was developed by Kuhn *et al.* [16]. This reverse engineering approach consists of a four steps. The first step is static analysis method consisting of parsing the target application's source code to extract the identifier names and comments used in each source code file. Secondly, Latent Semantic Indexing (LSI) is applied to compute the similarities between the source code files. Hereby, the so-called correlation matrix is obtained. Then, hierarchical clustering is used to rearrange the correlation matrix into a human-interpretable form. Finally, each cluster is labeled using the terms from the LSI step that best describe the entire cluster. An example output from this approach is presented in Figure 3.5. Note how the automatically generated labels presented on the right-hand side vary greatly in terms of communicative quality. Some, like "action, box, component, event, button, layout, Graphical User Interface (GUI)" provide

[5] https://git-scm.com

a good indication about the role and purpose of their cluster, while others, such as “start, length, integer, end, number, pre, count”, are too general and therefore fail in providing a useful description of a cluster. This limitation was also noted by Kuhn *et al.* [16], who presented the automatically generated output as a starting point for further investigation by the user of the tool who could then improve the labels by updating them manually.

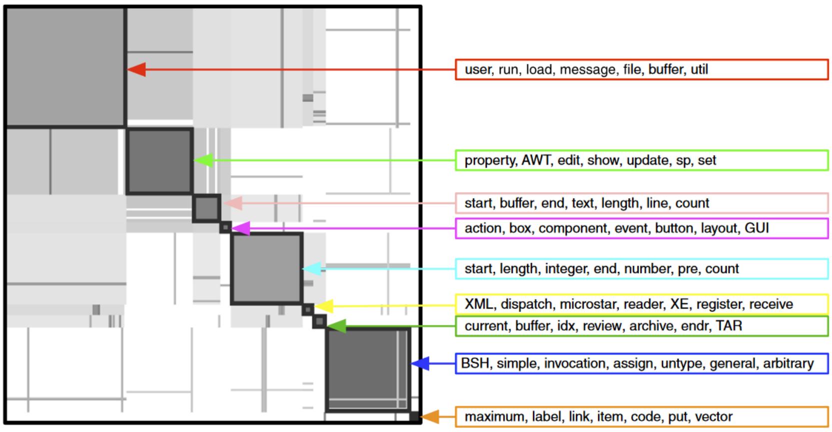


Figure 3.5: Example output from the semantic clustering approach. Obtained from Kuhn *et al.* [16].

MERGING EXECUTION TRACES

Ziadi *et al.* [19] developed a fully dynamic approach for reverse engineering UML sequence diagrams. This approach starts with dynamic analysis to collect execution traces[6]. Subsequently, these execution traces are transformed to graph-based representations and a k-tail merging algorithm is applied to merge the set of execution traces into one comprehensive graph. Finally, a transformation algorithm is used to generate a UML sequence diagram of the targeted application. An overview of this approach is provided in Figure 3.6 and an example output is presented in Figure 3.7.

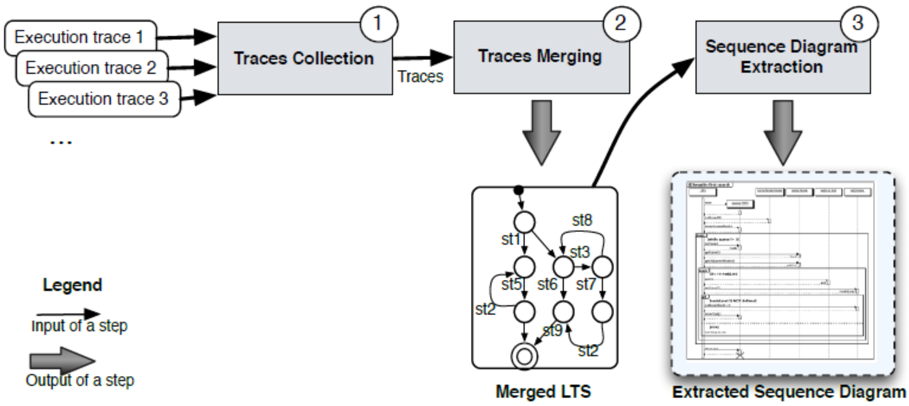


Figure 3.6: Overview of the reverse engineering approach from Ziadi *et al.* [19]. LTS stands for Labeled Transition System, which is the graph-based representation of the execution traces.

Similar to the semantic clustering approach, note the textual descriptions used in Figure 3.7. These are again made up of strings taken directly from the source code. Besides the work of Kuhn *et al.* [16] and Ziadi *et al.* [19], this is a pattern that was observed for many different existing SRE tools and methods[20–28], indicating an inherent limitation of traditional abstraction methods. Due to this limitation, fully automated reverse engineering of high-level diagrams - like SysML activity diagrams - remained out of reach. However, recent advancements in LLM technology present a novel opportunity to develop reverse engineering systems capable of addressing this gap.

[6] An execution trace is a process log of the sequence of operations executed by a program during a run

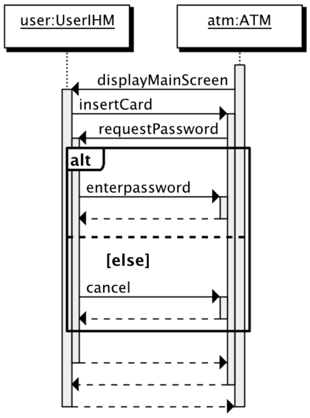


Figure 3.7: Example of a UML sequence diagram that can be generated using the reverse engineering approach from Ziadi *et al.* [19].

3.2.2. LARGE LANGUAGE MODEL-BASED REVERSE ENGINEERING

Recent developments in LLM technology present a novel opportunity to automate the high-level abstraction step of the SRE process. The latest LLMs are trained using vast amounts of general data, which typically includes various programming languages as well. Moreover, highly capable, specialized programming models - called big code models - have become available too. Most of the development attention of big code models lies on the automated generation of source code, based on natural languages instructions. So far, using LLMs for SRE purposes has received far less attention. At the start of this thesis project, no relevant publications were found where large language models were used for the reverse engineering of high-level software architecture[7].

A significant advancement in recent LLMs is their expanded input capacity. As of November 2023, GPT-4 Turbo can process up to 128,000 tokens, approximately equivalent to 96,000 words of English text. This represents a substantial increase from the 2,048-token limit (about 1,500 words) of the original GPT-3 model released in 2020 [29]. This dramatic expansion in context window size enables the processing of extensive codebases and complex systems within a single analysis, thereby enhancing the potential for automating high-level abstraction in SRE.

In the context of the REProcess tool, a key capability of state-of-the-art LLMs is their proficiency in interpreting source code, even when encountering syntax from Python packages like ParaPy, for which they were not explicitly trained. Current advanced models achieve this by leveraging the semantic information contained in the syntax, combined with interpolation and extrapolation of similar functionalities and patterns learned from the vast array of Python packages in their training dataset. Naturally, this approach introduces a margin of error, raising questions about its feasibility and effectiveness for reverse engineering ParaPy KBE applications. To address these concerns, a feasibility study was conducted at the start of this project. The study yielded promising results, confirming the potential of using state-of-the-art LLMs for developing the REProcess tool.

3.3. PROJECT OUTLINE

The overarching goal of this thesis is to address several key KBE application development challenges including the tendency of becoming black-box applications, the limited project-to-project knowledge transfer and the limited reuse of existing KBE applications. To this end, this thesis builds upon the foundation established by the DEFAINE project, with the overarching objective of advancing the implementation of the MDKBE approach. Specifically, this research contributes to the development of the Reverse Engineer Process model (REProcess) tool, an essential component of this approach. A visualization of the purpose of the REProcess tool is presented in Figure 3.8.

[7] Based on a Scopus search on 01-05-2023 using the following search query: "TITLE-ABS-KEY("large language models" AND ("reverse engineering" OR re-engineering OR reengineering) AND (software OR program OR "source code" OR application OR "object oriented"))

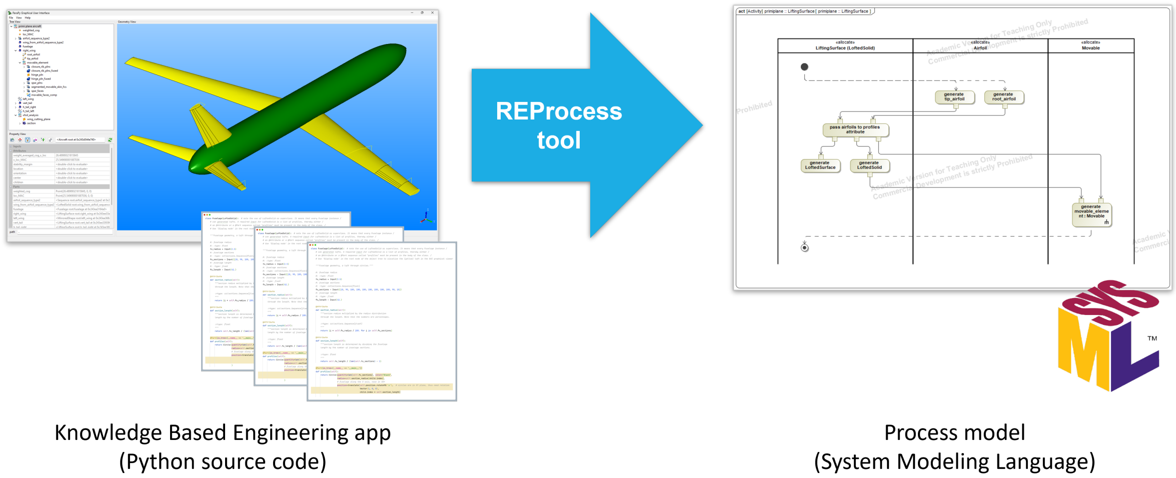


Figure 3.8: Visualization of the purpose of the REProcess tool.

3.3.1. PROJECT OBJECTIVES

The project follows an MBSE approach, structured around two key objectives:

1. **System Specification & High-Level Design**
   - Synthesize a high-level system specification and design of the REProcess tool including **system requirements and software architecture**.
2. **Prototype Development & Testing**
   - Investigate the feasibility and effectiveness of utilizing **state-of-the-art LLMs** as a **novel SRE approach** to perform **automated, high-level abstraction**.
   - Design and implement a **proof-of-concept prototype**, referred to as the **REProcess prototype**, capable of reverse engineering the engineering workflows embedded in **ParaPy KBE applications**.

3.3.2. SCOPE

The primary focus of this research is the **exploration and implementation of the LLM-based reverse engineering method**. Given this focus, not all features of the complete REProcess tool are implemented in this project. Three major aspects remain **outside the scope** of this thesis:

- **SysML model generation:** To enable importing the reverse-engineered process model into SysML modeling tools like CATIA Magic, a particular type of data file is required[8] which requires an extensive data transformation algorithm. Developing this transformation algorithm lies outside the scope of this thesis project.
- **Multi-workflow integration:** While the implemented prototype can process complex workflows, including interactions between multiple classes, Python modules, and external toolboxes (e.g. flow solvers, Finite Element Method (FEM), etc.), it is currently limited to handling **one workflow at a time**.
- **Training custom LLMs:** This research assesses the capabilities of several open-source LLMs but any training/fine-tuning of custom LLMs was not feasible due to a lack of training data.

To maintain clarity, this thesis adopts the following terminology:

- The *REProcess tool* refers to the complete envisioned system, integrating all features required for the MDKBE approach. This includes the SysML model generation and multi-workflow integration.
- The *REProcess prototype* refers to the developed implementation, specifically focusing on the LLM-based reverse engineering method.

[8] XML Metadata Interchange (XMI) data format, specified in [30]

3.3.3. RESEARCH QUESTIONS

Following the MBSE approach, the research questions are structured to address the high-level specifications, the feasibility of LLM-based reverse engineering, and the practical implementation of the reverse engineering method:

1. What are the system specifications for the envisioned REProcess tool?
2. To what extent can state-of-the-art LLMs be used to reverse engineer a process model that captures the engineering workflows embedded in KBE applications?
3. How can a practically useful reverse engineering tool be built around the LLM-based reverse engineering method?

3.3.4. REPORT STRUCTURE

The structure of this report follows the adopted MBSE methodology, consisting of two phases which correspond to the two main project objectives:

- **Phase 1: REProcess Tool System Specification & High-Level Design**
  - chapter 4 defines the system requirements and presents a set of example activity diagrams to specify the desired output from the REProcess tool.
  - chapter 5 details the conceptual system architecture and data processing structure.
- **Phase 2: REProcess Prototype Development & Testing**
  - chapter 6 outlines the prototype-specific requirements.
  - chapter 7 describes the design and implementation of the REProcess prototype.
  - chapter 8 presents the test results generated for the verification and validation of the implemented REProcess prototype.
- **Conclusions & Recommendations**
  - chapter 9 summarizes key conclusions, discusses limitations and open issues and provides recommendations for future work.

OVERVIEW OF APPENDICES

In addition to the report chapters outlined above, a series of appendices are added at the end of this report. An overview of the appendices related to the aerospace part of this combined thesis is provided below:

- **Appendix A** contains background information about KBE and MBSE for readers who are less familiar with these concepts.
- **Appendix B** contains a user manual for the implemented REProcess prototype.
- **Appendix C** presents the full set of test cases that were performed to verify and validate the REProcess prototype.

4

REPROCESS TOOL SYSTEM SPECIFICATIONS

This chapter provides a high-level specification of the complete REProcess tool, constituting the first step of the adopted MBSE approach. While implementation of the complete system specification of the REProcess tool presented in this chapter lies outside the scope of this thesis work, the findings are useful to sharpen the vision and understanding of the REProcess tool. To this end, the first section presents a set of manually created SysML activity diagrams belonging to an example process model. These provide a better understanding of the output to be generated by the REProcess tool. Subsequently, an overview and discussion of the system requirements defined for the REProcess tool is presented in section 4.2.

4.1. EXAMPLE PROCESS MODEL

This section presents a set of SysML activity diagrams corresponding to a single process model that was created manually using the CATIA Magic modeling platform[1]. They were created using an iterative process, in parallel with the synthesis of the system requirements presented in section 4.2. The purpose of these diagrams is to better understand and define the desired output from the REProcess tool. Naturally, this is a key aspect of the system requirements for the REProcess tool. Later in this project, during the implementation of the REProcess prototype, these diagrams were also used as a baseline for comparison with automatically generated results.

KBE APPLICATION USED FOR MODELING EXERCISE

The KBE application used for this modeling exercise was created for the design of Unmanned Aerial Vehicle (UAV) drones and is aptly called the “Modular UAV app”. This KBE application was also used during the previous thesis work of Fernandes [12], which focused on the implementation of the Translation Engine (step 2 of the MDKBE approach). A screenshot of the Modular UAV app’s GUI is presented in Figure 4.1. Part of the previous work with the Modular UAV app by Fernandes [12] was the creation of a SysML knowledge model, including several activity diagrams. The availability of this existing SysML model was a key motivation to select the Modular UAV app. Additionally, it was found to be well-structured, readily available and it incorporates a broad range of KBE features including generation of complex geometries, utilization of a simulation toolbox (XFoil Analysis Module), and automated generation of PDF reports. The existing knowledge model created by Fernandes [12] was used as a baseline to generate the diagrams presented in this section.

4.1.1. EXAMPLE ACTIVITY DIAGRAMS

The most important characteristic of the example activity diagrams is that they all correspond to the same high-level functionality of the Modular UAV app, namely the aerodynamic analysis of the UAV’s propeller geometry using blade element theory. In other words, they collectively model **a single engineering workflow**. Thereby, the diagrams provide a good indication of the most defining characteristic of the process model, namely its **multi-layered, hierarchical structure**. This, together with several other key characteristics, is discussed further after the introduction of the diagrams themselves.

The **top-level activity diagram**, outlining all (high-level) steps of the workflow, is presented in Figure 4.2. Note how both the “Generate main blade” and “Do Blade Element Theory Analysis” nodes in Figure 4.2 contain a

[1] https://www.3ds.com/products/catia/catia-magic

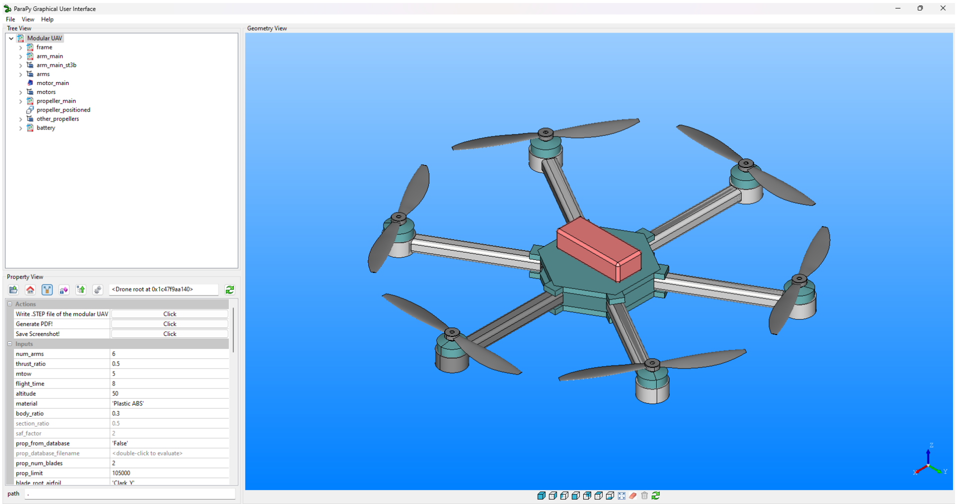


Figure 4.1: Screenshot of GUI of the Modular UAV applications developed and previously used by Fernandes [12].

rake symbol in the bottom-right corner. This indicates that both of these nodes represent activities which are defined in **lower-level activity diagrams**. The activity diagram corresponding to the “Generate main blade” node is presented in Figure 4.4. The activity diagram corresponding to the “Do Blade Element Theory Analysis” node is presented in Figure 4.3. All three diagrams were created based on a highly detailed review of the corresponding source code. The last activity diagram (Figure 4.3) was generated from scratch. The others were created by adapting existing diagrams from Fernandes [12].

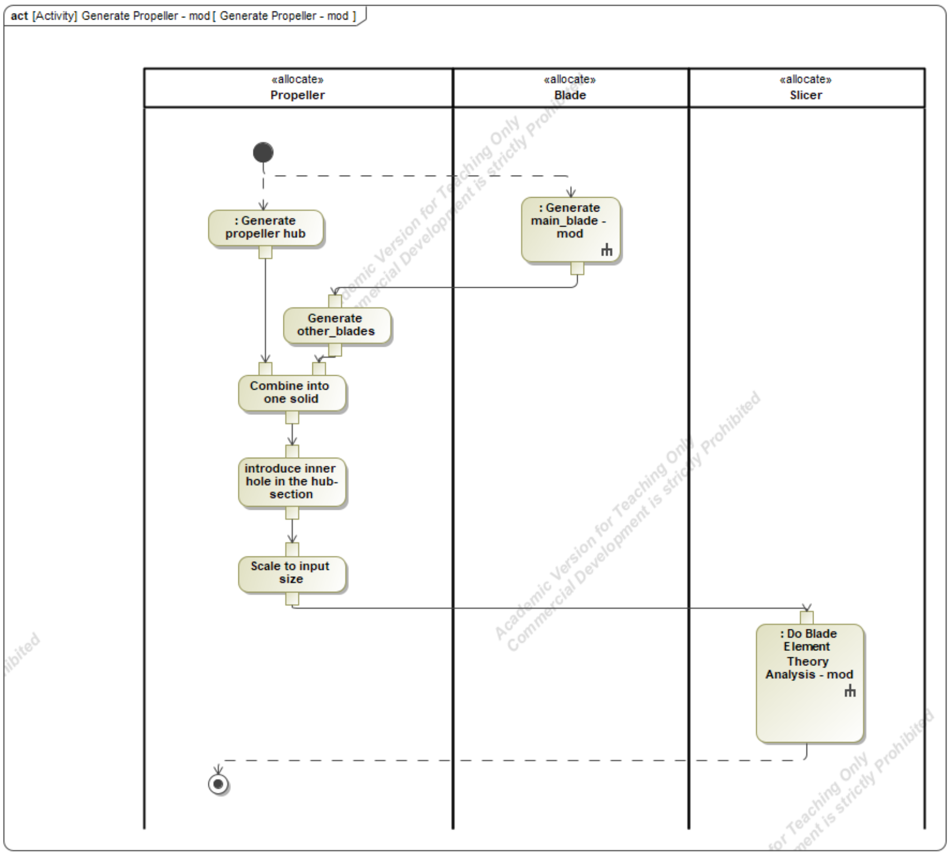


Figure 4.2: The “Generate Propellers” activity diagram. Adapted from [12].

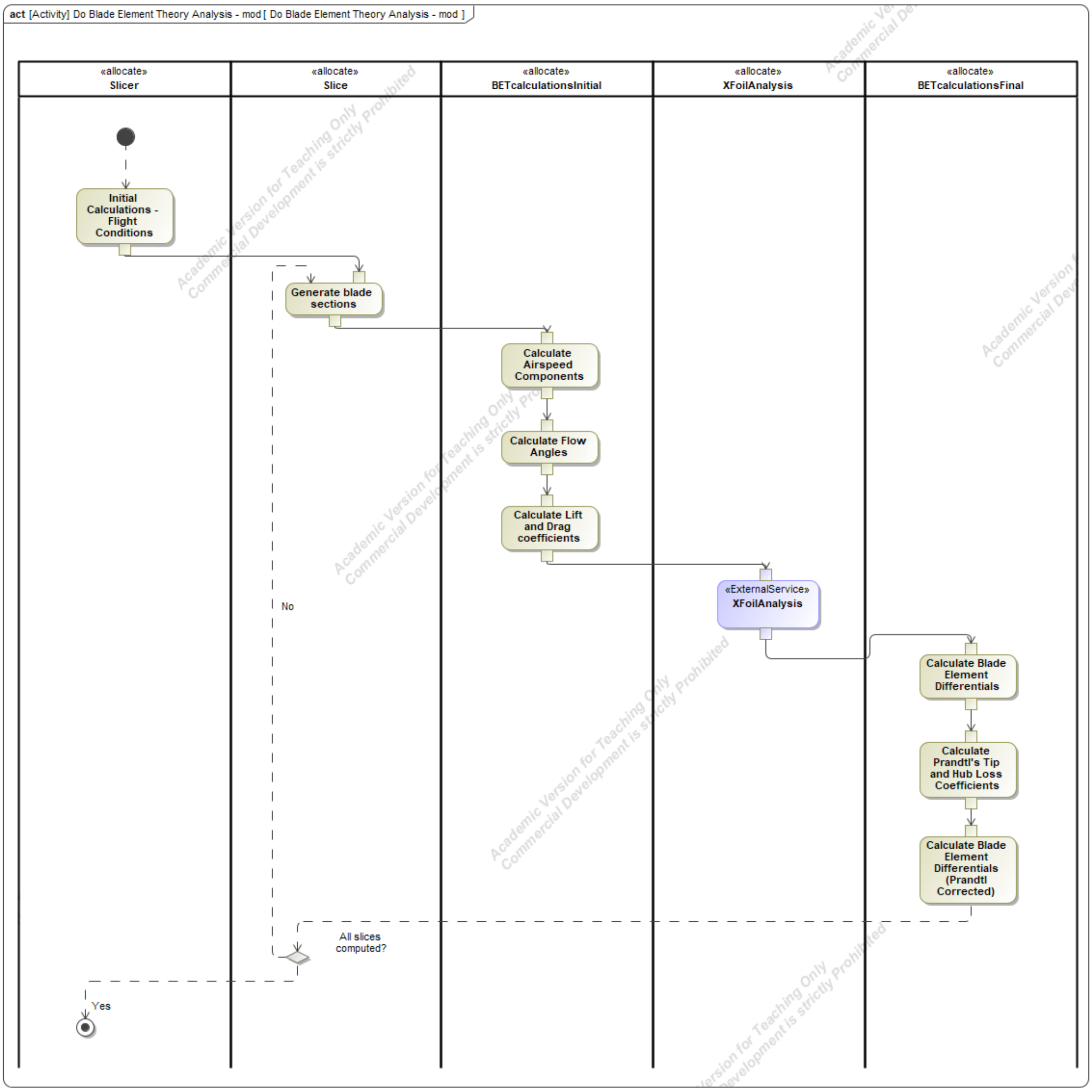


Figure 4.3: The "Execute Blade Element Theory Analysis" activity diagram. Adapted from [12].

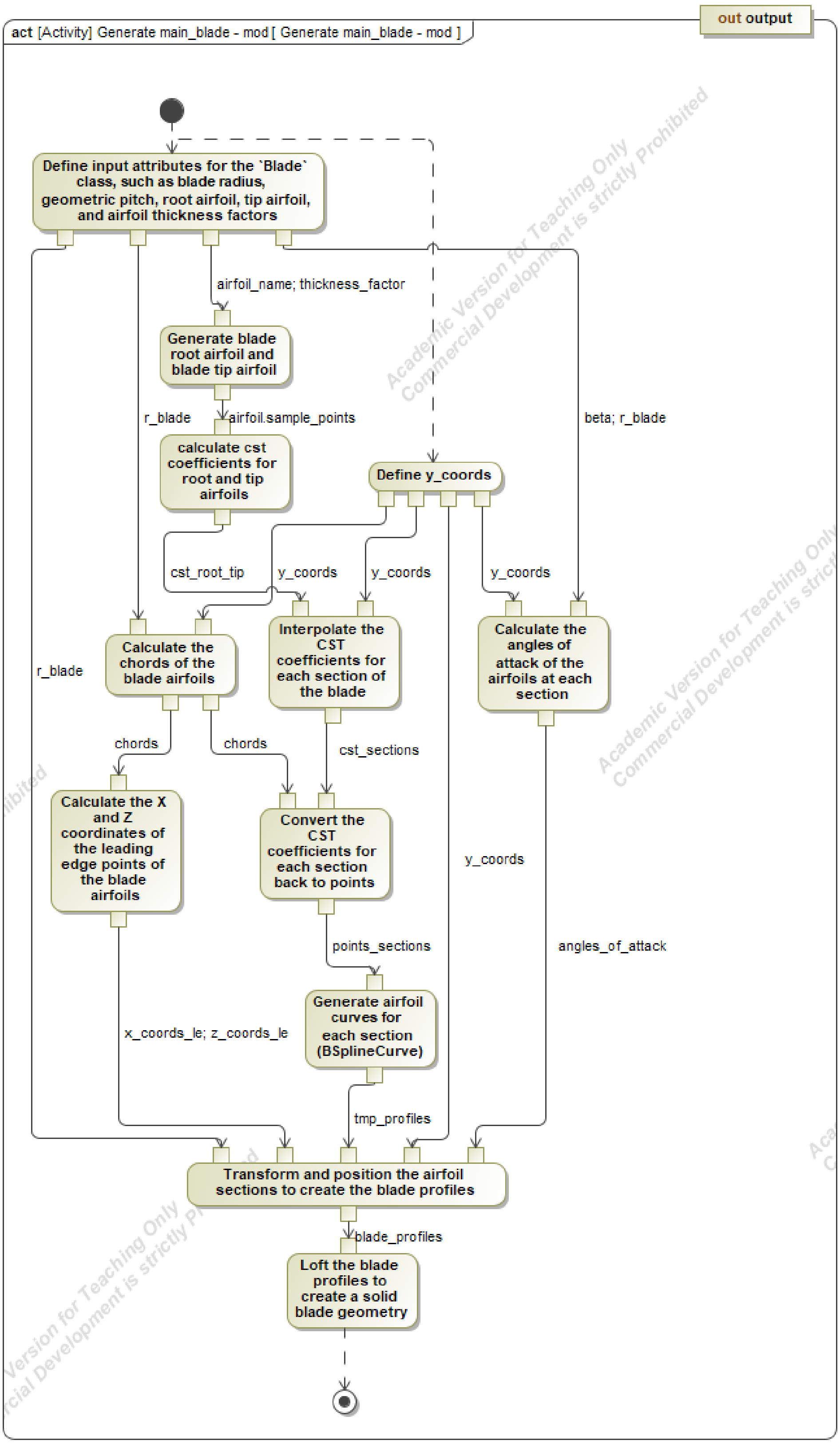


Figure 4.4: Additional activity diagram specifying the "Generate main_blade" activity. This diagram corresponds to a lower level of abstraction due to the presentation of additional details such as the data flow between nodes.

4.1.2. KEY CHARACTERISTICS OF THE PROCESS MODEL

This subsection presents the key characteristics of process model and the activity diagrams that have to be generated by the envisioned REProcess tool. The manually generated example activity diagrams are used to support these discussions. Some of the key characteristics, such as the hierarchical structure, align with previous findings from the DEFAINE project. Others are novel characteristics corresponding to important changes or additions, which includes the adaptations made to the original diagrams from Fernandes [12]. The underlying rationale for these design decisions is also provided.

MULTI-LAYERED, HIERARCHICAL STRUCTURE

A defining characteristic of the process model and its activity diagrams is the hierarchical structure, where the engineering workflows embedded in a KBE application are represented at multiple levels of abstraction. This structure mirrors the hierarchical nature of software and system design, where complex processes are broken down into manageable sub-processes, each corresponding to a different level of detail.

Top-level activity diagrams, e.g. Figure 4.2, provide a high-level overview of the engineering workflow, focusing on the primary functional steps. Lower-level diagrams then decompose each step into more detailed sub-processes, gradually transitioning towards a representation that is closer to the underlying source code. This stepwise functional decomposition is visualized on the left-hand side of Figure 4.5. A corresponding decomposition of the product model is shown on the right. This side-by-side comparison highlights the relation between activities and blocks (i.e. allocation) from each level in the hierarchical breakdown.

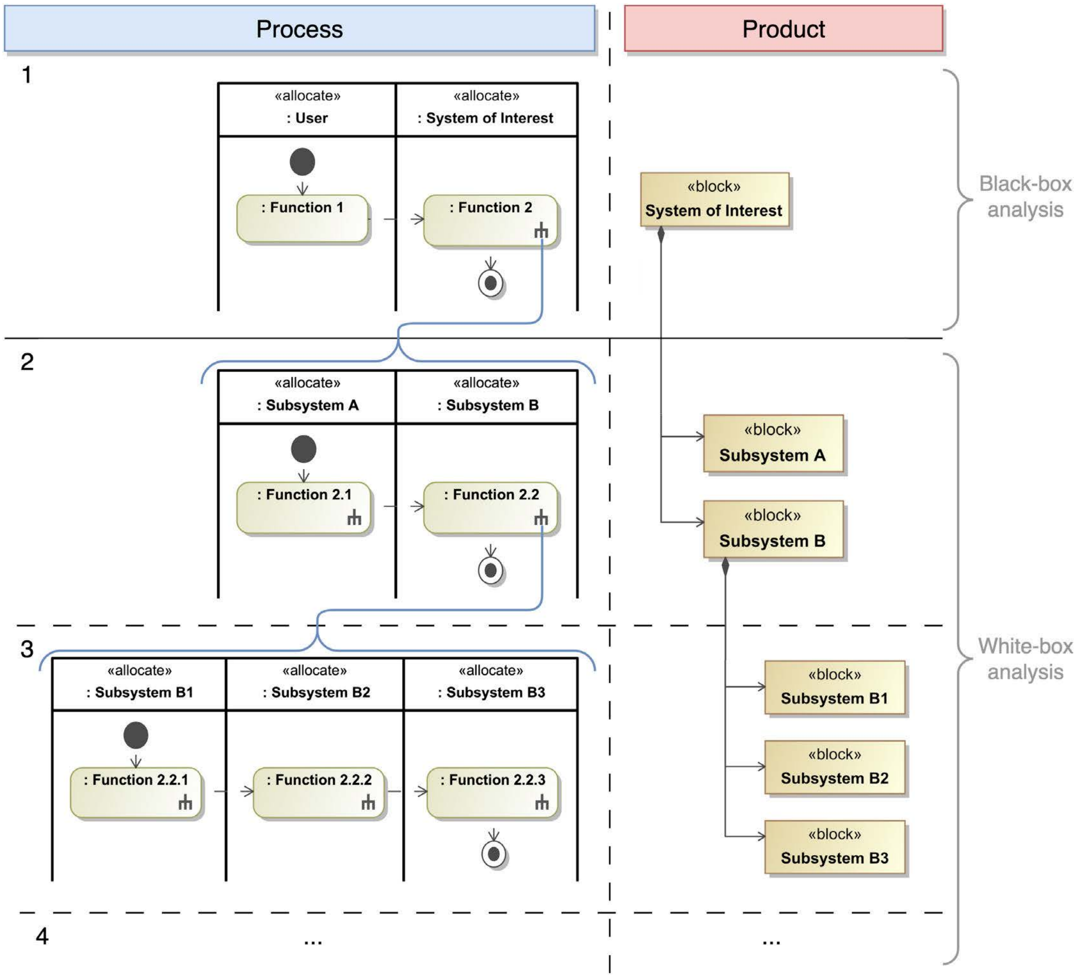


Figure 4.5: Visualization of the hierarchical, multi-layered structure of activity diagrams from the process model (left) and block definition diagrams from the product model (right). The horizontal layers highlight the relation between activities and blocks at each level in the hierarchical breakdown. Obtained from [12], which adapted the figure from [31].

Using a **multi-layered, hierarchical structure** for the process model has a number of **key benefits**:

- **Alignment with MBSE approach:** One notable advantage of generating diagrams at multiple levels of abstraction is that it aligns with the hierarchical approach often employed in MBSE methodologies (e.g. the V-model). As such, it will be familiar and intuitive for engineers and software developers to understand. Furtermore, modeling platforms like CATIA Magic are specifically built to support such hierarchical structures. A great example is the navigation feature that allows users to easily “zoom in” on a particular function block by double-clicking it to open the corresponding lower-level activity diagram.
- **Cognitive effectiveness:** The size and complexity of industry-level KBE applications means that displaying the entire engineering workflow in a single diagram would result in massive diagrams that could overload a user with too much information. In other words, they would have very low *cognitive effectiveness*. One of the key benefits of generating activity diagrams at according to a multi-layered, hierarchical structure is that it servers as a means to divide the information over multiple diagrams while ensuring the user can still easily navigate to the specific information they are looking for, thereby increasing cognitive effectiveness of the model.
- **Facilitating (partial) re-use:** The modularity resulting from the hierarchical structure improves the traceability between model and code and facilitates re-use of (parts of) existing KBE applications, which is one of the overarching objectives of the MDKBE approach.
- **Tailor to user-specific needs:** Different types of users of the REProcess tool will have different needs and objectives. For example, a KBE developer working on implementing a new algorithm might use the REProcess tool to search within applications developed by their colleagues for similar implementations. This requires examining detailed, low-level diagrams, as well as inspecting metadata attached to various model elements (e.g. the specific data parameters that are passed between sub-functions). On the other hand, a customer who ordered the development of a KBE application may use the REProcess tool to verify that the engineering workflow they specified was correctly embedded in the application. For this task, they are not interested in implementation-specifics but only require high-level diagrams that provide an overview of the workflow. By generating the process model using a hierarchical structure, the REProcess tool can be used flexibly and meet the specific needs of various types of users.

Final note: when moving from higher to lower levels of abstraction, the focus shifts from describing what is achieved (e.g. “generate a mesh around the wing”), to how it is achieved (e.g. “1) define mesh boundaries; 2) select meshing parameters; 3) generate point cloud”) and by which part of the KBE applications (i.e. functional allocation). Correspondingly, an interesting way to view the source code of the KBE app itself is to simply regard it as the functional representation at a very low level of abstraction. By taking this alternative perspective, the sharp distinction and between model and code is reduced and it becomes easier to see them as two parts of the same whole.

CAPTURE ALL MAIN FUNCTIONALITIES

KBE applications typically incorporate multiple distinct functionalities, such as geometry generation, data export, high-level parameter computation (e.g., total system weight, manufacturing cost, or energy efficiency), and design optimization. Rather than being combined into a single, continuous workflow, these functionalities are typically implemented as independent yet interrelated processes, which can be triggered individually through the application’s user interface (e.g. by clicking a button).

Since a KBE application consists of multiple, distinct workflows, its process model must reflect this modular structure rather than attempt to integrate all functionalities into a single monolithic diagram. To achieve this, a top-level SysML package diagram is required to provide a structured overview, organizing functionalities into logically grouped packages. Each main functionality can then be represented through a high-level SysML activity diagram, which can be further decomposed into lower-level activity diagrams as needed.

SWIMLANES

A very noticeable characteristic of the adapted SysML are the swimlanes, i.e. the vertical partitioning of the diagram headed by `<allocate>`. As stated in Friedenthal *et al.* [11] [page 259]: *“A set of activity nodes [...] can be grouped into an activity partition (also known as a swimlane) that is used to indicate responsibility for execution of those nodes. A typical case is when an activity partition represents a block or a part and indicates that any behaviors invoked by call actions in that partition are the responsibility of the block or the part. The* *use of partitions to indicate which behaviors are the responsibilities of which blocks specifies the functional requirements of a system or component defined by the block."*

In the original activity diagrams, simlanes were only used to allocate actions to *external* tools. However, while adapting the original diagrams, it was found to be quite challenging to correlate the actions to their corresponding source code segments. To address this issue, the design decision was made to place *all* action nodes within swimlanes **if** the actions in an activity diagram are defined in multiple different Python classes. Due to the swimlanes it becomes very explicit how the execution flows through the various classes, thereby increasing traceability between source code and model.

DATA FLOW CONNECTOR TYPE

Data flow connectors (solid lines) are used as the default type of node connectors. This represents a change with respect to the control flow connectors (dotted lines) used in the original activity diagrams. The rationale for making this change is that data flow connectors allow specifying the data parameters flowing through them. Moreover, data flow connectors are deemed to better represent the backwards chaining mechanism of KBE systems, where a preceding method in the dependency chain is invoked if no valid data for a particular slot is available.

METADATA

The inclusion of metadata represents another key characteristic of the process model. In fact, it is a defining characteristic of SysML models in general. The metadata can be attached to entire diagrams (e.g. the diagram name) as well as the nodes (e.g. actions or blocks) and edges (i.e. node connectors). Some notable examples of the metadata attached to various model elements are functional allocation (also represented visually by swimlanes), requirements allocation, and the specification of comments to attach the corresponding snippets of KBE application source code snippets to actions or blocks. Another, very clear example of metadata being attached to model elements is the data flow, represented by annotations of parameter names next to each data flow connector, shown in Figure 4.4.

4.1.3. THE COMPREHENSIVE PROCESS MODEL

The combination of characteristics outlined above is really what sets a diagram from a SysML model apart from "regular" software visualizations created by many other reverse engineering methods. To highlight this, often the term *comprehensive process model* is used in subsequent parts of this report instead of just referring to it as the process model. Note, however, that both terms refer to the same thing. The additional term merely serves the purpose of highlighting its key characteristics, namely the multi-layered, hierarchical structure, capturing all main KBE application functionalities, and the inclusion of a wide array of metadata. Together, these characteristics ensure clarity and traceability, as well as facilitating the (partial) re-use of existing KBE applications.

4.2. SYSTEM REQUIREMENTS FOR REPROCESS TOOL

The system requirements for the envisioned REProcess tool were synthesized based on information from several key sources: the project objectives outlined in subsection 3.3.1, DEFAINE documents where the reverse engineering step from the MDKBE approach was described [6, 9, 32], and discussions held with supervisors during the early stages of this thesis project. The exercise of manually generating (part of) a process model presented in the previous section also provided key insights. The resulting overview of system requirements is presented in Table 4.1. Each of the requirements is described in more detail in the remainder of this section.

Table 4.1: System requirements for the complete REProcess tool.

Identifier	Description
SR-01	The design of the REProcess tool shall be suited to ultimately be integrated into the envisioned MDKBE approach (Figure 3.1).
SR-02	The REProcess tool shall automatically generate a *comprehensive process model* of the engineering workflows embedded within the analyzed KBE application.

SR-03	The visualization of large, complex engineering workflows shall be divided into a set of *hierarchically structured diagrams*, representing parts of the workflow at *various levels of abstraction*.
SR-04	The generated process model shall be defined using SysML.
SR-05	The generated process model shall include activity diagrams to visualize the engineering workflows.
SR-06	The REProcess tool shall work for KBE applications developed with the Parapy KBE system.
SR-07	The REProcess tool shall be suited for the analysis of industrial KBE applications.
SR-08	Workflows to be reverse engineered by the REProcess tool shall be executable parts of the KBE application.
SR-09	The REProcess tool shall generate cognitively effective activity diagrams.
SR-10	The smallest individual processing step that should still be captured in the process model is the evaluation of user-defined part or attribute slots.
SR-11	If a process model for the analyzed application exists, the REProcess tool shall combine the original and reverse engineered model.
SR-12	The internal data structure used to represent the process model shall facilitate the integration of interrelated workflows.
SR-13	The conceptual design (ie, software architecture) shall allow for a tiered development approach, where each phase can be concluded by a standalone functional prototype.

SR-01: INTEGRATION INTO MDKBE APPROACH

The design of the REProcess tool shall be suited to ultimately be integrated into the envisioned MDKBE approach (Figure 3.1).

Due to the limited scope of this thesis work, the complete REProcess tool and the prototype implemented during this thesis work will mostly be considered and tested as stand-alone tools during their development process. However, it is crucial to keep in mind the long term goal of integrating the tool into the envisioned MDKBE approach (Figure 3.1) because this drives a lot of other requirements and design decisions. Therefore, it is included explicitly as the first system requirement.

SR-02: COMPREHENSIVE PROCESS MODEL

The REProcess tool shall automatically generate a comprehensive process model of the engineering workflows embedded within the analyzed KBE application.

The term “comprehensive process model” was defined in subsection 4.1.3 and refers to the complete functional software architecture, defined as a combination of the following features: capturing all main KBE application functionalities according to a multi-layered, hierarchical structure, and the inclusion of a wide array of metadata including allocation of functions and requirements.

SR-03: HIERARCHICAL DIAGRAM STRUCTURE

The visualization of large, complex engineering workflows shall be divided into a set of hierarchically structured diagrams, representing parts of the workflow at various levels of abstraction.

The hierarchical diagram structure mentioned in this requirement description goes hand-in-hand with the functional breakdown described for SR-02 above. However, whereas SR-02 described the functional breakdown as a required feature for the model itself, SR-03 is specifically about the *diagrams* of the model (i.e. the model views). Essentially, this system requirements states that the diagrams used to define and describe the process model should match the hierarchical structure resulting from the functional breakdown.

SR-04: USING THE SYSTEMS MODELING LANGUAGE

The generated process model shall be defined using SysML.

As described in chapter 3, the decision to use SysML as the main language to describe the model was already made during the DEFAINE project. That design decision was simply translated to this system requirement for the REProcess tool. The standardized data format used to save, import and export SysML models is called XML Metadata Interchange (XMI) [11]. Hence, stating that "the generated process model shall be defined using SysML" implicitly means that the REProcess tool will have to generate an XMI file that defines the process model. For more information about SysML and the XMI data format, the reader is referred to [11] and [33] respectively.

SR-05: SYSML ACTIVITY DIAGRAMS

The generated process model shall include activity diagrams to visualize the engineering workflows.

Similar to the adoption of SysML, this requirement represents a decision that was already made during the DEFAINE project. When considering the following description from [11], one can easily understand why Activity diagrams are the logical choice to represent the reverse engineered workflows: *"The activity diagram is the primary representation for modeling flow-based behavior in SysML and is analogous to the functional flow diagram that has been widely used for modeling system behavior. Activities provide enhanced capabilities over traditional functional flow diagrams, such as the capability to express their relationship to the structural aspects of the system (e.g., blocks, parts)."* [p.205].

SR-06: PARAPY KBE SYSTEM

The REProcess tool shall work for KBE applications developed with the Parapy KBE system.

While many KBE systems share fundamental characteristics, such as object-oriented programming and demand-driven evaluation [34], their implementations vary significantly. As a result, developing a generic reverse engineering tool is impractical. The REProcess tool is therefore tailored for ParaPy, leveraging its built-in dependency tracking and slot evaluation mechanisms. However, the conceptual design may serve as a foundation for future tools aimed at other KBE systems.

SR-07: INDUSTRIAL KBE APPLICATIONS

The REProcess tool shall be suited for the analysis of industrial KBE applications.

The following implications of focusing on analyzing industrial KBE applications are identified:

- **Decent level of code quality:** Large companies standardly use quite rigorous and mature DevOps processes, focusing on adherence to coding standards. Typically they also make use of (custom) code checking tools and code review practices. Moreover, on average their KBE developers will be relatively experienced (or they rely on knowledge from external KBE experts like developers from ParaPy). Thereby, it is assumed that the target KBE applications to be reverse engineering by the REProcess tool feature a decent level of code quality. This is beneficial from a reverse engineering perspective, because it means more information will be present that is useful for reverse engineering purposes (e.g. source code comments, meaningful parameter naming, etc.), and it is present more consistently.

- **Proprietary source code:** A downside of analyzing industrial KBE applications, from a reverse engineering perspective, is that they will often demand confidential data treatment because a lot of proprietary information is embedded in the source code. This means using LLMs hosted on external online servers (e.g. ChatGPT or OpenAI API) is not possible. Instead, downloadable, open-source LLMs will have to be used that run on private servers or local machines.
- **Large, complex applications:** Professional KBE apps developed and used in the industry can feature large, complex engineering workflows. Part of this complexity is interfacing with one or more external analysis and simulation tools such as flow simulation tools (e.g. XFoil[2] or Ansys CFX[3]) and structural analysis tools (e.g. Ansys Mechanical[4]). The REProcess tools should therefore be capable reverse engineering KBE applications that incorporate these tools.

SR-08: FOCUS ON EXECUTABLE PARTS

Workflows to be reverse engineered by the REProcess tool shall be executable parts of the KBE application.

This system requirement is essentially a scope limitation that also happens to align well with the characteristics of dynamic analysis methods. To perform dynamic analysis, all source code included in the analysis must be executable to allow capturing data at runtime. Thus, in practical terms, this requirement states that it should be possible to instantiate the analyzed KBE app and evaluate all slots included in the analysis scope without resulting in errors or crashes.

The phrase "executable parts of the KBE application" indicates that any parts of the KBE app excluded from analysis do *not* have to be executable code. This is made possible by the object-oriented programming style and demand-driven slot evaluation, which allows for non-functional code to exist in the code base without causing errors as long as it is not actively called.

SR-09: COGNITIVELY EFFECTIVE DIAGRAMS

The REProcess tool shall generate cognitively effective activity diagrams.

The term *cognitive effectiveness* stems from the field of cognitive science. It is defined as "a combination of the speed, ease and accuracy with which information can be extracted from a representation" [35, 36]. The term is used in this thesis project to refer to the capability of generated diagrams to improve a user's comprehension of the reverse engineered KBE application. To achieve this, the REProcess tool will break down complex workflows into hierarchically structured diagrams, ensuring users can navigate different levels of abstraction without being overwhelmed. Key design choices — such as swimlanes for functional allocation and data flow visualization — improve readability and traceability.

SR-10: CODE-TO-MODEL MAPPING CARDINALITY

The smallest KBE workflow step that should be captured as an individual model element in the process model is the evaluation of user-defined part or attribute slots.

The term *cardinality* is used to refer to the mapping between KBE source code and knowledge model elements. For the process model, the evaluation of a ParaPy attribute or part slot is defined as the smallest step (i.e. lowest level of abstraction) that should still be captured as an individual element (i.e. an action) in the process model. The definition of input slots is considered too minor to be relevant as a separate process step. Note,

[2] https://web.mit.edu/drela/Public/web/xfoil
[3] https://www.ansys.com/products/fluids/ansys-cfx
[4] https://www.ansys.com/products/structures/ansys-mechanical

however, that the control and data flow associated to input slots is still considered relevant information. Thus, when removing the nodes representing input slots from reverse engineered workflows, care should be taken to transfer the associated control and data flow information to the node that represents the evaluation of the ParaPy part to which that input slot belongs. This is explained in more detail in subsection 5.2.3.

SR-11: COMBINE ORIGINAL AND REVERSE ENGINEERED MODEL

If a process model for the analyzed application exists, the REProcess tool shall combine the original and reverse engineered model.

This system requirement was derived from SR-01. In case the REProcess tool is used as an integrated part of the roundtripping system envisioned in the MDKBE approach, two different versions of the process model will exist: the "original" process model that is no longer up-to-date with the KBE source code after step 3, and the reverse engineered model generated by the REProcess tool. Despite being outdated, the original model can still contain valuable information that was not included in the KBE source code. For example, the allocation between requirements and processes, or certain comments that were added to diagrams of the initial knowledge model. This necessitates developing a functionality to identify and merge corresponding elements from the old into the updated knowledge model.

SR-12: INTERNAL DATA STRUCTURE

The internal data structure used to represent the process model shall facilitate the integration of interrelated workflows.

A proper internal data structure is crucial for ensuring the scalability and integration capabilities of the REProcess tool, particularly as it handles large and complex engineering workflows. Moreover, it will contribute to the development process of the tool itself because it facilitates accessing and storing data, as well as aiding developers in comparing and spotting differences between implementation variants.

SR-13: TIERED DEVELOPMENT APPROACH

The conceptual design of the REProcess tool shall allow for a tiered development approach, where each phase can be concluded by a standalone functional prototype.

This requirement aligns with the second main objective of this thesis project, namely the development of a stand-alone REProcess prototype. Moreover, by designing the REProcess tool according to this requirement, its implementation can be spread out more easily over multiple, separately conducted projects. This will facilitate the continued development of the REProcess tool, after the development of the REProcess prototype in this thesis project is concluded.

5

CONCEPTUAL DESIGN OF THE REPROCESS TOOL

The REProcess tool is envisioned as a software system for reverse engineering KBEapplications. This chapter outlines its conceptual design, formulated based on the system requirements defined in the previous chapter. The design process follows the MBSE methodology, ensuring a structured, high-level approach to system development. The chapter begins with a high-level functional breakdown of the tool (section 5.1), defining its key capabilities and the sequence of steps required to construct a comprehensive process model. The most critical of these operations is the LLM-based SRE method, which forms the core of the REProcess tool. The conceptualization of this novel SRE method is discussed in section 5.2. Other key steps are the identification main KBE app functionalities and the integration of individual reverse engineering results obtained from the SRE method. These are discussed in section 5.3. The last key step, exporting a SysML model, is discussed in Figure 5.5.

Finally, an integrated overview of the conceptual REProcess tool design is presented in section 5.5 and some preliminary conclusion based on the findings in this chapter are discussed in section 5.6.

5.1. HIGH-LEVEL FUNCTIONAL BREAKDOWN

This functional breakdown, presented below, was created based on the high-level system specification presented in the previous chapter.

1. **Identify main KBE app functionalities:** As described in the specification of SR-02, the desired comprehensive process model should capture all main functionalities of a KBE app. Therefore, the first high-level functionality to be implemented by the REProcess tool involves identifying all the main functionalities of the analyzed KBE application. The input for this step is the KBE application source code, and the output is a list of main functionalities. This list serves as the main input for the next step.

   Note that the implementation of this functionality lies outside the scope of the REProcess prototype developed in this thesis work. However, some conceptual design options and suggestions are provided in subsection 5.3.1.

2. **Individually reverse engineer each main functionality:** The second high-level step constitutes individually reverse engineering each main KBE application functionality from the identified list. The set of example activity diagrams from section 4.1 serve as a reference for the expected output from each of these reverse engineering runs. Note, however, that representing the reverse engineered workflows as SysML activity diagrams is defined as a *separate* REProcess tool functionality which is outlined further below. Instead, the output from each reverse engineering run is an internal representation of the reverse engineered workflow that is stored in the so-called information base of the REProcess tool.

   The development of this key REProcess tool feature was the **primary focus** of this research as it directly corresponds to the design and implementation of the **novel LLM-based SRE method** introduced in chapter 3. The exact purpose of this novel SRE method is to reverse engineer individual workflows embedded in ParaPy KBE applications. Its high-level conceptual design is discussed in section 5.2. A detailed discussion of its implementation in the REProcess prototype is presented in chapter 7.

3. **Integrate results:** Once all main KBE application functionalities have been reverse engineered individually, the comprehensive process model (defined in subsection 4.1.3) is created by merging all of the individual workflows stored in the information base. A key aspect of this merging process is the identification of duplicate (sequences of) steps. The common occurrence of such duplicate (sequences of) steps relates to the interconnected nature of KBE applications. Correspondingly, the merger of duplicate (sequences of) steps typically results in a significant reduction of the overall size of the model. More importantly, through this integration process, the set of individual workflows is transformed into a single, interconnected representation: the comprehensive process model.

   Note that the implementation of this functionality lies outside the scope of the REProcess prototype developed in this thesis work. However, some conceptual design options and suggestions are provided in subsection 5.3.2.

4. **Export SysML model:** The final high-level REProcess tool functionality can be regarded as a data transformation. It consists of exporting the internal representation of the reverse engineered process model as an XML Metadata Interchange file, which is a file format can be imported in SysML modeling platforms like CATIA Magic [33].

   Note: Implementation of this functionality lies outside the scope of the REProcess prototype developed in this thesis work. However, some conceptual design options and suggestions are provided in section 5.4.

A high-level concept of the REProcess tool software architecture is defined based on this functional breakdown. This architecture consists of three main software components: the core reverse engineering method, the model composer, and the XMI writer. The **core reverse engineering method** implements the novel LLM-based SRE method (functionality 2 listed above). Its conceptual design is presented next, in section 5.2. Functionalities 1 and 3 are allocated to the **model composer**, which is discussed in section 5.3. Subsequently, section 5.4 outlines the XMI writer that should implement the 4th high-level REProcess tool functionality. Finally, an integrated view of the conceptualized REProcess tool architecture is presented in section 5.5.

5.2. THE LLM-BASED REVERSE ENGINEERING METHOD

This section presents an overview of the conceptual design of the LLM-based reverse engineering method. The purpose of this method is to perform reverse engineering of individual KBE application workflows. In the first subsection, a brief description of key design options is presented, along with the rationale for particular design decisions. The three subsequent subsections describe the sequence of program analysis and abstraction methods that collectively define the conceptual design of the proposed LLM-based SRE method.

5.2.1. CONCEPTUAL DESIGN OPTIONS

The traditional SRE process consists of two key steps: program analysis (extracting program structure and execution flow) and abstraction (deriving high-level descriptions from lower-level representations). The first important design choice consisted of choosing between using an LLM for both of these steps (analysis and abstraction) or restrict its use solely for the abstraction step.

In the context of the REProcess tool, the analysis step must determine the process flow of a ParaPy KBE application — that is, the sequence of execution steps. To assess the feasibility of using an LLM for program analysis, some tests were conducted. Several ParaPy class definitions were provided to an open-source LLM[1]), accompanied by an instruction to deduce the execution sequence. The experiment was then repeated after modifying the order of slot definitions in the source code. Due to ParaPy's object-oriented paradigm and backward-chaining execution mechanism, the order of slot definitions does not dictate execution order [34]. However, despite explicitly specifying this in the instruction to the LLM, it still incorrectly assumed a top-to-bottom execution order, demonstrating its inability to accurately infer control flow. Based on these findings, it was concluded that current open-source LLMs are not well-suited for the analysis step and should instead be leveraged exclusively for the abstraction step, i.e. generating natural language descriptions from extracted code elements.

[1]Llama2: `https://huggingface.co/docs/transformers/model_doc/llama2`

Given that the LLM is unsuitable for program analysis, an alternative method was required. Three main approaches were considered:

- Utilizing existing reverse engineering tools
- Leveraging built-in ParaPy dependency tracking methods
- Developing a custom analysis tool from scratch

Given the substantial effort required for custom development, the first two approaches were prioritized. A review of existing reverse engineering tools—such as MoDisco [25], AgileUML [26, 37], Scalpel [27], and ModMoose [38] revealed significant limitations. Most of these tools were not optimized for Python/ParaPy, relied only on static analysis methods, lacked high-level process abstraction capabilities, or were closed-source and non-extensible.

Fortunately, the second approach—leveraging ParaPy's built-in precedents analysis method—proved highly promising. This method directly accesses ParaPy's dependency tracking system, a core feature of KBE systems [39], providing an accurate and robust representation of execution flow. Given its alignment with the objectives of REProcess tool, precedents analysis was selected as the primary analysis method for extracting execution sequences.

5.2.2. STEP A: PRECEDENTS ANALYSIS

Precedents analysis was selected as the main program analysis method for the REProcess tool. This is a dynamic analysis method that consists of instantiating the KBE application and tapping directly into the dependency tracking system, an key feature of KBE systems [39]. In case of the Parapy KBE system, this can be done by calling an Application Programming Interface (API) method called `get_precedents_tree()` [40]. The `get_precedents_tree()` method has to be executed on a target attribute of an instantiated Parapy object. See the example (Listing 1) provided below. In this example, the analysis target is the center of gravity of the `fused_boxes` part. To obtain the precedents tree for this attribute, the `get_precedents_tree()` method can be called in two ways. Either by running it on the `fused_boxes_cog` attribute (example A, line 35), or by directly running it on the `cog` attribute of the `example_obj.fused_boxes` object (example B, line 36).

The output of `get_precedents_tree()` is a highly nested dependency tree, formatted as a dictionary of dictionaries. A partial example of a dependency tree is shown below in Listing 2. This example corresponds to the output from `precedents_tree_example_A` (line 35 in the previous example). Notice how *cache objects* (e.g. `Cache fused_boxes_cog at 0x1fba365d720>`) are used to define each node. Node connections are established by adding nodes to the list corresponding to the `children` key. The nodes listed as children are preceding nodes, meaning they lie upstream relative to the current node. In other words, children nodes are executed *before* their parent node. The Parapy system also provides a built-in method to generate a visualization of the precedents tree using Graphviz[2]. This method is called on line 37 in Listing 1. The output of this method is visualized in Figure 5.1. Note that this visualization represents the same graph (partially) shown in Listing 2.

DATA TRANSFORMATION TO NETWORKX GRAPH

After obtaining the call tree, a data transformation is applied to convert it to a graph format. The result from this data conversion is the so-called *raw precedents graph*. More specifically, the tree is transformed to a directed graph, which is defined using the NetworkX Python library[3]. This facilitates all downstream manipulation and visualization steps, including the merger of individual workflows into a comprehensive model.

5.2.3. STEP B: GRAPH TRANSFORMATION

The output obtained from the precedents analysis essentially forms the starting point of the reverse engineering method. Looking at Figure 5.1, however, it is clear that a lot of work is still needed before it starts to resemble an activity diagram similar to those presented in section 4.1. Two obvious aspect are the visual layout and the textual node descriptions, which are addressed further down the line. The aspect addressed in this step of the proposed SRE method is the structure of the graph itself.

[2] https://graphviz.org/
[3] see https://networkx.org/

```python

from parapy.core import *
from parapy.geom import Box, TransformedShape, OXY, FusedSolid
from parapy.gui import display

class MyClass(Base):
    width = Input(5)
    length = Input(5)
    height = Input(5)

    @Part
    def start_box(self):
        return Box(width=self.width,
                   length=self.length,
                   height=self.height)

    @Part
    def transformed_box(self):
        return TransformedShape(shape_in=self.start_box,
                                from_position=OXY,
                                to_position=OXY(x=1, y=2, z=3))

    @Part
    def fused_boxes(self):
        return FusedSolid(shape_in=self.start_box,
                          tool=self.transformed_box,
                          color='green')

    @Attribute
    def fused_boxes_cog(self):
        return self.fused_boxes.cog

parapy_obj = MyClass()

precedents_tree_example_A = parapy_obj.get_precedents_tree("fused_boxes_cog")
precedents_tree_example_B = parapy_obj.fused_boxes.get_precedents_tree("cog")

print(parapy_obj.show_precedents_tree("fused_boxes_cog", "PRC_tree__fused_boxes_cog.png"))

```

Listing 1: Source code example of a Parapy class, including calls to `get_precedents_tree` and `show_precedents_tree` methods.

Again referring to Figure 5.1, note that there are significantly more nodes - 33 in fact - than the **four** attribute and part slots defined in the corresponding source code (Listing 1). The additional nodes cluttering the diagram can come from several main sources. The first is the definition of input slots. As defined in SR-10, definition of input slots is considered too minor of a step to warrant representation as an individual model element (i.e. node) in the process model. Another source of clutter is the ParaPy KBE system itself. Many nodes in Figure 5.1, correspond to the internal workings of the ParaPy KBE system. Some examples are the nodes denoted by the terms `TopoDS_Shape`, `builder`, `TOPODIM`, or `TOPOLEVEL` on their bottom row. Collectively, these nodes are referred to as **implementation details**. The goal of this graph transformation step is to remove these implementation details and rearranges the overall graph structure into something better resembling an activity diagram. The main types of methods applied by the graph transformation algorithm are filtering (based on heuristics) and graph edge contraction. Correspondingly, the graph transformation is considered an abstraction method. This chapter will not go into further detail about the algorithm itself. A detailed specification of the implemented algorithm is presented in chapter 7.

An example of the difference between the raw and transformed graph is provided in Figure 5.2. This comparison highlights the differences in **graph structure**. Node content is not meant to be readable because it is not

```
precedents_tree = {
    'node': <Cache fused_boxes_cog at 0x1fba365d720>,
    'children': [
        {
            'node': <Cache fused_boxes at 0x1fba365ce20>,
            'children': []
        },
        {
            'node': <Cache cog at 0x1fba365d8a0>,
            'children': [
                {
                    'node': <Cache TopoDS_Shape at 0x1fba365e1a0>,
                    'children': [
                        {

                            < OMMITTED FOR BREVITY >

```

Listing 2: Truncated example of the `precedents_tree_example_A` (line 35 in Listing 1).

relevant here. This example clearly shows the reduction in node count. Another change achieved by the graph transformation algorithm is that the workflow graph now has a singular starting point, located at the top of the transformed graph. This is an example of how the algorithm transforms the graphs in order to make them better resemble an activity diagram. The graph transformation is described in full detail in section 7.3.

The output of the graph transformation step forms the first main ingredient to generate a SysML activity diagram of the analyzed workflow. Specifically, the structure of the transformed graph forms the blueprint for an activity diagram. It contains all important steps and connectors/dependencies. The other main ingredient, still missing at this point, is the set of the textual descriptions describing each of the performed computation steps represented by the nodes. These are generated in the subsequent step.

5.2.4. Step C: LLM-based abstraction

After the transformed precedents graph is obtained, the LLM-based abstraction step is performed. In essence, this comprises of formulating a prompt for each node in the transformed precedents graph and running this through an LLM - a process called *inference* - to get the corresponding node description. An example of such an LLM prompt is provided in Listing 3, which shows their typical 3-part structure. The first is the **System Prompt** which is used to provide general context and instructions to the LLM. Subsequently, the **User Message** contains the specific request that the LLM should address. Finally, the **Assistant Response** header indicates where the LLM should start formulating its output. For additional information about prompt engineering, the reader is referred to documentation from OpenAI[4] or Google[5].

In this example prompt, observe that both the system prompt and user message contain natural language text instructions, in addition to the source code snippet. Here, system prompt is *"Your task is to reverse engineering an existing knowledge based engineering application."*, and the user instruction (directly below the source code) is *"Write a concise summary (maximum 10 words) of the source code provided above"*. Due to the inherent freedom of natural language, there are infinitely many ways how these instructions could be re-formulated. As a consequence, there is a lot of design freedom related to this aspect of the reverse engineering system. How well an LLM responds to a particular prompt formulation mainly depends on the formatting of the prompts it was trained with, which differs for each LLM. Tuning the prompt formulation can therefore be best done iteratively until a satisfactory formulation is found.

A textual description for every node is obtained by iteratively looking up the corresponding source code snippet from each node in the transformed precedents graph and injecting it into a prompt that is passed to the LLM for inference. Once all individual prompts are processed by the LLM, the main ingredients to generate a workflow diagram are obtained. The graph structure, obtained from the transformation algorithm, forms the

[4] https://platform.openai.com/docs/guides/prompt-engineering
[5] https://cloud.google.com/discover/what-is-prompt-engineering

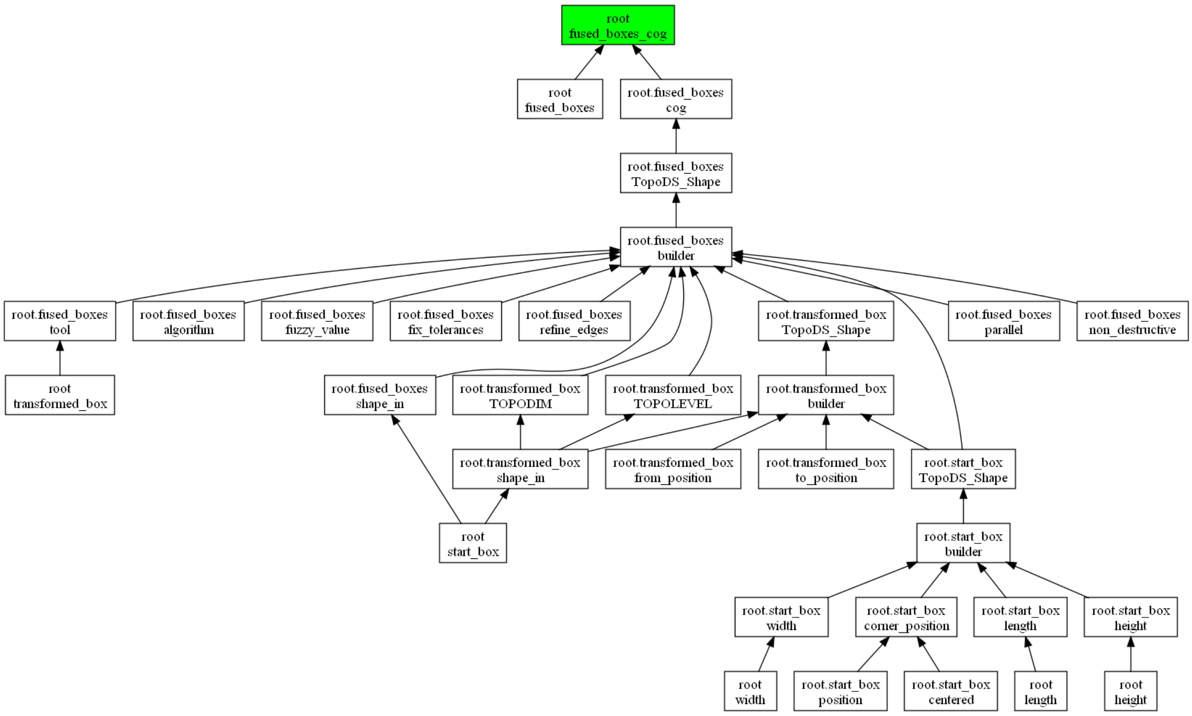


Figure 5.1: Example diagram generated by calling `show_precedents_tree` on the `fused_boxes_cog` attribute (line 37 in 1). The first line of each node is the qualified name of the Parapy object, and the second line is the name of the slot.

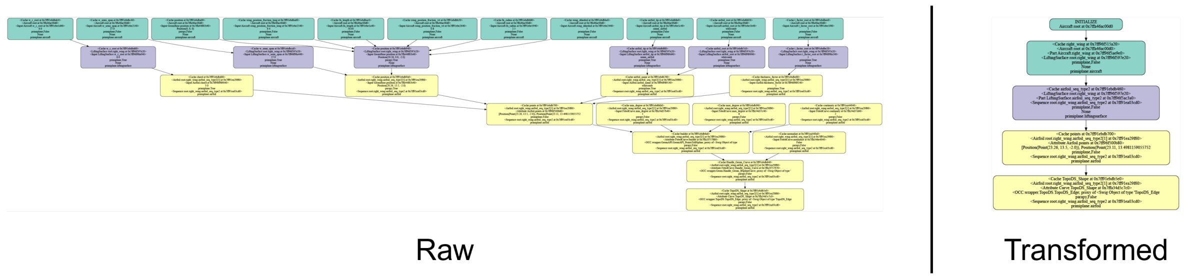


Figure 5.2: Comparison between the raw and transformed precedents graph that highlights the differences in graph structure. Node content is not meant to be readable because it is not relevant here. **Coloring** is used as a substitute for SysML swimlanes and refers to the ParaPy class corresponding to the nodes.

blueprint of the graph and the nodes can be "filled" with the descriptive summaries produced by the LLM. Meanwhile, the original node representation data is all saved as metadata to ensure a rich and complete model representation of the workflow is obtained. The graph in which all of this information is combined will be referred to as the *activity graph*. This name refers to the fact that an activity diagram can be generated based on all information stored in this data object.

5.3. THE MODEL COMPOSER

The model composer performs two REProcess tool functionalities. First, it identifies a list of main KBE application functionalities. This list represents the reverse engineering targets on which the method outlined in the previous section are called. This first functionality is described in subsection 5.3.1. Subsequently, once all individual workflows have been reverse engineered, the second task of the model composer is the integration of the results. This is functionality is outlined in subsection 5.3.2.

Note: the implementation of the functionalities outlined in this section falls outside the scope of this work. They are discussed to provide a better understanding of the overall envisioned system. Furthermore, the proposed conceptual design options provide a starting point for follow-up work.

```

### System Prompt
Your task is to reverse engineering an existing knowledge based engineering application.

### User Message
    @Attribute
    def x_loc_MAC(self):
        # ADDED BY JK
        taper = self.w_c_tip / self.w_c_root
        y_bar = 2*self.w_semi_span/6 * (1+2*taper) / (1+taper)
        x_bar_MAC = tan(radians(self.sweep)) * y_bar
        x_bar_ac = 0.25*(2/3) * self.w_c_root * (1+taper+taper**2)\
                   / (1+taper)

        x_MAC = self.wing_position_fraction_long * self.fu_length + \
                x_bar_MAC + x_bar_ac

        return x_MAC

Write a concise summary (maximum 10 words) of the source code provided above.

### Assistant Response
```

Listing 3: Typical example of an LLM prompt.

5.3.1. IDENTIFY MAIN KBE APPLICATION FUNCTIONALITIES

As described in the specification of SR-02, the desired comprehensive process model should capture all main functionalities of a KBE app. Therefore, the first step of the REProcess tool's internal workflow involves identifying all of these main functionalities within the targeted KBE application. The input for this step is the KBE application source code, and the output is the list of main functions to which the core reverse engineering method will be applied in the subsequent step.

PROPOSED CONCEPTUAL DESIGN

A very straightforward approach to implement this step would be to simply consider the computation of all attribute and part slots defined in the main class of the KBE application as main functionalities. The main class is the one typically used to instantiate the KBE application. For example, the `Drone` class of the Modular UAV app, or the `Aircraft` class of the Primiplane KBE application (example below). The rationale behind this heuristic approach is based on the observation that KBE developers tend to define important slots in this main class to make them easily accessible to users interacting with the KBE application. Of course, this will not hold for all cases, but due to its simplicity and effectiveness it is considered a great starting point for the implementation of this functionality in the REProcess tool.

PRIMIPLANE EXAMPLE USE CASE

As an example, consider the KBE application called *Primiplane*[6]. The Primiplane app is capable of generating aircraft geometry, with a specific focus on movables (ailerons, elevators and rudder), as well as running XFoil analysis[7] to compute basic lift and drag characteristics. A screenshot of the app is provided in Figure 5.3. The main class of this application - the one used to instantiate the GUI presented in Figure 5.3 - is the `Aircraft` class. The list below is obtained by following the main functionality identification approach proposed above. A brief description of each slot, including the corresponding class, is provided for reference. The asterix (*) denotes standard classes from the ParaPy system itself.

1. `fuselage` - Part slot that generates the fuselage geometry by instantiating the `Fuselage` class.
2. `right_wing` - Part slot that generates the right wing geometry by instantiating the `LiftingSurface` class.

[6] Obtained from [41]
[7] https://web.mit.edu/drela/Public/web/xfoil

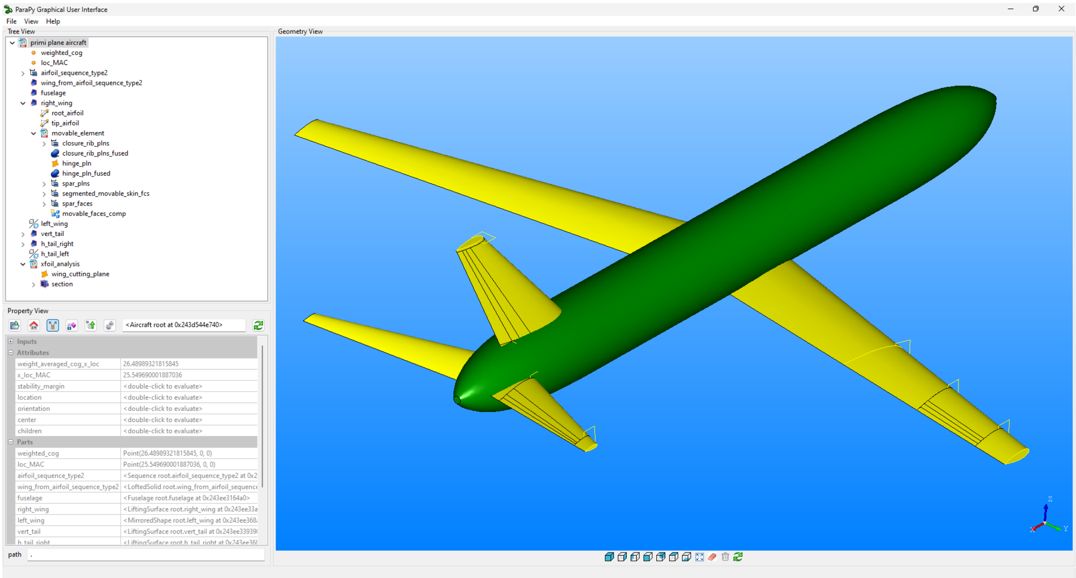


Figure 5.3: Screenshot of the GUI of the Primiplane KBE app

3. `left_wing` - Part slot that generates the left wing geometry by mirroring the right wing using the `MirroredShape*` class.
4. `vert_tail` - Part slot that generates the vertical tail geometry by instantiating the `LiftingSurface` class.
5. `h_tail_right` - Part slot that generates the right horizontal tail geometry by instantiating the `LiftingSurface` class.
6. `h_tail_left` - Part slot that generates the left horizontal tail geometry by mirroring the right tail using the `MirroredShape*` class.
7. `xfoil_analysis` - Part slot that runs the XFoil analysis on the right wing airfoil geometry by instantiating the `XfoilAnalysis` class.

By iteratively applying the SRE method defined in section 5.2 to each of these slots, a set of individual KBE application workflows is obtained. Notably, the occurrence of several duplicate sequences of workflow steps can already be deduced from the slot descriptions added to the set main KBE application functionalities above. For instance, the part geometries from both the `left_wing` and `h_tail_left` are created by mirroring their right-side counterparts, indicating a direct dependency. Correspondingly, the workflow to generate the `left_wing` part will consist of the exact same workflow to generate `right_wing` part plus one additional mirroring step. The same holds for the `h_tail_left` part. Furthermore, the `xfoil_analysis` slot also takes the `right_wing` as input, meaning there are not two but three separate occurrences of the `right_wing` workflow stored within the in the REProcess tool information base after the iterative reverse engineering process is complete. This highlights the need to **integrate the results** from the individual reverse engineering runs into one single representation, which is the focus of the step outlined below.

5.3.2. Integrate results

The objective of this high-level REProcess tool functionality is to integrate all the individual workflows created by the iterative application of the proposed SRE method and thereby generate the **comprehensive process model**.

Proposed conceptual design

To start, an empty “master” graph is created that will contain the process model. Then, an iterative process is followed to merge all of the individual activity graphs into the master graph. This merging process can be implemented using the `compose` function from the NetworkX library[8]. By definition, graph composition merges any identical nodes present in both graph while preserving the edges from both graphs. An example of this process is provided in Figure 5.4. As shown, the S and t-nodes, which were present in both graphs, have been merged into one respectively while preserving the incoming and outgoing edges from both of the original graphs. The result is a single, connected graph.

An important note regarding the NetworkX `compose` function is that its default behavior when merging two identical nodes is to only keep the metadata from one of the original nodes. The metadata from the other node is simply discarded. Therefore, some custom logic must be added before the using the `compose` function itself that identifies identical nodes and takes care of correctly merging the attached metadata.

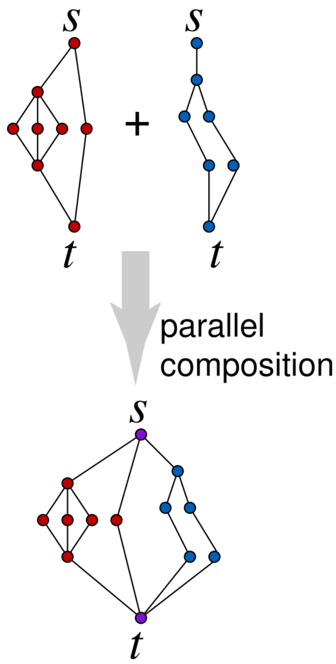


Figure 5.4: Example of graph composition. Adapted from [42]

Three different scenarios can occur when composing a reverse engineered activity graph into the process model graph. These are defined based on the degree of overlap between the two graphs:

1. The first scenario is where the activity graph turns out to be a complete sub-graph of another workflow that was already composed into the model graph. This indicates that the workflow represented by the current activity graph is in fact not a main function. Instead, it should be placed one level lower in the abstraction hierarchy. Naturally, the opposite could also occur. An example of this scenario was previously encountered when

2. The second scenario is that there is absolutely no overlap between the two graphs, aside from the first node representing the initialization of the KBE app. This indicates that the activity workflow being composed into the process model indeed represents a stand-alone functionality, and implies it should indeed be regarded as main functionality of the analyzed KBE application.

3. Finally, there is the mixed scenario where there is partial overlap between the two graphs. This can take multiple forms. The simplest is when both workflows share an initial sequence of lower-level steps, but somewhere the graphs diverge and arrive at a unique end-point. Alternatively, two already-diverged graphs can also “converge” again for some limited sequence of steps, only to then diverge again later. Larger, complex graphs can also exhibit combinations of both. In any case, an overlapping sequence of steps indicates that it collectively provides one higher-level functionality. The same is true for each of sequence of unique steps.

[8] https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.operators.binary.compose.html

Identification of these different scenario's can help to make sense of the interrelations between all of the individually reverse engineered workflows and place them at the right level in the hierarchical structure of the process model. Doing so will also reduce to total amount of main functions with respect to the list originally defined during step 1. The remaining list will only contain functions that are really distinct from each other with regard to the output they provide and the processes that are invoked. Accordingly, the updated list represents the different *use cases* that the KBE app can provide. An overview of these use cases can be represented using a SysML use case diagram. This diagram is a functional representation of the analyzed KBE app at the highest level of abstraction.

5.4. THE XMI WRITER

The XMI generation is the final step of REProcess tool workflow. This step should be viewed as a data transformation, from the internal representation of the process model (a NetworkX graph) to the formal XMI data format specified by the Object Management Group (OMG) [33].

Note: the implementation of the functionalities outlined in this section falls outside the scope of this work. They are discussed to provide a better understanding of the overall envisioned system. Furthermore, the proposed conceptual design provides a starting point for follow-up work.

PROPOSED CONCEPTUAL DESIGN

Implementation of this step can be achieved by extending the product model reverse engineering tool for which an XMI writer was already developed by [32]. This has two main benefits. Firstly, the functionality to setup the overall structure and general content (e.g. specification of the SysML profile) of the XMI file does not have to be developed. This can be handled completely by the existing tool. Secondly, using the full product model reverse engineering tool capability first will mean the model is already populated with all of the blocks that represent the classes. Consequently, the REProcess tool can refer to these same classes when creating the swimlanes in the activity diagrams that specify the functional allocation.

An important remark regarding the implementation of this step pertains to the XMI data format. Unfortunately, while it is the standard for exchanging SysML models, it has one major flaw: a standardized data format for the definition of *model views* (i.e. diagrams) is not included in the official specification from OMG. It only provides a standardized format for the definition of the model itself. As a result, different modeling platforms (Magic Systems of Systems Architect (MSOSA), Enterprise Architect, Papyrus, etc.) each implemented their own variation. Correspondingly, the implementation of this step can not be completely platform-agnostic. A way to (partially) address this issue that was proposed in the DEFAINE project is to use JSON as an intermediate data format [32]. This is visualized in Figure 5.5. Note that Cameo is the former name of the CATIA MSOSA platform. A noteworthy alternative option that lies on the horizon is to implement the XMI writer for SysML v2, which is planned for release in early 2025[9].

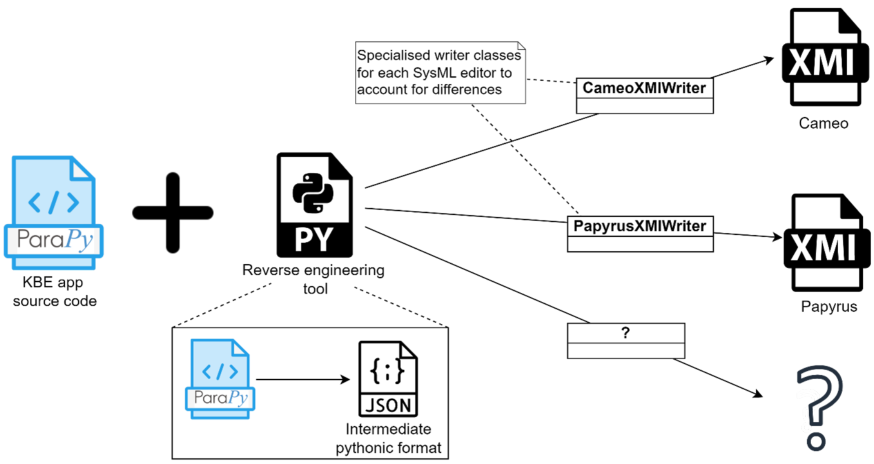


Figure 5.5: Implementation concept for the XMI writer developed in DEFAINE for the product model reverse engineering tool. Obtained from [43].

[9] https://www.edumax.pro/blog/what-is-new-in-sysml-20

5.5. REPROCESS TOOL ARCHITECTURE

An integrated view of the REProcess tool software architecture is provided in Figure 5.6. The three structural software components are represented by the three vertical swimlanes. Within these, the nodes correspond to the high-level REProcess tool functionalities discussed previously in this chapter. The allocation of main these functionalities to the structural software components is indicated using the swimlanes.

- **Core reverse engineering method:** This software component performs the reverse engineering of individual main KBE application functionalities (functionality 2 defined in section 5.1). The conceptual design of the LLM-based SRE method was presented in section 5.2. This component directly correlates to the implementation of the novel LLM-based SRE method. Hence, the REProcess prototype (presented in chapter 7) can essentially be regarded a stand-alone implementation of this software component.
- **Model composer:** The model composer performs two of the high-level REProcess tool functionalities defined in defined in section 5.1, namely step 1 (identifying main functionalities) and step 3 (integrating the results). Therefore, the model composer can be considered as a so-called wrapper function of the core reverse engineering method. This is also reflected by the activity flow in Figure 5.6. Both the steps before and after the execution of the core reverse engineering method are allocated to the model composer. The output generated by the model composer is referred to as internal representation of the comprehensive process model.
- **XMI writer:** The final part of the REProcess tool is the XMI writer, which is tasked with generating the output file that can be imported by a modeling tools like CATIA Magic. To achieve this, the XMI writer has to perform a data transformation, from the internal representation of the composed process model to the XMI data format.

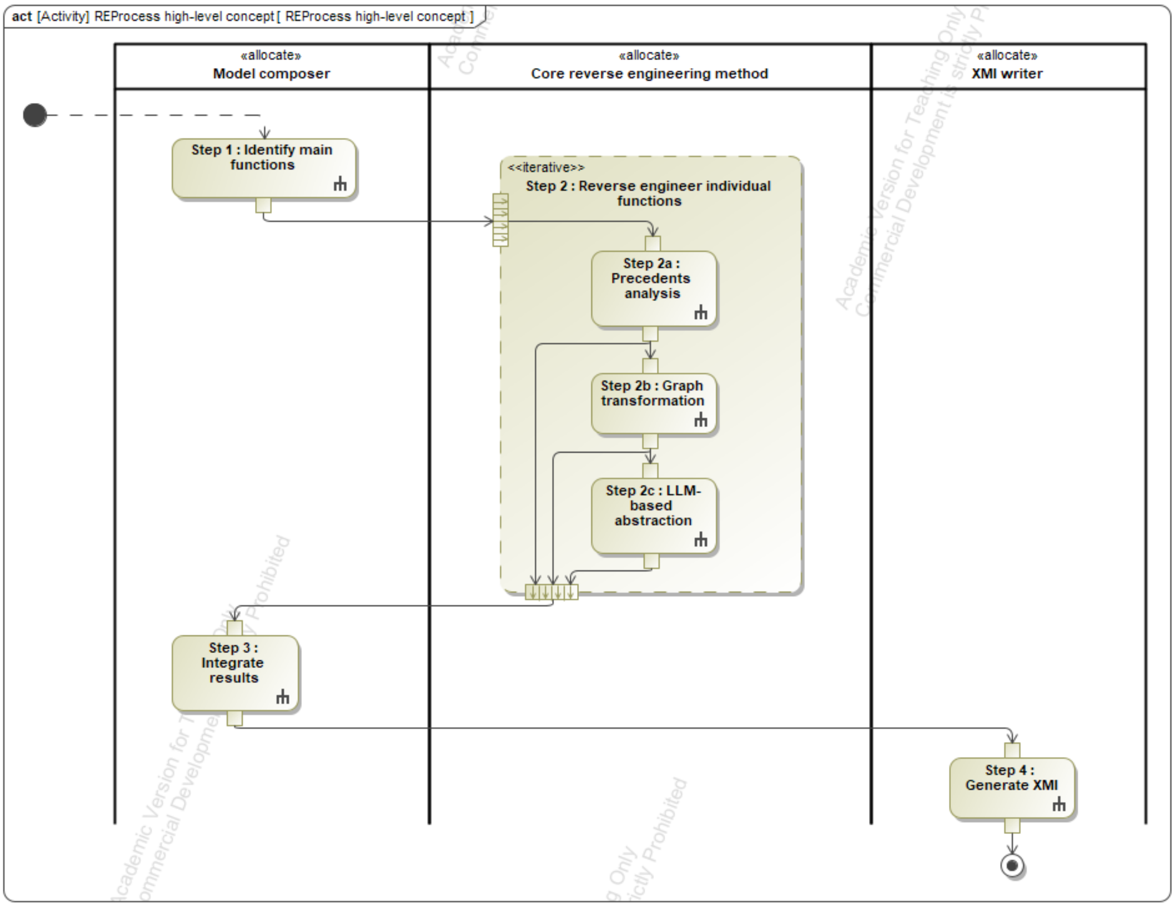


Figure 5.6: Integrated representation of the proposed conceptual design for the REProcess tool. The vertical swimlanes indicate the allocation of the process steps to the main software components of the REProcess tool.

5.6. PRELIMINARY CONCLUSIONS BASED ON CONCEPTUAL DESIGN SPECIFICATION

Based on the findings presented in this chapter, a number of preliminary conclusion can be drawn:

1. The primary research objective of this thesis is to evaluate the feasibility of using state-of-the-art LLMs to reverse engineer process models from ParaPy-based KBE applications. A key conclusion from this conceptual design phase is that LLMs cannot be reliably used for the program analysis step due to their inability to accurately infer execution order in object-oriented, backward-chaining environments like ParaPy. Instead, the precedents analysis method—which directly leverages the dependency tracking mechanism inherent to KBE systems—was selected as a robust and reliable alternative.

2. Furthermore, it was established that structuring and abstracting the extracted execution flow into a meaningful representation is a non-trivial challenge. The graph transformation step plays a crucial role in simplifying complex execution structures, while the LLM-based abstraction step enables the automatic generation of concise, natural language descriptions for each workflow step.

3. This chapter has also indicated the challenge and complexity of developing the complete REProcess tool envisioned for the MDKBE. It requires knowledge and skills about many different topics, namely: KBE systems (both general and Parapy-specific), the Systems Modeling Language, the XML Metadata Interchange format, Software Reverse Engineering methods, graph theory, Large Language Models, development of (object-oriented) software applications, data engineering, acquiring/setting up hardware to run the system locally, and cognitive science to adequately test and improve the performance of the developed system.

NEXT STEPS

The next phase of this research will focus on the implementation and evaluation of the LLM-based reverse engineering prototype. The REProcess prototype will serve as a proof of concept, testing the feasibility of the proposed analysis-abstraction workflow and identifying further improvements. The next chapter presents a detailed specification of the prototype requirements. The design and implementation of the REProcess prototype are presented in chapter 7.

6

REPROCESS PROTOTYPE SPECIFICATIONS

Having established the conceptual design for the REProcess tool, this research now moves to the second phase: development of the prototype that implements and investigates the core reverse engineering functionality, namely the reverse engineering of individual KBE workflows.

This chapter presents detailed specification of the REProcess prototype. An overview of synthesized requirements is presented in the first section below, followed by an in-depth discussion of each requirement. The second part of this chapter goes into detail about the graph transformation algorithm of the REProcess prototype. This so-called black-box specification approach specifies in significant detail **what** the graph transformation algorithm should achieve. An introduction of the approach, along with a glossary defining **key terminology** used during all subsequent stages of the design is presented in section 6.2. Thereafter, the **desired behavior** of the algorithm for every type of node in the precedents graphs is defined in section 6.3. The question of **how** this is implemented is answered in the next chapter.

6.1. REPROCESS PROTOTYPE REQUIREMENTS

Table 6.1: List of requirements for the REProcess prototype implementing the core reverse engineering method.

Identifier	Description	Derived from
PR-01	The REProcess prototype shall be capable of reverse engineering individual workflows performed by a KBE application.	SR-02, SR-03
PR-02	The REProcess prototype shall work for KBE applications developed using the Parapy KBE system.	SR-06
PR-03	The REProcess tool shall capture the following source code features in the reverse engineered activity graphs: • All main types of Parapy slots: inputs, attributes, parts & sequences of parts • Usage of functionalities from the kbeutils library • Slots without precedents	SR-06, SR-07, SR-10
PR-04	The REProcess tool shall be capable of analyzing Parapy KBE applications without errors that include the following source code features: • All features mentioned above for PR-03 • DynamicType • .validators • Regular (i.e. non-decorated) class methods • KBE modules located in different folders	SR-06, SR-07
PR-05	The source code of the target KBE application shall be at least of decent quality.	SR-07

PR-06	The REProcess prototype shall accept all main types of user-defined slots as the reverse engineering target: inputs, attributes & parts.	SR-06, SR-07
PR-07	The targeted slot shall be defined either at root level, or lower in the product hierarchy.	SR-02
PR-08	The REProcess prototype shall only include parts of the KBE application into the model that are involved in the computation of the target slot.	SR-08
PR-09	The REProcess prototype shall visualize the reverse engineered workflows using diagrams that are semantically similar to SysML Activity diagrams.	SR-05
PR-10	The REProcess prototype shall be capable of generating partial workflow diagrams to break down the visualization of a large, complex workflow into multiple, more manageable diagrams.	SR-03, SR-09
PR-11	The REProcess tool shall be deterministic, i.e. repeating the reverse engineering process with identical inputs shall lead to the same output.	SR-01
PR-12	The REProcess tool shall not make use of web services from external parties such as cloud computing from AWS, Microsoft Azure, etc.	SR-07
PR-13	The REProcess prototype shall be capable of reverse engineering one KBE application workflow within 1 hour.	SR-01

PR-01: INDIVIDUAL WORKFLOWS

The REProcess prototype shall be capable of reverse engineering individual workflows performed by a KBE application.

This prototype requirement is basically a formalized way of stating that the REProcess prototype should implement the core reverse engineering method (step 2) of the concept outlined in chapter 5. As such, it corresponds directly to the second of the main objectives that were formulated for this thesis project (subsection 3.3.1).

PR-02: PARAPY APPS

The REProcess prototype shall work for KBE applications developed using the Parapy KBE system.

This requirement is the direct equivalent of SR-06 (Parapy KBE system), only on a prototype level. Therefore, the discussion of SR-06 also applies to this requirement (see section 4.2). It was added to the prototype requirements for the purpose of completeness.

PR-03: CAPTURE KBE FEATURES

The REProcess tool shall capture the following source code features in the reverse engineered activity graphs:

- *All main types of Parapy slots: inputs, attributes, parts & sequences of parts*
- *Usage of functionalities from the kbeutils library*
- *Slots without precedents*

The relation between this prototype requirement and SR-06 (Parapy KBE system) is due to the list of Parapy features used in its definition, making this requirement Parapy-specific. Furthermore, it is related to SR-10 due to usage of the term “correctness”. SR-10 stated that the reverse engineered process model should *correctly* represent the KBE application, and provided a high-level definition of what this means. However, this high-level definition is far from sufficient to guide the implementation of the REProcess prototype. After all, the behavior of a Parapy KBE application due to the evaluation of an input slot or a part slot is markedly different, so it makes sense that the "correct" way to represent them in the reverse engineered process model differs as well. Therefore, the high-level definition from SR-10 was used as a starting point to synthesize more precise specifications for each of the listed Parapy features. These specifications define the correct way to represent each Parapy feature in the reverse engineered process model.

- **Inputs:** According the definition of correct representation provided for SR-10, input slots do not represent a computation step that is significant enough to capture in the model as a standalone step. Instead, the correct way to capture input slots in the process model is to represent them as data objects flowing between the main computation steps. To better explain this, the following two textual descriptions are used as an analogy:

  - *A process consists of three steps: First, compute attribute A. Next, use this attribute to define input slot X for part B. Finally, create part B.*

  - *A process consists of two main steps: First, compute attribute A. Then, create part B (with its input slot X defined using attribute A).*

  The latter of these descriptions is analogous to way that input slots should be represented in the model, as this description does not treat the definition of the input slot as a separate step. Instead, the information regarding the input slot is treated as a side-note by placing it between brackets. For the actual process model, this translates to storing input slots as metadata to the associated nodes and connectors. When stating that the data flow of input slots should be correctly captured, this is what is meant.

  Furthermore, SR-10 also mentioned that the control flow of inputs slots should be captured correctly. This refers to the way that node connections (i.e. graph edges) should be defined in the process model. Specifically, it is considered incorrect to define any node connection by referencing input slots due to the implicit nature of this approach. Instead, the part to which a particular input slot belongs should be used to define these connections.

- **Attributes:** The evaluation of an attribute slot is considered to represent a significant computation step, meaning it should be represented by a corresponding action node in the process model. Moreover, the source code of the attribute should be attached to the action node as metadata in order to improve traceability between model and code. Any slot that provides input for computation of the attribute should be captured in the model by node connectors flowing from those nodes to the node representing the attribute slot. Similarly, node connectors should also be included to all the downstream slots that use the computed value of the attribute as input for their computations. The attribute slot should be attached as metadata to these downstream node connections.

- **Parts:** The evaluation of a part slot is considered to represent a significant behavior step that should be captured in the process model. Correctly representing these steps in the process model is defined similar as for attributes in many regards: adding the source code as metadata, and incorporating all of the node connections to represent the data and control flow aspects. A crucial difference between attributes and parts, however, is that parts are defined as class objects. This is particularly relevant when the class used to create the part is one of the classes that was developed as part of the KBE application itself. In that case, the part creation step invokes a lot of additional behavior that should also be represented in the process model. Since all of this behavior is contained, or "nested", within the part creation step, it is referred to as lower-level behavior.

  For the process model, it is important that the diagrams effectively communicate this nested behavior. The previously presented example diagrams presented in section 4.1 showed two ways to achieve this. The first is to use swimlanes to represent the different parts. A benefit from this solution is that the behavior of multiple different parts can be captured in one visualization. Moreover, the control and data flow between nodes from the different levels can be represented. The alternative to using swimlanes is to define a separate activity diagram that describes the lower-level behavior and link it (indicated by the fork symbol) to the part creation node from the higher-level diagram.

  Of course, parts can also be created using standard classes from Parapy (e.g. `FittedCurve()` or `LoftedSolid()`). In that case, the lower level behavior is not considered relevant to include in the process model because it is standardized behavior from the KBE system. Therefore, it is considered adequate to represent the creation of such a part by a single node, similar to how an attribute is represented.

The distinction between various types of parts and classes, and the specific behavior desired from the REProcess prototype corresponding to each of these types, is one of the most complex aspects of the REProcess prototype. Therefore, this topic is further elaborated in section 6.3, after the other prototype requirements have been discussed.

- **Kbeutils features:** These are add-on features offered by Parapy that provide specialized analysis, synthesis and optimization functionalities. For example, a dedicated meshing toolbox is available, along with XFoil and AVL for aerodynamic analysis, MSC NASTRAN and ANSYS Mechanical for structural analysis to name a few[1]. The explicit inclusion of kbeutils features reflects the high priority that was placed on developing this feature. This priority was derived from SR-07 (suited for industrial KBE applications), based on the consideration that the additional capabilities provided by the kbeutils toolboxes provide are essential for more advanced, industrial KBE applications.

  Since kbeutils features are provided by Parapy, they represent standardized behavior. In this regard, they are similar to Parapy geometry classes. Accordingly, the lower level behavior of these features does not have to be represented explicitly in the process model. However, it is important that the process model captures how these features are integrated into the reverse engineered application.

- **Slots without precedents:** When a slot is defined that does not have any precedents, this should be correctly reflected in the model by ensuring there are no data flow connectors flowing to this node. While this is situation is quite rare, it can occur in KBE applications and should thus be accounted for.

The final aspect of correctly representing the KBE features involves the textual descriptions of attributes and parts to be generated by the LLM. This was defined by the following set of criteria:

- The descriptions should be concisely formulated, using 10 words at maximum.
- The descriptions should adopt a high-level functional perspective, focusing on *what* the step entails rather than *how* it was implemented.
- The description should mention usage of standardized methods/approaches/tools where applicable. For example, “Compute lift and drag coefficient using XFoil”.
- The descriptions of attributes and parts should be clearly distinguishable from each other.
- The descriptions should be accurate with respect to the source code. I.e. they may not contain falsehoods.

PR-04: TOLERATE KBE FEATURES

The REProcess tool shall be capable of analyzing Parapy KBE applications without errors that include the following source code features:
- *All features mentioned above for PR-03*
- *DynamicType*
- *.validators*
- *Regular (i.e. non-decorated) class methods*
- *KBE modules located in different folders*

The word “tolerate” is used to refer to the situation where the REProcess prototype should be capable of successfully reverse engineering a KBE app where a feature is present (and may be invoked during a reverse engineered workflow) but does not have to be captured correctly in the output. In other words, these feature may be ignored but should not lead to any errors. This means both fatal errors, which stop execution of the reverse engineering process, as well as errors in the representation of KBE features that should have been correctly represented according to PR-03. Hence, this requirement specifies features that occupy a middle ground between features listed in PR-03, and features that are completely excluded from the scope

[1] Complete list available here: https://parapy.nl/features/simulation-toolbox

of the REProcess prototype. The specific features listed for PR-04 are discussed individually below. For more background information regarding the listed features, the reader is referred to the Parapy documentation [40].

- **All features from PR-03:** These are included explicitly for completeness. As discussed, the ability to correctly capture features (PR-03) is a step beyond tolerating them.
- **DynamicType:** The DynamicType feature is one of the key methods how dynamic behavior can be implemented using the Parapy KBE system. It allows selecting the class used to create a part at run-time, based on a conditional statement. There are two main ways how this feature can be used. The conditional statement can either be linked to user input, to allow users to switch between different modes while interacting with the GUI of the KBE app. Alternatively, internal logic can be implemented to autonomously switch between different modes based on certain conditions and thresholds.

  When the DynamicType feature is used in a reverse engineered workflow, the dynamic aspect of its behavior does not have to be represented perfectly in the model. Specifically, the control behavior (i.e. abstraction of the conditional statement + different workflows related to each option) do not have to be captured. Only the path that was actually executed during a particular run has to be correctly represented in the model.
- **Validators:** As the name suggests, these can be defined in the source code of a KBE application to validate whether a provided input value meets certain conditions. There are two syntax variants to specify a validator, of which one meets the criteria posed in the previous chapter to be classified as a significant step. Despite this, they are not considered as such because they actually define lower-level behavior of input slots. Since the declaration of input slots is considered to not represent a significant step, its lower level behavior is also excluded.
- **Regular class methods:** This refers to class methods that are not decorated to define them as a Parapy slot, but are simply implemented as a class method of a regular object-oriented Python program. This is allowed by the Parapy system and does not lead to any errors. However, the main consequence of not being decorated as a Parapy slot is that these regular class methods are not included in the dependency-tracking system, which means they are also not returned by the `get_precedents_tree()` method. As such, the inability to capture the computation steps corresponding to regular class methods in the reverse engineering output is essentially a limitation of the system resulting from the conceptual design choice to use `get_precedents_tree()` as the main analysis method.
- **KBE modules located in different folders:** The source code of more elaborate and complex KBE apps is usually structured using some sort of hierarchical folder structure. This is relevant and should be accounted for because the KBE app has to be instantiated by the REProcess prototype in order to run the dynamic precedents analysis. Due to the nature of Python import statements, a specific implementation is required in order to prevent breaking runtime errors.

PR-05: QUALITY CODE

The source code of the target KBE application shall be at least of decent quality.

The concept of "decent quality code" was operationalized by reviewing the most essential Python and Parapy coding guidelines [40, 44], as well as taking into account specific limitations imposed by the conceptual design decisions presented in the previous chapter. This resulted in the following list of criteria used to define KBE code that is at least decent quality:

- **Meaningful parameter naming:** For example, $length$ instead of l, or $air\_density$ instead of rho. Aside from being good coding practice, the textual information embedded in meaningful parameter names is a key source of information for an LLM to make sense of a piece of code and perform abstractive summarization. Without it, the LLM-based abstraction step will perform worse.
- **Well-segmented code:** This is defined as at least 90% of attribute slots being defined with less than 100 lines of code. Note that both numbers used to quantify this criteria are "ball-park figures" and should be used more as an indication/guideline than as a hard boundary. What this criteria aims to capture is that

defining a single Parapy attribute using 1000 lines of code is generally considered bad practice, as it is not in line with the object-oriented programming style and detracts from the legibility of the code.

Moreover, such massive attribute source code definitions also pose a problem for the conceptualized software design because the evaluation of an attribute (or part) slot was defined to be the smallest, still-relevant step to include in the process model. Correspondingly, the conceptualized REProcess tool always summarizes an attribute as a single step, regardless of whether it is defined using 20 or 1,000 lines of code. In case of the latter, however, this is most likely too big of a reduction in information. As a result, the generated process model would be overly simplified, or in other words, its "resolution" would be too low to properly represent the workflow.

- **Well-structured code:** This refers to the typical hierarchical class structure used for KBE applications. One class should be the main class, which uses several other application classes to create its main parts. Moreover, each of the application classes of the KBE application should represent some larger, more high-level function.

  Such a code structure is a rather natural thing to do, so most applications will meet this criteria. It is important for the conceptualized reverse engineering system because the slots representing the main functions are expected to be defined in such a main class.

- **Docstrings:** Additional information about attribute and part slots should be added as docstrings, rather than using comments on the line above the start of the class method declaration. This complies with the PEP style guide [45]. In addition, it is useful for the LLM prompt generation because docstrings will be included when the source code of a slot is parsed, while comments placed above are not. Similar to having useful parameter naming, the additional contextual information provided in the docstring will improve the abstraction performance of the LLM.

This prototype requirement was derived from SR-07. The rationale is that the source code of industrial KBE applications is generally expected to be of higher quality than that of applications developed in educational or academic contexts. This is primarily due to the greater experience of developers, both individually and due to the presence of larger teams, as well as the implementation of professional development practices such as code reviews and standardized development pipelines.

PR-06: TARGET SLOT TYPES

The REProcess prototype shall accept all main types of user-defined slots as the reverse engineering target: inputs, attributes & parts.

Recall that the target slot refers to the slot on which the `get_precedents_tree()` method is called. Thus, this requirement stipulates that any input, attribute, or part slot should be a valid input for which the corresponding workflow can be reverse engineered by the REProcess prototype. Note that, while the evaluation of an input slot itself might not represent a significant computation step, the workflow executed to arrive at a particular input value can still be interesting for a user of the REProcess tool.

This requirement is particularly comprehensive for part slots because, as will be explained in the implementation chapter, several different categories of parts exist, each with unique underlying behavior. For example, there is a significant difference between specifying a singular part and a part sequence. In Parapy, this is achieved by using the *quantify* argument when defining a part. Although this change is small and the source code definition of singular parts and part sequences looks very similar, it triggers large differences under the hood.

In order to meet this prototype requirement, a solution had to be developed that accounts for all possible variations in underlying Parapy behavior of different types of part slots.

PR-07: TARGET SLOT LEVEL

The targeted slot shall be defined either at root level, or lower in the product hierarchy.

The level of the target slot refers to its nesting depth relative to the root object, so it is equivalent to Parapy's concept of `tree_level`. For example, in the primiplane application, the aircraft itself is the root object, created using the Aircraft class. Slots of the root object, such as the `fuselage` or the `right_wing` part, are considered "root level target slots". In contrast, a lower level target slot, such as the center of gravity of the right wing, would be referred to as "aircraft.right_wing.cog" in the product hierarchy.

According to SR-02, the REProcess prototype should be capable of reverse engineering any slot evaluation workflow, regardless of the target slot's level in the product hierarchy. Given that different implementations of the `get_precedents_tree()` analysis method are required to handle the two possible variations (root vs. lower level target slot), this prototype requirement was established.

PR-08: ONLY INVOLVED SLOTS

The REProcess prototype shall only include parts of the KBE application into the model that are involved in the computation of the target slot.

This requirement relates to several system-level requirements, notably SR-08 and SR-09. SR-08 specifies that only the operational parts of the KBE application need to be reverse engineered. By focusing strictly on the components involved in the computation of the target slot(s), the risk of encountering errors from unimplemented or faulty slots is minimized.

Additionally, this requirement aligns with SR-09, as including only the actively used parts of the KBE application helps to focus and reduce the overall information load, thereby enhancing cognitive effectiveness.

PR-09: ACTIVITY DIAGRAM-LIKE VISUALIZATIONS

The REProcess prototype shall visualize the reverse engineered workflows using diagrams that are semantically similar to SysML Activity diagrams.

Naturally, this prototype requirement was defined as a substitute for SR-05, which specified that the reverse engineered workflows should be visualized by means of activity diagrams. As indicated by the REProcess software concept, meeting this system requirement would involve implementing the XMI writer, which is extensive and beyond the scope of this thesis project. Instead, this prototype requirement stipulates that the diagrams should generated that are semantically similar to SysML activity diagrams. This means a substitute should be found for all diagram elements discussed in the specification of SR-05, namely:

- A clear initial node that defines the start of the workflow.
- Action nodes to specify each behavior step.
- Data flow connectors to indicate how computed slot values are used by subsequent processes.
- Metadata attached to both action nodes and the connectors to include additional key information (e.g. source code corresponding to an action node).
- Swimlanes to add structure and present the allocation of functions to structural software components (i.e. classes).

PR-10: PARTIAL VISUALIZATIONS

The REProcess prototype shall be capable of generating partial workflow diagrams to break down the visualization of a large, complex workflow into multiple, more manageable diagrams.

This is the requirement is the prototype equivalent of SR-03, which stated that a set of hierarchically structure diagrams should be used to visualize large, complex workflows. It is also related to SR-09 because doing so also contributes to the cognitive effectiveness of the diagrams. The main difference between the SR-03 and PR-10 is that for the prototype, the functionality does not have to be automated. For the prototype, it is acceptable that the partial visualization functionality works based on user input. Specifically, the user should specify which parts of the complete workflow should be included in each partial visualization. Based on this input, the REProcess prototype should generate the corresponding diagrams.

The development of the feature specified by this prototype requirement is expected to provide two benefits. Firstly, it demonstrates the conceptualized approach to generate a set of hierarchically structured diagrams, which is a key aspect of the envisioned system. In addition, it unlocks the ability to assess the cognitive effectiveness of generating a set of hierarchically structured diagrams instead of a single, very large diagram. Therefore, a follow-up study can be performed using the REProcess prototype to evaluate possible implementations of the hierarchical breakdown approach, which would be super valuable input for subsequent development of the complete REProcess tool.

PR-11: DETERMINISTIC PROCESS

The REProcess tool shall be deterministic, i.e. repeating the reverse engineering process with identical inputs shall lead to the same output.

This prototype requirement directly related to the conceptual design decision to use an LLM-based abstraction approach. Anyone who previously used a LLM-based tool such as ChatGPT will recognize that asking the same question twice does not result in identical answers. A simplified explanation for this behavior is that the underlying LLM is given some "artistic freedom" by introducing small stochastic variations in the answer formulation process. The purpose of these variations is to make the generated answers sound less "mechanical" due to being repetitive and predictable. Of course, this is only an issue when the LLM is formulating long answers, which does is not the case for this LLM application in REProcess tool. On the contrary, there are very good reasons why the opposite behavior is desired for the REProcess prototype (and tool). The following two scenario's underline why it is important that a deterministic core reverse engineering method is implemented that generates the same output when identical input is supplied multiple times.

- In practice, it is extremely likely that some parts of a KBE application will be analyzed by the REProcess prototype multiple times while remaining unchanged. For example, when a KBE developer is focused on implementing a certain function in one part of the code while leaving the rest of the app untouched. When this developer uses the REProcess tool/prototype several times during the implementation process to evaluate the effect of their changes, it would be very confusing if the tool generates a completely different output even for functionalities that they did not touch during their work.
- As stated in SR-01, the final goal for the REProcess tool is to be integrated in the envisioned MDKBE approach, which includes generating a model database. For this database, change management is important. It is undesirable to have a particular KBE function is represented multiple times in the model database while the underlying source code is actually identical. This requires the method to be deterministic.

PR-12: NO EXTERNAL COMPUTING

The REProcess tool shall not make use of web services from external parties such as cloud computing from AWS, Microsoft Azure, etc.

This requirement was derived from SR-07 which stated that the REProcess tool should be suited for analyzing industrial KBE applications. In virtually all cases, these applications will be proprietary software. Because using cloud computing services from external parties poses a confidentiality risk, using these services is not an acceptable for the REProcess tool nor the prototype.

PR-13: COMPUTATION PERFORMANCE

The REProcess prototype shall be capable of reverse engineering one KBE application workflow within 1 hour.

Performing the workflow reverse engineering within 1 hour of computation time is important for future integration into the MDKBE approach (SR-01). In fact, much better performance will be required to make the envisioned development approach practically feasible. A KBE developer who makes changes to a Python module should not have to wait more than a couple of seconds before the behavioral impact of their changes is processed by the REProcess tool. However, there are a lot of performance gains that can still be achieved between the implementation of the prototype and the complete tool (see the discussion in chapter 9). Meanwhile, the relentless improvement of computing power and LLM capabilities will also lead to a reduction in the required computation time. Based on the large potential for improvements, the computational performance target was set at a maximum of 1 hour for the prototype. This should leave sufficient margin to ensure that the general approach is feasible for future integration into the MDKBE approach.

6.2. INTRODUCTION OF THE BLACK-BOX SPECIFICATION APPROACH

Based on the presented set of prototype requirements, the desired behavior of the graph transformation algorithm was specified using the so-called black-box approach. This specification can be regarded as an intermediary step between the prototype requirements and the actual software implementation. It forms the blueprint for the implementation of graph transformation algorithm.

The general approach consists of reviewing the input data, in this case the raw precedents graphs generated by the `get_precedents_method()`, and correlating patterns and features from this input data to the desired output. This can be best explained using the example presented in Figure 6.1.

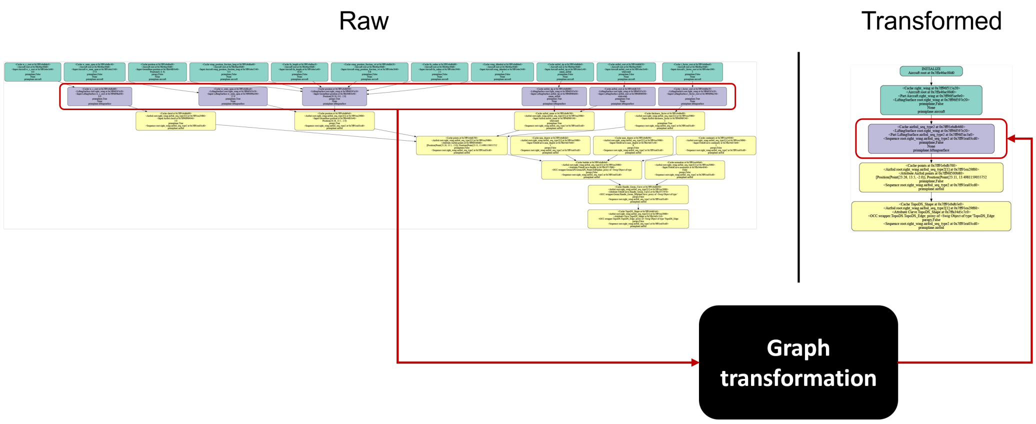


Figure 6.1: Example showing how a black-box approach was used to specify the desired functionality of the graph transformation algorithm.

In this example, the node highlighted in the transformed graph on the right-hand side represents the creation of a part called `right_wing`. According to the previously provided specification (PR-03) this is how the creation of a part should be represented in the model. However, as highlighted on the left-hand side in the example, the generation of the `right_wing` part was represented in the raw precedents graph by a total of six nodes. Closer inspection would reveal that each of these nodes corresponds to the initialization of an input slot of the `right_wing` part. The black-box representation provides an elegant and concrete way to combine all of this information into a very concrete specification that describes one of the functionalities to be implemented by the graph transformation algorithm. Namely, the graph transformation algorithm should contract all of the input nodes into their corresponding part node.

The black-box behavior specification approach was applied to each of the Parapy features listed in prototype requirement PR-03 to obtain a complete set of specifications. However, before presenting these specifications, first some additional terms and concepts are introduced regarding the Parapy KBE system and the node objects from the raw precedents graph. These are used to characterize various aspects of the precedents graphs and are crucial for the subsequently presented discussion of the desired black-box behaviors.

6.2.1. DEFINITION OF TERMS AND CONCEPTS

In order to adequately discuss the desired black-box behavior of the abstraction methods to be implemented for the REProcess prototype, it is crucial to first define a number of terms and concepts that can be used to characterize and discuss various aspects of the precedents graph and the Parapy KBE system.

PARAPY VS. APPLICATION OBJECTS

The first concept to introduce is the distinction between Parapy and application objects:

- **Application node/slot/class/function/etc.:** In the context of the reverse engineering method, the terms "application node/slot/class/function/etc." are used to refer strictly to things for which an explicit source code definition is present in the KBE application code itself. In other words, anything *not* defined in the code of the Parapy KBE system itself. The opposite of "application" nodes/slots/classes/functions are "Parapy nodes/slots/classes/functions". This categorization is dichotomous, meaning a particular node/slot/class/function/etc. is either an "application" or a "Parapy" thing.
- **Parapy node/slot/class/function/etc.:** in the context of the the reverse engineering method, the terms "Parapy node/slot/class/function/etc." are used to refer to nodes, slots, classes, functions from the Parapy KBE system itself. For example, all the Parapy geometry classes such as `Box()`, `LoftedSurface()`, or `Point()` are Parapy classes. Some other notable examples are `Sequence()`, `DynamicType()`, `Base()`, `GeomBase()`, and any classes or functions imported from the kbeutils package. Essentially, anything *not* defined explicitly in the source code of the KBE application.

  It is important to note that even when an object is instantiated using an application class, this object can still have Parapy slots when these were added into the application class via inheritance from a Parapy class. A good example would be the `LiftingSurface(LoftedSolid)` class from to the previously used Primiplane example. Here, the inheritance from `LoftedSolid()` introduces numerous additional slots such as color, position, volume, faces, edges, builder, and children, to name a few. These are all categorized as Parapy slots, even though they are properties of an application object. Note however that Parapy slots can still be overwritten with a custom specification in the LiftingSurface() class itself. In that case, they would again be designated as application slots.

  As a general rule, designating something as a "Parapy node/slot/class/function/etc." means it is not considered relevant to include in the process model. These are the "implementation details" mentioned in the discussion of step A from subsection 5.2.2, and should generally be removed from the raw precedents graph by the graph transformation algorithm. A few important exceptions do exists, however. Most notably the so-called geometry node, which will be discussed later in this section.

TERMS AND CONCEPTS RELATED TO PRECEDENTS GRAPH NODES

Previously, when discussing precedents graphs the focus was on the structural perspective. From hereon, however, the representation and characteristics of the individual nodes also becomes relevant. Therefore, this subsection is dedicated to providing an understanding of key terms and concepts used to describe nodes from the precedents graph. The explanations will be presented by referring to the annotated close-up of a precedents graph node[2] presented in Figure 6.2.

The first important note to make regarding the visualized node representation is that the nodes from the raw precedents graph are actually defined solely by the cache object presented on the first row (annotated by "Node"). All information presented on the additional rows are properties of this cache object. In fact, each node's cache object has dozens more properties[3] than those that are shown here. Note that having access to all of this data is a direct result from using `get_precedents_tree()` as the main analysis method, because that method is what actually returned the nodes as cache objects. Having access to all of this additional data from the KBE system is therefore one of the key benefits of using this approach.

[2] For the interested reader, this particular node is the third purple node from the left highlighted in Figure 6.1.

[3] Even more when also including nested properties. Inspecting all of these can be best done by using the variable window in PyCharm's Python console

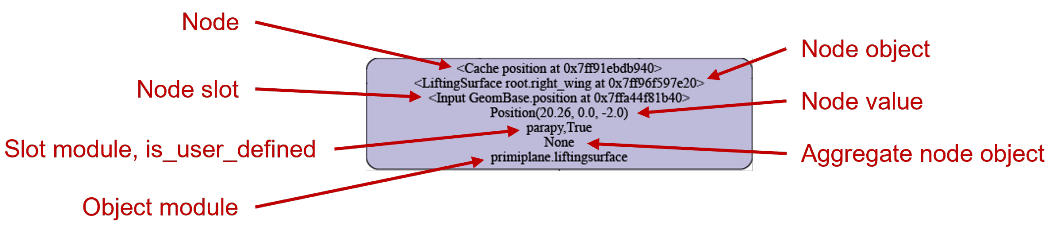


Figure 6.2: Annotated example of a precedents graph node, highlighting the final selection of additional node properties that were found to be relevant for reverse engineering purposes

The specific node properties shown in Figure 6.2 were selected over the course of the development process of the REProcess prototype. They all present information that is somehow important for the reverse engineering methodology. Each of these properties is described below.

- **Node:** This property can be accessed using the `<node>` syntax. Each node in the raw precedents graph is a defined as a cache object. Hence, they can be regarded as the fundamental building blocks of the graph. A cache object is included in the precedents graph corresponding to every slot evaluation that was executed during the analyzed workflow.

- **Node object:** This property can be accessed using the `<node>.obj` syntax. It contains the class name and instance of this node's parent object. Given that each node corresponds to the evaluation of a slot, it is crucial to also know to which part object the slot belongs. This information is provided by the node object.

  The node object in the annotated example above indicates that this particular node represents a slot from a part that was created using the `LiftingSurface()` class. Furthermore, it provides the qualified name of the parent object ("root.right_wing"), which specifies both the name of the parent part, and its location in hierarchical product tree.

- **Node slot:** This property can be accessed using the `<node>.slot` syntax. As can be observed from Figure 6.2, it provides information both regarding the type of Parapy slot (input/attribute/part) as well as the slot data itself. From a reverse engineering perspective, the information regarding slot type is very useful. As will become apparent later in this chapter, differentiating between the different slot types is an essential capability for the reverse engineering method.

  Additionally, the slot data itself is also essential. Observe how the slot from the annotated example shows that the source code definition of the position slot was inherited from the `GeomBase()` class. This indicates that the data from the node slot can be used to differentiate between Parapy slots and applications slots. Specifically, this is achieved by using the data from *slot module* and *is_user_defined*, which are sub-properties of the node slot itself. Both are introduced further below in this list.

  By combining the information from the node object and node slot it really becomes apparent what a particular node represents. In Figure 6.2, for example, combining this information enables the understanding that this node represents the definition of the `right_wing` position.

- **Node value:** This property can be accessed using the `<node>.value` syntax. As the name suggests, it contains the value of the slot represented by the node. In Figure 6.2, for example, it indicates that the position of the `right_wing` part is defined at XYZ position: 20.26, 0, -2.0.

  At this point, one might wonder why the numerical value of a slot is so interesting from a reverse engineering perspective that it was was included in the selection of node properties to be visualized. This is a just question because the answer is, *it isn't*. However, there is one particular type of nodes, namely part nodes, for which the value property does contain key information. This will be further explained during the introduction and discussion of the part node concept, later in this chapter.

- **Slot module:** This property can be accessed using the `<node>.slot._module` syntax. Note how this syntax indicates that, as mentioned before, the so-called slot module is a sub-property of the node slot. Furthermore, it should be noted that the value of the slot module displayed in Figure 6.2 is an abbreviated version of the full output, which is the full qualified name indicating the specific Python module used to instantiate the object. For differentiation purposes however, only the first part is relevant so the rest is omitted. Correspondingly, the only possible return values of slot modules are "parapy", "kbeutils", or the name of the analyzed KBE application's root directory (for the node from Figure 6.2 this would be "primiplane"). Accordingly, the slot module can be used to easily determine whether a node represents a parapy or application slot. The relevance of this distinction was discussed in the previous subsection.

- **is_user_defined:** This property can be accessed by using the following syntax: $< node > .obj.is\_user\_defined(str(< node > .slot.\_\_name\_\_))$. Note that this differs from the previous properties because $is\_user\_defined$ technically is a method instead of a property itself. As a consequence it has to be called, instead of the property value being available directly. The returned value by calling this method is a boolean ($True$ or $False$). To explain this method, the following description was taken from the Parapy documentation [40]:

  User-defined means that user has overwritten the default value for a slot, either by:

  1. *Passing it into the constructor of this object, e.g.* $Box(width = 1)$*; changing the value programmatically at runtime, e.g.* $obj.width = 1$*; or by changing the value in the GUI.*
  2. *Defining a child rule in @Part expressions,*
  3. *The slot is defaulting and the user defined a slot higher up in the composition tree that encapsulates the computation.*
  4. *The slot is_required and the user defined a slot higher up in the composition three that is trickling down its value.*

  From this description, it is concluded that the $is\_user\_defined$ value can be very relevant for reverse engineering purposes. When it is $True$, it means the data represented by a node might be relevant for users of the REProcess tool who aim to understand the behavior of the KBE application, even when it is a Parapy node. Note that a $True$ value for $is\_user\_defined$ therefore provides an exception to the general rule where all Parapy nodes are considered implementation details.

  The node from Figure 6.2 is a good example of this case. The source code from the Primiplane app that defines the $right\_wing$ part is presented below. Note that the position slot is passed explicitly into the constructor. Hence, the positioning of the $right\_wing$ part is custom behavior, unique to this particular KBE app which makes it relevant to capture (as metadata) in the reverse engineered process model. However, as shown in Figure 6.2, the position slot appears as a Parapy node in precedents graph because the source code definition itself is still inherited and not overwritten by a custom definition in the `LiftingSurface()` class. This explains the added value of evaluating $is\_user\_defined$ alongside the slot module property.

```

    @Part
    def right_wing(self):
        return LiftingSurface(
            pass_down="airfoil_root, airfoil_tip, w_c_root, w_c_tip,"
                      "t_factor_root, t_factor_tip, w_semi_span, "
                      "sweep, twist",
            position=rotate(
                translate  # longitudinal and vertically translation w.r.t. fuselage
                (self.position,
                 "x", self.wing_position_fraction_long * self.fu_length,
                 "z", self.wing_position_fraction_vrt * - self.fu_radius),
                "x", radians(self.wing_dihedral)),
            # wing dihedral applied by rigid rotation
            mesh_deflection=0.0001,
            mov_start=self.wing_mov_start,
            #: spanwise position of inboard section, as % of lifting surface span
            mov_end=self.wing_mov_end,
            #: spanwise position of outboard section, as % of lifting surface span
```

```
            h_c_fraction=self.wing_h_c_fraction,
            # hinge position, as % of chord
            s_c_fraction1=self.wing_s_c_fraction1,
            # frontspar position, as % of chord
            s_c_fraction2=self.wing_s_c_fraction2
            # back spar position, as % of chord
            )

```

- **Aggregate node object:** This is a sub-property of the node object and can be accessed using the `<node>.obj.aggregate` syntax. The default value defined for the aggregate property is *None*. The only exception occurs when this node corresponds to one of the instances from a part sequence. In that case, the value of this property will be a back-pointer to the sequence object [40]. As such, this property is very useful to identify sequences in the raw precedents tree. Based on this identification, specialized behavior can be implemented to ensure the sequences are correctly represented in the model. This will be further elaborated later in this section.

- **Object module:** This is a sub-property of the node object and can be accessed using the `<node>.obj.__module__` syntax. This property is similar to the previously discussed slot module but, as the name suggests, now the value of this property is determined by the "origin" of the object rather than the slot. Or, put differently, whether the class used to instantiate the node object is an application class or a Parapy class. Due to this difference, the object module is simpler because it is not as dynamic. Contrary to specific slots, entire Parapy classes will not be overwritten by application-specific definitions. If an object is a Parapy object, it will stay a Parapy object.

  During the previous introduction of the ParaPy vs. application characterization, it was already mentioned that objects created with application classes can have ParaPy slots. Observe that Figure 6.2 presents an example of this occurring in practice: the slot module indicates this node is a Parapy slot, while the object module shows its an application object (Primiplane).

  For reverse engineering purposes, the most useful aspect from assessing the value of the object module is that it can be used to identify all nodes corresponding to Parapy objects. This means it can be used to remove all detailed, lower-level behavior associated with Parapy classes.

6.3. BLACK-BOX BEHAVIOR SPECIFICATION PER NODE TYPE

Three main types of nodes appear in the precedents graphs, namely input, attribute an part nodes. As discussed, these can be differentiated based on the node slot. An example of an input node was already provided in Figure 6.2. Examples of nodes from the other two main types are presented in the annotated precedents graph from Figure 6.3. This diagram is a close-up of the transformed graph previously depicted in Figure 5.2. Note that, aside from containing attribute and part nodes, this example also introduces two special (sub-)types: the initialization and geometry nodes. These node (sub-)types were identified/defined during the development of the REProcess tool. Additionally, it was found that the definition of part sequences, as opposed to singular parts, presents unique challenges for the graph transformation algorithm, which also requires specialized behavior. This section is dedicated to defining the required black-box behavior of the graph transformation algorithm for each of these types of nodes/Parapy features.

6.3.1. BLACK-BOX BEHAVIOR FOR INPUT SLOTS

According to PR-03, the declaration of input slots does not reflect a computation step that is significant enough to capture in the model as a standalone step. Therefore, only the control flow and data flow aspects of input slots should be captured in the process model.

In concrete terms, this means all input nodes present in the raw precedents graph should be removed. Before doing so, however, the data represented by the input node itself should be attached as metadata to all of its downstream node connectors (i.e. outgoing graph edges) in order to retain the data flow information related to the input node. This should only be done if this data is relevant. The relevance of the data is determined by reviewing both the slot module and the $is\_user\_defined$ property. The data is considered relevant when the slot is either an Application slot, or $is\_user\_defined$ is *True*. When this is the case, the data should be stored as metadata, and the input node should be contracted into its part node. When the data is not relevant,

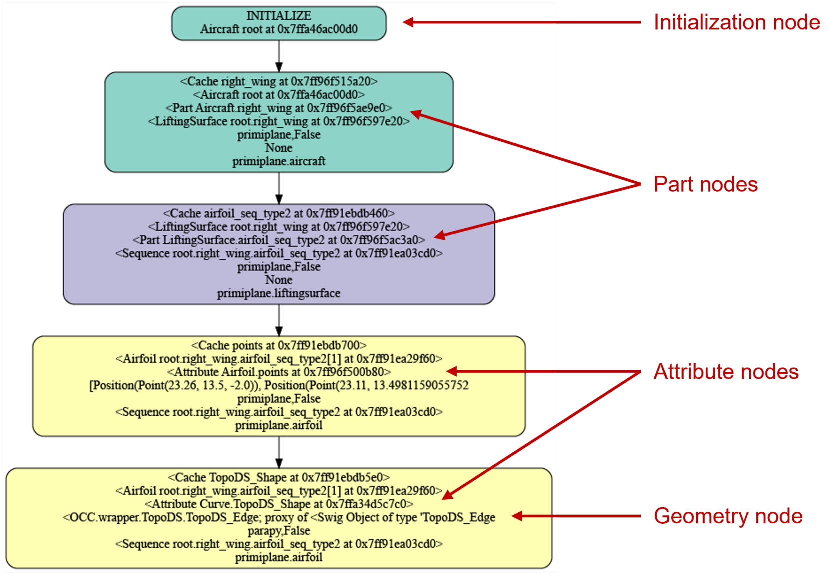


Figure 6.3: Annotated transformed precedents graph, presenting examples of attribute and part nodes, as well as so-called initialization and geometry nodes.

it can discarded. Note that, when discarding a node, care should be taken to not break dependency chains. Therefore, direct connections (i.e. graph edges) should be added between the nodes directly upstream and downstream of the input node before removing the node itself.

A final important note is that generally a corresponding part node is present in the raw precedents graph for all input nodes. However, in certain situations this may not be the case. In those case, the corresponding part node should be added into the graph in order to subsequently use that as the merge target for the input node contraction. This approach can be used solves all exceptions, except one special case. Due to complicated reasons related to the implementation of the Parapy KBE system, it is not possible to introducing a part node for the root object. Therefore, a work-around was invented in the form of the so-called initialization node.

THE INITIALIZATION NODE

The initialization node (also referred to as init node, or initial node) is exceptional in the regard that it is "manually" defined as a string object, rather than being a Parapy cache object like all of the other nodes. This string is composed of the word "INITIALIZE", followed by a string representation of the root object.

The initialization node serves a dual purpose in the transformed precedents graph. Firstly, it represents the starting point of the workflow, similar to the initial node in SysML. As defined in the specification of PR-09, such a single starting point is required. The second purpose is more practical and related to the implementation. Namely, the initial node provides a substitute target wherein all input nodes from the root object can be contracted. In this regard, the initial node fulfills a similar role as a part node. The need for a substitute originates from the *get_precdents_tree*() method, which does not include a part node for the root object. Introduction of the initialization node provides a solution to this issue.

6.3.2. BLACK-BOX BEHAVIOR FOR ATTRIBUTE SLOTS

The evaluation of an attribute slot represents a significant computation step, meaning an action node should be included in the process model for every application attribute slot that was evaluated during the analyzed workflow. Moreover, the source code defining these attribute should be attached as metadata to its representing action node to improve traceability. Any other slots used used for computation of an attribute should be captured in the model by node connectors flowing from those nodes to the action node that represents the attribute slot evaluation.

Generally speaking, nodes representing Parapy attribute slots should be removed from the graph using the same procedure outlined above for removing input slots to ensure dependency chains are preserved. One notable exception to this rule are geometry nodes, which are a special kind of attribute slots.

GEOMETRY NODES

Geometry nodes, or geom nodes for short, are a special type of attribute nodes. Even though they represent Parapy slots, it was determined that some should be kept in the precedents graph to represent the geometry of a part. Their occurrence in the precedents graph is an artifact of the Parapy implementation. Intuitively, one might assume that a part node itself describes the geometry. This would be analogous to having a physical part in the real world, where the object and its geometry are inextricably linked. However, this intuitive assumption does not apply to Parapy. There, the part node only represents the instantiation of the part object, which acts as a container for all aspects related to the part. The actual part geometry is defined as properties, i.e. lower-level slots, of this part object. The nodes appearing in the raw precedents graph corresponding to these geometry slots are referred to as geometry nodes. Here, note the explicit usage of plural forms, which relates to the fact that multiple geometry nodes appear for each part object. Some examples of geometry nodes are listed below.

- TopoDS_Shape (shown in Figure 6.3)
- builder
- Handle_Geom_Curve
- TopoDS_Vertex

The specific combination of geometry nodes that appear in the raw precedents graph for a given part object depends on the type of geometry of the part. For example, a Handle_Geom_Curve node appears when the part geometry is some sort of curve which would be the case when for instance the $FittedCurve()$ class is used. The TopoDS_Shape and builder nodes were found to be the most general and commonly occurring types of geometry nodes.

For the transformed precedents graph, it was determined that one of geometry nodes should be kept in the graph in order to represent a part's geometry generation step in the process model. The other geometry nodes should be removed, similar to regular Parapy attribute nodes. The specific node kept in the transformed graph to represent the geometry will be referred to as *the* geometry node (singular form). *-+ ““““““““““‘

6.3.3. BLACK-BOX BEHAVIOR FOR PARTS SLOTS

In the transformed graph, part nodes represent the creation of a part, which includes both the instantiation and initialization of the object. One of the key discoveries made during the development of the REProcess prototype was the identification of several different types of part slots. Each type was found to behave differently from a reverse engineering perspective. This posed a major complicating factor for the development of the REProcess tool and prototype. Each type requires different processing behavior to represent ensure they are correctly represented in the reverse engineered process model. The different types of parts that were identified are described below. For each type, a definition of the “correct” way to capture it in the process model is also provided.

- **Type 1: Parts created with Parapy classes.** Examples are the `Box()`, `FittedCurve()`, or `avl.Component()` classes.

  Given that all of these classes are standardized, it is assumed that users of the REProcess tool are not interested in the internal behavior of parts instantiated with a type 1 class. Therefore, the correct way to capture type 1 parts in the process model is defined as using a single node to represent the creation of the part. All lower-level nodes representing internal behavior of the type 1 class should be *contracted* into the part node.

- **Type 2: Application classes inheriting from type 1.** By inheriting from type 1 classes, some of the behavior from these classes is standard Parapy behavior. Moreover, in the specific case where a class inherits from a parapy.geom.occ class, which was found to occur quite often in practice, the part created with the type 2 class will have an associated geometry. Some examples of type 2 classes are `Airfoil(BSplineCurve)` from the Modular UAV app [12], as well as `Fuselage(LoftedSolid)` and `Airfoil(FittedCurve)` from the Primiplane app[4] [41].

  The correct representation of type 2 parts was defined as keeping all custom, application-specific internal behavior in the process model while removing all the standardized processing steps related

[4] Specifically the Primiplane app from tutorial 8 of the KBE course material.

to the type 1 ParaPy class that was inherited. There is one important exception to this general rule, namely the generation of the part geometry. Previously, for type 1 parts, the creation of a part and all intermediary steps up to and including the generation of its geometry were contracted into the part node. A beneficial side-effect of this approach is that it makes intuitive sense to a REProcess tool user that is less familiar with object-oriented programming and/or KBE systems. After all, creating a part's geometry "feels" synonymous to creating the part itself. However, for KBE systems this is not actually the case. In reality, the "creation" of a part corresponds only to the instantiation and initialization of a Python object using the specified class. The geometry of the created part object is contained in one of its slots (most commonly called "TopoDS_Shape"). Evaluation of this geometry slot is only performed when its slot value is demanded and after the dependency tracking system has resolved all preceding computations required to get the necessary inputs. Based on these considerations, the correct representation in the process model of type 2 parts with geometry slots was defined to be as follows: 1) a part node should be present to indicate the instantiation and initialization of the part; 2) each application specific, lower-level processing step should be represented with a separate node in the process model; and 3) a geometry node should be included to explicitly represent the part geometry generation step.

A final interesting note regarding the "generate geometry" nodes from type 2 classes is that they will generally be the final node of the lower-level behavior. This can be explained by considering that these nodes tend to embody the "purpose" of the class. The previously named examples explain this quite well. For instance, the main purpose of the `Fuselage(LoftedSolid)` class is to generate the fuselage geometry. This is already suggested by the class name (Fuselage), but the inheritance from the parapy.geom class makes it extra clear. Should the geometry generation not have been the main purpose of the Fuselage class, it is much more likely that the KBE developer would have opted to inherit from the more generalized approach to inherit from the ParaPy Base or GeomBase class instead (which would have made it a type 3 class).

- **Type 3: Application classes inheriting from (Geom)Base**. For any application class, the only alternative to inheriting from a type 1 class is to inherit from the Base or GeomBase class instead. Otherwise, the KBE application class would not be a functioning part of the KBE system. Some examples of type 3 classes are `XFoilAnalysis(Base)` and `Blade(GeomBase)` classes, both from the Modular UAV app [12].

  Based on practical experience gained during the implementation phase, two main categories of type 3 classes were identified. First, there are classes similar to type 2 classes, which still have the main purpose to generate part geometry. However, for these classes, the KBE developer has opted to create a type 1 part within the class definition of the type 3 part to contain the geometry. For example, the previously mentioned `Blade()` class has a part slot called `my_blade` where the representative geometry of `Blade()` class is generated. Alternatively, there is also a group of type 3 classes that are not focused on geometry generation. Instead, these often perform some sort of analysis function, such as the previously mentioned `XFoilAnalysis(Base)` class.

  For both categories of type 3 classes, the desired behavior of the graph transformation algorithm is similar as for type 2 classes: all relevant application nodes from these classes need to be preserved, while the inherited Parapy slots should be removed. Of course, a key difference is that type 3 classes will not have a geometry node. Another key aspect where differences between type 2 and 3 classes will emerge is the textual descriptions generated by the LLM. This will become apparent later, in subsection 7.4.2.

6.3.4. BLACK-BOX BEHAVIOR FOR PART SEQUENCES

Part sequences are relevant to discuss separately because they also introduce unique patterns in the precedents graph that have to be accounted for by the prototype implementation. Part sequences can be created for all three part types presented above. Thus, the REProcess implementation will have to account for sequences of type 1, 2 and 3 parts. Technically, parts sequences could contain a mixture of different part types. However, this is quite rare. Due to the associated complexity, this case is excluded from the scope of this thesis work. The identification and processing of nodes belonging to part sequences is greatly facilitated by the aggregate node object property.

In Parapy, defining a part sequence in the application source code can be achieved simply by specifying the `quantify` argument when creating a part. Although this change is small, and the source code definition of singular parts and part sequences looks very similar, it triggers large differences under the hood. Specifically, the sequence class is invoked as a wrapper and all of the individual part instances are stored in the slots of the sequence object. Importantly, a unique, stand-alone part object is created and evaluated for every sequence instance. Even when the geometry is identical for all parts and only the position of the part instances differs, the geometry is still re-computed for every single part. Correspondingly, a lot of duplicate sub-graph tend to be present when part sequences are generated which greatly clutters up the raw precedents tree.

The desired black-box behavior for sequences of type 1 parts is very straightforward. Since none of the lower-level behavior is relevant to include in the process model, the lower-level nodes of all sequence instances should simply be contracted into the part node. This tends to yield a massive reduction in overal node count.

By contrast, the processing of sequences of type 2 & 3 parts is significantly more complex. A perfect solution would be to compare the sub-graphs of all sequence instances to identify all unique segments and capture this information in the process model. However, after determining empirically that the lower-level behavior of the sequence instance tends to be identical, the simplifying assumption was made that the lower-level behavior of all sequence instances may be represented by one particular instances. To limit the impact of this simplifying assumption, a mitigation measure will be implemented. Namely, the user should be able to specify the specific sequence instance that will be used to represent the lower-level behavior of all sequence instances. Thereby, the user can still reverse engineer different sequence instance variations and compare these by performing multiple runs.

7

REPROCESS PROTOTYPE IMPLEMENTATION

The conceptual design for the core reverse engineering method of the REProcess tool was already presented in chapter 5, namely a sequence of dynamic precedents analysis, graph transformation, and LLM-based source code abstraction. Subsequently, each of these functions was further defined by the detailed prototype specifications from the previous chapter. Additionally, this the prototype specifications introduced some additional functionalities that are required to turn the core reverse engineering method into a functional standalone tool. This chapter presents how all of these functionalities were implemented in the REProcess prototype. First, an overview of the software architecture of the prototype implementation is presented in section 7.1. This overview is followed by a series of sections dedicated to describing the implementation of each of the main functionalities.

7.1. SOFTWARE ARCHITECTURE OF THE REPROCESS PROTOTYPE

An overview of the structural architecture of the prototype implementation is presented in Figure 7.1. It shows the allocation of main functionalities to the software components of the prototype implementation. An object-oriented programming style was used for the prototype implementation, in order to match with the implementation of the Parapy KBE system itself.

The leftmost swimlane in Figure 7.1 represents the user of the tool. The three actions in this column represent three commands that a user needs to enter to execute the complete reverse engineering process. Each of these commands relates to the initialization of one of the classes that were implemented for the REProcess prototype. The ReverseProcessModel class represented in the second swimlane is the main, overarching class of the implementation. From this class, both the DataflowAnalyzer and LLMPrompter classes are invoked. The DataflowAnalyzer class implements all of the functionalities related to the graphs, meaning both the precedents analysis method as well as the graph transformation algorithm. As the name suggests, the LLMPrompter class implements the LLM-based source code abstraction method. A more detailed description of each of the main functionalities is provided in the list below. This list also indicates the sections where the implementation of the specific functionalities will be further elaborated.

1. **Precedents analysis** (section 7.2)
   (a) Call `get_precedents_tree()` method
   (b) Convert tree data to graph
2. **Graph transformation algorithm** (section 7.3)
   (a) Merge/remove duplicate sequence instances
   (b) Merge/remove input nodes
   (c) Merge/remove remaining Parapy nodes
   (d) Introduce and merge into initialization node

   (e) Add additional metadata
   (f) Relabel to ID graph
3. **LLM-based source code abstraction method** (section 7.4)
   (a) LLM selection and implementation
   (b) Prompt engineering
4. **Output generation** (section 7.5)
   (a) Diagram visualization
   (b) Writing output file

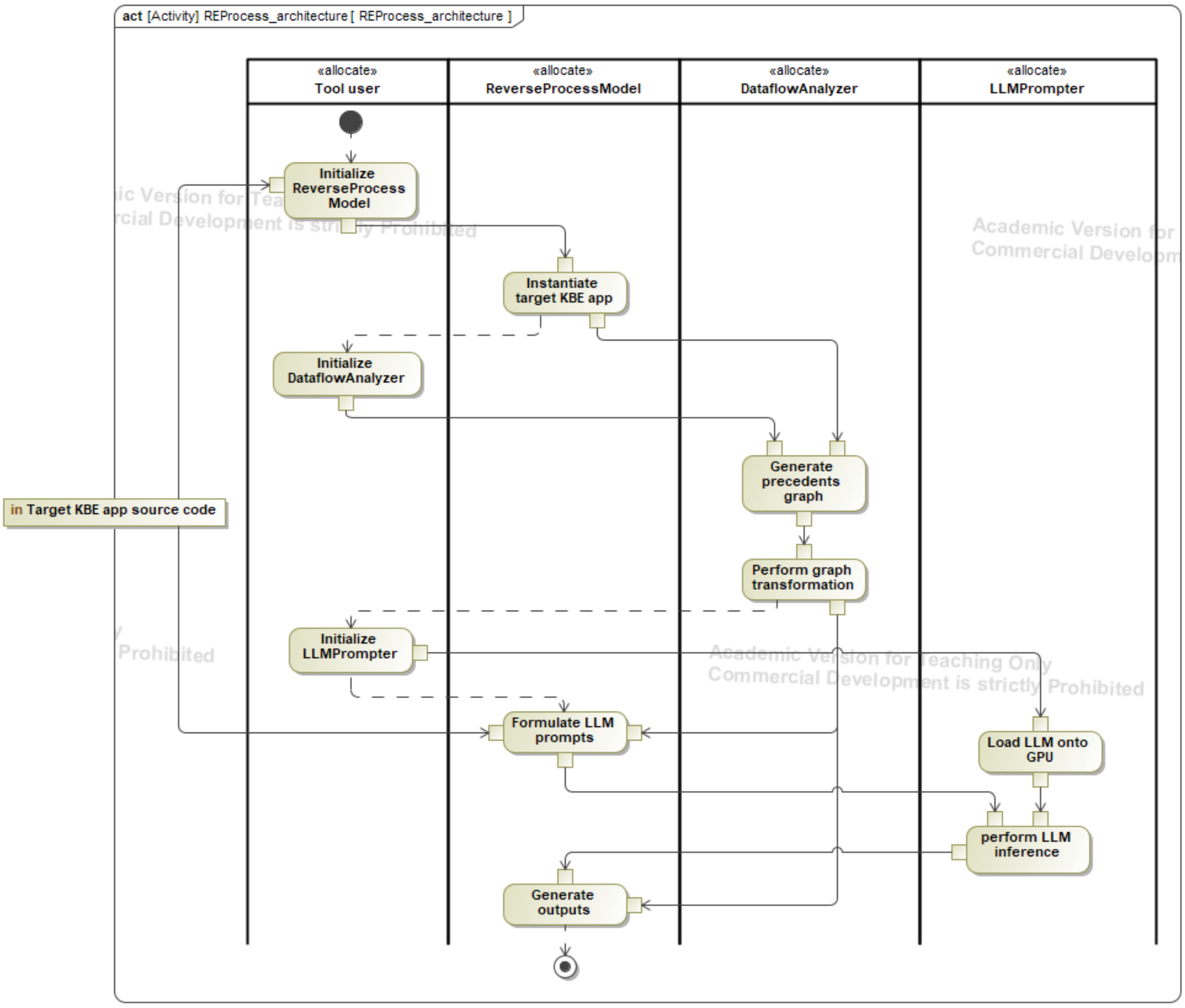


Figure 7.1: Software architecture implemented for the REProcess prototype.

7.2. PRECEDENTS ANALYSIS IMPLEMENTATION

The precedents analysis is the first main step of the core reverse engineering method. This step is responsible for creating the raw precedents graph. This section details the implementation of the precedents analysis method, highlighting the key components and logic used to generate the graph.

7.2.1. CALLING THE GET_PRECEDENTS_TREE METHOD

The precedents analysis is implemented by the `generate_precedents_graph` method from the Dataflow-Analyzer class. It was designed to handle syntax differences when utilizing the `get_precedents_tree()` method across different Parapy slot types, including Parts, Sequences, Attributes, and Inputs.

The `generate_precedents_graph` method first checks whether the target slot's qualified name indicates a nested location within a part hierarchy. This is done by searching for a dot (.) in the name. If a nested location is detected, the target part path is extracted, and the `target_attr_name` is set to the last component of the qualified name. After determining the target slot, the method differentiates between the target slots categorized as parts (including Sequences) and attributes/inputs. This differentiation is important as it dictates the approach used for subsequent steps in generating the precedents graph.

PROCESSING OF PART SLOTS, INCLUDING SEQUENCES

For target slots identified as part objects, the method checks whether the object is a sequence or a singular part. If a sequence object is identified, the user is prompted to specify the sequence instance to analyze. When the target target slot is a part, it checks whether this part has a `TopoDS_Shape` slot. If this is the case, it is assumed the user meant to target this slot for the reverse engineering analysis. When no `TopoDS_Shape` slot is found, the user is prompted to specify a more specific target slot from within the class of the previously specified part object. This is necessary because the `get_precedents_tree()` method does not work properly when a part is provided as the target.

PROCESSING OF ATTRIBUTES AND INPUTS

For attributes and inputs, the processing is a lot simpler. These can be used by the `get_precedents_tree()` method as-is. However, it should be noted that the `get_precedents_tree()` method does not work correctly when *targeting* input slots. The precedents from within the application class of the input slot are correctly identified and returned, but this is where the tracking stops. Any preceding slot evaluations executed within objects higher up in the product hierarchy to provide a value for the targeted input slot are not returned. When evaluating `root.right_wing.get_precedents_tree(root_profile)`, for example, any slot evaluations that precede `root_profile` but occurred before the `right_wing` part was instantiated are not returned by `get_precedents_tree()` even though they should have been added to the precedents tree. To account for this, the following warning is provided to the user when the target slot is identified as an input slot:

```
input("\nWARNING: Parapy's get_precedents_tree() method, which forms the basis of "
    "the current reverse engineering method, does not search for data "
    "dependencies outside of the class of the Input slot that was provided as "
    "the analysis target. Therefore, the activity diagram will be incomplete. "
    "Press enter to continue anyway.")
```

7.2.2. CONVERTING THE PRECEDENTS TREE TO GRAPH FORMAT

The transformation of the precedents tree into a graph structure is a crucial step in the precedents analysis process. This functionality is implemented in the `transform_nested_precedents_dict_to_obj_graph` method of the DataflowAnalyzer class. The transformation process begins by initializing a directed graph, `obj_graph`, using the NetworkX library's `DiGraph()` function[46]. Subsequently, a recursive helper function, `add_nodes_edges`, traverses the `raw_precedents_tree` (nested dictionary) and populates the created graph object with nodes and edges.

7.3. GRAPH TRANSFORMATION ALGORITHM

The desired behavior of the graph transformation algorithm was already extensively discussed in section 6.3. For the implementation, the main remaining challenge was to figure out the order of the processing steps. Based on extensive iterations, the order of steps presented in this section was determined.

7.3.1. MERGE/REMOVE SEQUENCE NODES

The first step in the graph transformation algorithm involves merging or removing duplicate sequence instances in the precedents graph. This process ensures a cleaner and more concise representation of the KBE application's structure by eliminating redundant sequence nodes. This functionality is implemented in the `merge_sequence_into_part_node` method from the DataflowAnalyzer class. This method follows the following three steps:

1. **Criteria for removing sequence instances** The `merge_sequence_into_part_node` method begins by defining criteria for removing sequence instances. This involves checking if a node belongs to a sequence object. If the node is a sequence instance, it is marked for removal unless it is belongs to the so-called *representative sequence instance*. This is the case when it is either the primary sequence instance (“child_0”), or if it belongs to the specific instance specified in the analysis target.

2. **Merging type 1 sequence nodes** Sequence nodes corresponding to type 1 parts are merged into their corresponding part nodes. The merge target is identified based on the aggregate object of each node (see Figure 6.2). If this identification method finds multiple aggregate objects, a user warning is issued to investigate the inconsistency.

3. **Removing type 2 and type 3 sequence nodes** Sequence nodes corresponding to type 2 and 3 parts are removed from the graph. The removal is based on the criteria defined earlier, ensuring that only redundant sequence instances are eliminated. Note that in this particular instance, a simple removal approach is used. In the previous chapter, the concept of dependency chain-preserving node removal was introduced but that does not have to be applied here because the dependency chain from the representative sequence instance remains untouched.

An example of the input (raw precedents graph) and output from the merge sequence step is presented in Figure 7.2. The block of blue nodes highlighted in the top figure shows the typical graph structure of node sequences. A close-up Figure 7.3 of this block is provided in Figure 7.3. This close-up also shows the identified merge target of the blue nodes. The location of this merge target in the resulting output graph is highlighted by the red arrow from Figure 7.2.

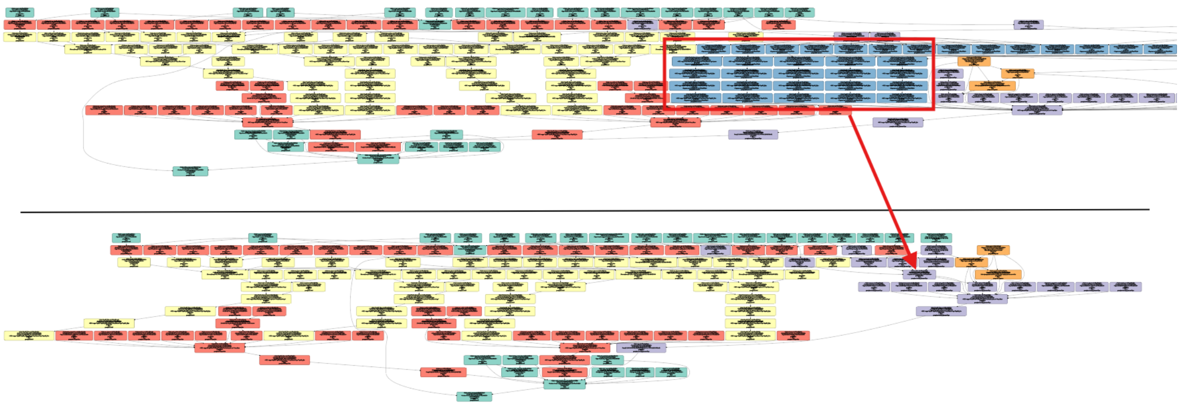


Figure 7.2: Illustration of the sequence merging process. On top: the raw precedents graph. On bottom: the output graph after the merging step on the bottom. Annotations highlight merging of sequence nodes into their aggregate object. Close-up of the sequence nodes provided in Figure 7.3.

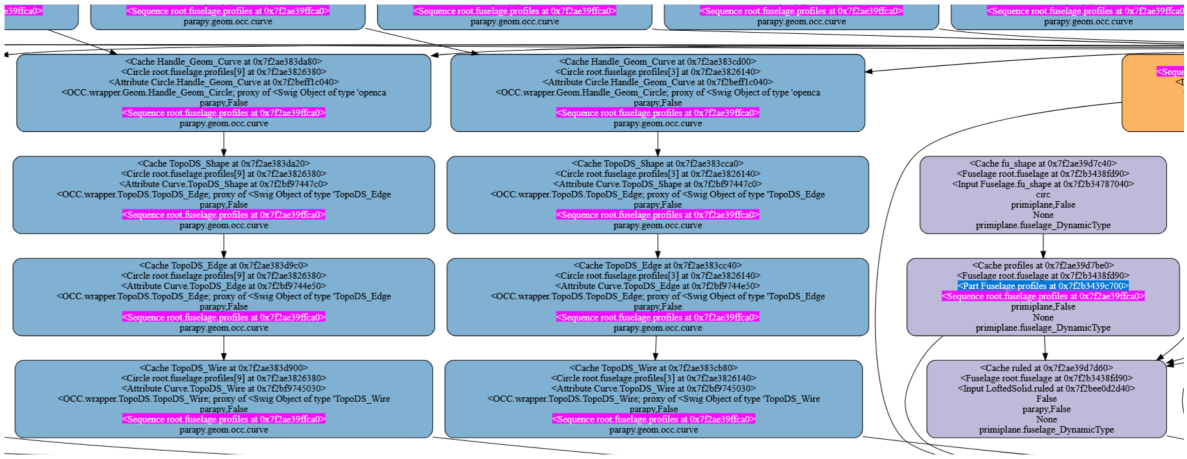


Figure 7.3: Close-up of the sequence nodes from Figure 7.2. The aggregate slots from the sequence objects are highlighted. On the right-hand side, the node merge target is identified (`<Cache profiles at 0x7f2ae39d7be0>`). The criteria used to highlight the merge target: a *part* node (third row) with `node.value` (fourth row) equal to aggregate object from the sequence nodes.

7.3.2. MERGE/REMOVE INPUT NODES

The second step in the graph transformation algorithm involves the merger and removal of input nodes. This process is implemented in the `merge_inputs_into_part_node` method of the DataflowAnalyzer class. The goal of this step is to ensure that input nodes are either merged into their corresponding part nodes, or removed from the graph[1]. Aside from removing redundant implementation details, a second key benefit from merging input nodes into their part node is that the flow of information starts to better resemble a activity diagram. This is illustrated in the example presented in Figure 7.4. In the top figure, a block of input slots is highlighted that is merged into the corresponding part node, indicated by the blue arrow to the node in the same graph. The result of this merging step is that this part node moves to a much more "upstream" position in the graph, as shown by the arrow. This transformation is desired because the part node can then later be used to represent the initialization of the part.

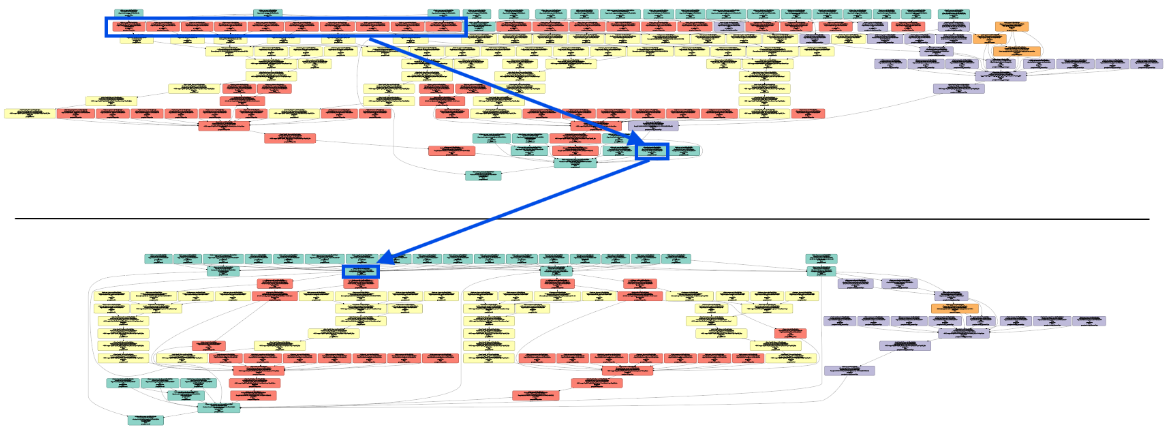


Figure 7.4: Illustration of the input merging process. On top: precedents graph after the previous sequence merging step. On bottom: the output graph after the input step. Annotations show how the merging of input slots to their corresponding part node, identified in the same graph, leads to rearranged graph shown on the bottom. The rearranged position of the part node used as merge target is indicated by the blue arrow in the bottom graph.

CRITERIA TO MERGE INPUT NODES

The decision to merge or remove input nodes depends on whether the input nodes represent relevant information that should be captured in the process model. The criteria below are used to identify nodes **relevant** for

[1]The input nodes corresponding to the root object form an exception. These will be processed later, by the introduce_initialization_nodes method

merging. If one of criteria these are not met, the node is non-relevant for merging and are later removed by the algorithm.

- The node must be of type `cache.Cache`, ensuring it is a valid cache object within the graph.
- The node slot must contain the string "Input". This indicates that the node is an input node.
- A node is completely excluded from merging/removal if it is designated as the reverse engineering target node (i.e. input parameter of `get_precedents_analysis` method). This ensures that the primary node of interest remains intact throughout the analysis.
- A node is relevant for merging if its slot does not originate from the `parapy` or `kbeutils` modules. A complementary merge criteria is if it is explicitly defined by the user, which is identified through the `is_user_defined` method from the ParaPy KBE system.

PROCESSING OF INPUT NODES

Input nodes identified as relevant are subsequently processed to store them as metadata and transform the graph structure. For each input node identified as relevant, the incoming edges are analyzed to determine the source nodes contributing to the input. These nodes are captured in a dictionary format, which is then assigned to the `received_inputs_dict` attribute of the corresponding part node. Additionally, input nodes identified as relevant are appended to the `data_flow` attribute of its outgoing edges. This step ensures that the relevant data represented by input nodes is captured in the process model, despite being merged into another node.

Following metadata storage, the algorithm proceeds to merging and/or removing the input nodes. This process is implemented by the steps outlined below. Note that when a node is identified as non-relevant, the first four steps - which correspond to node merging - are skipped.

1. **Identifying the part node:** Each input node is examined to locate its corresponding part node, the intended merge target. Identification of the merge target is performed by looking for the presence of a “Part” identifier in the node slot, combined with the node value being equal to the node object of the input node.
2. **Handling aggregate objects:** If no merge target is found in the previous step, the algorithm performs another search to check if a merge target in the form of a part sequence node can be found instead. This involves checking if the node’s value aligns with the aggregate attribute of any part nodes.
3. **Introducing part nodes:** When a valid merge target was still not found, the algorithm attempts to actively introduce a part node into the graph.
4. **Node merging:** Once a valid part node is identified, the input node is merged into it using the `nx.contracted_nodes` function provided by the NetworkX library. This operation combines the attributes and edges of the input node with those of the part node, effectively integrating the input node’s data into the part node. Associated metadata is also transferred to the part node, consolidating all important aspects into the part node.
5. **Node removal:** In cases where a node was identified as non-relevant, or when a suitable part node could not be identified for merging, the algorithm performs node removal. This is performed by the `remove_node_with_path_preserve` function, also defined in the DataflowAnalyzer class. This function ensures that the edges connected to the input node are rerouted to maintain connectivity, thereby avoiding disruptions in the graph’s dependency chains. This is achieved by adding direct edges between all nodes directly upstream and downstream of the input node, effectively creating “bypasses”. When the input node is subsequently removed, these bypasses keep the previous dependency chains intact.
6. **Process tracking:** Throughout the process, counters are maintained to track the number of input nodes successfully merged and removed. These counters provide feedback on the algorithm’s performance and are logged for verification purposes.

7.3.3. MERGE/REMOVE REMAINING PARAPY NODES

The third step in the graph transformation algorithm focuses on merging and removing remaining Parapy nodes that were not addressed in the previous steps. It is implemented by the `remove_all_parapy_nodes` method from the DataflowAnalyzer class. This process further refines the graph representation by eliminating nodes that are either redundant or considered implementation details. An example of this step is presented in close-ups corresponding to the input (Figure 7.5) and output (Figure 7.6 of this step. An overview of the complete output graph of is presented in Figure 7.7. Note that, compared to the bottom graph of Figure 7.4, several other sequences of nodes have also been merged. For example, the tall **vertical stacks of yellow nodes** from Figure 7.4 are no longer present in Figure 7.7. The implemented process performing this step is described below.

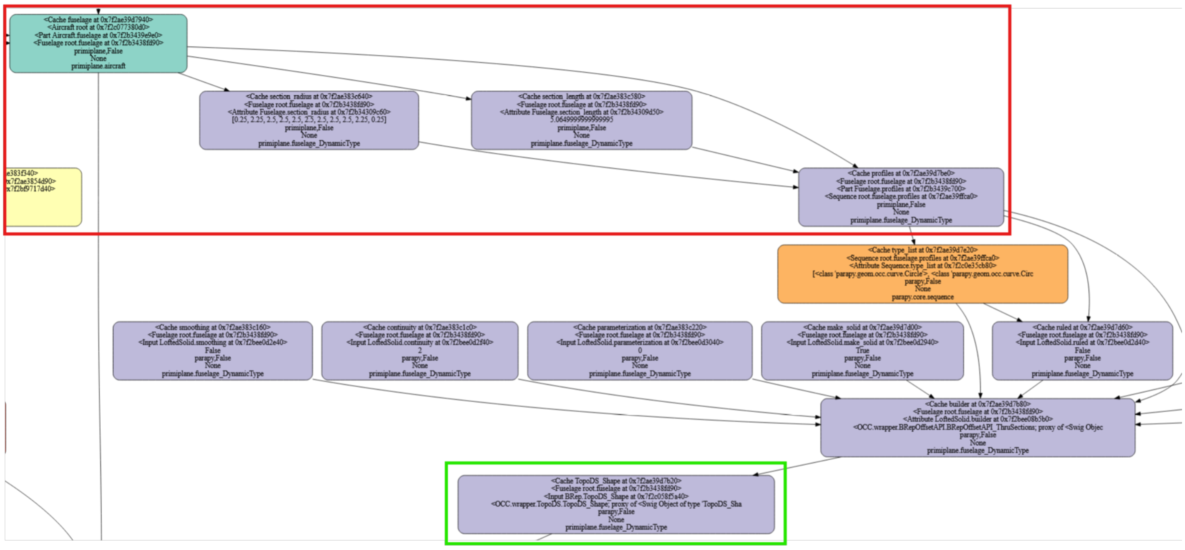


Figure 7.5: Close-up of the right-hand side of Figure 7.4, which is the graph used as input for this step. This is before the merging/removing of ParaPy nodes is performed. The nodes highlighted in this figure with red and green boxes are also present in Figure 7.6.

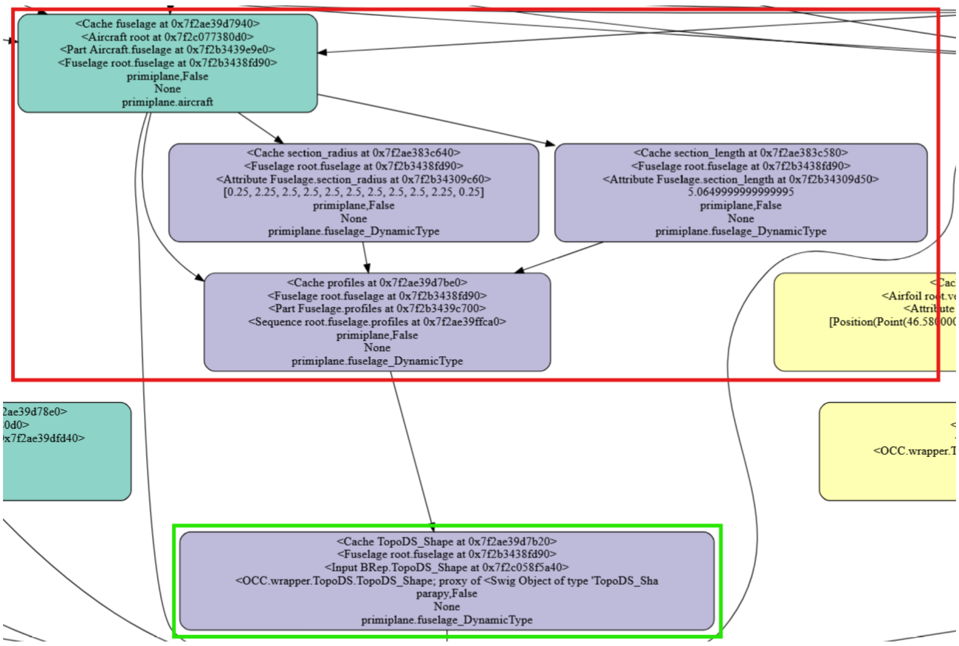


Figure 7.6: Close-up of the graph obtained after the merging/removing of ParaPy nodes. Note how the highlighted nodes (red and green boxes) remained in the graph after this processing step, but the nodes **in between** are removed.

PROCESSING OF PARAPY NODES

The process to merge and remove remaining Parapy nodes involves the following steps:

1. **Identification of type 2 node objects:** As discussed in section 6.3, the geometry nodes of type 2 objects present a key exception to the general rule that all Parapy nodes should be removed. Therefore, the first step is to identify all type 2 part objects present in the graph and select one of the geometry nodes from each of these parts that will represent the geometry generation step in the final process model. The combination of type 2 part objects and their geometry nodes is stored in the type_2_object_to_geom_node_mapping dictionary, which subsequently saved as metadata (in the form of a graph attribute).

2. **Removing/merging the remaining Parapy nodes:** The implementation of this step involves iterating through the graph and using the slot module and is_user_defined properties to identify and remove all remaining Parapy nodes. Naturally, the other exception criteria are also checked before nodes are processed. This includes the previously identified type 2 geometry, as well as checking if the node is not the analysis target node.

   For specific cases, Parapy nodes are merged, namely when they are geometry nodes. In case of the node corresponds to a type 1 part (sequence), its part node is used as the merge target. In case of type 2 geometry nodes, the selected geometry node is used as the merge target for the other geometry nodes. Both these merging operations are implemented to ensure the control and data flow are correctly represented in the final process model.

   In all other cases, Parapy nodes are simply removed using the same `remove_node_with_path_preserve` method that was used to remove nodes in the previous algorithm step.

Similar to the previous algorithm step, the number of nodes merged and removed is again recorded during this step for verification purposes. The final output of this step is presented in Figure 7.7.

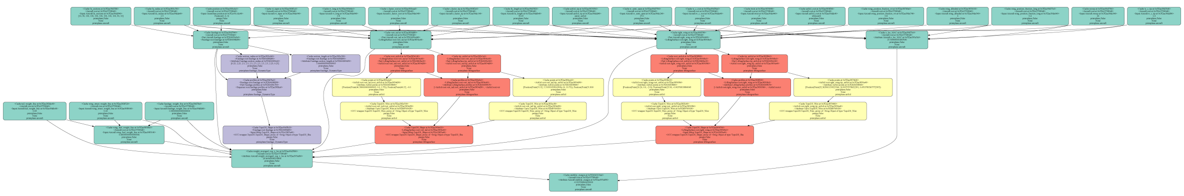


Figure 7.7: Overview of the output obtained from the merge/remove ParaPy nodes step. Figure 7.6 is a close-up of the four purple nodes on the left-hand side of this figure.

7.3.4. INTRODUCE AND MERGE INTO INITIALIZATION NODE

The fourth and final main processing step in the graph transformation algorithm involves introducing the initialization nodes and merging all input slots from the root object into it. The reasons to implement this behavior were discussed previously, in section 6.3. This step is implemented by the `introduce_initialization_nodes` method from the DataflowAnalyzer class. The example output is presented in Figure 7.8. Note that only the top row has changed, compared to Figure 7.7, and now shows the initialization node.

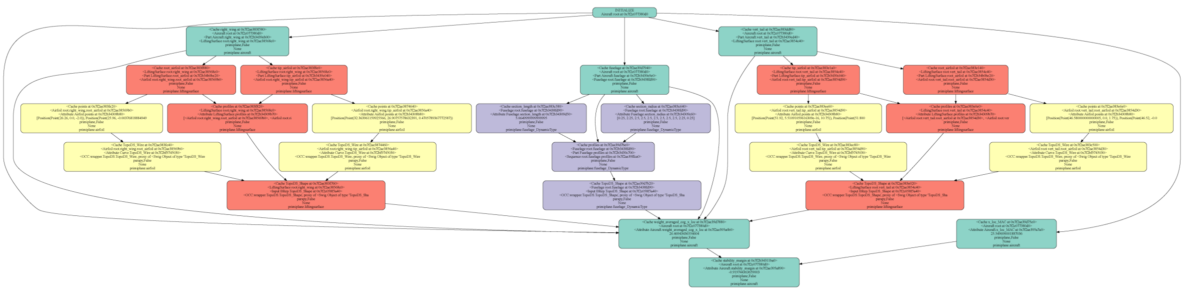


Figure 7.8: Example output from introduce initialization node step. Note how all nodes on the top row of Figure 7.7 are merged into the initialization node.

PROCESSING OF INIT NODE

The process to introduce and merge into initialization nodes is carried out as follows.

1. **Cluster nodes by object:** The first task is to cluster nodes based on their associated objects. This step groups nodes that belong to the same object, specifically focusing on root objects in the graph. Each root object cluster will later be represented by an initialization node. Note that this step can identify multiple root-level objects. This is necessary because the usage of functions from the kbeutils library was found to sometimes lead to the initialization of a separate root object, instead of defining it as lower-level behavior of an existing object in the part object hierarchy. Note that a graph attribute is used to store the "original" root object as a metadata property. This ensures an explicitly-defined, singular starting point for the reverse engineered workflow is captured in the process model (which is required to comply with PR-09).

2. **Create initialization nodes:** Subsequently, an initialization node is introduced for each root object in the graph. This node acts as a placeholder that will replace all input nodes associated with the root object. The initialization node is assigned attributes to track the object it represents and is marked with a specific node type for identification. This node type is stored as metadata each init node.

3. **Merge input nodes:** All input nodes related to the root object are merged into their respective initialization node, using a similar approach as when merging inputs into their part nodes.

4. **Handle disconnected initialization nodes *(optional)*:** If an initialization node is found to be disconnected from the rest of the graph after the previous step, an additional edge is introduced to connect it to the most likely node that it should have been connected to. This node is determined by making some assumptions. Correspondingly, a user warning is returned to make the user aware of this exceptional situation. Note that this only occurs in exceptional, highly specific cases.

5. **Merge stand-in target node (optional):** In the special case where the user provided a part node as the analysis target and the algorithm corrected this to the geometry slot of the part, a TopoDS_Shape node is still present in the graph after all previous processing steps. In this step, this TopoDS_Shape node is merged into its part node in order to comply to the desired behavior from the user.

Throughout this step, the number of nodes merged is recorded to monitor the effectiveness of the simplification process and ensure that the resulting graph maintains the necessary data flow information.

7.3.5. ADDING ADDITIONAL METADATA

This step is listed under the graph transformation algorithm because it is also implemented in the Dataflow-Analyzer class. However, it does not involve any graph transformation steps. As the name suggests, this step only involves the storage of a lot of additional metadata attributes. The purpose of this additional metadata is to enrich activity graph and prepare it for the output generation. As will be discussed later, all data in the generated output files is "flattened" to string representations. This means that all of the properties from the cache objects defining the nodes can no longer be accessed. Therefore, all properties that could be relevant for the process model are stored as additional graph, node and edge attributes. The method implementing this functionality is the *add_metadata_to_graph_and_nodes*. An illustrative example of attached metadata is provided in the next subsection. An overview of the captured metadata parameters is provided below:

GRAPH ATTRIBUTES

- `part_node_list`: A list of all application part nodes.
- `kbe_instance`: Stores a string representation of the KBE instance associated with the graph.
- `kbe_target_obj`: Stores a string representation of the object used to call the get_precedents_method.
- `target_slot_qualified_name`: The fully qualified name of the target slot.
- `target_attr_name`: Stores the name of the attribute used as the target of the get_precedents_method.
- `seq_instance_target_idx`: Stores the sequence instance index when a particular sequence instance was specified as the revers engineering target.
- `sequence_part_to_representative_obj_mapping`: A dictionary that maps sequences to their representative objects.

- `ppy_obj_set`: The set of all Parapy objects present in the transformed precedents graph.
- `part_type_2_3_dict`: A dictionary mapping the part nodes of type 2 and type 3 parts to their part objects.
- `graph_type`: Used to differentiate this graph from the so-called ID graph which will be introduced in the subsequent subsection.

NODE ATTRIBUTES

- `source_code`: The node slot source code is added as a string value to each node
- `node_type`: The type of this node, based on the specification presented in section 6.3. Appended by "_sequence" for sequence instance nodes
- `ppy_cache`: String representation of the Parapy cache object representing the node in the precedents graph
- `tree_level`: Level of this node's object in the part hierarchy (0 indicates the root level)
- `parent_objs`: Ordered sequence of parent objects, starting from the node object up until the root object.
- `node_obj_class`: Class of the node object.

EDGE ATTRIBUTES

- `contraction`: Nested dictionary created by the `nx.contracted_nodes()` function. Used to capture the edge attributes corresponding to each original edge when edges overlap due to a node contraction operation.
- `data_flow`: Dictionary of relevant Parapy slots flowing through a graph edge.

7.3.6. RELABEL TO ID GRAPH

The final step of the graph transformation algorithm involves relabeling all graph nodes to a qualified string representation, resulting in the so-called ID graph. The need to relabel the activity graph nodes is related to PR-11 (deterministic process). As-is, the activity graph does not comply to this requirement due to the the memory address[2] included in the cache objects representing the part nodes. This memory address will differ for every re-run of the analysis. Another undesirable effect of this node definition style is that it is not user friendly. As discussed in section 6.2, several additional properties need to be presented in order to clarify the exact part of the KBE app that a node represents. Therefore, a relabeling method (named `relabel_obj_to_id_graph`) was implemented in the DataflowAnalyzer class.

Figure 7.9 illustrates the output from this step. Note that, aside from the node ID's themselves, two rows with additional information are printed in this illustrated example. A tooltip showing the metadata attached to the final node is also presented. This tooltip is shown when hovering the mouse button on a node (only works when opening the SVG file of a graph). The mouse can also be hovered on the node edge to display the data flow (ParaPy slot names).

PROCESSING STEPS

The `relabel_obj_to_id_graph` method implements a comprehensive relabeling process. In addition to relabeling the node representations in the graph itself, it also has to process all of the metadata that was attached to the graph, as well as the individual nodes and edges. Below is a detailed breakdown of each operation involved in the relabeling process:

1. **Initialization and Input Definition:** The `relabel_obj_to_id_graph` method begins by defining lists of node, edge, and graph attributes that require relabeling. These lists include attributes such as `received_inputs_dict`, `data_flow`, and `init_node`. Additionally, mappings are initialized to maintain a relationship between the original cache objects and the newly assigned identifiers.
2. **Build relabel mappings:** For every node cache object, a unique string identifier is generated and and stored in a mapping dictionary. The string identifiers are generated by the `relabel_cache_to_id`

[2](i.e. the underlined part in: `<Cache right_wing at 0x7f460853e620>`

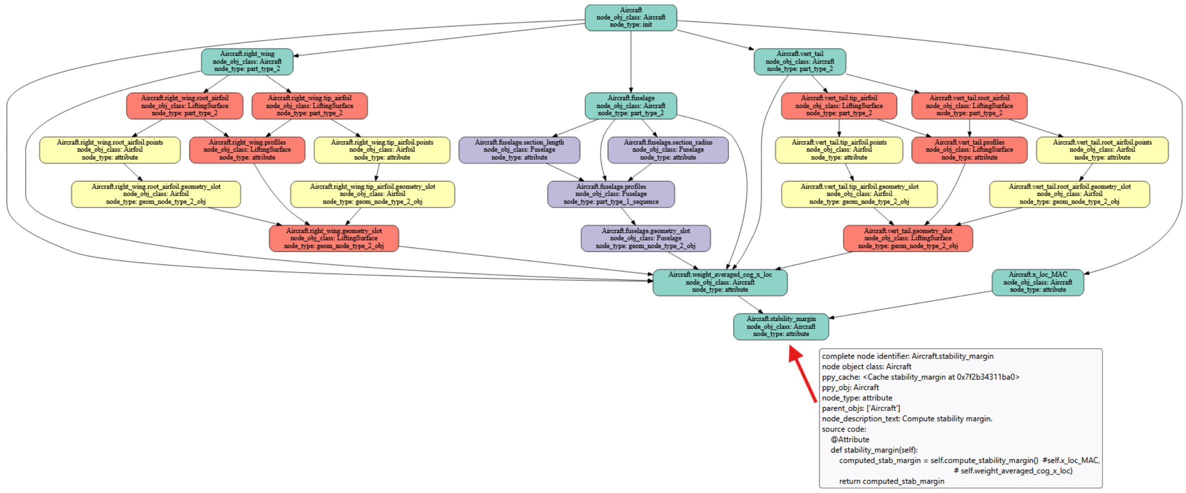


Figure 7.9: Example ID graph generated as output from this processing step. The tooltip showing the metadata attached to the final node is also presented.

method from the DataflowAnalyzer class. Essentially, this method defines each node as a fully qualified name that represents its location in the part hierarchy. The class name of the root object is used (instead of "root") as the first component of the qualified name. This ensures multiple root objects can be differentiated when the edge-case is encountered where additional root objects were initialized (by kbeutils) and captured in the activity graph. Note that initialization nodes are also relabeled by the `relabel_cache_to_id` method. Since init nodes represent the creation of the root object itself, their qualified name only consists of this first component. For other nodes, the qualified name consists of additional components separated by "." dots. The final ID component is the name of the node slot (except for geometry nodes, which use "geometry_slot" instead of TopoDS_Shape, Handle_Geom_Curve, etc.). Optionally, the qualified name includes additional components in between the root class and slot name. These are names of node *objects* (i.e. parts), which indicate how lower-level nodes are nested in the overall product hierarchy. Some examples of these qualified names are provided below:

- `Aircraft` *(initialization node)*
- `Aircraft.right_wing` *(part node)*
- `Aircraft.right_wing.root_airfoil` *(part node)*
- `Aircraft.right_wing.root_airfoil.points` *(attribute node)*
- `Aircraft.right_wing.root_airfoil.geometry_slot` *(geometry node)*

The relabeling mappings are stored as graph metadata for both forward and reverse conversions, facilitating subsequent transformations and maintaining clear traceability between original and relabeled nodes. Furthermore, the `graph_type` attribute is set to "id".

3. **Node Relabeling:** The NetworkX `relabel_nodes()` function is used to relabel the graph, based on the mapping specified in the relabeling dictionary.

4. **Metadata processing:** After the graph nodes themselves are converted, iterative functions execute the relabeling of the metadata attributes. Only explicitly listed attributes are relabeled, because the purpose of some attributes is to refer back to the original cache objects.

7.4. IMPLEMENTATION OF THE LLM-BASED ABSTRACTION METHOD

This section will cover the implementation of the LLM-based abstraction method. The hardware aspect is discussed first, as this drives design decisions regarding the software implementation which is presented thereafter. The latter is broken down into two main aspects: the prompt engineering, and the LLM inference. Note that, although both are considered part of the LLM-based abstraction method, only the latter is implemented in the LLMPrompter class. As depicted in Figure 7.1, the prompt engineering is (mostly) implemented within the *perform_LLM_abstraction* method of the overarching ReverseProcessModel class.

A final important note before diving into the LLM implementation, is that not every textual node description is generated by the LLM(!), because not all nodes from the transformed precedents graph have an associated definition in the application source code. Specifically, this applies to types of special nodes, i.e. the initialization and geometry nodes. For these two node type, a template based approach is used. This will be further discussed in section 7.5.

7.4.1. ACQUIRING COMPUTATIONAL RESOURCES

Implementation work related to the LLM-based abstraction method started with the question of how to get access to the required computational resources. This issue was the first to address because it drives subsequent design decisions. For instance, the available amount of GPU vRAM determines the range of Large Language Model sizes that can be chosen from.

For this thesis project, the computational resource requirement was characterized as *"high availability, medium power"*. The emphasis on high availability corresponds to the iterative development approach that was applied in order to deal with the exploratory nature of this thesis project. The medium power aspect was determined based on a previously conducted feasibility study, which indicated that models between 10-20B nodes should be more than capable of delivering the desired output. Note that these models were used as-is during this project, so the workstation was only used LLM inference meaning no additional power was required for training or finetuning purposes[3].

Three main options were identified to access the required computational resources:

- **DelftBlue[4]:** This is the massive CPU and GPU cluster from TU Delft. It is available for all students and researchers from the TU Delft. Unfortunately, the computation jobs submitted by students are assigned lower priority in the job que. As a result, it even occurred during experimentation with DelftBlue that jobs went backwards in the que due to the large amount of TU Delft researchers being given priority. This option was disregarded due to the (extremely) low availability.
- **Hugging Face inference endpoints[5]:** These provide a production solution for deploying any Transformers, Sentence-Transformers, and Diffuser models from the Hugging Face Hub. The inference endpoints run on dedicated, auto-scaling infrastructure and are kept separate from the Hugging Face Hub source repositories, ensuring high security and reliability. Therefore, they could be considered despite PR-12 (no external computing). However, it was determined the monthly billing structure with the flexible pricing scheme did not fit within the financial constructs of TU Delft. Therefore, this option was disregarded.
- **High-performance workstation:** The third option consisted of acquiring a dedicated, high-performance workstation for this and future projects. This option was chosen after the other two were eliminated. For the GPU, an NVIDIA RTX 6000 Ada was selected (48 GB of vRAM) which more than fulfilled the requirements. It even provided additional opportunities in terms of LLM selection. Remote desktop access was setup so the machine could be operated remotely.

7.4.2. PROMPT ENGINEERING

In the REProcess tool, prompt engineering plays a crucial role in generating effective and meaningful abstractions of the analyzed KBE application. Moreover, the design space of this aspect is massive due to the virtually infinite possibilities of formulating the natural language instructions. On one hand, this poses a great challenge for the implementation, but it also provides many highly interesting opportunities. For the REProcess tool, this is further amplified by the fact that the prompt engineering acts as an interface between

[3] Although this would be an interesting research avenue to pursue in future research.
[4] https://doc.dhpc.tudelft.nl/delftblue/
[5] https://huggingface.co/inference-endpoints/dedicated

multiple analysis and abstraction methods. An interesting way to look at the prompt engineering is to regard prompts as containers wherein relevant information from multiple sources/analysis methods is combined. In case of the REProcess prototype, the nodes in the transformed graph dictate the prompts that are formulated. The corresponding source code is obtained by the `inspect` Python module (which could even be considered a static analysis component). And, as will be discussed later, different prompt variants are generated based on the node's metadata (this is done by the keyworded response start feature). This is visualized in Figure 7.10

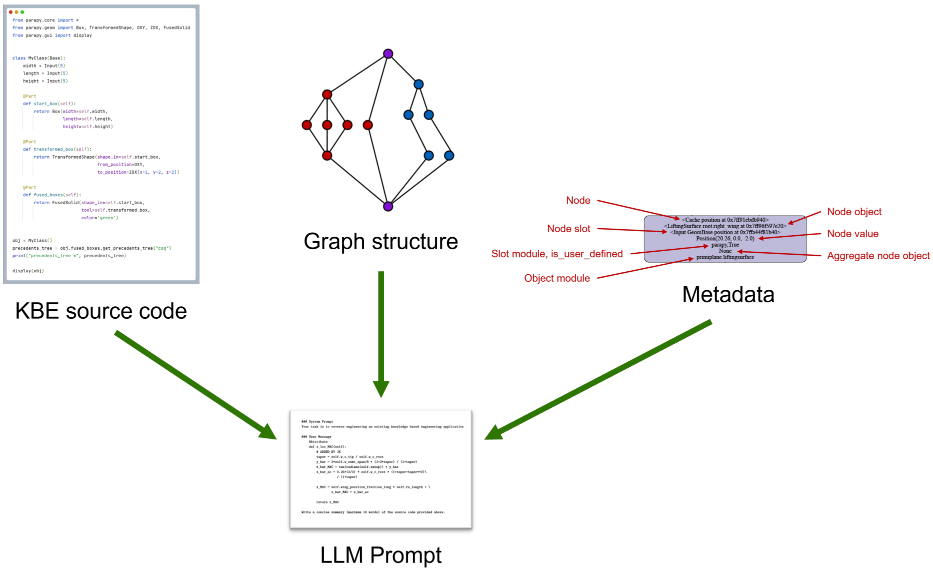


Figure 7.10: Visualization of the information collection characteristic of prompt formulation.

Of course, there are many more possible implementation of this information combination processes which indicates that the interface role of the prompt engineering system from the REProcess tool/prototype increases the number of design options even further. Unfortunately, pursuing all of these opportunities was not possible within the scope of this thesis project. This subsection will only outline the specific prompt engineering strategies employed in the tool, focusing on user prompts, system prompts, and the keyworded response start feature. Additionally identified ideas and opportunities will be presented later, in the discussion and final recommendations.

USER PROMPT FORMULATION

The user prompt is the specific part of the instruction to the LLM. When using ChatGPT as an analogy, the entire chat message is the user prompt. In case of the REProcess prototype, the user prompt is built up from parts: the source code snippet (`code_snip`) and the user instruction (`user_instruction`). A special user prompt generation feature implemented for the REProcess prototype is the so-called prompting context mode. Essentially, this context mode allows a user to switch between two different modes of generating the user prompt. In the "method" mode, the code snippet injected into each user prompt is that of the specific node *slot* (i.e. the class method definition). Alternatively, in the "class" mode, the source code of the entire class corresponding to the node *object* is used. Of course, while the provided source code changes, the goal for the LLM in both cases is still to generate a description for one specific node. Therefore, a slightly different user instruction (and system prompt) is used for the second case which instructs the LLM to focus on generating a description of a specific node, instead of the entire class. Examples of each prompt context mode are provided below:

Method prompt context example

```
### User Message

    @Attribute  # required input to the BSplineCurve superclass
    def control_points(self):
        airfoil_file = self.airfoil_name + '.dat'
        file_path = AIRFOIL_DIR / airfoil_file
        with open(file_path, 'r') as file:
```

```
            point_lst = []
            for line in file:
                # the cartesian coordinates are directly interpreted \
                # as X and Y coordinates
                x, y = line.split(' ', 1)
                point_lst.append(self.position.translate(
                    "x", float(x) * self.chord,  # the x points are scaled \
                    # according to the airfoil chord length
                    "y", float(y) * self.chord * self.thickness_factor))  \
                    # the y points are scaled according to the thickness factor
        return point_lst
```

```
Describe the task performed by the source code provided above using a maximum of 10 words.
```

Class prompt context example

```
### User Message
```

```
class Airfoil(BSplineCurve):

    """Airfoil geometry, a curve through points."""

    #: airfoil name
    #: :type: string
    airfoil_name = Input()
    #: chord [m]
    #: :type: float
    chord = Input(1.)
    #: thickness factor [%]
    #: :type: float
    thickness_factor = Input(1.)

    mesh_deflection = 1e-5
    degree = Input(10)          # Note (JK): without Input() this causes an
    # error. Likely due to different ParaPy version or Python version?

    @Attribute  # required input to the BSplineCurve superclass
    def control_points(self):
        airfoil_file = self.airfoil_name + '.dat'
        file_path = AIRFOIL_DIR / airfoil_file
        with open(file_path, 'r') as file:
            point_lst = []
            for line in file:
                # the cartesian coordinates are directly interpreted \
                # as X and Y coordinates
                x, y = line.split(' ', 1)
                point_lst.append(self.position.translate(
                    "x", float(x) * self.chord,  # the x points are scaled \
                    # according to the airfoil chord length
                    "y", float(y) * self.chord * self.thickness_factor))  \
                    # the y points are scaled according to the thickness factor
        return point_lst
```

```
Describe the task performed by the control_points method defined in the class source code provide above
using a maximum of 10 words.
```

When comparing the examples above, note the difference in the user instruction (at the end of each prompt). In case the class prompt context mode, the `control_points` method is mentioned explicitly. In terms of performance, the differences between the prompting methods is characterized as follows:

- The class prompt context method is a more complex assignment. It asks more from the instruction-following capability of the LLMs (focus on a specific part of the code). Also, larger inputs are provided in this case so the computation performance is (slightly) worse. Correspondingly, the used LLMs should be trained with larger context windows because KBE class definition can be very extensive. However, well-trained LLM should be able to generate better abstractions by having access to **more contextual information**. Especially when method source code is very short (just a few lines of code).
- The method prompt context is **simpler**, with better expected computational performance due to smaller inputs. The concise, simple nature of the prompt means there is less risk of instruction misinterpretation by the LLM that could lead to hallucinations.

In addition to the development and assessment of the two different prompt context modes, the following design aspects of the user prompt generation were also investigated.

1. **User instruction formulation:** Naturally, the specific formulation of the user instruction was considered and iterated quite extensively. The formulation(s) presented in the user prompt examples above reflect the final iteration.
2. **Ordering of code snippet and user instruction:** The ordering of the source code snippet and user instruction was also assessed. It is known LLMs can have trouble "remembering" specific instructions when large inputs are provided, leading to hallucinations. Based on experimentation where the order of the code snippet and instruction was swapped, it was indeed found that it was best to provide the contextual information first, followed by the specific instruction to the LLM.

System prompt formulation

System prompts can be used to provide high-level information describing the LLM's role and context in order to shape its response style. The system prompt is used to provide general information and is appended in front of every user prompt. In case of the REProcess prototype, the system prompt positions the LLM as a reverse engineering assistant, guiding it to understand and abstract the KBE application's behavior. Two variations were implemented, tailored for the method and class modes. The final variants, presented below, were the result from multiple iteration sequences.

- **Method context system prompt:** "Your task is to reverse engineer an existing knowledge based engineering application. The application is written in Python using object-oriented programming. I will provide code snippets of a class method from this application. Your job is to summarize this source code using a maximum of 10 words."
- **Class context system prompt:** "Your task is to reverse engineer an existing knowledge based engineering application. The application is written in Python using object-oriented programming. I will provide the source code of a class from this application. Your job is to summarize the source code of a specific method from this class definition, using a maximum of 10 words."

Keyword response start feature

The final interesting prompt engineering feature that was implemented is the so-called keyworded response start. This is a novel feature that enhances the specificity and relevance of the LLM output by forcing it to start its response with certain predefined keywords. This aids in generating accurate, concise and functionally meaningful descriptions. By default, this feature is turned on in the REProcess prototype implementation but it can also be turned off by setting the `keyword_response_start` input parameter to `False` when calling the `perform_LLM_abstraction` method (the third user command). The implementation of the keyworded response start feature is presented below. This source code, taken from the DataflowAnalyzer class, shows the different keywords used for each node types as well as presenting some additional options that were considered:

```
if keyword_response_start:
    # Good options for...
    #  - Attributes:    Compute, Determine, Calculate,
    #  - Parts:         Generate, Initialize (= SITUATIONAL, only type 2/3 parts!),
    #  - @Inputs:       Compute, Define,
    #  - Context_dep.:  Analyze, Use, Assess, Perform
```

```

    if (self.dataflow.act_graph.nodes[node_i]['node_type'] == 'attribute'):
        response_start = 'Compute'
    elif (self.dataflow.act_graph.nodes[node_i]['node_type']).startswith(
            'part_type-1'):
        response_start = 'Generate'
    elif (self.dataflow.act_graph.nodes[node_i]['node_type']).startswith('part'):
        response_start = 'Create'
    elif (self.dataflow.act_graph.nodes[node_i]['node_type'] == 'input'):
        response_start = 'Determine'
```

Note that implementation of this feature does not simply add the response start as an additional element in the user prompt. Instead, the entire prompt is formatted first, including the LLM response start cue, and then the response start is appended. This “tricks” the LLM into thinking it has already formulated a part of its response so it simply continues from there. This implementation was considered to be most robust.

7.4.3. IMPLEMENTATION OF THE LLMPROMPTER CLASS

The list presented below provides an overview of the important, general aspects of the LLMPrompter class implementation. The remainder of this subsection will go into more depth about the specific LLMs selection, and the used model configuration(s).

- Hugging Face platform: used to select and acquire open-source LLMs.
- Transformers library: used for LLM instantiation and inference.
- PyTorch + CUDA 12.1: to enable GPU-based inference.
- Bitsandbytes library: used for model quantization.
- inspect module: to extract KBE source code snippets.

LLM SELECTION

The REProcess prototype was built using a model-agnostic implementation approach in order to account for the extremely rapid development pace of Large Language Models. In other words, the specific LLM used by the system is easily interchangeable, allowing it to benefit from the continuous improvements in LLM performance. In the REProcess prototype, swapping LLMs can be done simply by changing the `llm_checkpoint` input parameter passed into the `perform_LLM_inference function` (see section B.1). Moreover, this implementation approach meant that the final decision for a particular model could be made later, based on a comparison of actual, reverse engineered diagrams created by the prototype.

Selecting an appropriate LLM was one of the essential aspects of the REProcess prototype development. The Hugging Face platform was used for this purpose. Aside from offering an extensive collection of open-source, state-of-the-art models, another key feature from this platform are the leaderboards. Specifically, the Big Code Models Leaderboard[6] was found to be most useful. It ranks the big code models on the Hugging Face Hub based on several key benchmark scores. Based on these rankings, four LLMs were selected for implementation and testing in the REProcess prototype:

- **CodeLlama-13b-Instruct-hf:**[7] The CodeLlama models are code-specialized versions of the general-purpose Llama 2 model. This is the instruction-tuned variant with 13B nodes, which fits on the workstation GPU without quantization. Trained on 16K token sequences. Additional inference improvements shown on inputs of up to 100,000 tokens [47].
- **CodeLlama-34b-Instruct-hf:**[8] This is the 34B variant of the CodeLlama model described above, which requires 8-bit quantization to fit on the workstation GPU.
- **Phind-CodeLlama-34B-v2:**[9] Fine-tuned version of the CodeLlama-34B model above, trained on an additional set of high-quality programming-related data.

[6] `https://huggingface.co/spaces/bigcode/bigcode-models-leaderboard`
[7] Model page: `https://huggingface.co/meta-llama/CodeLlama-13b-Instruct-hf`
[8] Model page: `https://huggingface.co/meta-llama/CodeLlama-34b-Instruct-hf`
[9] Model page: `https://huggingface.co/Phind/Phind-CodeLlama-34B-v2`

- **Deepseek-coder-33b-instruct:**[10] Instruction-tuned version of the Deepseek coder model, which was trained from scratch. It has a context window size of 16K tokens. Like the 33B models above, it requires 8-bit quantization to fit on the workstation GPU.

From an implementation perspective, it should be noted that a key difference between these models is that they use different prompt formats. In order to account for these differences, the LLMPrompter class implements a `do_prompt_formatting` method that checks the specific model type and applies the appropriate format. Examples of two different prompt formats are shown in the code snippet below, which was taken from the `do_prompt_formatting` method. Note that the REProcess prototype implementation was developed before the `apply_chat_template` was introduced, which can be used to automate this task.

```
def do_prompt_formatting(self, system_prompt, code_snip_i, user_instuction):
    [...]

    if self.checkpoint == "Phind/Phind-CodeLlama-34B-v2":
        self.formatted_prompt = (
            "### System Prompt\n"
            f"{system_prompt}\n\n"
            "### User Message\n"
            f"{code_snip_i}\n"
            f"{user_instuction}\n\n"
            "### Assistant Response\n"
        )
        # Define llm_response_start_cue to be equal to the final line of the formatted prompt:
        self.llm_response_start_cue = "### Assistant Response\n"

    elif "codellama/CodeLlama-" in self.checkpoint and "Instruct-hf" in self.checkpoint:
        self.formatted_prompt = f"""\
            [INST] <<SYS>> {system_prompt} <</SYS>> \
            {code_snip_i} \n
            {user_instuction} \
            [/INST]\n"""
        # Define llm_response_start_cue to be equal to the final line of the formatted prompt:
        self.llm_response_start_cue = "[/INST]\n"

    [...]

    return self.formatted_prompt
```

Based on experimentation, the `Phind-CodeLlama-34B-v2` model was selected due to its proficiency in handling complex code interpretation tasks, including Parapy-specifc syntax such as `DynamicType`. Also its ability to follow the specific instructions provided in the prompt (e.g. limiting to a maximum of 10 words) stood out compared to the other models.

LLM CONFIGURATION

Once a set of models was selected, the next step was to determine the optimal parameters and settings to maximize the tool's efficiency and output quality. The implementation of the `LLMPrompter` class provided the foundational structure for this process. Key parameters such as quantization and generation configuration were adjusted and tested iteratively.

- **Quantization configuration:** The model was loaded with an 8-bit quantization setting, as specified by the `BitsAndBytesConfig` with `load_in_8bit=True`. This approach significantly reduces memory usage while maintaining a high level of precision in output generation. Although a 4-bit quantization was considered, the 8-bit configuration provided a better balance between performance and accuracy.

[10] Model page: https://huggingface.co/deepseek-ai/deepseek-coder-33b-instruct

- **Output generation configuration:** The output generation parameters were carefully tuned to enhance output quality. Key settings included:
  - `num_beams=2`: This parameter enabled beam search, an effective decoding strategy that evaluates multiple sequences simultaneously, thus improving the likelihood of generating the most relevant descriptions.
  - `max_new_tokens=500`: This constraint ensured that the model's output was concise, preventing overly verbose responses that could detract from the clarity of the generated abstractions.

7.5. OUTPUT GENERATION IMPLEMENTATION

7.5.1. TEMPLATE-BASED TEXT GENERATION FOR SPECIAL NODES

As mentioned in the discussion of the algorithm steps above, geometry nodes and an initialization node are kept in the graph to represent specific important steps in the KBE application workflow. For these two nodes, relevant KBE application source code is not available so the LLM based abstraction method cannot be used to generate the textual descriptions of these nodes. This is highlighted in the example provided below (Figure 7.11). The tooltip for the final node is shown which presents the corresponding source code. The last parameter shown in this tooltip shows the source code from the `TopoDS_Shape` slot, which is defined within the KBE system itself. This code is not specific to the KBE application itself, so providing it to an LLM to let it generate a textual node description cannot lead to a useful output. Instead, a template-based method is used to generate the corresponding node description: `f"Generate <node name> geometry using <class name of node object>"`. A similar template based approach is used for the initialization node, included at the top of Figure 7.11.

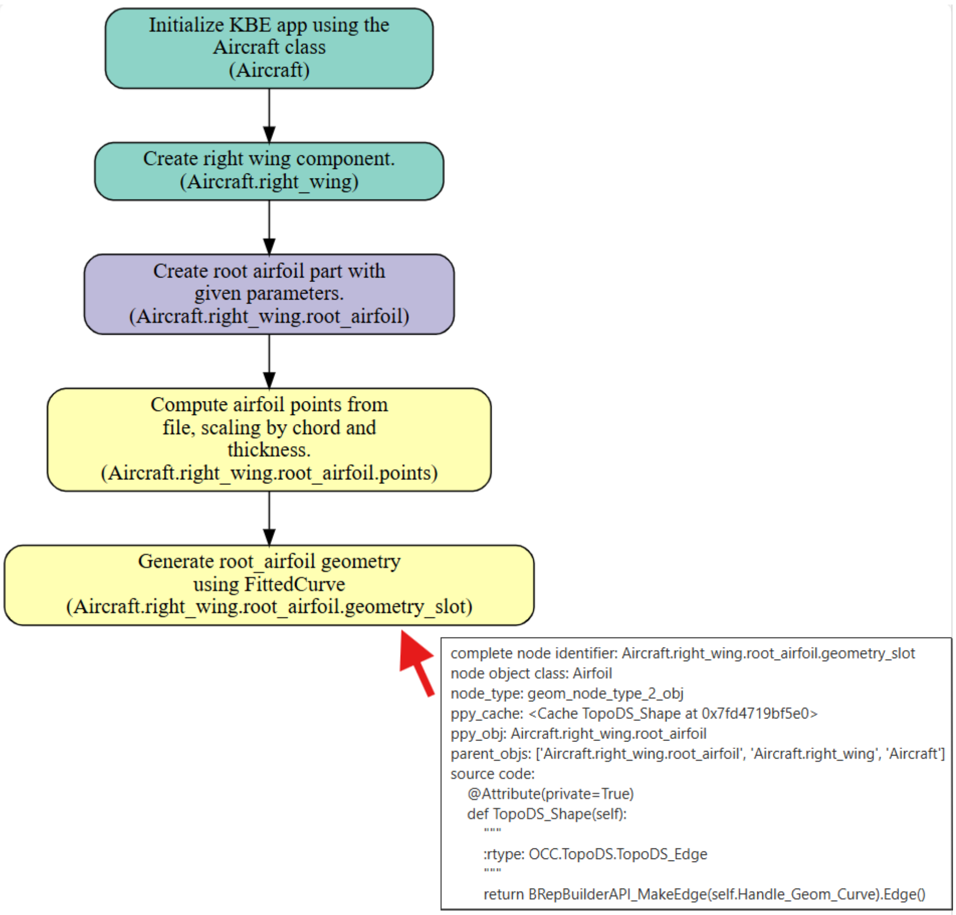


Figure 7.11: Example activity diagram generated using the reverse engineering method. A tooltip corresponding to the final node is depicted that shows the metadata attached to this node.

7.5.2. DIAGRAM VISUALIZATION

Diagrams are generated with Pygraphviz[11], which is a dedicated graph visualization tool. A dedicated function is integrated into NetworkX to convert the NetworkX graph to a Pygraphviz graph (`nx.nx_agraph.to_agraph`). Subsequently, Pygraphviz is used to generate the diagrams (using the "dot" layout engine).

1. **Node shapes** are rounded rectangles. This is most semantically similar to action nodes in activity diagrams.

2. **Coloring** of the nodes is used as a substitute for SysML swimlanes. The color assignment is performed based on alphabetical sorting of class names. Additionally, the class of the root object is always set to green. Both measures increase consistency between different diagrams, and therefore cognitive effectiveness.

3. **Data flow** is captured in the edge tooltips. These are shown when hovering on them with the cursor. This ensures the desired data is present in the diagram for users that would like to review it, while hiding it in the diagram itself to prevent cluttering the diagram with a lot of additional information.

4. **Tooltips** are shown when hovering the cursor on specific nodes or edges to present additional, more specific information to the user. For example, the node type, object class, and corresponding class method source code are shown when hovering on nodes. Edge tooltips present the data flow by listing all of the slots flowing through a connector. For connectors flowing to/from Parapy parts, this includes all individual input slots. These tooltips are a substitute for the specification window that can be opened when using modeling platforms such as CATIA Magic Systems of Systems Architect to inspect the metadata associated to nodes and node connectors.

7.5.3. WRITING OUTPUT FILES

In addition to the visualizations, two output files can be generated by the REProcess prototype: the raw LLM data (including performance metrics), and DOT file that captures the full activity graph. Aside from the graph structure, this DOT file includes all relevant metadata of the graph, nodes and edges as well as the textual node descriptions generated by LLM.

7.5.4. PARTIAL VISUALIZATION METHOD

The `visualize_partial_graph()` method is an additional feature that was implemented in the ReverseProcessModel class to provide another key functionality to the tool user. This method was implemented to mimic a key feature of the SysML models represented in modeling platforms like CATIA Magic Systems of Systems Architect, namely the ability to "collapse" a whole bunch of lower-level behavior into a single node and represent this in a separate activity diagram.

The `visualize_partial_graph()` method mimics this as follows. Once called by the user (by calling `<process_model>.visualize_partial_graph()`), it prints a numbered list with all the class names present in the reverse engineered activity graph and prompts the user to provide a list of the specific classes that they would like to include in the partial visualization. Based on this list, it generates a partial graph that only contains nodes from the specified classes. All other nodes are contracted into their respective part nodes. Results from this functionality are presented in the next chapter.

[11] https://pygraphviz.github.io

8

REPROCESS PROTOTYPE RESULTS

This chapter presents the key results and findings obtained with the REProcess prototype, highlighting its capabilities and performance characteristics. section 8.1 briefly outlines the verification approach, detailing the selection of test cases and their alignment with the system requirements. Following this, section 8.2 presents the key findings, including example diagrams and qualitative insights into the prototype's ability to reverse engineer KBE application workflows. These results highlight the completeness of the extracted metadata, the capability of hierarchical diagram generation, and the ability to swap LLMs without altering the tool's core implementation. Additionally, the impact of several key features of the REProcess prototype are also discussed, along with an analysis of computational performance across different test cases. Finally, some remaining issues and limitations identified based on the results are discussed in section 8.3.

8.1. VERIFICATION

A set of test cases was composed for the verification of the implemented REProcess prototype. Two KBE applications were used for the bulk of these test cases, namely the Primiplane app[1] and the Modular UAV app [12]. A few test cases analyzed other KBE applications, including the Asystor app [48] and a small, custom developed application (ref_KBE_JK).

Note: all reference KBE applications used in this research are included in the separately provided source code repository of the REProcess prototype.

The approach used to define the test cases consisted of two main steps. First, an initial set of test cases was created that covers the full range of target slots specified by the combination of prototype requirements 6 and 7. Concretely, this meant two target slots were included for every type of Parapy slot: inputs, attributes, parts, and Sequences of parts. Per slot type, one test case targeted a slot from the root object, while the other targeted a lower-level slot nested within another part object (i.e. lower in the product hierarchy). For both parts and part sequences, test cases were included for all three part types defined in subsection 6.3.3 (i.e. type 1, 2, and 3).

The second step consisted of assessing this initial set of test cases and determining which of the other prototype requirements were not yet covered. Based on this assessment, some changes and additions were made. Some target slots were swapped out for others in order to include additional special Parapy features into the test plan. For example, an instance of `DynamicType()` was manually introduced into the Fuselage class of the Primiplane application, since this feature was not used in either of the two original KBE applications. Finally, some specialized test cases were added to cover edge cases and specific design features implemented in the REProcess prototype. For example, analyzing the same test case both with and without using keyworded response start feature.

A complete overview of the verification test set is provided in Appendix C. The remainder of this chapter presents and discusses a number of diagrams that were selected to highlight key features and obtained results.

[1] Specifically, the Primiplane app from tutorial 8 of the KBE course material [41]

8.2. KEY RESULTS

This section presents several example diagrams and other REProcess tool outputs that showcase its capabilities and highlight key features of the implementation.

COMPREHENSIVE METADATA

Figure 8.1 presents an annotated example of one of the generated activity diagrams. The figure was composed by combining four screenshots to include the four tooltips, which can normally only be viewed one at a time, by hovering the mouse on the respective component (only works for SVG files). Two of the tooltips showcase the metadata attached to the nodes themselves, while two others show the data flowing through the node edges. This visualization showcases the comprehensive metadata attached to the model elements, highlighting how the REProcess tool is capable of reverse engineering a model of the software, rather than just generating a particular software visualization,

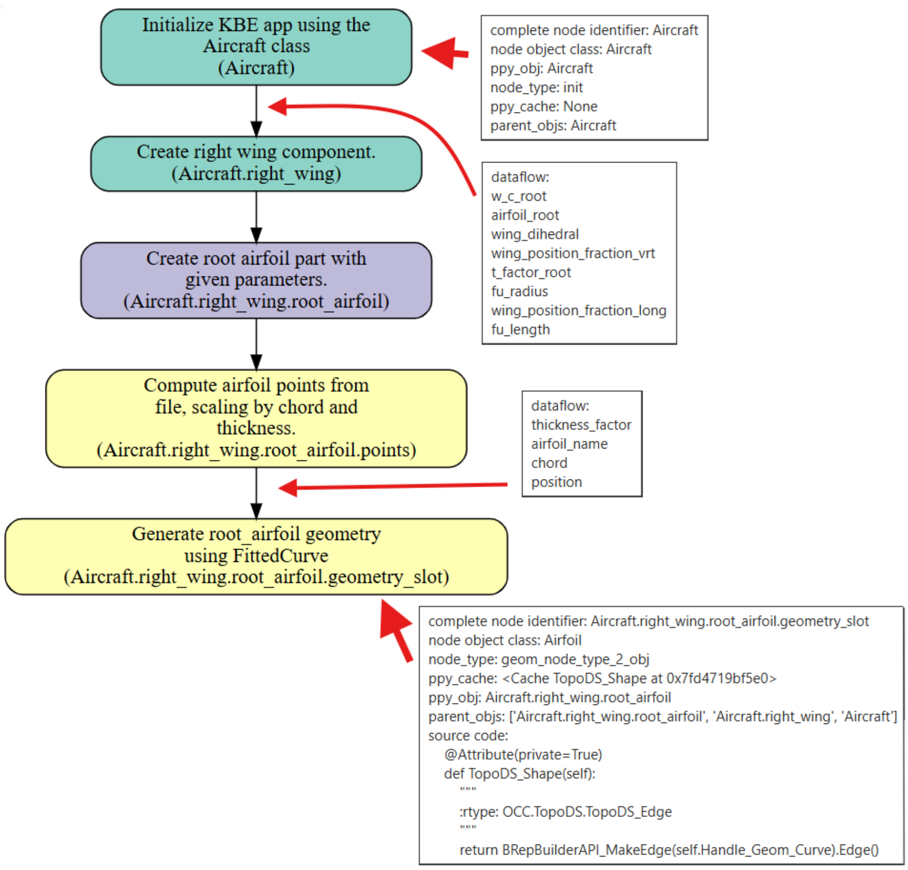


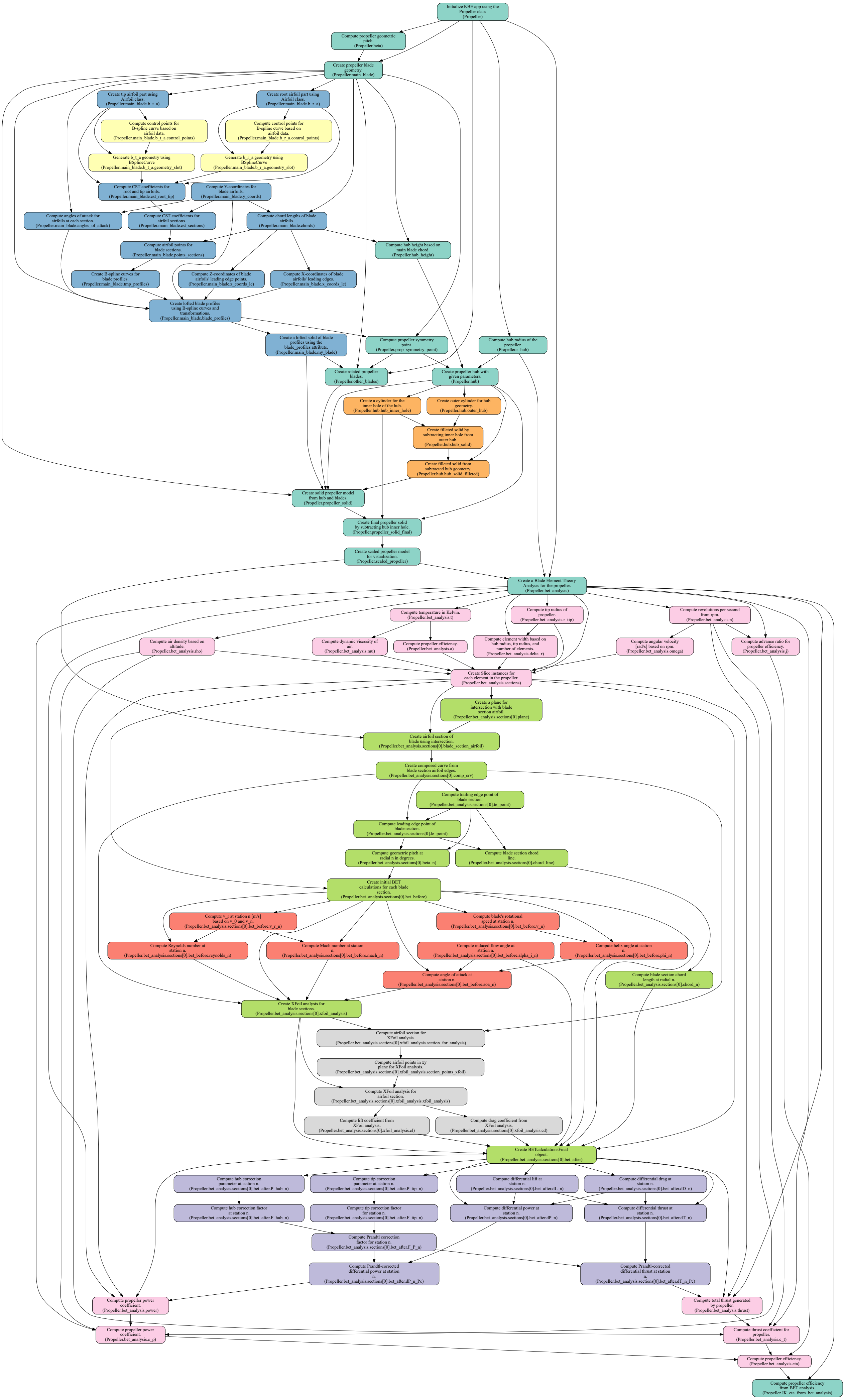
Figure 8.1: Activity diagram showing four tooltips to highlight the comprehensive amount of attached metadata.

PARTIAL DIAGRAMS (MODULAR UAV)

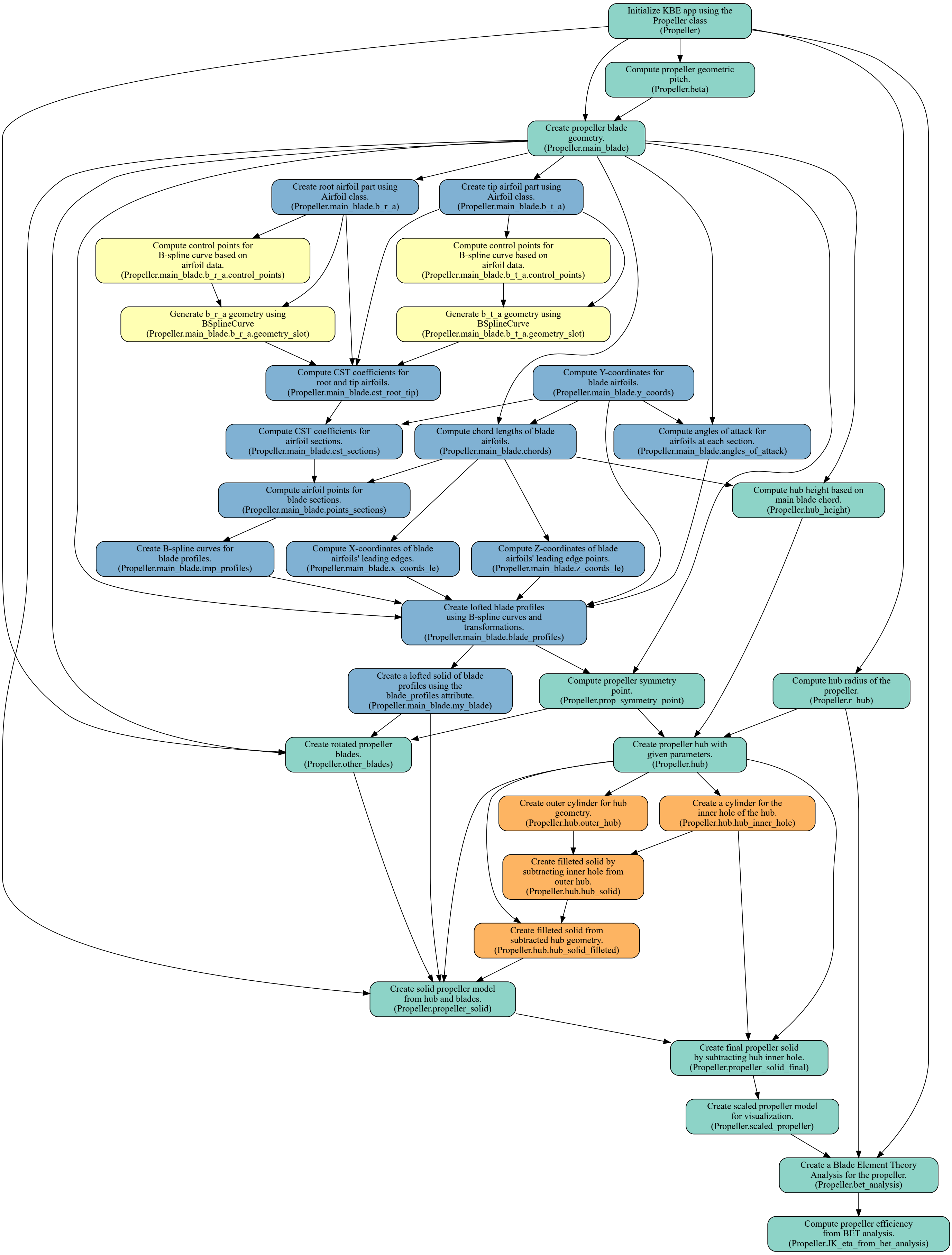
One the following three pages, a set of diagrams is presented that showcase the **partial diagram generation** feature of the REProcess prototype. The reverse engineering target is the `Propeller.JK_eta_from_bet_analysis` slot from the modified Modular UAV application. To allow readability, the reader is referred to the digital version of this document, as it contains the PDF-versions of these diagrams that support high levels of magnification to make the node content legible.

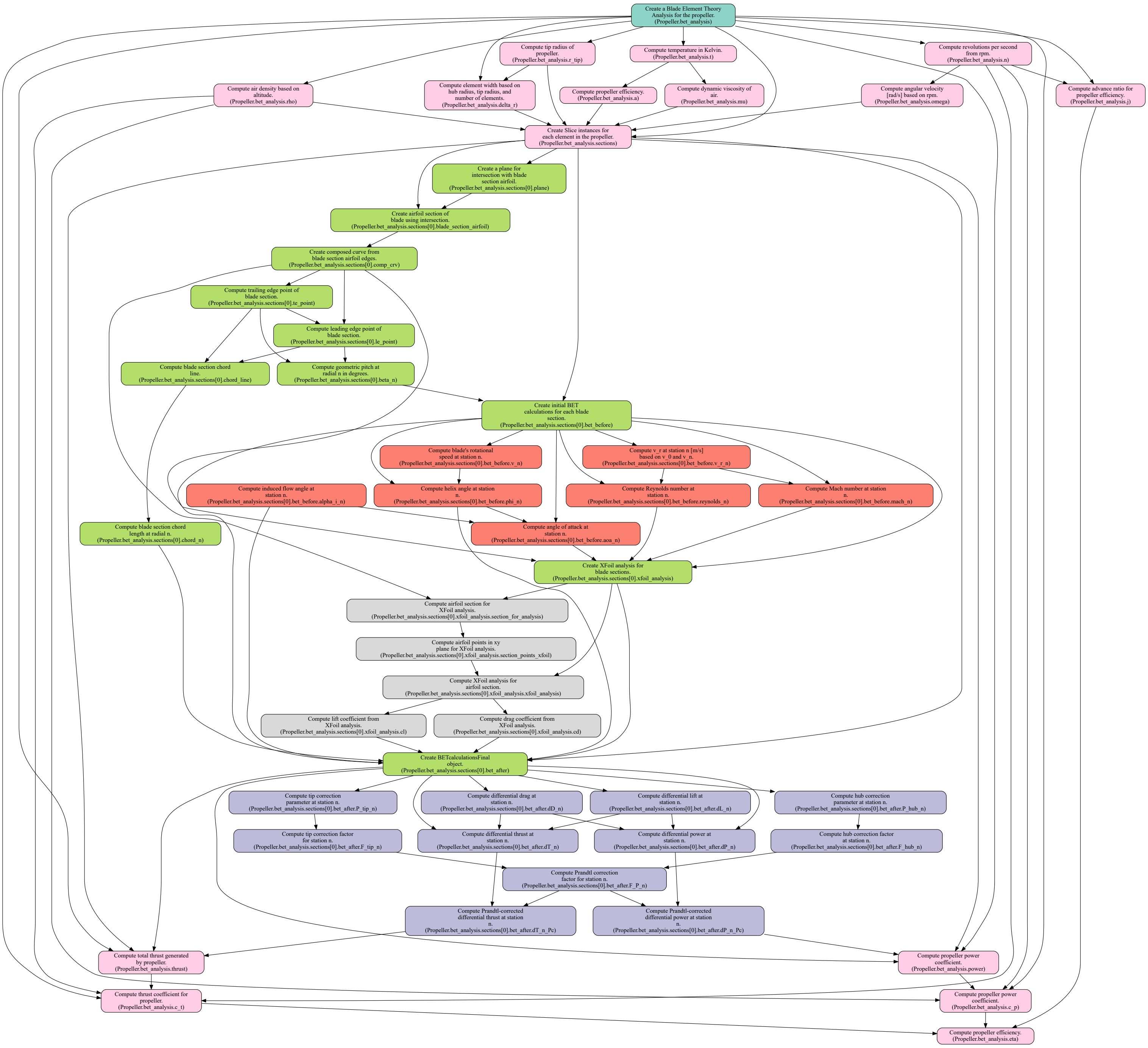
The first diagram represents the complete output, i.e. all relevant steps from the entire workflow. The two subsequent diagrams respectively focus on the first and second segments of this comprehensive workflow. These segments are divided by the centrally positioned "Create a Blade Element Theory Analysis for the propeller. Propeller.bet_analysis)" node (dark green, right above the cluster of pink nodes). Collectively, this set of diagrams show the REProcess tool capability to generate a set of hierarchically structured diagrams corresponding to a single workflow, similar to the set of SysML activity diagrams presented in section 4.1 that exemplified the so-called **comprehensive process model** defining the desired output from the REProcess tool.

An additional key remark about these diagrams is that they present the largest and most complex KBE application workflow from the test cases included in the verification set. This is highlighted by the size of its precedents graph obtained after the first reverse engineering step. It consisted of roughly 2,300 nodes (!). To better understand the magnitude of this number, consider that the diagram on the next page, which represents the entire workflow and includes all relevant steps, already seems large and complicated. However, the amount of nodes in this diagram is “just” 80. This massive reduction in node count, combined with the fact that the resulting diagram **accurately** reflects the internal process steps, is deemed a significant achievement.



Initialize KBE app using the Propeller class (Propeller)
Compute propeller geometric pitch. (Propeller.beta)
Create propeller blade geometry. (Propeller.main_blade)
Create tip airfoil part using Airfoil class. (Propeller.main_blade.b_t_a)
Create root airfoil part using Airfoil class. (Propeller.main_blade.b_r_a)
Compute control points for B-spline curve based on airfoil data. (Propeller.main_blade.b_t_a.control_points)
Compute control points for B-spline curve based on airfoil data. (Propeller.main_blade.b_r_a.control_points)
Generate b_t_a geometry using BSplineCurve (Propeller.main_blade.b_t_a.geometry_slot)
Generate b_r_a geometry using BSplineCurve (Propeller.main_blade.b_r_a.geometry_slot)
Compute CST coefficients for root and tip airfoils. (Propeller.main_blade.cst_root_tip)
Compute Y-coordinates for blade airfoils. (Propeller.main_blade.y_coords)
Compute angles of attack for airfoils at each section. (Propeller.main_blade.angles_of_attack)
Compute CST coefficients for airfoil sections. (Propeller.main_blade.cst_sections)
Compute chord lengths of blade airfoils. (Propeller.main_blade.chords)
Compute airfoil points for blade sections. (Propeller.main_blade.points_sections)
Compute hub height based on main blade chord. (Propeller.hub_height)
Create B-spline curves for blade profiles. (Propeller.main_blade.temp_profiles)
Compute Z-coordinates of blade airfoils' leading edge points. (Propeller.main_blade.z_coords_le)
Compute X-coordinates of blade airfoils' leading edges. (Propeller.main_blade.x_coords_le)
Create lofted blade profiles using B-spline curves and transformations. (Propeller.main_blade.blade_profiles)
Create a lofted solid of blade profiles using the blade_profiles attribute. (Propeller.main_blade.my_blade)
Compute propeller symmetry point. (Propeller.prop_symmetry_point)
Compute hub radius of the propeller. (Propeller.r_hub)
Create rotated propeller blades. (Propeller.other_blades)
Create propeller hub with given parameters. (Propeller.hub)
Create a cylinder for the inner hole of the hub. (Propeller.hub.hub_inner_hole)
Create outer cylinder for hub geometry. (Propeller.hub.outer_hub)
Create filleted solid by subtracting inner hole from outer hub. (Propeller.hub.hub_solid)
Create filleted solid from subtracted hub geometry. (Propeller.hub.hub_solid_filleted)
Create solid propeller model from hub and blades. (Propeller.propeller_solid)
Create final propeller solid by subtracting hub inner hole. (Propeller.propeller_solid_final)
Create scaled propeller model for visualization. (Propeller.scaled_propeller)
Create a Blade Element Theory Analysis for the propeller. (Propeller.bet_analysis)
Compute temperature in Kelvin. (Propeller.bet_analysis.t)
Compute tip radius of propeller. (Propeller.bet_analysis.r_tip)
Compute revolutions per second from rpm. (Propeller.bet_analysis.n)
Compute air density based on altitude. (Propeller.bet_analysis.rho)
Compute dynamic viscosity of air. (Propeller.bet_analysis.mu)
Compute propeller efficiency. (Propeller.bet_analysis.a)
Compute element width based on hub radius, tip radius, and number of elements. (Propeller.bet_analysis.delta_r)
Compute angular velocity [rad/s] based on rpm. (Propeller.bet_analysis.omega)
Compute advance ratio for propeller efficiency. (Propeller.bet_analysis.j)
Create Slice instances for each element in the propeller. (Propeller.bet_analysis.sections)
Create a plane for intersection with blade section airfoil. (Propeller.bet_analysis.sections[0].plane)
Create airfoil section of blade using intersection. (Propeller.bet_analysis.sections[0].blade_section_airfoil)
Create composed curve from blade section airfoil edges. (Propeller.bet_analysis.sections[0].comp_crv)
Compute trailing edge point of blade section. (Propeller.bet_analysis.sections[0].te_point)
Compute leading edge point of blade section. (Propeller.bet_analysis.sections[0].le_point)
Compute geometric pitch at radial n in degrees. (Propeller.bet_analysis.sections[0].beta_n)
Compute blade section chord line. (Propeller.bet_analysis.sections[0].chord_line)
Create initial BET calculations for each blade section. (Propeller.bet_analysis.sections[0].bet_before)
Compute v_r at station n [m/s] based on v_0 and v_n. (Propeller.bet_analysis.sections[0].bet_before.v_r_n)
Compute blade's rotational speed at station n. (Propeller.bet_analysis.sections[0].bet_before.v_n)
Compute Reynolds number at station n. (Propeller.bet_analysis.sections[0].bet_before.reynolds_n)
Compute Mach number at station n. (Propeller.bet_analysis.sections[0].bet_before.mach_n)
Compute induced flow angle at station n. (Propeller.bet_analysis.sections[0].bet_before.alpha_i_n)
Compute helix angle at station n. (Propeller.bet_analysis.sections[0].bet_before.phi_n)
Compute angle of attack at station n. (Propeller.bet_analysis.sections[0].bet_before.aoa_n)
Compute blade section chord length at radial n. (Propeller.bet_analysis.sections[0].chord_n)
Create XFoil analysis for blade sections. (Propeller.bet_analysis.sections[0].xfoil_analysis)
Compute airfoil section for XFoil analysis. (Propeller.bet_analysis.sections[0].xfoil_analysis.section_for_analysis)
Compute airfoil points in xy plane for XFoil analysis. (Propeller.bet_analysis.sections[0].xfoil_analysis.section_points_xfoil)
Compute XFoil analysis for airfoil section. (Propeller.bet_analysis.sections[0].xfoil_analysis.xfoil_analysis)
Compute lift coefficient from XFoil analysis. (Propeller.bet_analysis.sections[0].xfoil_analysis.cl)
Compute drag coefficient from XFoil analysis. (Propeller.bet_analysis.sections[0].xfoil_analysis.cd)
Create BETcalculationsFinal object. (Propeller.bet_analysis.sections[0].bet_after)
Compute hub correction parameter at station n. (Propeller.bet_analysis.sections[0].bet_after.P_hub_n)
Compute tip correction parameter at station n. (Propeller.bet_analysis.sections[0].bet_after.P_tip_n)
Compute differential lift at station n. (Propeller.bet_analysis.sections[0].bet_after.dL_n)
Compute differential drag at station n. (Propeller.bet_analysis.sections[0].bet_after.dD_n)
Compute hub correction factor at station n. (Propeller.bet_analysis.sections[0].bet_after.F_hub_n)
Compute tip correction factor for station n. (Propeller.bet_analysis.sections[0].bet_after.F_tip_n)
Compute differential power at station n. (Propeller.bet_analysis.sections[0].bet_after.dP_n)
Compute differential thrust at station n. (Propeller.bet_analysis.sections[0].bet_after.dT_n)
Compute Prandtl correction factor for station n. (Propeller.bet_analysis.sections[0].bet_after.F_P_n)
Compute Prandtl-corrected differential power at station n. (Propeller.bet_analysis.sections[0].bet_after.dP_n_Pc)
Compute Prandtl-corrected differential thrust at station n. (Propeller.bet_analysis.sections[0].bet_after.dT_n_Pc)
Compute propeller power coefficient. (Propeller.bet_analysis.power)
Compute total thrust generated by propeller. (Propeller.bet_analysis.thrust)
Compute propeller power coefficient. (Propeller.bet_analysis.c_p)
Compute thrust coefficient for propeller. (Propeller.bet_analysis.c_t)
Compute propeller efficiency. (Propeller.bet_analysis.eta)
Compute propeller efficiency from BET analysis. (Propeller.JK_eta_from_bet_analysis)

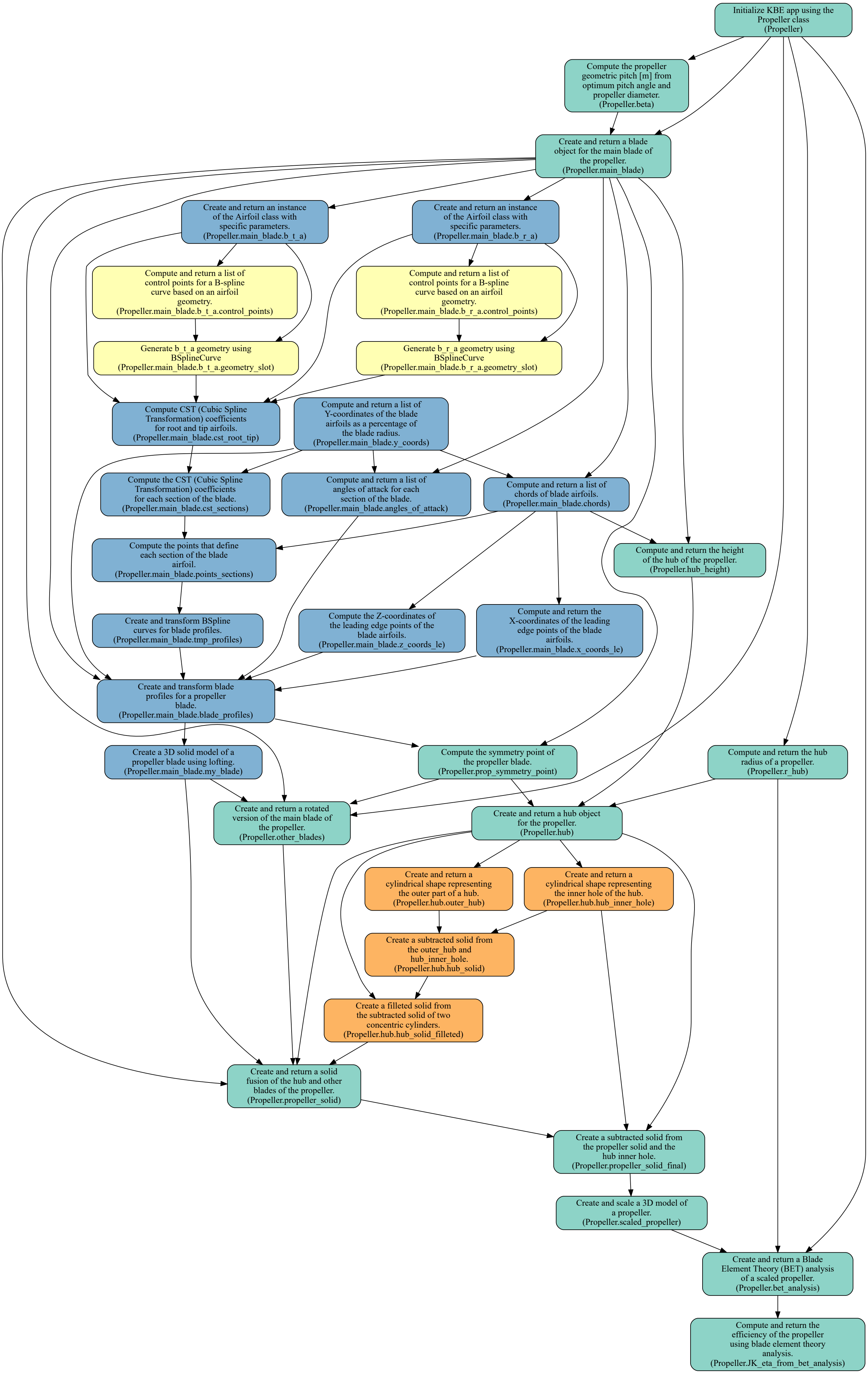

Initialize KBE app using the Propeller class (Propeller)
Compute propeller geometric pitch. (Propeller.beta)
Create propeller blade geometry. (Propeller.main_blade)
Create root airfoil part using Airfoil class. (Propeller.main_blade.b_r_a)
Create tip airfoil part using Airfoil class. (Propeller.main_blade.b_t_a)
Compute control points for B-spline curve based on airfoil data. (Propeller.main_blade.b_r_a.control_points)
Compute control points for B-spline curve based on airfoil data. (Propeller.main_blade.b_t_a.control_points)
Generate b_r_a geometry using BSplineCurve (Propeller.main_blade.b_r_a.geometry_slot)
Generate b_t_a geometry using BSplineCurve (Propeller.main_blade.b_t_a.geometry_slot)
Compute CST coefficients for root and tip airfoils. (Propeller.main_blade.cst_root_tip)
Compute Y-coordinates for blade airfoils. (Propeller.main_blade.y_coords)
Compute CST coefficients for airfoil sections. (Propeller.main_blade.cst_sections)
Compute chord lengths of blade airfoils. (Propeller.main_blade.chords)
Compute angles of attack for airfoils at each section. (Propeller.main_blade.angles_of_attack)
Compute airfoil points for blade sections. (Propeller.main_blade.points_sections)
Compute hub height based on main blade chord. (Propeller.hub_height)
Create B-spline curves for blade profiles. (Propeller.main_blade.tmp_profiles)
Compute X-coordinates of blade airfoils' leading edges. (Propeller.main_blade.x_coords_le)
Compute Z-coordinates of blade airfoils' leading edge points. (Propeller.main_blade.z_coords_le)
Create lofted blade profiles using B-spline curves and transformations. (Propeller.main_blade.blade_profiles)
Create a lofted solid of blade profiles using the blade_profiles attribute. (Propeller.main_blade.my_blade)
Compute propeller symmetry point. (Propeller.prop_symmetry_point)
Compute hub radius of the propeller. (Propeller.r_hub)
Create rotated propeller blades. (Propeller.other_blades)
Create propeller hub with given parameters. (Propeller.hub)
Create outer cylinder for hub geometry. (Propeller.hub.outer_hub)
Create a cylinder for the inner hole of the hub. (Propeller.hub.hub_inner_hole)
Create filleted solid by subtracting inner hole from outer hub. (Propeller.hub.hub_solid)
Create filleted solid from subtracted hub geometry. (Propeller.hub.hub_solid_filleted)
Create solid propeller model from hub and blades. (Propeller.propeller_solid)
Create final propeller solid by subtracting hub inner hole. (Propeller.propeller_solid_final)
Create scaled propeller model for visualization. (Propeller.scaled_propeller)
Create a Blade Element Theory Analysis for the propeller. (Propeller.bet_analysis)
Compute propeller efficiency from BET analysis. (Propeller.JK_eta_from_bet_analysis)


Create a Blade Element Theory Analysis for the propeller. (Propeller.bet_analysis)
Compute tip radius of propeller. (Propeller.bet_analysis.r_tip)
Compute temperature in Kelvin. (Propeller.bet_analysis.t)
Compute revolutions per second from rpm. (Propeller.bet_analysis.n)
Compute air density based on altitude. (Propeller.bet_analysis.rho)
Compute element width based on hub radius, tip radius, and number of elements. (Propeller.bet_analysis.delta_r)
Compute propeller efficiency. (Propeller.bet_analysis.a)
Compute dynamic viscosity of air. (Propeller.bet_analysis.mu)
Compute angular velocity [rad/s] based on rpm. (Propeller.bet_analysis.omega)
Compute advance ratio for propeller efficiency. (Propeller.bet_analysis.j)
Create Slice instances for each element in the propeller. (Propeller.bet_analysis.sections)
Create a plane for intersection with blade section airfoil. (Propeller.bet_analysis.sections[0].plane)
Create airfoil section of blade using intersection. (Propeller.bet_analysis.sections[0].blade_section_airfoil)
Create composed curve from blade section airfoil edges. (Propeller.bet_analysis.sections[0].comp_crv)
Compute trailing edge point of blade section. (Propeller.bet_analysis.sections[0].te_point)
Compute leading edge point of blade section. (Propeller.bet_analysis.sections[0].le_point)
Compute blade section chord line. (Propeller.bet_analysis.sections[0].chord_line)
Compute geometric pitch at radial n in degrees. (Propeller.bet_analysis.sections[0].beta_n)
Create initial BET calculations for each blade section. (Propeller.bet_analysis.sections[0].bet_before)
Compute blade's rotational speed at station n. (Propeller.bet_analysis.sections[0].bet_before.v_n)
Compute v_r at station n [m/s] based on v_0 and v_n. (Propeller.bet_analysis.sections[0].bet_before.v_r_n)
Compute induced flow angle at station n. (Propeller.bet_analysis.sections[0].bet_before.alpha_i_n)
Compute helix angle at station n. (Propeller.bet_analysis.sections[0].bet_before.phi_n)
Compute Reynolds number at station n. (Propeller.bet_analysis.sections[0].bet_before.reynolds_n)
Compute Mach number at station n. (Propeller.bet_analysis.sections[0].bet_before.mach_n)
Compute blade section chord length at radial n. (Propeller.bet_analysis.sections[0].chord_n)
Compute angle of attack at station n. (Propeller.bet_analysis.sections[0].bet_before.aoa_n)
Create XFoil analysis for blade sections. (Propeller.bet_analysis.sections[0].xfoil_analysis)
Compute airfoil section for XFoil analysis. (Propeller.bet_analysis.sections[0].xfoil_analysis.section_for_analysis)
Compute airfoil points in xy plane for XFoil analysis. (Propeller.bet_analysis.sections[0].xfoil_analysis.section_points_xfoil)
Compute XFoil analysis for airfoil section. (Propeller.bet_analysis.sections[0].xfoil_analysis.xfoil_analysis)
Compute lift coefficient from XFoil analysis. (Propeller.bet_analysis.sections[0].xfoil_analysis.cl)
Compute drag coefficient from XFoil analysis. (Propeller.bet_analysis.sections[0].xfoil_analysis.cd)
Create BETcalculationsFinal object. (Propeller.bet_analysis.sections[0].bet_after)
Compute tip correction parameter at station n. (Propeller.bet_analysis.sections[0].bet_after.P_tip_n)
Compute differential drag at station n. (Propeller.bet_analysis.sections[0].bet_after.dD_n)
Compute differential lift at station n. (Propeller.bet_analysis.sections[0].bet_after.dL_n)
Compute hub correction parameter at station n. (Propeller.bet_analysis.sections[0].bet_after.P_hub_n)
Compute tip correction factor for station n. (Propeller.bet_analysis.sections[0].bet_after.F_tip_n)
Compute differential thrust at station n. (Propeller.bet_analysis.sections[0].bet_after.dT_n)
Compute differential power at station n. (Propeller.bet_analysis.sections[0].bet_after.dP_n)
Compute hub correction factor at station n. (Propeller.bet_analysis.sections[0].bet_after.F_hub_n)
Compute Prandtl correction factor for station n. (Propeller.bet_analysis.sections[0].bet_after.F_P_n)
Compute Prandtl-corrected differential thrust at station n. (Propeller.bet_analysis.sections[0].bet_after.dT_n_Pc)
Compute Prandtl-corrected differential power at station n. (Propeller.bet_analysis.sections[0].bet_after.dP_n_Pc)
Compute total thrust generated by propeller. (Propeller.bet_analysis.thrust)
Compute propeller power coefficient. (Propeller.bet_analysis.power)
Compute thrust coefficient for propeller. (Propeller.bet_analysis.c_t)
Compute propeller power coefficient. (Propeller.bet_analysis.c_p)
Compute propeller efficiency. (Propeller.bet_analysis.eta)

TESTING DIFFERENT LLMS

As discussed in , the REProcess prototype was built using a model-agnostic implementation approach in order to account for the extremely rapid development pace of Large Language Models. The specific LLM used by the system is **easily interchangeable**, allowing it to benefit from the continuous improvements in LLM performance. In the REProcess prototype, swapping LLMs can be done simply by changing the `llm_checkpoint` input parameter passed into the `perform_LLM_inference function` (see section B.1). For the comparison between two top-performing Big Code models from the corresponding HuggingFace leaderboard[2], another version of the second diagram from the previously presented set of three diagrams is provided on the following page, which was generated using the *Deepseek-coder-33b-instruct* instead of *Phind-CodeLlama-34B-v2*. The instructions, context and other model settings were identical. A comparison of the textual descriptions generated by these models shows the Deepseek model has a tendency to write longer descriptions compared to *Phind-CodeLlama-34B-v2*. Moreover, the descriptions from the Deepseek model produces are more focused on the source code steps, often refering to code constructs like lists, instances, and parameters, rather than abstracting higher-level functions. This highlights why the *Phind-CodeLlama-34B-v2* is preferred.

[2] `https://huggingface.co/spaces/bigcode/bigcode-models-leaderboard`, accessed on 6-20-2025


Initialize KBE app using the Propeller class (Propeller)
Compute the propeller geometric pitch [m] from optimum pitch angle and propeller diameter. (Propeller.beta)
Create and return a blade object for the main blade of the propeller. (Propeller.main_blade)
Create and return an instance of the Airfoil class with specific parameters. (Propeller.main_blade.b_t_a)
Create and return an instance of the Airfoil class with specific parameters. (Propeller.main_blade.b_r_a)
Compute and return a list of control points for a B-spline curve based on an airfoil geometry. (Propeller.main_blade.b_t_a.control_points)
Compute and return a list of control points for a B-spline curve based on an airfoil geometry. (Propeller.main_blade.b_r_a.control_points)
Generate b_t_a geometry using BSplineCurve (Propeller.main_blade.b_t_a.geometry_slot)
Generate b_r_a geometry using BSplineCurve (Propeller.main_blade.b_r_a.geometry_slot)
Compute CST (Cubic Spline Transformation) coefficients for root and tip airfoils. (Propeller.main_blade.cst_root_tip)
Compute and return a list of Y-coordinates of the blade airfoils as a percentage of the blade radius. (Propeller.main_blade.y_coords)
Compute the CST (Cubic Spline Transformation) coefficients for each section of the blade. (Propeller.main_blade.cst_sections)
Compute and return a list of angles of attack for each section of the blade. (Propeller.main_blade.angles_of_attack)
Compute and return a list of chords of blade airfoils. (Propeller.main_blade.chords)
Compute the points that define each section of the blade airfoil. (Propeller.main_blade.points_sections)
Compute and return the height of the hub of the propeller. (Propeller.hub_height)
Create and transform BSpline curves for blade profiles. (Propeller.main_blade.tmp_profiles)
Compute the Z-coordinates of the leading edge points of the blade airfoils. (Propeller.main_blade.z_coords_le)
Compute and return the X-coordinates of the leading edge points of the blade airfoils. (Propeller.main_blade.x_coords_le)
Create and transform blade profiles for a propeller blade. (Propeller.main_blade.blade_profiles)
Create a 3D solid model of a propeller blade using lofting. (Propeller.main_blade.my_blade)
Compute the symmetry point of the propeller blade. (Propeller.prop_symmetry_point)
Compute and return the hub radius of a propeller. (Propeller.r_hub)
Create and return a rotated version of the main blade of the propeller. (Propeller.other_blades)
Create and return a hub object for the propeller. (Propeller.hub)
Create and return a cylindrical shape representing the outer part of a hub. (Propeller.hub.outer_hub)
Create and return a cylindrical shape representing the inner hole of the hub. (Propeller.hub.hub_inner_hole)
Create a subtracted solid from the outer_hub and hub_inner_hole. (Propeller.hub.hub_solid)
Create a filleted solid from the subtracted solid of two concentric cylinders. (Propeller.hub.hub_solid_filleted)
Create and return a solid fusion of the hub and other blades of the propeller. (Propeller.propeller_solid)
Create a subtracted solid from the propeller solid and the hub inner hole. (Propeller.propeller_solid_final)
Create and scale a 3D model of a propeller. (Propeller.scaled_propeller)
Create and return a Blade Element Theory (BET) analysis of a scaled propeller. (Propeller.bet_analysis)
Compute and return the efficiency of the propeller using blade element theory analysis. (Propeller.JK_eta_from_bet_analysis)

METHOD VS. CLASS CONTEXT MODE

To showcase the difference between the method and class context mode, Table 8.1 below presents a selection of ten paired responses that best illustrate the impact of using different levels of context when querying the LLM. These example were obtained from a test with the Modular UAV application on the JK_eta_from_bet_analysis attribute.

Table 8.1: Comparison of LLM output: slot context vs. full class context

Response with slot context	**Response with full class context**
Compute r_hub by dividing d_prop by 2 and multiplying by 0.08.	Compute propeller hub radius.
Compute propeller diameter using thrust, density, limit, and coefficient.	Compute propeller diameter based on thrust coefficient and propeller limit.
Compute thrust coefficient based on propeller attributes.	Compute thrust coefficient for propeller.
Compute hover thrust using MTOW and num_arms.	Compute required thrust per motor for hovering.
Compute atmospheric density based on altitude.	Compute air density at altitude.
Compute CST coefficients for airfoil root and tip boundaries.	Compute CST coefficients for root and tip airfoils.
Create Airfoil object with given parameters.	Create root airfoil part using Airfoil class.
Create B-spline curve with points, deflection, and color.	Create B-spline curves for airfoil profiles.
Create LoftedSolid object with blade_profiles, max_degree 2, and mesh_deflection 1e-5.	Create a lofted solid blade geometry.
Create solid propeller with blades and hub.	Create solid propeller model.

The examples above highlight a number of **key differences**:

- **Level of Detail:** The responses generated with full class context tend to be more concise and conceptually complete, whereas the slot context responses are often more specific but sometimes lack higher-level meaning. For example, the slot context response explicitly describes the computation of the propeller hub radius formula, whereas the full class context response generalizes it to a direct statement of intent.
- **Contextual Awareness:** In cases where dependencies exist between different parts of the class, the slot context mode lacks awareness of how a method interacts with the broader system. For instance, when computing thrust coefficient, the slot context response only references “propeller attributes” generically, whereas the full class context response acknowledges the computation in a broader aerodynamic framework.
- **Abstraction vs. Precision:** The slot context responses frequently include implementation details such as numerical parameters, while the full class context responses provide higher-level descriptions of their purpose. This is evident in the responses for computing CST coefficients, where the slot context response emphasizes numerical constraints while the full class context response provides a broader description of airfoil characteristics.
- **Function Naming and Readability:** Responses in the full class context mode often align with human-readable descriptions of engineering concepts, whereas slot context responses may be more technical and specific to their immediate computation. This difference is apparent in how the creation of B-spline curves is phrased—the full class context response provides a clear description, while the slot context response includes additional implementation-specific details.
- **Response Efficiency and Generalization:** The full class context mode enables the LLM to infer general relationships, avoiding redundant descriptions across different method calls. This is beneficial in reducing unnecessary verbosity in responses and promoting a more structured representation of a knowledge-based engineering application.

Overall, while the slot context mode provides specific, implementation-heavy responses, the full class context mode offers a more holistic and structured understanding of the functions within the KBE application. This indicates that for applications requiring high-level process modeling, the full class context mode is the preferred approach.

KEYWORDED RESPONSE START

Similar to the analysis above, Table 8.2 present a series of paired examples between LLM output. In this case the difference is the use of the keyworded response start feature. These example were obtained from a test with the Modular UAV application on the JK_eta_from_bet_analysis attribute.

Table 8.2: Comparison of LLM output: keyworded_response_start True vs. False. Both cases used class method context mode.

Response with keyworded_response_start = True	**Response with keyworded_response_start = False**
Compute propeller hub radius.	The r_hub method calculates the hub radius of the propeller in meters.
Compute CST coefficients for root and tip airfoils.	The cst_root_tip method calculates the CST coefficients for the upper and lower root and tip airfoils.
Create root airfoil part using Airfoil class.	The b_r_a method creates an Airfoil object for the root airfoil of the blade.
Create a lofted solid blade geometry.	The my_blade method creates a 3D model of a blade using lofted solid geometry.
Create solid propeller model.	The propeller_solid method creates a 3D solid model of the propeller by fusing the hub and blade parts.

The comparison presented in Table 8.2 illustrates the impact of enabling the keyworded response start feature in LLM outputs. When the keyworded response start feature is disabled, the LLM produces short, functional summaries (e.g., “Compute propeller hub radius”), which are succinct and to the point. These responses focus purely on the action being performed without unnecessary elaboration. The non-keyworded responses tend to be longer and contain additional contextual information, making them more informative but also potentially harder to parse quickly. For technical documentation where clarity on function-level responsibilities is crucial, enabling keyworded response start is advantageous.

COMPUTATIONAL PERFORMANCE

An indicative overview of the computational performance of the REProcess prototype for various test cases is presented below in Table 8.3. Note that some cases, like “Modular UAV, eta” exhibited unexpectedly long load times related to occasional hardware performance fluctuations, but these are considered outliers that do not reflect systematic inefficiencies in the method. The attained level of performance meets the requirements for prototype.

Table 8.3: Computational performance of \ac{reprocess} prototype accross various test cases

Test case [KBE app, target slot name]	**Model loading time**	**Total time (load + run)**	**Average tokens/sec**
Primiplane, stability_margin	16 sec	1 min, 13 sec	61.18
Primiplane, left_wing	16 sec	43 min, 20 sec	67.68
Primiplane, right_wing	16 sec	48 min, 18 sec	92.5
Modular UAV, bet_analysis	16 sec	27 sec	13.23
ref_KBE_JK, cog	17 sec	33 sec	40.58
Modular UAV, scaled_propeller	17 sec	2 min, 41 sec	43.96
Primiplane, movable_faces	22 sec	1 min, 50 sec	81.62
Modular UAV, eta	4 min, 9 sec	8 min, 39 sec	100.8
Average	**46 sec**	**13 min, 22 sec**	**62.7**

8.3. REMAINING ISSUES AND LIMITATIONS

A list of encountered issues and limitations is presented below:

1. **Sporadic LLM misinterpretations:** Although not very common, some cases of source code misinterpretation were identified. For example:
   - Drone.propeller_main.main_blade.cst_sections (method context mode) standard CST coefficient method not mentioned, and description is objectively wrong. This is an example of LLM under performing in core task. Expected cause is a lack of context.
   - Drone.propeller_main.hub.hub_solid (class context mode) -Objectively wrong description. This is a case of instruction misinterpretation. It seems the LLM is mixing up hub_solid and hub_solid_filleted, which is defined directly below this part in the code. This is an example of a more complex LLM instruction (class context mode) going wrong.
2. **Nested slot referencing:** This term refers to the situation where the value of a lower-level slot is used in the definition of another slot, instead of the top-level slot itself. For example, in the underlined text of this source code statement: `weight_averaged_cog_x_loc = self.fuselage_weight_frac * self.fuselage.cog.x`. When this occurs, it poses a challenge in terms of correctly drawing connectors because it could mean lower level computations had to be performed of which the output is then passed back up to higher levels in the product hierarchy. Conversely, this can also result in node connectors that flow directly from higher level nodes to nodes that are multiple levels lower in the product hierarchy. It is debated whether this is a feature or a bug. On one hand, it is a more detailed and accurate representation of the application behavior. On the other hand, it leads to an increase in the overall amount of node connectors and makes the diagrams feel less organized because they do not follow the standard behavior where data to lower-level nodes always flows via the respective part nodes.
3. **Precedents analysis issue:** `get_precedents_tree()` method only includes inputs to a part in the output from `get_precedents_tree()` if they were **strictly** necessary for the computation of the target slot. This means an input that is provided to a part that is the target object does not by definition show up in the precedents tree, while intuitively this would be expected. For example when the `get_precedents_tree()` method is called on an attribute or sub-part of a type 2 target object, only the inputs and their precedents used to compute the target slot of the target object are included. This imposes challenges and limitation on the *completeness* of this analysis. However, this is a typical concern for dynamic analysis methods [49].

   Another issue related to the `get_precedents_tree()` method is that it does not work as expected for input slots. This was observed in both test cases using a lower-level input slot as the reverse engineering target. These findings indicate that the precedents related to particular input slots are not traced beyond the scope of their own class. In other words, preceding slots belonging to parent objects are not returned. This is an inherent issue of the ParaPy `get_precedents_tree` method that cannot be solved easily without their involvement. Fortunately, while this finding does present a limitation on feasible analysis targets, the impact is low because reverse engineering the workflows related to input slots has less practical, real-world applications. Reverse engineering the workflows corresponding to attributes and parts is much more relevant from a practical perspective.
4. **LLM context overload warning:** For the class prompt context mode, some LLM prompts (test case produced a warning that the input was larger than the advisable context length. These test cases were examined in additional detail. However, no deviating outputs were discovered. It was concluded the setting to throw this warning was set conservatively.

Despite these limitations, the REProcess tool successfully extracts structured process models for complex engineering workflows. Future work can refine certain aspects, such as dependency tracing and response accuracy, to further enhance model completeness. The next chapter will present the conclusions of this research, and outline recommendations for future improvements.

9 Conclusions & Recommendations

The rapid advancement of Generative AI (GenAI) has introduced transformative possibilities across various engineering disciplines, including software development, design automation, and model-based systems engineering. In the context of knowledge-based engineering (KBE), GenAI offers promising capabilities for automating tasks such as code generation, process abstraction, and reverse engineering. However, integrating GenAI into engineering workflows presents key challenges, particularly in dependency tracing, interpretability, and computational performance. Addressing these challenges is critical to leveraging AI-driven automation while ensuring model traceability, maintainability, and efficiency in complex engineering applications.

The aerospace industry, characterized by intricate engineering processes and stringent verification requirements, has increasingly turned to model-based approaches to enhance automation and knowledge integration. The Model-Driven Knowledge-Based Engineering (MDKBE) approach presents a structured methodology that combines the advantages of knowledge-based engineering with Model-Based Systems Engineering (MBSE) principles. By structuring computational workflows in a model-driven manner, MDKBE aims to improve efficiency, standardization, and adaptability in aerospace engineering applications.

A central challenge within MDKBE is the lack of explicit workflow documentation in KBE applications. While KBE systems effectively automate complex design processes, they often result in workflows that are difficult to trace and interpret. This limitation hinders system verification, application reuse, and broader integration with MBSE frameworks. The ability to automatically extract structured process representations from these applications is therefore essential for improving model transparency and lifecycle management.

9.1. The REProcess Tool and Research Questions

Given the rapid evolution of GenAI, Large Language Models (LLMs) have emerged as a promising tool to address these challenges. LLMs have demonstrated strong capabilities in understanding and generating structured representations of software, making them a potential solution for automating process model extraction in KBE applications. However, due to the novelty of this approach, significant uncertainties remain regarding the effectiveness, reliability, and practical feasibility of using LLMs for reverse engineering KBE workflows.

To explore this potential, this research focused on developing a novel method for reverse engineering KBE applications by leveraging dynamic analysis, graph transformation techniques, and LLM-based source code abstraction. The primary objective was to advance process model extraction and representation, bridging the gap between software engineering and model-based approaches in aerospace KBE systems. This led to the conceptual design and prototyping of the REProcess tool, which implements core reverse engineering functionalities.

The research was guided by three main questions:

1. What are the system specifications for the envisioned REProcess tool?
2. To what extent can state-of-the-art LLMs be used to reverse engineer a process model that captures the engineering workflows embedded in KBE applications?
3. How can a practically useful reverse engineering tool be built around the LLM-based reverse engineering method?

9.2. ADDRESSING THE RESEARCH QUESTIONS

9.2.1. SYSTEM SPECIFICATIONS FOR THE REPROCESS TOOL

The envisioned REProcess tool was conceptualized with a set of high-level specifications to ensure its practical utility in aerospace engineering contexts. The primary system requirements were as follows:

- **Process Model Extraction**: The tool should reconstruct structured process models from unstructured KBE applications.
- **Interoperability**: It should ensure compatibility with widely used modeling languages such as SysML and MBSE frameworks.
- **Scalability**: The tool must support various levels of detail, from high-level process overviews to detailed functional descriptions.
- **Visualization and Usability**: The system should provide clear and interpretable visualizations of the extracted workflows, aiding engineers in understanding and refining processes.

An investigation into the reverse engineering capabilities of LLMs, specifically related to (ParaPy) KBE application code, determined their capacity to abstract meaningful natural language representations of KBE source code. However, they lacked the capability to extract the processing flow directly from the source code. Building upon the insights, a high-level design of the REProcess tool was conceptualized consisting of three core components:

- **Core Reverse Engineering Method**: This method reverse engineers individual workflows of the analyzed KBE application. It consists of a sequence of dynamic precedents analysis, graph transformation, and LLM-based source code abstraction.
- **Model Composer**: This component performs a high-level analysis to determine the set of workflows that should be reverse engineered. It integrates the separate outputs into a comprehensive process model and incorporates previously existing SysML model data where applicable.
- **XMI Writer**: Implements functionality to transform the internal representation of the process model into the SysML format and output it as an XMI file. This allows users to inspect and interact with the model in platforms like CATIA Magic Systems of Systems Architect.

These different components underscore the complexity of implementing a fully functional MDKBE system, requiring expertise in KBE systems, software engineering, model-based systems engineering, graph theory, generative AI, and cognitive science for usability testing and refinement.

9.2.2. REPROCESS PROTOTYPE IMPLEMENTATION

The REProcess tool effectively implemented the core reverse engineering method. It performs the following four core functionalities:

- **Dynamic precedents analysis**: Extracts dependency trees from KBE applications, using the `get_precedents_tree` method built into the ParaPy KBE system.
- **Graph Transformation**: Highly complex, heuristical algorithm that converts raw extracted precedents tree data into a structure representing engineering workflows captured in the process model.
- **LLM-Based Abstraction**: Applies generative AI techniques to summarize individual steps from the reverse engineering workflows and generate natural language text descriptions.
- **Visualization method**: Generate process diagrams to aid user interpretation and validation.

9.3. LIMITATIONS & OPEN ISSUES

1. Proper scientific **evaluation and testing** of the tool was identified as one the major remaining challenges. The concept of cognitive effectiveness was identified as a way to operationalize the tool's performance. However, applying this concept in an evidence-based validation and/or iteration approach was found be impossible due to practical reasons. The setup of such a user experiment requires a huge amount of work. First, a test has to be devised that enables quantifying the added value of the REProcess tool. This test should measure the improvement in a user's program comprehension, as well as the response time in order to evaluate the cognitive effectiveness of the diagrams. Naturally, this also requires a control group, performing the same test without access to the diagrams generated by tool. Creating such a test is already a big task. Then, a second challenge is to actually find a statistically significant amount of willing and motivated test participants. The combination of these practical factors prevented a proper, user-based validation of the current prototype.

2. This project was marked by a **lack of training data and rigorous performance/quality metrics**. However, despite these challenges, a functional and decently performing tool was developed capable of providing added value in the engineering process. This is a crucial conclusion because it goes against one of the fundamental arguments of data scientists and AI experts to disregard new project ideas and product concepts. Typically, they consider the lack of available training data and performance/quality metrics as a deal-breaker. However, this project shows that the recent improvements in foundational model capabilities are beginning to debunk this traditional notion.

   Furthermore, this opens up the possibility for a new development route. If a decent Minimum Viable Product can be created with an existing model, the resulting AI product can subsequently double up as a data collection method by collecting input from users. For example, like the thumbs up/down system from ChatGPT. With these approaches, a "proper" dataset can be created to train or fine-tune a dedicated LLM. This approach turns the traditional Machine Learning workflow on its head, and therefore represents a **paradigm shift** for many developers of AI systems.

3. **Testing challenges:** A big challenge related to the testing of the LLM performance in the REProcess prototype is the need for "real-life" examples. These are required to properly assess the interpretative capability of the LLM. Simplified test cases, often used to test other aspects of KBE systems, are not sufficient because a meaningful context is very important for an LLM to be able to properly interpret the source code. When the source code contains an equation that can be recognized and identified by the LLM, this provides valuable information about the purpose of that piece of source code. For example, the source code of an Attribute called "re" might contain the equation to calculate the Reynolds number. Based on the parameter name alone the LLM might not have recognized that this attribute represents the Reynolds number. However, the equation(s) in the source code definition of this attribute provide this additional information, based on which the LLM can identify this. In doing so, the LLM is then also able to infer that this application has something to do with flows, making it likely that it models some type of aircraft or ship. Because an LLM uses this kind of logical inference to make sense of what it has to do and generate better responses, it is necessary to test the prototype with "real-life" applications of which the source code makes actual sense. In contrast, the typical testing approach for showcasing and/or testing certain features of KBE applications would be to use use extremely basic KBE applications that are written as succinctly as possible to only include the particular feature of interest. Unfortunately, based on the reasoning above, testing with such basic examples is deemed to lead to unrepresentative results regarding the LLM-based abstraction part of the prototype. To assess this aspect properly, well-written "real-life" KBE applications need to be used. This poses a challenge because only a very limited number of such applications is available for this research.

   Moreover, assessing the quality of the output is more difficult as well. For one, this is because no hard criteria such as metrics can be used. The quality of the generated descriptions has to be gauged manually with a judgment call. Consequently, an effort by the researcher to properly understand the analyzed application has to be made first before the output generated by the prototype tool can be properly assessed.

4. **A mismatch between the SysML modeling paradigm, and the dynamic behavior of KBE applications:** The flexibility provided by the dynamic behavior of KBE applications is a strength from a user/developer perspective, but it also introduces a big challenge for modeling/reverse engineering. It means that a

particular behavior observed during execution of the KBE app can't be *generalized* with 100% certainty. This is known to be an inherent challenge of reverse engineering programs that are written using a declarative programming style, but it becomes extra apparent when trying to generate a process SysML model of the reverse engineered application. The reason is that generalization is one of the core mechanics on which the SysML modeling language is built. For example, creating an action node and naming it " : initialize KBE application" means the node will specify the generalized behavior of initializing the app. This means when an action node with the same name (" : initialize KBE application") is created in a second, unrelated activity diagram of the same overarching model, it is automatically correlated and linked to the similarly-named node from the first activity diagram. As a result, all associated metadata like output data pins and underlying behavior is added to the node of the second diagram. Unfortunately, this modeling behavior is not desired in this situation because the goal is actually to define a different, unrelated scenario. This example illustrates the fundamental mismatch when using activity diagrams to model specific scenario's. Activity diagrams are meant to specify the generalized behavior of a system which is, by definition, not context-specific. On the other hand, the reverse engineering workflows are in fact instances of the system behavior, which **could** be highly context-specific. Here, the word "could" is key because in practice the observed behavior is very often not context-specific. This makes it very appealing to simply generalize single, reverse engineering workflows since it works for most cases. However, when it does not, the fundamental mismatch of using SysML activity diagrams to model context-specific behavior suddenly becomes troublesome.

There are some possible workarounds to mitigate this fundamental mismatch such as to always include context-specific pointers when formulating behavior names. For example, use " : Initialize KBE app for the geometry generation use case" instead of just " : Initialize KBE app". In addition, control flow elements could be used added to the diagrams to indicate multiple context-specific variations exist. Such solutions have pro's and cons of course. The main benefit is, of course, that the mismatch issue is resolved, which means SysML can still be used to correctly represent the process model. However, the diagrams will become more difficult to read (i.e. lower cognitive effectiveness) as a result of adding more text and/or (types of) model elements to the diagrams.

A concrete issue related to this discussion is the question whether the desired to merge two nodes where the exact same behavior occurs, but with different input and output values. A good example would be the generation of `root_airfoil` and `tip_airfoil` parts in the Primiplane app (e.g. test case R-2.1). Here, both of these parts are created twice: once for the `right_wing` part and another time for `vert_tail`. Here, is it desirable to represent each airfoil generation step once in the diagram, meaning the `right_wing` and `vert_tail` both refer to the same node for these steps, or twice (as is done now)? This question is left as an open research question, to be addressed in follow-up research. The overarching research topic to which this question belongs the definition of a detailed ontology for the reverse engineered process model. This would look similar to the previous work from Fernandes [12], but now focused specifically on the process model.

9.4. RECOMMENDATIONS

The following list summarizes the most important recommendations based on the information, results and experiences from this project:

9.4.1. RECOMMENDATIONS FOR IMPROVING THE REPROCESS PROTOTYPE

1. Developing a preliminary version of the model composer class would provide a lot of added value and would be quite easy to implement. The set of main application functions could simply be defined as all top-level slots from the main KBE application file. Then implementing a loop to execute the core reverse engineering method for each slot is very straightforward. Finally, the *compose()* function from the NetworkX library[1] would be a great and easy way to implement the model integration step. The only real issue with using this function is that it overwrites metadata when merging two graphs. Fortunately, this could be addressed by a simple function that combines the metadata from both graphs in dictionaries, and assigns these dictionaries as metadata to each respective model element instead. Implementation

[1] NetworkX documentation: https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.operators.binary.compose.html#compose

of these additional features would already allow the REProcess prototype to generate a *comprehensive* process model, instead of individual workflow diagrams.

2. **Computational performance improvements:** The following list of design improvement were identified that could be implemented in order to improve the computational performance of the REProcess system.

   (a) LLM batch processing

   (b) First compose the model (when generating an integrated process model) and identify all duplicate nodes. Subsequently, only run LLM for the set of unique nodes included in the entire composed graph.

   (c) Change management system + (vector) database to store and re-use previous LLM runs.

   (d) Use smaller LLM, but finetune this model to achieve comparable quality. Additional prompt iteration could also be performed to improve performance of smaller model.

   Note that these aspects would improve the computational performance of the LLM-based abstraction method. However, the `get_precedents_tree()` method was actually found to be the biggest bottleneck in terms of computational performance. This will have to be addressed by Parapy itself, since it concerns an API method which is part of the KBE system itself.

3. **Few-shot prompting:** this is a technique where several context-specific questions-answer pairs are included in the prompt in order to improve LLM performance. This technique is particularly useful in scenarios with limited labeled data. For the REProcess tool, it is considered to have huge potential in two aspects. Firstly, it can be used as a means to instruct the LLM how to deal with Parapy-specific syntax elements, such as kbeutils functionalities or DynamicType instances. Thereby, it presents a simpler alternative to generating a large dataset and finetuning the model. Furthermore, by providing node type-specific examples of answers (i.e. for inputs, attributes, type 1 parts, type 2, etc.) few shot prompting can be used as way to achieve a similar result as with the keyworded response start feature. Namely, the LLM can be steered to use a particular style of answer formulation.

   Note: the LangChain library[2] was identified as an interesing option for the implementation of more intelligent prompt engineering methods.

4. **Variable mapping cardinality:** Mapping cardinality refers to the mapping between nodes in the transformed precedents graph, and action nodes in the reverse engineered process model. A strict one-to-one mapping was implemented for the current REProcess prototype. However, implementing additional mapping cardinalities is a very interesting options. For example, a many-to-one mapping would work well in case of the left_wing and right_wing from Primplane. For these, it would make sense to represent them together as "generate wing geometry" at a higher level. Conversely, a one-to-many mapping would also be interesting to address the case where attribute slot definitions are super extensive. For example, the Asystor app features a single attribute of more than 1000 lines of code(!). In the current implementation, the LLM is instructed to summarize this entire definition in just 10 words. Naturally, this means the behavior must be massively oversimplified so it is captured well in the process model. These examples indicate the benefits of implementing a variable mapping cardinality.

   In terms of implementation, several options were identified which mostly vary in how much intelligence/responsibility is asked from the LLM versus using some heuristics to more strictly guide the process. On one end of the spectrum is the LLM-centric implementation. This would use a more open user instruction, such as "Described the functionality performed by this class method using a bulleted list". This leaves it up to the LLM to decide how many bullet points (i.e. action nodes) it uses. On the other hand, the strictly guided option would implement some heuristics to look for breakpoints or, oppositely, merge cues and format the prompts based on these. For example, a for-loop defined using more than 10 lines of code could be defined as a pattern that should be attributed a dedicated action node. Then a custom prompt would be generated where the LLM is instructed to focus solely on describing this particular for-loop. An example of a mid-range option would be to count the number of lines of code, and introduce slight prompt modifications. For example:

[2] https://www.langchain.com/

- if slot source code is less than 50 lines of code -> prompt: "generate a single description of the provided source code"
- if slot source code is 51-200 lines of code -> prompt: "generate a description of the provided source code, formatted as a bulleted list. Use a maximum of 3 bullet points".
- if slot source code is 201-500 lines of code -> prompt: "generate a description of the provided source code, formatted as a bulleted list. Use a maximum of 5 bullet points".
- etc.

5. **Implement additional output formats:** An attempt was made to generate outputs in other graph data formats than DOT, namely GML and GraphML. This would enable the opening the graphs in other interactive graph tools such as yED[3]. Unfortunately the associated NetworkX functions (`write_gml` and `write_graphml` respectively) were found to be incapable of dealing with the complexity of the graph data. Specifically, the complex data types attached as metadata, and the usage of any other punctuation mark than underscores in strings. There still is great potential in implementing this export functionality, because it allows using advanced graph manipulation tools like yED. However, it will require a significant effort to implement the necessary data transformation function that converts all important metadata in the activity graph to formats that can be handled by the output writing functions from NetworkX without losing important aspects. For example, simply converting a dictionary to its string representation is not possible when only underscores and alphanumerical characters are allowed.
6. **Partial visualization method improvements**:
   (a) A nice feature would be to generate two or more diagrams based on the provided user input instead of just one. This could be achieved by prompting the user to specify the classes to be included in the top-level diagram. Through the process of elimination this implicitly defines the lower-level classes as well. Therefore, their diagram(s) could also be generated. Determining how many diagrams are required could be done by checking whether the removal of all top-level nodes leads to one connected sub-graph, or multiple disconnected sub-graphs. In case of the latter, each disconnected sub-graph should be represented in a separate graph.

   (b) A simpler alternative to the current prompt, where users are asked to specify the individual classes to be included in the partial visualization, is to ask for a product tree level. Here, providing a value of 0 would mean only the nodes from the root object are included; a value of "1" would also include all nodes nested within each parts of the root object; etc. This is a more holistic approach to provide this input and could be easier and more user-friendly.

   (c) Finally, the partial visualizations could also be generated based on some heuristic algorithm. For instance, by looking at the amount of nodes in a diagram. An optimization could be implemented that breaks down the full activity graph into a set of smaller sub-graphs such that each diagram is closest to a target amount of nodes (e.g. 15-20 nodes per diagram). Thereby, each diagram would present roughly the same amount of information to the user. In order for this to truly apply, however, it would be advisable to first implement the variable mapping cardinality. Without this feature, the number of nodes could be a poor representation for the amount of information represented in a diagram.
7. Implement the current feature to generate an activity diagram within the current Parapy GUI. Could be introduced relatively easy as an extra option when right-clicking a particular part (in the viewport), or attribute (in the property window). The resulting SVG could be displayed in a pop-up window.

9.4.2. OTHER FOLLOW-UP RESEARCH OPPORTUNITIES

In addition to implementing improvements of the tool, several other interesting areas for future work are also identified:

1. **Dissemination of the REProcess prototype:** due to the established benefits of the current REProcess implementation, it can already support the design of KBE applications in its current form. This highlights the potential for making the tool widely accessibly and actively disseminating it.

[3] https://www.yworks.com/yed-live/

2. **Start collecting training data for training/fine-tuning custom LLMs on ParaPy** A labelled dataset is required, consisting of examplary question-answer pairs. To start, this would probably be high-quality natural language descriptions of ParaPy source code snippets (e.g. individual slots, or entire class definitions). A more advanced option would be to capture the knowledge model data insstead of the source code. Note that these custom LLMs can be trained for both model-to-code and code-to-model applications, which could be applied respectively for step 2 and step 4 of the proposed MDKBE approach.
3. Dedicate a follow-up research project to defining an exact ontology for the reverse engineered process model, in order to address the mismatch between SysML and the execution behavior KBE applications.
4. The Aerospace Engineering faculty should implement a structural solution to support novel AI-based research. Current facilities are lacking, and AI education and training of students and staff is required to stay at the forefront of developing advanced design methodologies.

9.5. FINAL REFLECTIONS

This research highlights the transformative potential of LLM-based reverse engineering in aerospace engineering. The REProcess tool provides a promising approach to capturing and structuring engineering workflows, paving the way for improved knowledge management and process optimization. While challenges remain—such as model accuracy, computational efficiency, and integration complexity—the findings of this thesis suggest that AI-powered reverse engineering can significantly contribute to the future of automated engineering analysis.

Future research should focus on refining the tool's capabilities, exploring domain-specific LLM fine-tuning, and conducting user studies to evaluate its effectiveness in real-world applications. Additionally, interdisciplinary collaboration between AI researchers and aerospace engineers will be crucial to ensuring the successful adoption of such tools within industry practices.

The development of the REProcess tool marks an important step towards AI-powered engineering, offering a glimpse into a future where generative AI enhances the efficiency and effectiveness of complex, safety-critical systems in aerospace and beyond.

PART II

COMMUNICATION THESIS

10

COMMUNICATION THESIS INTRODUCTION

10.1. COMMUNICATION CONTEXT

One of the main findings from the aerospace part of this thesis was the significant complexity associated with developing GenAI-powered software systems for domain-specific applications like the envisioned MDKBE approach. Knowledge and skills from a wide range of different domains is required. Continued development of the MDKBE approach, for instance, requires expertise from domains including engineering, programming, Knowledge Based Engineering (KBE), (generative) AI, modeling languages, and user interface design, among others. In order to cover this broad range of required expertise and tackle this comprehensive challenge, it was concluded that further development of the GenAI-powered software systems requires a closely collaborating, interdisciplinary team.

This demand for interdisciplinary collaboration extends far beyond the development of software tools for the MDKBE approach. It is, in fact, a defining characteristic of projects that seek to apply GenAI solutions to domain-specific problems. When GenAI is employed to (partially) *automate* work practices within the context of engineering *complex and critical systems*[1], it is particularly important that this collaboration goes well because the effectiveness of the collaboration is a key determinant for the quality of the developed solutions, as well as the required time and cost. However, these kinds of interdisciplinary collaborations are known to face a multitude of challenges [50].

10.1.1. CHALLENGES IN INTERDISCIPLINARY GENAI PROJECTS

Effective interdisciplinary collaboration in GenAI projects plays a pivotal role in determining project outcomes. Yet, achieving such collaboration is inherently difficult [51]. For example, differences in terminology and disciplinary jargon often impede communication, while conflicting objectives and priorities can undermine alignment. Additionally, trust issues between team members and unequal power dynamics further complicate the collaborative process [50, 52]. Good communication between the different experts involved is essential to overcome these inherent challenges.

In addition to general interdisciplinary collaboration challenges, these GenAI projects also face an array of challenges related specifically to the technology itself. Some, like the black-box nature of the technology and the potential for hallucination, were already discussing in chapter 2. Yang *et al.* [53] describe two additional key factors that make Human-AI interaction uniquely difficult to design together with end users: the uncertainty surrounding AI's capabilities, and the complexity of its output. These factors play a major role in the perception of GenAI technology. Lupetti and Murray-Rust [54] found that a lack of understanding can even lead to enchantment and lead to a magical perception of GenAI technology. Other, more high-level challenges also impact GenAI projects. An extensive overview of is provided in the literature review by Dwivedi *et al.* [55]. Some indicative examples they describe are presented below, providing an indication about the complexity faced by the interdisciplinary teams working on GenAI projects.

[1] Defined in chapter 2

- **Data issues** Data-related issues are among the most critical barriers to effective AI implementation. The lack of sufficient, high-quality data is one of the major reasons why projects with great ideas have to be abandoned. A common cause for this is that relevant data is frequently dispersed across many different teams, departments and organizations meaning it is too difficult to centralize and standardize the data.
- **Organizational and managerial challenges** AI integration is often hampered by internal organizational issues. A lack of in-house AI expertise, and fear under employees to be automated are common problems. Furthermore, organizations frequently lack clear strategies for implementing AI or fail to align AI solutions with their operational needs.
- **Legal, privacy and security challenges** The legal landscape surrounding AI remains underdeveloped, presenting a major hurdle. Ambiguities in accountability and governance create uncertainties for organizations deploying AI technologies. Furthermore, concerns over privacy, security, and compliance with regulations hinder progress.

10.1.2. SCOPE OF THIS THESIS PROJECT

The wide array of challenges faced by interdisciplinary teams working on GenAI also provides some insight to the many different types of stakeholders that play a role in these projects, such as data scientists, AI developers, managers, (engineering) domain experts, IT, legal experts, privacy and security officers, etc. Due to practical consideration (limited time and investigative capacity) this thesis will focus on the two most central and essential stakeholders in these collaborative GenAI projects: the domain experts and AI experts.

1. **Domain experts** who work on, or interact with complex, critical systems (e.g. aerospace, nuclear, medical technology, etc.). In addition, these experts are involved in AI-driven projects that aim to improve current work practices in terms of efficiency, quality, speed, safety, etc.
2. **AI experts** working on novel tools and methods in close collaboration with domain experts from group 1. Here, the term AI expert is used to refer to a variety of AI-specific professional roles. Depending on the context (academic or business domain), they might call themselves AI researchers, AI developers, AI engineers, or data scientists, for example.

   Note: When referring to these experts the term AI is used instead of GenAI because they typically have a broader background in AI, machine learning and/or data science, rather than being limited to GenAI specifically. Thus, even though the scope of this research focuses specifically on GenAI projects, these experts are referred to as AI experts to reflect their broader domain knowledge.

In addition to forming the core of the collaborative GenAI projects, another key part of the rationale to focus on these two types of stakeholders was that they have the most overlap with the knowledge and expertise gained during the aerospace part of this thesis. Essentially, both of these roles were fulfilled during the first part of this combined thesis. By focusing on the collaboration and communication between these two groups, this previous experience is put to optimal use.

10.2. PRIOR RESEARCH

A limited amount of prior research focused specifically in the interdisciplinary collaboration in AI projects between domain experts and AI experts. The most notable contribution is the work by Piorkowski *et al.* [56]. Their study investigated how AI developers navigate communication challenges when collaborating with multidisciplinary teams. Using the lens of shared mental models, they identified three key themes: knowledge gaps, trust building, and expectation management. The authors reported that mismatched expertise across roles often leads to significant communication barriers. For example, domain experts frequently struggle to understand complex AI concepts such as model algorithms and evaluation metrics, while AI developers find it challenging to translate these into actionable insights relevant to the domain experts' workflows. To address these gaps, AI developers in the study employed strategies such as informal education sessions, iterative clarification through examples, and artifact-based communication tools like slide decks and shared documentation. These efforts aimed to align team members' mental models, mitigate misunderstandings, and foster effective collaboration.

Unfortunately, as noted by Piorkowski *et al.* [56] themselves, their study also had a critical limitation: it focused exclusively on the perspective of AI developers, neglecting the viewpoint of domain experts. Other qualitative studies tend to follow a similar pattern. For example, Sadek *et al.* [57] performed four workshops involving 17

AI researchers to develop Responsible AI design guidelines, but no domain experts were involved. Furthermore, previous studies that do involve domain experts focus on other domains or other, "non-generative" types of AI. For example, the training neural networks to detect anomalies in medical images, and the interaction between doctors and this technology, is well-studied [58].

This leaves a significant gap in understanding the bi-directional dynamics of these collaborations. Addressing this gap and understanding interdisciplinary GenAI projects from the side of engineering domain experts is essential to better support interdisciplinary teams in their collaborative effort to develop novel GenAI-powered engineering systems.

10.3. PROJECT OUTLINE

The high-level objective of this thesis project is to contribute to the development of GenAI-powered engineering systems by improving the interdisciplinary collaboration between AI experts and domain experts through the design of a communication tool.

In order to develop a truly impactful solution first a deep understanding of the existing problem(s) must be obtained. Given the gap in the understanding of the specific interactions and communication problems between domain experts and AI experts, the initial focus of this thesis project was to perform an in-depth investigation of the interdisciplinary collaboration between AI experts and domain experts from *both* of their perspectives. This aligns well with the double diamond approach, which was therefore adopted [59]. A visualization of the double diamond approach is presented in Figure 10.1. The blue boxes below the figure represent the main findings. The green boxes describe the main activities that were carried out during these phases.

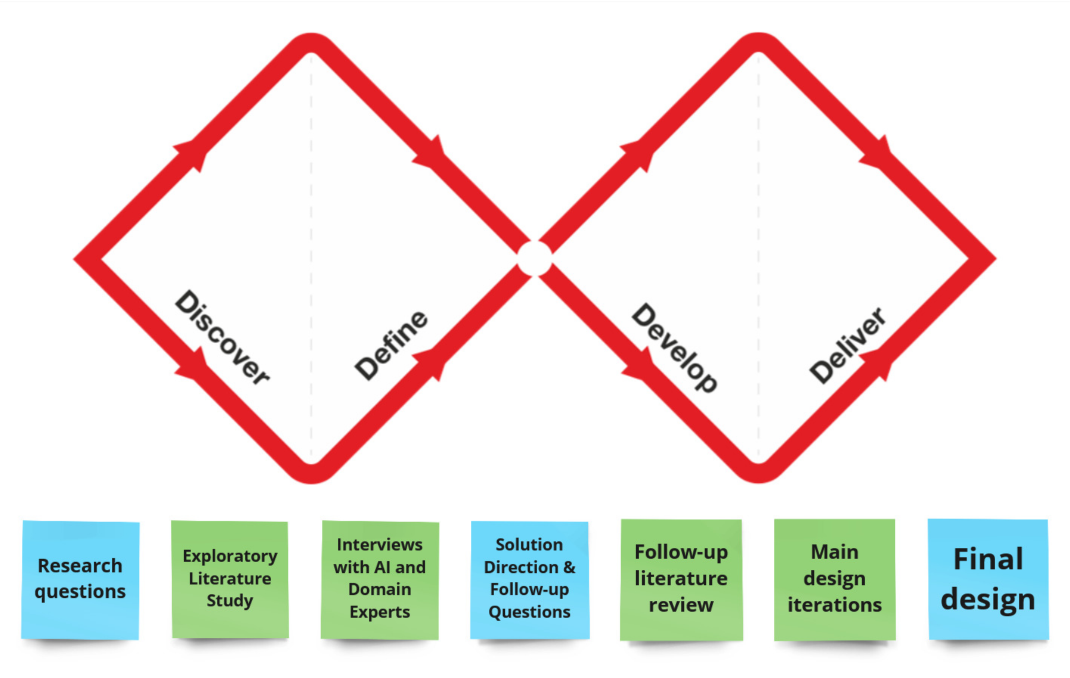


Figure 10.1: High-level project approach mapped to the double diamond. Figure adapted from [59].

10.3.1. RESEARCH QUESTIONS

The first two phases of the double diamond (discover and define) are aimed at understanding the problem. For this thesis project, this corresponded to answering the following research question:

RQ-1: IDENTIFY KEY CHALLENGES

What are the key collaboration and communication challenges between AI experts and domain experts in interdisciplinary AI projects, and how do these relate to each other?

A series of structured interviews with domain experts and AI experts was conducted to find an answer to this research question. This type of qualitative research is well-suited to obtain the deep insights about the problem that are required to develop a truly impactful solution. The choice for a *structured* interview approach was motivated by the need to compare the interviews conducted with a variety of participants from multiple organizations and backgrounds. Moreover, the structured approach facilitated identification of the most critical problems so that these could be prioritized during the subsequent design and implementation of the tool.

In addition to determining the key collaboration and communication challenges, the structured interviews were also used to determine a general solution direction. While preparing for the interviews, a list of solution ideas was collected on the side. Subsequently, these ideas were presented and discussed with participants at the end of the interviews. In this way, the structured interviews served a dual purpose by also providing an answer to the second main research question which is presented below. Strictly speaking, this represents a minor deviation from the double diamond approach as this normally postpones ideation and selection of solutions to the second phase. However, in this specific case it was possible due to the knowledge and experience gained during the aerospace thesis project. By also incorporating this second topic into interviews, the opportunity to collect input from AI experts and domain experts was used to the fullest extent.

RQ-2: DETERMINE SOLUTION DIRECTION

Which communication tool idea has the most potential to facilitate the collaboration between AI experts and domain experts?

PHASE 2:

In the second phase of the double-diamond approach (develop and deliver), the focus shifted from understanding the problem to developing the solution - in this case designing and implementing the selected communication tool idea. This corresponds to the following research question:

RQ-3: COMMUNICATION TOOL DESIGN

How to implement the selected communication tool concept such that it provides real added value?

Determining how to implement the tool such that it has real added value, will require determining the best application as well as exploring which similar tools already exist. Hence, research question 3 invoked two additional related questions (RQ-4 & RQ-5) which are presented below. Two main types of research activities were conducted to answers these. The first was another literature review, aimed at finding communication tools similar to the selected solution idea. The second type of activity constituted an iterative design process where the input from interviews, literature, and personal experience were used to develop iterations and presented these to professionals in order to collect additional input and feedback. The last of these major iterations was used to perform the concept validation. Through this iterative process, an answer to RQ-3 was obtained too.

RQ-4: DETERMINE DESIGN FOCUS

Which specific use case should the morphological chart be geared towards?

RQ-5: IDENTIFY SIMILAR EXISTING TOOLS

To what extent do similar tools exist, both in scientific and non-scientific contexts?

10.3.2. REPORT STRUCTURE

Before any interviews could be conducted, first the structured interview protocol had to be developed. To this end, a literature study was performed to identify common collaboration challenges and build a conceptual framework. These findings are presented in chapter 11. Subsequently, the results and analysis of the conducted interviews are presented in chapter 12. A solution direction was determined based on the interview results. This solution direction is presented in chapter 13, along with some additional follow-up questions that were prompted by the interview results and had to be answered next.

Phase two of the double diamond approach (develop and deliver) consisted mainly of two types of design-based research activities. The first was a second literature review, now aimed at finding communication tools similar to the solution idea that was selected based on the interview results. These findings are presented in chapter 14. The second type of activity was a series design iterations. Here the knowledge and experience from the previous aerospace thesis was combined with insights from the interviews and literature to iteratively develop a series of prototypes. More details about the approach, the main iteration cycles and the insights obtained from these activities is provided in chapter 15. Additionally, the final step of this approach, namely the concept validation sessions, are also discussed in this chapter. The finalized design of the communication tool is presented in chapter 16. Lastly, chapter 17 wraps up the science communication part of this combined thesis report with the conclusions, limitations and recommendations.

A separate, third part of this report presents a number of high-level findings and conclusions that resulted from the integration of the both parts of this combined thesis project. These are presented in chapter 18.

11

CONCEPTUAL FRAMEWORK AND INTERVIEW PROTOCOL

This chapter presents the findings from the problem exploration phase. The end goal of the activities from this phase was to create the protocol used to conduct the interviews with AI experts and domain experts. Three main research activities were conducted to achieve this, namely 1) reviewing existing literature about related research; 2) composing a conceptual framework by identifying and selecting suitable collaboration and communication theories/models; and 3) conceptualizing a range of communication tool ideas that could solve identified challenges and could be further developed during the second phase of the project. These activities are described in section 11.1, section 11.2 and section 11.3 respectively. Finally, the interview protocol itself is presented in section 11.4.

Note: a different research direction explored initially during this work about the role of visualization in making software workflow visualization more comprehensible, which identified a gap research gap but could not be explored due to practical limitations. Ssee Appendix D for details about this study.

11.1. EXPLORATIVE LITERATURE REVIEW

The literature review served a dual purpose: first, to identify studies that employed **similar, qualitative methods**, and second, to find state-of-the-art insights into **collaboration challenges and barriers** in interdisciplinary AI projects. The former was essential for informing the methodology and theoretical framework used in this research, while the latter provided valuable insights into the challenges faced by interdisciplinary teams working on AI projects.

11.1.1. PRIOR RESEARCH USING A SIMILAR APPROACH

To identify relevant studies, a series of targeted queries were performed in Scopus, focusing on interdisciplinary collaboration, communication, and mental models in the context of AI. An example of one of these queries is provided below:

> TITLE-ABS-KEY ("mental model" AND ((inter-disciplinary OR multi-disciplinary OR trans-disciplinary) AND (collaboration OR communication)) AND (survey OR interview))

KEY FINDINGS: PIORKOWSKI ET AL. (2021)

A particularly influential study was the work by Piorkowski *et al.* [56], which explored how AI developers address communication challenges in multidisciplinary teams. This study was deemed a "must-read" for its detailed insights into the practical challenges and solutions related to communication in AI projects.

The study by Piorkowski *et al.* [56] examined the communication challenges faced by AI teams and their stakeholders within the same organization but across different departments. Their research utilized the concept of **Shared Mental Models** and five realization principles from Scheutz *et al.* [60]. These are discussed later, in section 11.2. The study identified several key communication gaps in interdisciplinary AI teams:

1. **Knowledge gaps across roles**: These were addressed through bidirectional education and workshops designed to align domain-specific knowledge with AI concepts.
2. **Establishing mutual trust**: Regular contact and educational efforts were critical for building trust and fostering understanding.
3. **Expectation management**: Differences in mental models, such as mismatched expectations about AI performance, were mitigated through education and clear communication of goals and limitations.

The study also emphasized the importance of education as a key determinant of project success, highlighting the role of audience-adapted communication. Best practices included allocating longer planning periods, building shared documentation as both a conversational and knowledge management tool, and using iterative pilot projects to build trust and confidence among stakeholders. Design suggestions for improving collaboration tools included customizable documentation, better integration of data science and communication tools, and automating project updates through AI-powered tools.

11.1.2. KNOWN COLLABORATION AND COMMUNICATION CHALLENGES

The identification of previous work about collaboration and communication challenges in interdisciplinary (gen)AI project had a very practical purpose, namely to obtain a list of potentially relevant issues that could serve as a basis for discussion during the subsequent interviews. The main search query that was used is outlined below. Findings were limited to 2015 or later, only journals & books, and 5+ citations. This yielded 134 results. Subsequently, relevant articles were selected based on title and abstract.

> TITLE-ABS-KEY(((multidisciplinary OR interdisciplinary OR transdisciplinary OR “cross-functional”) AND (collaboration OR team OR communication) AND (barriers OR challenges OR enablers OR framework OR factors OR criteria)) AND ("generative ai" OR "artificial intelligence" OR ai) AND ((engineering OR development) AND (method OR tool OR approach OR framework)))

KEY FINDINGS: DWIVEDI ET AL. (2021)

The most important finding from the literature search aimed at identifying challenges was the work by Dwivedi *et al.* [55]. Based on an extensive literature review and input from experts, they collected a comprehensive overview of challenges and categorized these in 7 different categories. Their results are presented in Table 11.1 below. In the right-hand column, the challenges that were incorporated into the interview protocol are highlighted in bold. These were selected based on relevance for the specific context and scope of this thesis.

Table 11.1: AI Challenges from the literature [55]

AI Challenge	Details
Social challenges	Patient/Clinician education; **Cultural barriers**; Human rights; Country-specific disease profiles; **Unrealistic expectations towards AI technology**; Country-specific medical practices and **insufficient knowledge on values and advantages of AI technologies**.
Economic challenges	Affordability of required computational expenses; High treatment costs for patients; High cost and reduced profits for hospitals; Ethical challenges including **lack of trust towards AI-based decision-making** and unethical use of shared data.
Data challenges	**Lack of data to validate benefits of AI solutions**; **Quantity and quality of input data**; **Transparency and reproducibility**; Dimensionality obstacles; **Insufficient size of available data pool**; Lack of data integration and continuity; **Lack of standards for data collection**; Format and quality issues.
Organizational and managerial challenges	Realism of AI; Better understanding of health system needs; **Organizational resistance to data sharing**; **Lack of in-house AI talent**; **Threat of workforce replacement**; **Lack of strategy for AI development**; **Lack of interdisciplinary talent**.
Technological and technology implementation challenges	Non-Boolean nature of diagnostic tasks; Adversarial attacks; **Lack of transparency and interpretability**; Design of AI systems; AI safety; Specialization and expertise; Big data; Architecture issues and complexities in interpreting unstructured data.
Political, legal and policy challenges	Copyright issues; Governance of autonomous intelligence systems; **Responsibility and accountability**; **Privacy/safety**; National security threats from foreign-owned companies collecting sensitive data; Lack of rules of accountability in AI use; Costly human resources still legally required to account for AI decisions; Lack of official industry standards for AI use and performance evaluation.
Ethical challenges	**Responsibility and explanation of AI decisions**; Processes relating to AI and human behavior; Compatibility of machine versus human value judgments; Moral dilemmas and AI discrimination.

11.2. DEFINING THE CONCEPTUAL FRAMEWORK

Building the conceptual framework was an iterative process. A list of 48 communication theories and models obtained from the Communication Sciences website from UT Twente [61] was used as a starting point. This list was subsequently used as input to formulate a series of prompts for a SciSpace ChatGPT conversation used to narrow down the list to a selection of theories most suited for the context of this research. To enable SciSpace ChatGPT to make this assessment, an elaborate description of this research was included. Subsequently, the AI model was asked to suggest a list of additional theories that could be potentially useful to create the conceptual framework. Based on these initial filtering and exploration steps, a short-list of 15 communication theories and models was obtained. The complete prompts and corresponding results can be found in Appendix E.

The next steps were performed without AI assistance and consisted of verifying the findings based on their original sources and assessing the quality and usefulness of the theories. Selecting the specific theories and models to include in the conceptual framework was the real iterative part of the framework development. This part was performed in conjunction with the development of the interview protocol itself. The high-level approach that was followed was to review a current version of the interview protocol and determine its strengths and weaknesses. Subsequently, specific theories and models were selected that could be used to address identified weaknesses. A categorized overview of all theories and models that were considered at some point during this process is provided below. Given the overlap between these models, not all models were included in the final selection to prevent overload and redundancy. After several iterations between the interview protocol and conceptual framework, a final set of four were selected: Mental Model theory, the Collaborative Continuum, Relational Coordination theory, and the PMI Project Lifecycle Model. These are discussed in more detail in the remainder of this subsection.

- **Aligning teams**
  - Theory of Interactive Team Cognition [62]
  - Mental Model theory, including *Team Mental Models* [63] and *Shared Mental Models* [60].
  - Common Ground Theory: [62]
  - Expectancy Value Theory: [64]
  - Uncertainty Reduction Theory: [65]
- **Communication methods**
  - Media Richness Theory: [66]
  - Time-Space Matrix [67]
  - Computer Mediated Communication [68]
  - Contextual Design [69]
- **Collaboration maturity**
  - Collaborative continuum [70]
  - Relational Coordination Theory [71].
- **Technology adoption**
  - Diffusion of Innovations Theory [72]
  - Technology acceptance model [73]
  - Consumer Acceptance of Technology model [74]
- **Characterizing project phases**
  - V-model [11]
  - DevOps [75]
  - PMI Project Lifecycle Model [76]

RELATIONAL COORDINATION THEORY

Relational Coordination Theory, developed by Gittell and Ali [71], emphasizes the importance of high-quality communication and strong interpersonal relationships in coordinating work across teams. This theory focuses on how relationships influence performance in tasks that are complex and interdependent, highlighting the critical role of communication in such contexts.

This model is particularly useful for distinguishing between **intensive** and **sporadic** collaboration. Intensive collaborations are characterized by frequent, high-quality communication, shared goals, and mutual respect, while sporadic collaborations often involve more transactional and infrequent interactions. By examining these factors, the theory provides valuable insights into how teams interact and coordinate their efforts.

The decision to incorporate this model into the conceptual framework was based on several factors. As noted on its website[1]: "Relational coordination is particularly important for achieving these outcomes when work is highly interdependent, uncertain, and time constrained." This makes it highly applicable to the interdisciplinary collaboration between domain experts and AI experts under investigation in this research. Secondly, it is a validated model with a proven track record of analyzing coordination in complex projects. A **standardized and validated survey** of 7 questions is available which provides solid input for the interview protocol [77]. The use of categorical questions in this survey made this theory even more fitting, because this facilitates results comparison and prioritization of the most critical problems. Furthermore, in addition to its emphasis on communication, this theory also addresses other key collaboration aspects, namely shared knowledge, shared goals and mutual respect. By covering several areas, the inclusion of this theory simplified the conceptual framework by reducing the total number of theories and models required.

[1] https://heller.brandeis.edu/relational-coordination/about-rc/index.html

COLLABORATIVE CONTINUUM

The Collaborative Continuum Model conceptualizes collaboration as a spectrum, ranging from *Immuring* to *collaboration* and *integration*, with each stage reflecting varying levels of interaction, formality, and interdependence[78]. The various levels and their definitions are presented in Figure 11.1. This figure was obtained from the Teaglefoundation[2]

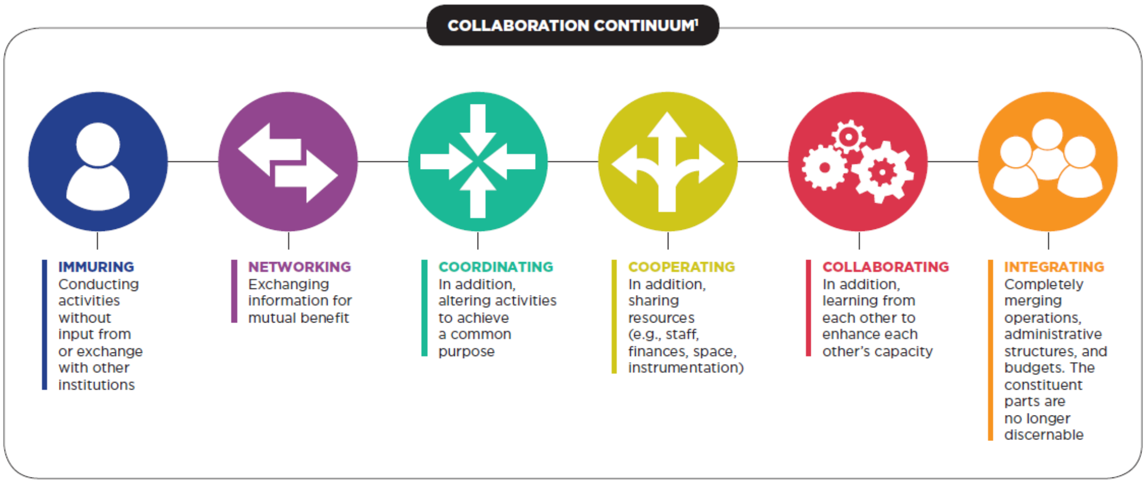


Figure 11.1: Collaboration Continuum model.

This model was particularly useful in this research for analyzing the varying levels of collaboration within interdisciplinary GenAI projects. It helped differentiate between teams that engage in occasional information sharing and those that work closely together to achieve shared goals, providing a structured framework to examine team dynamics in these projects.

PMI PROJECT LIFECYCLE MODEL

The PMI Project Lifecycle Model defines five key phases that represent the standard progression of a project: initiation, planning, execution, monitoring, and closing. This model provides a clear and structured framework for managing projects, ensuring that each phase supports specific goals. In the initiation phase, project objectives are defined, and feasibility is assessed. The planning phase focuses on developing detailed strategies, timelines, and resource allocations. Execution involves carrying out the planned activities, while monitoring ensures progress is tracked against established metrics. Finally, the closing phase formalizes the project's completion and reviews its outcomes [76].

This model was chosen as the basis for the interview protocol because of its simplicity and accessibility, making it ideal for guiding participants in identifying the current phase of their project. Unlike more complex models, such as the V-model, which includes up to nine detailed phases, the PMI framework avoids excessive elaboration that overcomplicates data collection.

THEORY OF SHARED MENTAL MODELS

The theory of Shared Mental Model (SMM) highlights the critical role of aligned knowledge, goals, and plans in facilitating effective collaboration and teamwork. The theory is operationalized through five core principles: consistency, reactivity, proactivity, coordination, and knowledge stability [60].

- **Consistency** Consistency involves maintaining stable, synchronized team knowledge by resolving conflicts caused by differing perceptions or incomplete information. This is achieved through mechanisms like conflict resolution and shared updates.
- **Reactivity** Reactivity is the ability to quickly detect and respond to unexpected changes in the environment or tasks. It ensures goals and strategies are adapted in real-time to maintain team effectiveness.
- **Proactivity** Proactivity focuses on anticipating issues and taking preemptive actions, such as identifying bottlenecks or offering clarification and assistance. This helps mitigate disruptions before they escalate.

[2] `https://www.teaglefoundation.org/Teagle/media/GlobalMediaLibrary/documents/resources/CollaborationContinuum.pdf`

- **Coordination** Coordination ensures teamwork through shared goals, clear task assignments, and transparent communication. It fosters open information sharing and cooperation to achieve collective objectives.
- **Knowledge Stability** Knowledge stability refers to recognizing the staleness of information and adjusting actions or communication accordingly. It ensures decisions are based on timely and reliable information.

11.3. COMMUNICATION TOOL IDEAS

Conducting the explorative literature research, developing the conceptual framework and iterating on the interview protocol also provided inspiration for potential solutions. Initially, these ideas were stored in a list that was kept on the side with the idea of revisiting them after the interviews had been conducted. In this approach, the interview results could be used to ideate additional solutions and then select one to work out in detail during the subsequent design phase. When the list started to form, however, this approach was reconsidered because it was deemed very interesting to use the list during the interviews themselves to gather participants' input on the ideas. The list of ideas that was incorporated into the interview protocol is presented below.

- **Design a morphological chart for AI-powered tools/methods:** A morphological chart could provide a structured way to explore and combine different attributes of GenAI-powered tools and methods. This visual framework allows teams to systematically generate, compare, and evaluate potential solutions based on key variables relevant to GenAI development.
- **Imagination sparker, e.g., curated list of inspirational projects or infomercial:** This tool would aim to stimulate creative thinking and innovation by showcasing a curated selection of successful GenAI projects, use cases, or promotional content. By exposing stakeholders to these examples, the tool could help inspire new ideas or highlight potential applications of GenAI in their domain.
- **Design a community for sharing resources/best practices (platform + interaction mechanics, promo, etc.):** This concept involves creating a dedicated platform where stakeholders can share resources, insights, and best practices related to GenAI development. Features such as discussion forums, collaborative tools, and promotional strategies would encourage interaction and foster a culture of knowledge exchange within the community.
- **Knowledge extraction approach aimed at domain experts:** This idea focuses on developing a tool or method for efficiently extracting and formalizing the tacit knowledge of domain experts. Such a tool would help bridge the gap between GenAI developers and domain experts by providing structured, accessible information to guide the development process.
- **Campaign to convince directors/managers to increase structural resource allocation to AI development processes:** This initiative would involve designing a targeted communication campaign to demonstrate the importance and long-term benefits of investing in GenAI development. By presenting compelling evidence and success stories, the campaign aims to persuade decision-makers to allocate more resources and support to these processes.

11.4. STRUCTURED INTERVIEW PROTOCOL

The interview protocol was designed according to a funneling approach, moving from broad, exploratory questions about the participant's AI-related projects to more specific questions addressing collaboration and communication challenges. This structure aligns with the research objective of identifying and analyzing key barriers to effective multidisciplinary collaboration. Below is an overview of the main blocks of questions, along with their purposes and connections to the conceptual framework.

1. **Background:** The purpose of these questions was to gather contextual information about the participant's current AI-related projects, their role in those projects, and the stakeholders involved. This block aimed to establish the engineering and organizational context, including the project phase and the nature of their collaboration with domain experts or other AI researchers. These questions helped to ground subsequent discussions in the participant's specific experiences.

   Link to conceptual framework: The five project phases from the Project Management Institute were used as response categories to question the current phase of the participant's main AI project.

2. **Collaboration questions:** This block focused on the nature and quality of the participant's collaboration with domain experts (Group 1) or AI researchers/developers (Group 2). Questions explored the role distribution, the level of collaboration, as well as rating the quality (scale of 1-10) and motivation of both sides (intrinsic vs extrinsic). In addition, this block aimed to establish the most relevant collaboration challenges/barriers, in terms impact and frequency, faced in practice within AI projects.

   Link to conceptual framework: The **Collaboration Continuum** was used as a scale for participants to characterize the current level of collaboration. This scale was also used to ask what collaboration level they would be striving for in the future. In addition, the **five PMI project phases** were used to ask if there was a particular project phase when this increased level of collaboration would be most beneficial.

   The list of **collaboration barriers** composed during the literature study (subsection 11.1.2) was used as the basis for the question about key collaboration challenges/barriers. The participants were asked to use this list to compose a top-3 of the most relevant and obstructive collaboration barriers in their project.

3. **Alignment questions:** These questions examined the extent to which the participant and their collaborators shared goals and knowledge, and felt mutual respect. The goal of these questions was to determine the alignment, or relational coordination, between project members with different disciplinary backgrounds and to identify the most critical are to improve.

   Link to conceptual framework: These questions were based on the **Relation Coordination Survey**, but adapted to the specific context of this project. They were also reformulated as statements, to be answered on a five-point Likert scale ranging from completely disagree to completely agree. Moreover, the questions about shared knowledge and mutual respects were each broken down into two questions to capture the perspective from each side separately. For example, the standard relational coordination survey question about shared knowledge was: "How much do people in these roles know about your role in [work process]?". The corresponding statements incorporated in the interview protocol were: 1) "The AI-specific knowledge of domain experts is sufficient to understand the project work"; and 2) "The domain-specific knowledge of AI professionals is sufficient to understand the project work".

4. **Communication questions:** This block explored the frequency, timeliness, accuracy, and problem-solving orientation of communication between the participant and their collaborators. Additional questions examined the use and reuse of standardized communication artifacts (e.g., documentation, diagrams) to facilitate information exchange. Participants were also asked to reflect on communication gaps and strategies for addressing miscommunication.

   Link to conceptual framework: Heavily based on the **Relational Coordination Theory's**.emphasis on effective communication patterns, while also addressing practical tools (e.g., artifacts) that support collaboration.

5. **Solution ideas:** This section sought to identify potential solutions to improve collaboration and communication. Participants were asked to provide feedback on specific design ideas (e.g., morphological charts, knowledge-sharing platforms) and to propose additional solutions. This block emphasized practical, actionable insights for enhancing multidisciplinary teamwork and GenAI tool development.

   Link to conceptual framework: Supports the study's applied focus by integrating participants' feedback into the iterative development of tools and strategies to address collaboration barriers.

11.4.1. COMPLETE INTERVIEW PROTOCOL

The finalized interview protocol is presented in Table 11.2. As stated before, this final interview protocol was developed iteratively. The final step of its development was the first interview itself, which was conducted with AI-2. Aside from gathering data, this sessions was also used to test and verify the protocol itself. Overall, the protocol was found to perform well. Several minor tweaks were made, such as the inclusion of some additional wrap-up questions and making the categorization more explicit so interviewees could better understand the progression and anticipate future questions.

Table 11.2: Complete interview protocol used for structured interviews

Nr	Questions	Prio	Related concepts & theories
	Introduction & background (10 min)		
1	Could you briefly describe the main project you are currently working on related to (Gen)AI?	H	
2	How does this project aim to improve or transform current work practices?	H	
3	Who are the end users of the developed AI tool? (overlap with group 1, internal/external)	H	
4	In what phase is your main AI project? *[Initiation, planning, execution, monitoring, completion]*	H	PMI project phases
5	What is your role in the AI project?	H	
6	Is your work data-, model-, or product-centric?	H	Mentioned in Piorkowski *et al.* [56]
7	How many AI researchers/developers (group 2) and domain experts (group 1) are involved?	H	
8	Are there other stakeholders that play a key role in the AI project?	H	
	Collaboration barriers (15 min)		
	The following questions are about your collaboration with domain experts/AI researchers		
9	Could you characterize the typical domain expert/AI researcher involved in the project?	H	
10	How would you describe the role distribution (yours vs. theirs) in this collaboration? (responsibilities, time investment)	H	
11	How would you define your level of collaboration with them? *[Immuring - networking - coordinating - cooperating - collaborating - integrating]*	H	Collaboration Continuum
12	How would you rate the overall quality of your collaboration with domain experts/AI researchers on a scale from 1-10?	H	
13	Do you feel intrinsically or extrinsically motivated to collaborate with them [DE or AI]? Please explain. *[Highly intrinsic - Mostly intrinsic - Mixed - Mostly extrinsic - Highly extrinisic]*	M	Relates to adoption attitude + power dynamics
14	Do you feel they [DE or AI] are more intrinsically or extrinsically motivated to collaborate with you? Please explain. *[Highly intrinsic - Mostly intrinsic - Mixed - Mostly extrinsic - Highly extrinisic]*	M	Relates to adoption attitude + power dynamics
15	If you feel a need for more collaboration with domain experts during the development process, what level of collaboration would you strive for? Please explain. *[Immuring - networking - coordinating - cooperating - collaborating - integrating]*	H	Collaboration Continuum
16	At what stage of the development process do you think this collaboration will be most beneficial? Please explain. *[Initiation - planning - execution - monitoring - completion]*	H	PMI project phases

Table 11.2: Complete interview protocol used for structured interviews

Nr	Questions	Prio	Related concepts & theories
17	From the following list of potential collaboration barriers related to AI projects, please select a top 3 of the ones most obstructive to your collaboration with domain experts: • *Differences in working culture* • *Differences in mental model* • *Threat of replacing human workforce* • *Unrealistic expectations towards GenAI technology (high or low)* • *Insufficient knowledge on value/advantages of GenAI technology* • *Lack of trust towards AI-based decision making* • *Transparency and reproducibility of output* • *Legal responsibility and accountability* • *Negative associations with AI due to bad prior experience with non-related tools* • *Quantity and/or quality of training data* • *Lack of data to validate benefits of GenAI technology* • *Lack of inhouse AI talent* • *Lack of multidisciplinary talent* • *Lack of AI development strategy* • *Restrictive AI laws (national + EU)* • *Limited computational resources* • *Other...?*	H	Dwivedi *et al.* [55]
18	Please explain your choices		
	Team alignment (5 min) *The following statements are about your alignment with the domain experts involved in your AI project*		
19	The domain experts share my goals for the AI project. *[Completely disagree - Mostly disagree - Neither - Mostly agree - Completely agree]*	H	Relational Coordination Survey, Coordination (SMM)
20	The AI-specific knowledge of domain experts is sufficient to understand the project work. *[Completely disagree - Mostly disagree - Neither - Mostly agree - Completely agree]*	H	Relational Coordination Survey, Coordination + Consistency (SMM)
21	The domain-specific knowledge of AI professionals is sufficient to understand the project work (of domain experts). *[Completely disagree - Mostly disagree - Neither - Mostly agree - Completely agree]*	H	Relational Coordination Survey, Coordination + Consistency (SMM)
22	The role of AI professionals within the project is respected. *[Completely disagree - Mostly disagree - Neither - Mostly agree - Completely agree]*	H	Relational Coordination Survey, Consistency (SMM)
23	The role of Domain Experts within the project is respected. *[Completely disagree - Mostly disagree - Neither - Mostly agree - Completely agree]*	H	Relational Coordination Survey, Consistency (SMM)
24	From these aspects (shared goals, shared knowledge, mutual respect), which one is the biggest challenge? Please explain and provide example.	H	
	Communication (15 min) *The following questions are about the communication between you and domain experts involved in your main AI project. When answering, please consider all forms of communication (meetings, phone calls, emails, etc.)*		
25	What are the main ways you communicate? (e.g. [Meetings (physical/online?), email, documentation, scrum board, ...])	M	

Table 11.2: Complete interview protocol used for structured interviews

Nr	Questions	Prio	Related concepts & theories
26	Who generally takes the initiative to communicate? *[Always me - Mostly me - Balanced - Mostly them - Always them]*	M	Relates to collaboration motivation, power dynamics
27	How frequently do you and domain experts communicate about your AI project? *[Not nearly enough - Not enough - Just the right amount - Too often - Much too often]*	H	Relational Coordination Survey
28	How timely is the communication between you and domain experts about your AI project? *[Never on time - Rarely on time - Sometimes on time - Often on time - Always on time]*	H	Relational Coordination Survey
29	Are you able to communicate accurately with domain experts about the project work (i.e. clear, free of misinterpretations)? *[Never - Rarely - Sometimes - Often - Always]*	H	Relational Coordination Survey
30	When problems occur within the project, do the domain experts tend to blame others or work with you to solve the problem? *[Always blame - Mostly blame - Neither blame nor solve - Mostly Solve - Always solve]*	H	Relational Coordination Survey
31	Out of the four aspects of communication, which one is the biggest problem in your project? Please explain and provide an example	M	
32	In what direction do you experience the greatest communication gaps? *[Overwhelmingly from DE to AI, Mostly from DE to AI, Similar in both directions, Mostly AI to DE, Overwhelmingly from AI to DE]*	L	
33	To what extent are you able to anticipate communication problems and proactively address them? *[Never - Rarely - Sometimes - Often - Very often]*	L	Proactivity (SMM)
34	To what extent are you able to recognize and address miscommunication while they occur? *[Never - Rarely - Sometimes - Often - Very often]*	L	Reactivity (SMM)
35	How often do you (re)use standardized communication artifacts (slidedecks, documentation, youtube links, etc.) to share information with domain experts? *[Never - Rarely - Sometimes - Often - Very often]*	L	Piorkowski *et al.* [56]
36	Can you share and/or describe some examples?		
37	To what extent do you (re)use standardized communication artifacts (e.g. intake forms, miro/canva boards, surveys, etc.) to collect information from domain experts? *[Never - Rarely - Sometimes - Often - Very often]*	L	Piorkowski *et al.* [56]
38	Can you share and/or describe some examples?		
	Ideas for improvement (15 min)		
39	If you could propose one improvement to the collaboration process between your team and domain experts, what would it be?	L	
40	In your opinion, which option would be most useful to improve the development of novel AI-powered tools & methods, and why? • *Motivating more domain specialists to become involved* • *Improve communication from AI developer to domain expert* • *Improve communication from Domain expert to AI developer*	H	
41a	Which idea(s) do you like best? And why? *Full list of design ideas:* • *Design a morphological chart for AI-powered tools/methods* • *Imagination sparker, e.g. curated list of inspirational projects or infomercial* • *Design a community for sharing resources/best-practices (platform + interaction mechanics, promo, etc.)* • *Knowledge extraction approach aimed at domain experts* • *Campaign to convince directors/managers to increase structural resource allocation to AI development processes*	H	List of design ideas

Table 11.2: Complete interview protocol used for structured interviews

Nr	Questions	Prio	Related concepts & theories
42a	Any additional comments regarding these options?	H	List of design ideas
41a	*Current favourite design idea is to design a morphological chart for AI-powered tools/methods:* • *Quick & convenient overview of key design decisions* • *Raise awareness of possibilities* What are your initial thoughts on such a tool? Added value? Existing similar tools?	H	Used instead of 41a during interviews with AI-1 and AI-3
42b	Which key design decisions do you think should be included?	H	Used instead of 42a during interviews with AI-1 and AI-3
	Wrap-up (5 min)		
43	Any final comments or insights?	H	
44	Would you like to stay in the loop for future updates?	H	Test interview with AI-2
45	Would you be open to testing & validation of the tool/method?	H	Test interview with AI-2
46	Do you know other interesting interviewees for my research?	M	Test interview with AI-2

12

INTERVIEWS WITH AI EXPERTS AND DOMAIN EXPERTS

This chapter is presents interview process and results. The first section describes the methodology used for selecting participants, conducting the interviews, and processing the responses. This is followed by a presentation of initial, high-level interview findings in section 12.2 outlining the most significant collaboration and communication challenges. Subsequently, section 12.3 provides a more in-depth analysis of these challenges, examining their underlying causes and interdependencies. section 12.4 presents the insights gathered from participants regarding potential solutions, contributing to the development of a structured communication tool to support interdisciplinary GenAI projects. Finally, a reflection on the interview process is provided in section 12.5.

12.1. METHODOLOGY

12.1.1. HUMAN RESEARCH ETHICS

In preparation of conducting the interviews, a number of steps were performed to comply with TU Delft ethical standards for conducting human research[1]. A data management plan was created, and a risk assessment and mitigation plan was setup. Subsequently, an the interview participant consent form was created according to these plans. This consent form is presented in Appendix G. A formal application was sent to TU Delft's Human Research Ethics Committee, which gave their approval to conduct the study.

12.1.2. PARTICIPANTS

The following criteria were used to select interview participants:

- **Balanced representation:** Efforts were made to ensure a balance between the number of AI experts and domain experts to equally capture both perspectives.
- **Active interdisciplinary research:** Participants were required to be actively engaged in an interdisciplinary project where the integration of AI knowledge and domain expertise is a core component.
- **Project diversity:** Instead of conducting a focused case study with participants from a single AI project, the aim was to interview professionals across a range of projects and contexts (academia, government, and industry). This broader approach was taken to help determine whether certain challenges are widespread or specific to particular projects.
- **Pairwise comparison**: For each included AI project, the aim was to interview both an AI expert and a domain expert. This approach ensures optimal comparison between the two perspectives under similar conditions. If interviewing both was not possible, attempts were made to recruit two AI experts or two domain experts to maintain consistency.

[1] https://www.tudelft.nl/over-tu-delft/strategie/integriteitsbeleid/human-research-ethics

INTERVIEWEE RECRUITMENT APPROACH

Two primary avenues were pursued to gather interview participants. The first involved reaching out to relevant professionals from the researcher's personal network. Conveniently, the aerospace part of the thesis project was initiated under the DEFAINE project, which also included various companies and research organizations. These connections were now approached to participate in the interviews. In addition, they were asked to suggest additional participants from their professional network to broaden the pool.

The second approach was to search for participants within TU Delft's AI Labs [2]. These AI Labs were founded specifically with the intention of facilitating interdisciplinary research between researchers from different faculties. A total of 24 AI Labs exist, each focusing on a specific application of AI. The descriptions of all 24 labs were reviewed to make an initial selection of labs that fit within the scope of this research. Subsequently, the individual research projects carried out within each of these labs were assessed in order to obtain a short-list of researchers and lab directors. These were contacted via email.

A one-pager describing the research and providing some indicative example questions was created and attached to the emails that were sent out to potential participants. Two versions of these

FINAL SET OF PARTICIPANTS

A final set of six participants was obtained. Three of these were AI experts, and the other three domain experts. They are divided over four different projects, and work in five different organizations. Two pairs working on the same project were found: one AI expert and domain expert pair (same organization), and one pair of two domain experts (different organizations). A brief overview of project descriptions is provided below:

- Developing text-to-CAD tool for designers and engineers.
- Developing a chat platform for internal use by employees. Broadly similar to ChatGPT-4o (i.e. agentic, connected to external tools and databases) but custom-built and with unique, domain-specific capabilities.
- Developing a (partially) autonomous robot to support emergency responders.
- Developing tools to (partially) automate engineering workflows and connect/integrate various domain-specific engineering tools (parametric design, analysis & simulations, etc).

ADDITIONAL INSIGHTS FROM PARTICIPANT SEARCH

Aside from recruiting interview participants, it was found that the exercise of reviewing a wide array AI projects also helped to sharpen the research scope. For instance, medical technology was initially considered as one of the most interesting domains to include in this research. At first glance, this field seemed to fit very well with the descriptive characteristics of being "complex and critical". However, after reviewing various project descriptions, it became evident that most AI applications in this domain primarily involve training neural networks for data analysis tasks, such as processing medical images to identify tumor cells.

Although promising, these projects were found to be a poor fit for the intended research focus but the reason for this misalignment was not immediately clear. After further deliberation, it was concluded that these projects typically require minimal collaboration between medical domain experts and AI experts. Namely, once a dataset of (labeled) medical images is provided by domain experts, AI specialists can often handle the remainder of the process independently. Such collaborations are *multi*disciplinary rather than *inter*disciplinary, and thus fall outside the scope of this research.

As a result, the search criteria were refined to emphasize projects that use AI to automate domain-specific work processes. This approach inherently requires detailed domain knowledge and necessitates close, interdisciplinary collaboration between AI experts and domain experts, thereby aligning very well with the research scope and objectives.

12.1.3. CONDUCTING INTERVIEWS

All interviews were conducted online, using Microsoft Teams. A slidedeck was created with the questions and response categories which was presented throughout the interview via screen sharing. An example of one of the slides that was used is shown in Figure 12.1.

[2] `https://www.tudelft.nl/en/ai/tu-delft-ai-labs`

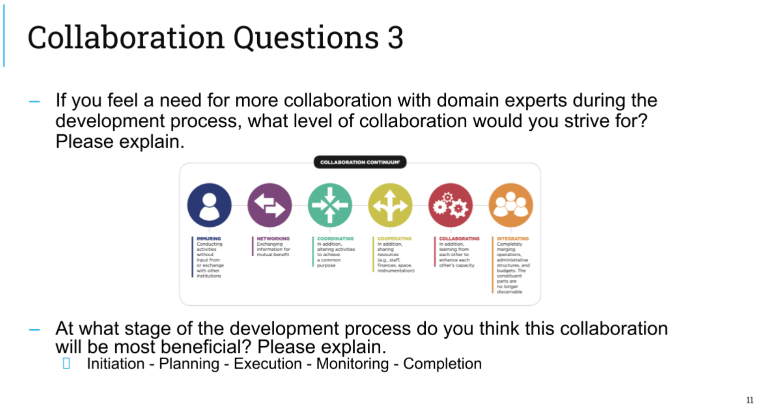


Figure 12.1: Example slide presented via screen sharing while conducting the structured interviews

While a time slot of 90 minutes was requested of all participants, only two participants could actually free up this amount of time for their interview. The four other participants had to be interviewed in 60 minutes. To make optimal use of the participant's available time, the questions were categorized based on priority. This prioritization was used for time management during the interviews. When falling behind on schedule, lower priority questions were skipped to ensure that at least the key questions of each topic could be answered.

- **Mandatory:** The consent form check, research introduction and background questions were considered mandatory elements of the interview. They lay a foundation for the rest of the interview and are required to make sense of all the other answers and place them in the proper context.

  Note: In case project members of the interviewee had been interviewed previously, an attempt was made to reduce the time spent on background questions. Instead of discussing the full set of background questions, a brief overview of their AI project was presented based on the understanding gained during the previous interview with their colleague. Then the interviewee was asked if their project was understood correctly. If confirmed by the interviewee, the background questions about their project were skipped. Personal background questions, like *"What is your role in the project?"*, were still asked.

- **Top priority:** The questions that directly addressed the research questions were assigned the highest priority. This includes the following (groups of) questions:
  - Questions based on the Collaboration Continuum, in particular the question about desired level of collaboration.
  - The question to compose a top 3 of most obstructive collaboration barriers.
  - Questions based on the Relational Coordination Survey (both the alignment and communication aspects).
  - Questions about improving collaboration with a communication tool. More specifically the best direction/purpose, and discussing the list of ideas with an emphasis on discussing the idea(s) with the highest potential.
  - Question to rate the overall collaboration quality.

- **Medium priority:**
  - Questions about intrinsic vs extrinsic motivation to collaborate.
  - Question about methods of communicating, and communication initiative.

- **Low priority:** Detailed questions about specifics communication aspects.
  - Direction of communication gaps.
  - Ability to anticipate miscommunications and react to them.

- (Re)use of standardized communication artifact.
- Communication tool ideas from participants to improve collaboration

12.1.4. CAPTURING AND PROCESSING INTERVIEW DATA

Processing the interview data was performed according to the following steps:

1. The interview recordings were reviewed and the auto-generated transcripts were cleaned up using Microsoft Stream (online). This is the default platform for watching back Microsoft Teams meeting recordings, and has the particularly useful feature where a text segment of the auto-generated transcript can be clicked to skip to that part of the recording. The guiding principle for cleaning up the transcripts was to produce text that was completely accurate with respect to the interviewees' statements while also being understandable in a standalone document. With this in mind, the following corrections were made:
   (a) Correcting misinterpreted words
   (b) Adding/correcting punctuation
   (c) Removing unfinished (parts of) sentences
   (d) Removing repeated words
   (e) Removing verbal fillers
2. While watching back the recordings during the previous step, a second activity performed in parallel was to collect the answers to all categorical questions in a large spreadsheet. In addition to these categorical answers, quotes with insightful explanations and examples were also added to their corresponding question topic. Notably, when interesting remarks were made about a particular question topic while answering a completely different question, these quotes were also stored according to topic. In such cases, the timestamps were also added.
3. In parallel with cleaning up the transcript, a systematic collection of interview responses was carried out. Answers to all categorical questions were documented in a comprehensive spreadsheet. Alongside these categorical responses, quotes that provided insightful explanations or examples were also captured in this spreadsheet. These quotes were stored according to topic. Thus, when an interesting remark about a particular topic was made while answering an unrelated question, it is stored as a quote under the questions that actually addressed that topic. In such cases, the timestamp is added too for easy reference. This dual approach ensured both structured data collection and preservation of nuanced, qualitative insights for further analysis.
4. Due to the substantial amount of categorical questions incorporated in the interview protocol, the completed spreadsheet provides a very clear overview of the results. Combined with the activity of reviewing all interview recordings, a very good understanding of the results was obtained. These pieces of information were used to derrive and formulate a number of preliminary high-level conclusions. Specifically, the key collaboration and communication challenges were identified, and a particular solution idea was selected.

12.2. HIGH-LEVEL INTERVIEW RESULTS

12.2.1. CHARACTERISTICS OF AI-DE COLLABORATIONS

Based on the background questions and overall understanding obtained from the interviews, the following characteristics of the collaborations between AI and domain experts were found:

- **Small teams:** With the exception of one AI expert, all AI projects were being performed in small core teams, consisting of 4-5 people.
- **Early project phase:** Generally, the projects were in the initiation or planning phase (5x). One interviewee was working on multiple projects in many phases.

- **Internal focus:** The generative AI systems that were being developed within the projects of the interviewees were typically intended for internal use. Moreover, the involved domain experts were often representative of the end users.
- **Currently at 'collaboration' level:** Most respondents placed themselves relatively highly on the Collaboration Continuum scale, namely at 'collaborating' (4x) or 'cooperation' (1x). One outlier placed themselves significantly lower, at the 'networking' level.
- **Overall very positive about collaboration quality:** The average grade given for overall quality of the interdisciplinary collaboration was an 8,0 ($n = 5$). Respondent AI-1 indicated they could not answer this question because it was too "bimodal". This respondent explained they were collaborating with two domain experts, and the collaboration quality with them was completely different ("Either very high, or very low").
- **Developer-client role distribution:** Typically the collaborations were initiated by the domain experts, who were looking for a solution to a problem they were dealing with. Subsequently, the AI experts ended up leading the collaborative project. This pattern was very clear from the interview question about who generally took the initiative to communicate. Except for DE-1, everyone responded this was most often the AI experts.

12.2.2. KEY COLLABORATION CHALLENGES

Two components of the interview were directed explicitly at identifying the key collaboration and communication challenges, namely the questions based on the relational coordination survey and the question about composing a top-3 of the most obstructive collaboration barriers. Since these were all categorical questions, aggregating their results was straightforward and formed the starting point of the results analysis.

RELATIONAL COORDINATION RESULTS

The aggregated results for the interview questions based on the Relational Coordination Survey are presented in Table 12.1. The first five questions relate to alignment (i.e. relational coordination) while the latter four are about communication specifically.

Table 12.1: Aggregated results for the interview questions based on the Relational Coordination Survey. Quantitative scoring and coloring based on prescribed methodology [71, 77]. Green is high (score > 4), blue is medium (score 3.5-4.0), and red is low (score < 3.5).

Nr	Question topic	AI expert responses ($n = 3$)	DE responses ($n = 3$)
1	Shared goals	4,0	4,7
2	Knowledge of domain experts about AI	2,7	3,5
3	Knowledge of AI experts about project-specific domain	3,3	3,2
4	Feeling respect for role of AI experts	4,7	5,0
5	Feeling respect for role of Domain Experts	4,3	5,0
6	Frequent communication	4,7	4,3
7	Timely communication	4,7	4,2
8	Accurate communication	3,8	4,0
9	Blaming vs. problem-solving communication	3,3	4,7

From the alignment questions, both questions 2 & 3 score very low indicating that **shared knowledge** is the main issue. Interestingly, the scores for the domain-specific knowledge of AI experts (question 3) are very much in agreement, but this is not the case the other way around (question 2). While domain experts rate their own knowledge about AI with an average score of 3.5, which is still considered medium, AI experts rate them significantly lower at just 2.7. The same holds for the first question, about shared goals. Although the averages are much higher, the AI experts are significantly more critical about domain experts than the other way around.

This could be explained by a number of reasons. Domain experts typically have a client role in these projects while AI experts are there to develop systems that solve their problems. As such, domain experts may have a particularly positive feeling about shared goals within the project because their problems are the ones being solved. Furthermore, the score difference for the knowledge of domain experts about AI may indicate that domain experts overestimate themselves.

From the communication questions, two main results jump out. First of all, the low score given by AI experts for blaming vs. problem-solving communication. Here, it is particularly important to understand the context and nuance. The following quote explains this well:

> **AI-2:** *"It depends mostly on what kind of expectations you've created, of course. I do think that they usually are very easy to blame the technology, so they don't necessarily blame the AI experts. But let's suggest we try something and the first iteration it didn't work. We weren't able to do it properly. Instead of blaming us, that usually doesn't happen, they say 'Oh, well, the technology is not there yet, it's too difficult, let's stop because it's over' instead of trying to really solve the problem. Usually what we find is that we need a few iterations to really get something right, and they are sometimes a little bit too willing to blame the technology."*

The second result to highlight is the relatively low score for accurate communication. On this, AI and domain experts are in agreement. In addition to the numerical scores presented in Table 12.1, four interviewees were asked specifically which of the four communication aspects from questions 6-9 was the biggest issue for collaboration. From these four interviewees, three responded it was communication accuracy and one said timeliness.

MOST OBSTRUCTIVE COLLABORATION BARRIERS

The aggregated answers to the question where interviewees had to compose a top-3 of most obstructive collaboration barriers are presented in Figure 12.2. The answers are grouped into three partially overlapping categories: social, knowledge, and practical barriers. The category of social barriers contains all cultural, interpersonal, and emotional factors. Knowledge barriers includes everything related to a lack of technical knowledge about (Gen)AI or specific domains. Finally, practical barriers are issues with a less personal, and more "external" origin. They impact collaboration but need to be addressed typically at the organizational level, or at an even larger scale. The categorization used in Figure 12.2 is based on the categories of AI challenges from Dwivedi *et al.* [55] where most of the response categories were taken from, namely: social, data, and organizational & managerial challenges. This categorization was adapted to provide more granularity on the (inter)personal level, separating social/emotional factors (related to "soft skills") from knowledge (related to "hard skills"). As a consequence all other barriers were grouped together in the practical category.

For each barrier, the number that is included indicates how many interviewees provided that answer. Not all interviewees adhered strictly to formulating a top-3. Some provided fewer or more answers. All responses were included, which explains why the total number of answers is slightly more than expected (21 instead of 18).

Several barriers were placed on the overlapping sections in Figure 12.2 to indicate that they are related to both aspects. For example, the *unrealistic expectations towards (Gen)AI* barrier was positioned in both the social and knowledge categories because it could result from a lack of knowledge and understanding of the technology, as well as from poor interpersonal communication. Similarly, a lack of multidisciplinary talent is clearly a knowledge-related issue, but it can also be a direct result of decisions at the organizational level, such as a poor hiring strategy or a lack of personal development programs. It could even be related to external factors outside the reach of organizations, such as a lack of suitable educational programs.

12.2.3. MAIN CHALLENGE: LACK OF SHARED KNOWLEDGE

In Figure 12.2, the knowledge barriers category contains the highest amount of responses and in Table 12.1, the questions about shared knowledge (both questions 2 & 3) received the lowest scores. Based on these results it was concluded that the **lack of shared knowledge** is the main challenge in the collaboration between AI and domain experts. Moreover, the lack of shared knowledge was found to lie at the root of many other challenges and issues. The previously presented quote from AI-2, for instance, showed how a lack of understanding about generative AI makes expectation management very important to prevent domain experts blaming the technology and giving up on the project after an unsuccessful first try.

Having established that the lack of shared knowledge forms the main overarching problem, we can zoom in on some additional, more specific factors like **expectation management** and **communication accuracy**. These

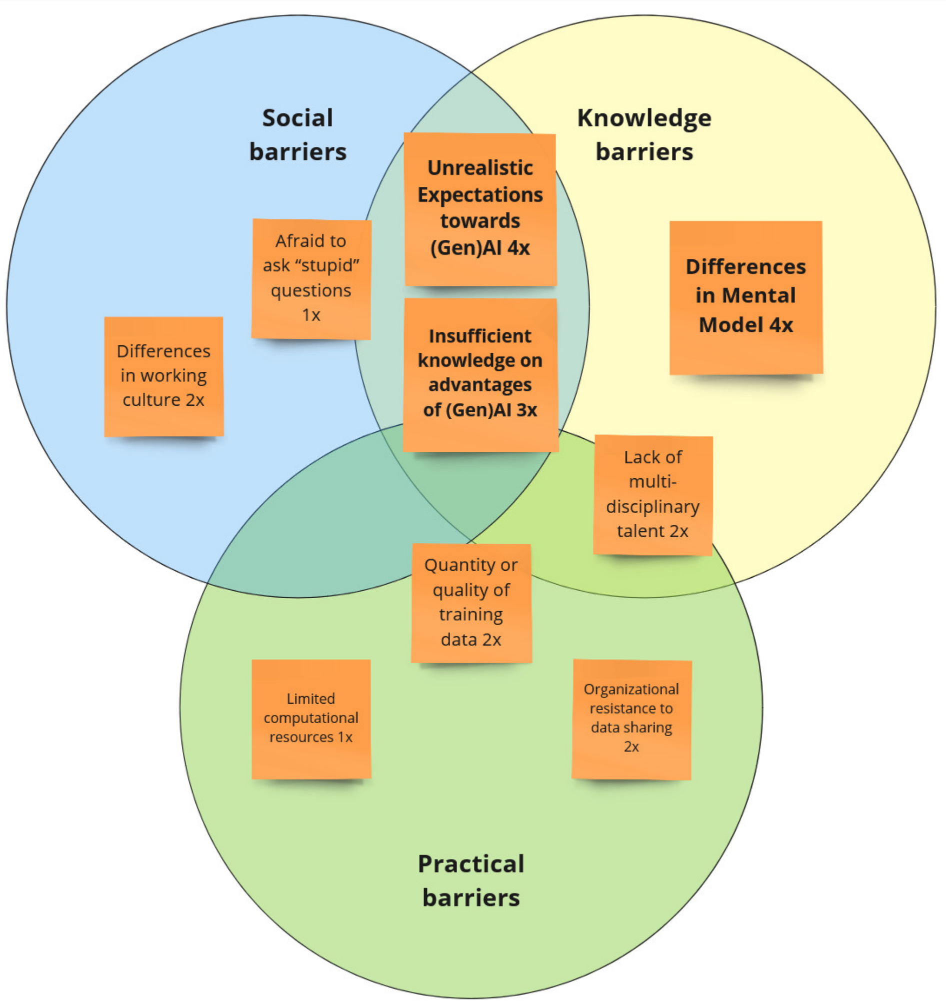


Figure 12.2: Venn diagram with the aggregated interview results for the question about most obstructive collaboration barriers. The numbers indicate how many interviewees provided that answer.

secondary issues, along with their relationship to the overarching problem of lack of shared knowledge, are discussed in the next section.

12.3. DETAILED PROBLEM ANALYSIS

Part of the secondary collaboration and communication issues discussed in this section could be regarded as "symptoms" of the lack of shared knowledge, while others act more like "compounding factors".

12.3.1. MISALIGNED EXPECTATIONS

Misaligned expectations are closely related to a lack of shared knowledge. In-depth analysis of the interview results revealed two different types of misaligned expectations. They differ in terms of their *cause* and relation to the lack of shared knowledge as well as the specific *context* in which they occur and their *impact*.

TYPE 1: MISUNDERSTANDING

The first type of misaligned expectations are those that exist without any communication between AI and domain experts about a particular issue has taken place. They are caused by a lack of awareness or incorrect assumptions about (aspects of) each other's domain. Thus, type 1 misaligned expectations are *directly related* to the lack of shared knowledge. The typical context of this type of misaligned expectations is before, and in the process of setting up, new interdisciplinary projects. Their occurrence typically decreases over the duration of the project.

Collaboration barriers from Figure 12.2 related to this type are *Unrealistic expectations towards (Gen)AI*, *Insufficient knowledge on advantages of (Gen)AI*, and the *Lack of multidisciplinary talent*. A selection of interview quotes that highlight this is provided below:

- **AI-1:** "What I found out is that people misunderstand what AI can do."
- **AI-2:** "Unrealistic expectations towards Gen AI technology is one of the biggest things. [...] Moravec's paradox, I'm not sure if you know it? What it implies is that people think some things are very hard to do for technical systems, so for example generative AI, which are in fact really easy to implement, and the other way around is just as likely. So sometimes they think 'Oh GenAI should be very good at this. It should be very easy to implement.' And then we [AI experts] look at the problems and are like 'This is so complicated, so complex, we're not able to do this.' So there's very much of a mismatch between those roles. And I think that is quite crucial."
- **AI-2:** "[...] if the base necessities aren't met to really bring a project a step forward, then you might as well not try it, because you are going to set expectations which are never able to be met. People will never be able to use it because it will die in a proof-of-concept phase, or in a pilot phase, and then you start to cut corners. So, you have on the one hand the expectations management of 'Can an AI do it?' but the other way around you have 'Can we do AI, as an organization?'. And for <my organization>, usually the answer is no for the more complex and the more advanced systems. [...] With GenAI especially, you need a very mature set of base requirements to be met before you can really do something properly, and that is something that you need to communicate effectively."
- **AI-3:** "I think one key AI concept that is misunderstood by domain experts is the capabilities of generative AI models. What it can do currently, what it cannot, and what outputs it can produce. So, for example, what kind of representation a generated CAD model is, and whether that is usable for manufacturing purposes or whether it can be only used for conceptualization and during the planning phase of the product development life cycle. The key point here is understanding the generative AI model technology itself, and the capabilities and the limitations of the model."
- **DE-2:** "The deeper you go into this, the more you realize there's actually quite a lot of tools that you need to be aware of. It's not just a single API and that's it."

These examples highlight that type 1 misaligned expectations have the biggest impact on the initiation of new collaborative AI projects which makes sense, when considering that this type of misaligned expectations exist *before* communication has taken place between AI and domain experts. Moreover, these interview results (and others) indicated that domain experts often have a poor understanding of the **capabilities** (strengths and weaknesses) of generative AI technology, as well as the **prerequisites** (data, quality metrics) and **required investment** (time, effort, money) to develop the generative AI systems. All these factors relate to the lack of shared knowledge and degrade the ability of domain experts to identify high-potential opportunities for generative AI projects.

TYPE 2: MISCOMMUNICATION

The second type of misaligned expectations are those resulting from miscommunications. These occur in scenarios where collaborations between AI and domain experts are already set up and they are actively working together on the design problem. This is one of the differentiating aspects between the two types of misaligned expectations.

Another key difference is that the lack of shared knowledge plays a different role. Instead of being a direct cause, here the lack of shared knowledge has an indirect effect by hindering the communication between collaborators. When working collaboratively on design problems, in-depth technical knowledge about particular topics from each others domain will be required. When this is lacking, collaborators need to educate each other on the spot. In and of itself, learning something new comes with a risk of misunderstandings which can lead to misaligned expectations. In addition, a lot of technical jargon is typically used in these design discussions which provides another way for a lack of shared knowledge to create misunderstanding. In particular, when a multitude of new technical terms are explained and immediately used in subsequent discussions.

Type 2 misaligned expectations correspond to the *Differences in mental model* barrier in Figure 12.2, as well as the lower scores for communication accuracy and blaming vs. problem-solving communication in Table 12.1. A selection of interview quotes that further highlight this issue is provided below:

- **AI-2:** "So one of the tools that we build is a B1 tool. B1 is a language level which is quite simple. [...] we said, 'well, Generative AI is very suited for this'. We did some preliminary tests and it was going quite well. But what we really found was, first of all, what is B1? Like can we quantify that level? That's very hard to do, and if you ask 10 experts, you get 11 opinions. So it's very much trying to find the middle ground and find a interpretation of B1 level that works for most people. So what we did was we said, 'OK, let's not set our goal at creating 100% B1 level texts. Let's just aim for 80% and help people to get started.' Those last 20 percents are usually the hardest part because then you really get to the finicky terms and the difficult terms that you have to still explain it in a very easy way. But you can at least get started and remove the easy-language writers block. That's what a lot of people have when they need to talk in a different way than what they're used to. So we set an expectation of 80% and that implies that suddenly we can measure what 80% B1 is. We did some tests with a lot of users, domain experts, and communication experts and we had a few metrics to score the texts. Then you could interpret those results as a percentage. So they interpreted 80% in such a way that if we have five metrics for B1 level and all texts scored four out of five metrics positively, then it's an 80% B1 text. That is the expectation we set. But that was not the way we wanted to validate or measure it. It was more of an abstract concept that we tried to get across. Instead of 'we guarantee that everything that comes out of this tool is B1' we just meant 'we'll get you started, and you get mostly there'. They were now interpreting the results that we measured with them as 'did we achieve our goal'. Our goal was to get people started in writing a text on B1 level, or close to it, but they said 'OK, we scored 76 percent so we didn't succeed'. But that was not the point, you know? So that is something where using those quantitative measures or words, it can really throw people off."
- **AI-2:** "We have a mismatch in the communication, which sets the wrong expectations, which results in people blaming the wrong parts of the problem. You can have very timely and very frequent communication, but if you can't really understand each other, you're missing the point entirely. [...] When something didn't work the way we expected, that is also expectation management in a sense, but [also] explaining the results. So why are the expectations not met? And do we think that we can solve that problem, or is it something we will have to accept? So it's just as much before we start as it is when we finish."
- *From these four aspects: frequency, timeliness, accuracy and problem-solving, which one would you say needs to be improved most?*
  AI-3: "I think it's communicating accurately, so making sure that we are on the same page, that we are referring to the same meaning of the same terminology, and we understand the domain of each other's roles."
- **AI-3:** "In the collaboration between domain experts and AI engineers, it's very important to emphasize understanding first of all the existing workflows of domain experts. And at the same time, it's also very important for domain experts to understand how the AI development life cycle, or the machine learning development life cycle works. So what the requirements are for better data, training data, etcetera. These are the main points that I think are very important to facilitate the collaboration."
- **DE-2:** "So obviously when we started looking at this kind of work, we didn't really know much about AI and all the different technologies and AI tools. Luckily a lot of them are Python-based but yeah, there was a phase where there was a very steep learning curve. Myself and <another domain expert>, we come from a design team. We still don't fully understand all the different AI terminology and tools out there but it's kind of familiar territory. So we are on a learning journey."

12.3.2. COMPOUNDING FACTORS

In addition to the previously described mechanism, other issues and barriers also relate to the misalignment in expectations. Collectively, these are considered compounding factors. An overview of these compounding factors is presented, along with example quotes.

LACK OF INTRINSIC MOTIVATION

Another important compounding factor that has not yet been discussed is a lack of intrinsic motivation to really engage in the project. This aspect was discussed with interviewees separately, and therefore not incorporated in previous results, like Figure 12.3 or Table 12.1. However, it is very important, as highlighted in the following quotations:

- **AI-2:** "It depends mostly on their [domain experts'] personal willingness to learn about the more technical side of things, because a lot of times they just throw something your way. They have a question, they have a problem, so they come to you. They think that we can solve it with AI and they just have to fill out the [intake] form and send it over, and then three months later everything is done and they can do exactly what they want without collaborating. That is not very often the case, but you sometimes feel that energy."
- **DE-2:** "In general, I would say it's quite difficult to find designers that are programmers as well, or are willing to become programmers. They are quite different skill sets. So it's a niche, within a niche, within a niche, right? If you want to do this, you need someone who's a designer. Within that they need to be a programmer, and within that they need to be fluent with AI and machine learning tools. So it's kind of difficult to hit all those three things. A typical designer, I would say, they're not interested to do programming."

External pressure

Aside from intrinsic motivation, building shared knowledge also requires having room for personal development and learning about new domains like (Gen)AI. A process called "cross-skilling" by one of the interviewees. However, based on the interviews, several external factors were identified that prohibit personal development, in particular of domain experts, and incentivize them to focus on other tasks and responsibilities. Different factors were found between industry and academic contexts, but their overall effect is largely the same.

Within industry, the external factor is typically management who put pressure on domain experts to focus only on their own tasks and core responsibilities. meanwhile, within in academia it is mostly individualism. This is highlighted by the following examples.

- **DE-1:** "I think if there was more time and if we had this [shared] end goal more from the start, then by now we would probably have reached that state of integrating. But in the end they also want to ensure that you develop into an independent researcher. Really integrating your work with another PhD student from the start is not very common, but ideally for this project that would have been the case from the start."
- **AI-1** "Just before doing the PhD, I worked four years as an analysis lead and I did that [developing engineering tools] on top of my responsibilities [...] and it's a really stressful situation. But we believe that building these tools will help us in the long term to answer the customers faster. That's why we try to invest the time to do it. And this is unofficial. It's not really recognized as your day-to-day job, if that makes sense, because they [management] will tell you: 'No, don't develop tools, just focus on the customer'. But if you start developing the tools on the side, this started in reality on the evenings and weekends outside of normal working hours, and they are really useful then you can bring it into your job and it saves you time. And then you can use that saved time to develop more of these tools. [...] Now it has gotten to the point that it's been recognized that, 'OK, these tools are a thing. You developing Python modules and releasing it somehow helps everyone else.' But in the beginning we couldn't. We pitched the business case, the improvement case, and asked them [management] to let us do that and it was: 'No, no, you cannot to do it.' [...] Like if you're trying to push a car with flat tires, it's like saying 'I'm not stopping to get new tires'. That's kind of the feeling that you get."

Differences in working culture

A number of interviewees also mentioned and described the effect of differences in working culture.

- **AI-2:** "My current role is more that of product owner, but I have a background in the data and the modeling world. So, I view the products through a data scientist's or an AI engineer's lens. We're trying to really see the questions that we get from our stakeholders as modular abstractions."
- **DE-1:** "I think that ties a bit into differences in working culture because, of course, in academia you need a research gap, and things should be novel, et cetera. But they [the client] were like, 'OK, but for us just this is fine. [...] It doesn't need to be so complex for us.'"
- **DE-3:** "With differences in mental model, I mean not being on the same page regarding problem definition. We [domain experts] have a concrete issue, and we are looking for a solution for that issue. We are not looking for anything else because we think it won't add any value, or maybe it's sensitive within the organization. [...] But when you ask a bunch of clever AI researchers and just explain your

situation, they give you a whole variety of solutions. However, I don't need everything. [...] I understand this [approach] from their point of view, but for us certain things are not that relevant."

AFRAID TO ASK "STUPID" QUESTIONS

The fear of asking stupid questions can also act as a compounding factor that reduces knowledge exchange between experts from different domains and, thereby, the process of building shared knowledge. Although, compared to the other factors, it was found to be less of an issue. One domain experts mentioned this as a factor, but only for the beginning of the project. Another domain expert even made an explicit comment that this was *not* an issue in their projects. Clearly, this factor can be a compounding factor but it was not a major issue within the projects of interviewees.

- **DE-1:** "In the beginning, it was hard that you don't have a lot of knowledge about the other field that your colleague is working on. So it might also hinder you a bit in terms of the questions you ask, or you might feel limited in terms of speaking out because you think some things are stupid or you're afraid to ask stupid questions, for example."
- **DE-2:** "If we do have questions or don't understand something, we will openly ask questions. There's no shame. [...] it's probably because of the nature of our business. We are hard-wired to be very helpful and ask questions to each other. I can understand it can be very different in other companies."

12.3.3. INTEGRATED REPRESENTATION OF THE KEY COLLABORATION CHALLENGES

The model presented in Figure 12.3 integrates all of the main collaboration and communication problems that were discussed. The red boxes contain the main issues and challenges that obstruct the collaboration between AI and domain experts. The dark blue boxes contain the compounding factors that put additional pressure on the core problems and can further aggravate the issues. The negative consequences that can result from this interplay of collaboration and communication challenges are depicted in yellow on the right-hand side.

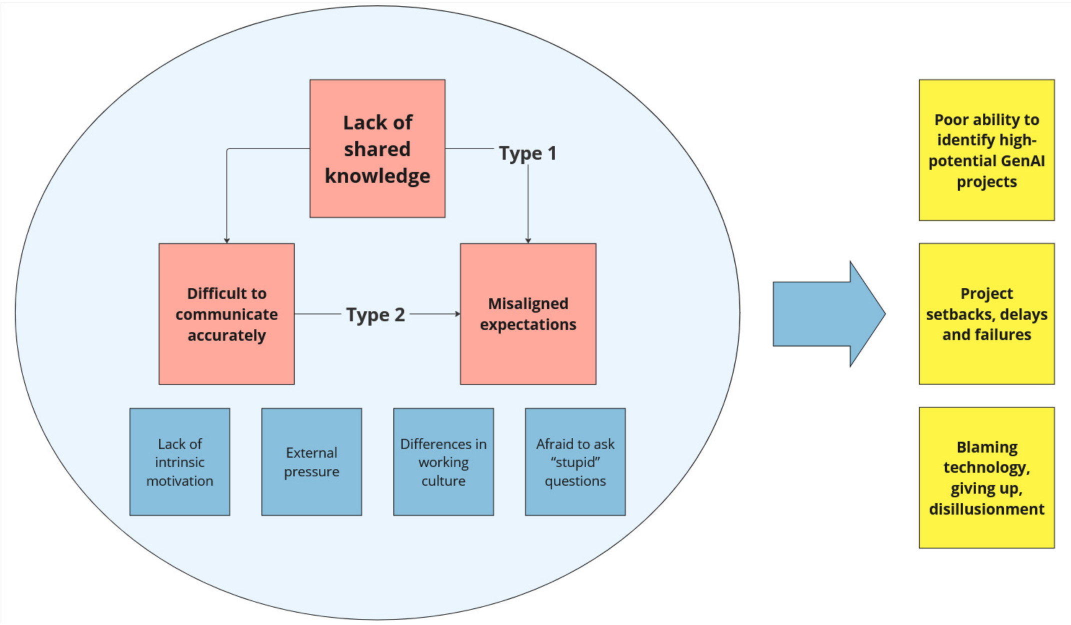


Figure 12.3: Integrated representation of the key interview results showing how the interrelations between the main collaboration and communication problems (red), compounding factors (dark blue) and resulting consequences (yellow).

ANSWER TO FIRST RESEARCH QUESTION

The problems contained in the blue circle in Figure 12.3 form the core of the problem. In other words, they are the *most critical* collaboration and communication challenges between AI experts and domain experts. Therefore, Figure 12.3 represents the answer the first research question: *What are the key collaboration and communication challenges between AI experts and domain experts in interdisciplinary AI projects, and how do these relate to each other?*

12.4. NEEDS AND INPUT ON SOLUTION IDEAS

In addition to investigating collaboration and communication problems, a part of the interviews was dedicated to discussing how the current situation could be best improved. These results are presented in this section.

FOCUS ON PROJECT INITIATION PHASE

A notable result was that all interviewees indicated they either wanted to move toward completely integrated teams of AI and domain experts in the future, or expand the amount of work performed at the 'collaborating' level on the Collaboration Continuum scale. In addition, multiple interviewees indicated that this close and intense collaboration is most important at the project *initiation* phase to really invest in establishing the objectives and scope of the project, getting to understand each other's perspective, and begin educating each other on important cross-disciplinary knowledge. A collection of related interview quotes is provided below:

- **DE-1:** "I think it makes these following steps [planning, execution, etc.] easier if you already start with integrating at the initiation phase."
- **DE-1:** "I think what would be nice and really work for such a project is to have like one or two full days in which you work together. So not planning one meeting of one or two hours, but a week in which you really stay together for two full days and you continuously discuss each other's work and expertise to basically speed up this process of creating a shared knowledge. Something to really reach this integration stage that we discussed at that early stage of the project."
- *If there is an improvement that you could propose to improve the collaboration process between you and these domain experts, what would it be?*
  AI-2: "I would say quality over quantity. So I would prefer a couple of goods deep-dive sessions in a physical environment where we really discuss the problem together, rather than a few more hours but online and in between different meetings. That is something that usually due to planning and scheduling isn't possible, but if we could do that, I think it would really improve our results."
- **AI-3:** "The phase where AI engineers collaborate most intensively with domain experts is in the beginning of the machine learning development cycle. [...] the most overlap between this [development] life cycle of AI engineers and the domain experts lies in the first part, which is identifying the data, preparing the data, and also identifying the challenge or the problem statement before starting the development itself. This is very important in order to analyze the data properly, and also to understand within the domain which data is needed in order to develop the machine learning system."
- **AI-3:** "The design team are the domain experts and they are also the end-users or the customers, so to say. And that requires them to be included during the initiation of the project, of course, in order to understand what they need, what they want. Their needs are requirements in general. And then also at the same time [their involvement is required] to understand the domain specific terminology, the domain specific data. All of these kind of things."

INCONCLUSIVE RESULTS ON COMMUNICATION TOOL OBJECTIVE

Before presenting the actual list of solution ideas, interviewees were asked about the most useful objective for a communication tool. They were given the choice to either focus on improving the communication quality in existing collaborations; or motivate more professionals to engage in collaborative AI projects.

In response, all three domain experts opted for engaging more professionals in AI projects, while two AI experts both chose to increase communication quality. The third AI expert did not provide input on this topic because the question had to be skipped due to time constraints. It is interesting that the responses are aligned per response category. Unfortunately, this division also means that the overall result is quite inconclusive and cannot be used to make a well-informed design decision.

- **DE-2:** "Engaging more professionals in these projects would be best, because then they themselves might say 'This is actually an interesting topic. I'm going to get some training. I'm going to do some learning. I'm going to try and show the work to other people in the company. Maybe I can get some more interest.'"

12.4.1. NEEDS HIGHLIGHTED BY INTERVIEWEES

The list of quotes provided below represents a collection of a variety of needs and specific areas for improvement that were described explicitly by the interviewees:

- **Little emphasis on communication: AI-2:** “So, for our current project I just finished up a report, which is partly business case and partly implementation plan on how to move forward. And I made it very clear that I think that it should be a program which includes one technical project, building the thing, and it should also include a project that is more of an implementation strategy which really invests heavily in learning and development, really invests heavily in educating our users and also our stakeholders, our domain experts. And if I suggest that it’s usually, ‘Yeah, yeah, we should do that’, but then I do get money for the technical solution part and I don’t get money for the more communication side of things. So, I’m able to pitch it, but it doesn’t always really land or cause any different policy changes or decision.”
- **Focus on interaction: AI-2:** “Create a list of inspirational projects. It really depends on whether or not the organization is homogeneous in their work processes. So, for example, I have a lot of different organizations with a lot of different work processes. If I create a item on my list that is a work process for <organization A>, people from <organization B> won’t always be able to map their problem to this problem. So, *we need to talk into more in abstractions that people are able to map their problem onto, and I think the morphological chart in that sense is more effective*. It doesn’t mean that it’s not handy to have those listings. Especially if you can make it into a list of demo’s rather than a text-based list. That really helps because people see the interaction. If you have a video, for example, it’s much easier for them to map their problem to the interaction rather than to the description of the project.”

12.4.2. INPUT ON SOLUTION CONCEPTS

The input on the presented list of solution ideas was more aligned. From the first four interviews, two interviewees responded that the morphological chart was their favorite idea, and a third respondent rated it a second. Furthermore, the community idea was reviewed as favorite once, and twice as second-best. The other ideas received significantly less attention.

Regarding the morphological chart, the following this were said:

- **AI-2:** “When I need to create new slides for a presentation, this is pretty much what I’m aiming for. It’s nice to have a word for it. So what are the design decisions? What are the design patterns? What is the flow chart, almost, of such a project? Where do you end up? So I think that is very effective during intake. Nobody will do this unguided. You really need somebody to guide you through such a chart. But that’s very effective to get very clear expectations very early.”
- **DE-1:** “I really like the morphological chart. I think it’s great that such a chart provides an overview of possibilities, but also the different combinations of them. I think that always works really well to have this overview and, in a way, that already can inspire people to even come up with novel methods, right?”
- **DE-2:** “If I wear my technical hat, the morphological chart would be really exciting.”

Regarding the idea to design a community, the following comments were made:

- **AI-2:** “That [a community] is the golden goal, I think, but my experience is communities are incredibly hard to curate [in practice]. I’m currently trying to curate a community for synthetic data within <my organization> and partners and such. And we have to pull so hard on everyone to come, to organize something, to give some information back to the community and not only come there to take information. That is just very hard. It takes a lot of time, it takes effort, it takes organization. [...] It should be the best solution, but it doesn’t really happen by itself.”
- **DE-1:** “I also really like the community for sharing resources and best practices. I think that also works really well. Again, it ties a bit into the other idea. I think it’s great if you have this overview, in this case of best practices, what worked well for certain people, and why it worked well, because it can facilitate creative thinking and generating new ideas.”
- **DE-2:** “I think overall though, a community, a design community will be excellent because it can help spark ideas for all the other ones. So I would go with that, personally.”

INPUT ON MORPHOLOGICAL CHART

After the first four interviews, the morphological chart was selected as the best option. Therefore, during the last two interviews, only input on this particular idea was gathered. A summary of these additional results is provided below:

- **AI-1:** "My initial thought about the morphological matrix, or the limitation of a morphological matrix, is that it assumes complete independence between the different categories. You need to add on some other methods to add and manage the incompatibilities. So that's difficult to do in general, because of the morphological matrix itself. So completely unrelated to AI, but I see that in the AI domain as well. [...] But yeah, I think in general some sort of categorization would be good. It doesn't necessarily need to be on every category, but the dependencies would be interesting."
- **AI-3:** "I really like the idea, especially related to the identifying the existing workflow without adopting AI in it. So the existing design workflow. And then identifying the areas in which AI can help improve the designer process or reduce the time, automate it, or streamline the design process or the workflow. So I need to find the areas in the designer process and then suggest the tools or the requirements that might be needed in order to implement that. So I really like this train of thought."

12.5. Reflection on the interview process

During the structured interviews, several unexpected patterns and challenges emerged that influenced data collection and analysis. These observations are discussed below, providing additional context for interpreting the interview findings.

Challenges in Interviewee Classification

The process of classifying interview participants into either AI experts or domain experts proved to be more complex than initially anticipated. Several interviewees did not fit neatly into either category, revealing the fluid and overlapping nature of expertise in interdisciplinary AI projects. For example, after contacting a lead researcher of a TU Delft AI Lab about interviewing an AI expert, they referred a colleague. However, during the interview, it became apparent that this individual was more aligned with the domain expert classification. The discrepancy arose due to differences in personal and institutional definitions of what qualifies as AI expertise (as discussed in chapter 2).

Similarly, another participant initially identified themselves as an AI researcher via email but, after explicit follow-up questions, turned out to have a professional background in engineering and manufacturing. Their self-classification was based on their **intent** to learn more about AI and reflected their current personal development efforts in the AI domain area rather than their core expertise. This mis-classification highlights a typical phenomena related to the novel nature of the GenAI field: many professionals are re-skilling themselves to become experts, but they do not have a formal background in the field.

Interviewee Self-Inconsistencies in Survey Responses

Another notable observation was the inconsistency between qualitative statements made during the interview and the responses provided to categorical questions. This was particularly evident in the Relational Coordination survey questions. For instance, in one interview, a participant repeatedly mentioned issues related to misaligned project goals between AI experts and domain experts. However, when directly asked about goal alignment in the survey, they provided a relatively high score ("Mostly agree")—contradicting their earlier statements. One possible explanation is that the structured nature of categorical questions prompted immediate, socially desirable responses, whereas open-ended discussions allowed for more reflective and nuanced answers.

Additionally, it is theorized that the unfamiliarity of technical experts with deeply discussing and analyzing communication and collaboration problems also played a role. It was observed that questions provoked a thinking process that continued after moving on from the current (categorical) question. They simply had not considered such questions before, so they did not have a readily available answer. Therefore, additional thoughts and conclusions that came after answering the initial thought-provoking question are then mentioned and infused into later answers. In other words, the questions need time to "perculate". Especially when being asked to give a critical categorical answer on the spot, like for the relation coordination questions, the natural reflex to give socially desirable answers takes over. Interestingly, this theory was found to have practical benefits. A pattern was observed where in many instances, interviewees addressed certain topics in an answer that were considered to be much more relevant and applicable to a question posed just a couple of minutes back. When taking this "perculation" effect account, entire sections of certain interviews which were previously considered to be rather confusing suddenly made a lot of sense.

Lack of differentiation in categorical answers

Another general observation regarding categorical questions is that (some) interviewees had a tendency to not differentiate their responses. Participants often provided similar ratings across different questions, even when their qualitative statements suggested varying levels of difficulty in specific collaboration areas. A feasible explanation is that participants were asked to evaluate entire groups (e.g., AI experts or domain experts) rather than individual collaborators. These group-level evaluations likely led to response averaging, reducing the differentiation between responses. Another, more direct explanation can be found in the definition of the Likert scale from the Relations Coordination Survey. Its response categories cover a very extreme range. For example, questions respectively have an lower and upper bound of "never" and "always". Such extreme answers are virtually never given in practice, which then limits the "viable" answers to just three remaining options. In future work, it is deemed worth looking at these scale and adjusting them to a more fine-grained and useful range of answer categories. These findings emphasize the need for careful interpretation of categorical survey responses and suggest that future studies could benefit from supplementing quantitative questions with structured qualitative follow-ups.

13
SOLUTION DIRECTION AND FOLLOW-UP QUESTIONS

The interview results presented in the previous chapter marked a significant step in the design process. A solution direction for the communication tool was selected on the basis of these results, which is presented in section 15.1. Here, the term *solution direction* is used because, although the type of communication tool was selected, a number of different use cases were also identified for which the tool could be designed. These use cases correspond to different conceptual design options. These are also presented in the first section. In order to define and select the actual communication tool *concept*, instead of merely the solution direction, a number of key follow-up questions were identified. These are presented in section 13.2, along with an outline of the research activities that were conducted to answer them.

13.1. SELECTED SOLUTION DIRECTION

The **morphological chart** was selected as the solution direction. This design decision was made based one three key considerations. Firstly, this idea was the overall favorite during the first four interviews[1] that were conducted (with AI-2, DE-1, DE-2, DE-3). The second important consideration has a practical nature, namely the potential to realize a positive impact. Given the huge challenge of setting up and curating a successful community, as described by interviewee AI-2, and the fact that there is currently no clear stakeholder to carry such a community and invest the required time and effort over a prolonged period of time, the community idea was considered to have less potential impact than the morphological chart idea. Finally, compared to the community idea, the morphological chart also integrates much better with the aerospace part of this thesis. In fact, there are several ways how the design process of this tool can benefit from the previously performed work. For one, the knowledge gained about GenAI can be applied directly to define and select the set of design features and respective design options to be included in the morphological chart. Furthermore, the personal experience of being an aerospace domain expert who wants to develop a GenAI-powered tool will help to empathize with the intended target audience. Lastly, the REProcess prototype developed for the aerospace thesis can serve as a very convenient test case during the development of morphological chart. By being its developer, the REProcess prototype is completely understood. Therefore, any version of the morphological chart can be quickly evaluated by using it to characterize the REProcess prototype. This consists simply of going down the list and selecting the design option(s) implemented for the REProcess prototype for each of the included design features. Due to these synergies between the aerospace thesis work and the morphological chart idea, the experience gained during the aerospace part of this thesis project is put to maximum use. For the community design idea, comparable natural synergies could not be identified.

Based on the considerations discussed above, it was concluded that the morphological chart is the best solution direction. Hereby, the second research question - *Which communication tool idea has the most potential to facilitate the collaboration between AI experts and domain experts?* - is answered

[1] In the last two interviews (AI-3 and the interview with AI-1) where only this design idea was presented, to enable having a more in-depth discussion in more depth, the feedback was also positive.

13.1.1. DESIGN REQUIREMENTS BASED ON INTERVIEWS

Based on the interview results, a set of high-level requirements were formulated to guide the design of the morphological chart. These requirements are closely linked to the critical issues that were included in Figure 12.3.

1. The morphological chart should facilitate the development of GenAI-powered systems used to (partially) automate engineering workflows within the context of complex, critical systems.
2. The morphological chart should help domain experts and AI experts to align their vision and expectations about GenAI-powered systems.
3. The morphological chart should focus on GenAI features and design options that domain experts often misunderstand or are unaware of.
4. The morphological chart should help domain experts to understand the prerequisites and investment required to develop GenAI-powered systems.

13.1.2. POSSIBLE USE CASES FOR MORPHOLOGICAL CHART

Initially, the selection of the morphological chart idea was believed to provide a very clear direction for the subsequent design and implementation phase. However, it was quickly realized that, while the overall *form* of the tool was now clear, the answer to the next main research question - *How to implement the selected communication tool concept such that it provides maximum added value?* - was not trivial. A key part of this was the realization that the morphological chart could be applied in a lot of different ways. In particular, two use cases were identified that were both deemed very interesting but have different implications for the design.

1. **Communication focus:** The morphological chart is placed on the table during intake meetings and initial design discussions between AI experts and domain experts. For online meetings, the equivalent would be to use a virtual whiteboard tool (e.g. Miro or Mural).

   Primary objective: Improving the accuracy of communication between AI experts and domain experts. The AI expert can guide the domain expert through the chart to explain all the different aspects that have to be considered. The options presented per design aspect provide a means for the domain expert to articulate what kind of tool they envision and what fits best with their context. This concept is more focused on addressing Type 2 misaligned expectations (defined in section 12.3).

2. **Education focus:** The morphological chart is presented to domain experts in settings where they do not interact with AI experts. For example, on posters placed near the coffee machine, in the hallway, etc.

   Primary objective: Educate domain experts on the capabilities (strengths and weaknesses), and the prerequisites and investment required to develop GenAI-powered tools. This should improve their ability to identify high-potential applications of GenAI within their own domain (inspiration and realistic expectations), as well as their ability to communicate with AI experts. This concept is more focused on addressing Type 1 misaligned expectations (defined in section 12.3).

In addition to the two primary use cases, a third possible use case that was identified is to use the morphological chart to **present** the design of implemented GenAI-powered tools in a clear and concise way. This way of using the morphological chart, however, could work in combination with either of the use cases presented above. In other words, this was not a "design-to" use case but rather an additional application of the final tool that will be viable either way.

13.2. FOLLOW-UP QUESTIONS

For either of the two primary use cases, the design of the communication tool would have to meet different requirements and the final design will look rather different. For example, when focusing on the second use case, a requirement along the lines of "The morphological chart should improve domain expert's ability to identify high-potential applications of GenAI within their domain." could be added to the list, but for the first use case this would not be a crucial requirement. On the other hand, when focusing on the first use case a greater emphasis on designing the usage process surrounding the morphological chart itself is required. At this early stage, that usage process was envisioned to be something along the lines of the methodology outlined below:

1. Sketch / visualize current engineering process/workflow
2. Formulate desired impact (work speed up, improve quality, increase job satisfaction, ...) to be obtained by novel GenAI]-powered tool.
   (a) Define using Key Performance Indicator (KPI)'s (current + target)
   (b) Specify the change w.r.t. current engineering workflow (using previously made diagram of current workflow).
3. Use morphological chart to discuss and define envisioned GenAI intervention.

Corresponding to this open question of what the specific focus of the morphological chart communication tool should be, the following main research question was formulated:

RQ-4: DETERMINE DESIGN FOCUS

Which specific use case should the morphological chart be geared towards?

Another key question that was raised at this point was deduced more directly from research question 3. Determining how to implement the tool such that it has real added value, will require finding out which similar tools already exist. Based on this information, the morphological chart can be designed to address a gap in existing work and thereby provide real added value. Additionally, performing this investigation could also help to find an answer to the research question 4. To guide this investigation, the following research question defined:

RQ-5: IDENTIFY SIMILAR EXISTING TOOLS

To what extent do similar tools exist, both in scientific and nonscientific contexts?

13.2.1. SUBSEQUENT RESEARCH AND DESIGN STEPS

The following activities were carried out to refine the content of the morphological chart and its accompanying methodology:

- **Follow-up literature study:** A follow-up literature study was aimed at identifying similar tools. This literature study directly relates to research question 5. The findings will also be useful as inspiration for the design itself In addition, the literature study can be used to identify a niche that is not yet covered by existing tools. Thereby, helping to answer research question 4. This literature study is presented in chapter 14.
- **Design based research:** An iterative approach was used to create the morphological chart and answer research question 4. Information collected from all previous research activities was used to develop and iteratively refine prototypes of the morphological chart. These prototype iterations were used during follow-up discussions (meetings/interviews) conducted with experts from the initial series of interviews as well as additional AI experts and domain experts. Their input was used to inform design improvements and in the end to determine an answer to research questions 3 and 4. These activities and their results are presented in chapter 15.
- **Final design:** The final design, which integrates the answers to all research questions, is presented in chapter 16.

14

Follow-up Literature Review

The aim of the follow-up literature review was to identify communication tools and frameworks developed by other researchers that fulfill a similar purpose as the envisioned morphological chart. This information serves a dual purpose. Firstly, it is useful to identify a specific gap that can be addressed through the design of the morphological chart and secondly, the existing tools and frameworks can be used as input and inspiration for later design work.

14.1. Methodology

The initial focus of this literature review was to identify similar communication tools and determine the specific gap. Once this was established, the scope of the search broadened and shifted towards finding and selecting work that can provide useful input for the design process.

14.1.1. Searches for similar tools

Google image search

Before diving into any scientific literature, it was recognized that insights and inspiration could also be obtained from assessing non-scientific sources. Large technical consultancies, for example, also write about AI and how to implement it. Therefore, a Google image search was performed first to find any existing morphological charts for GenAI. The following relatively simple queries were used:

- morphological chart generative ai
- morphological matrix generative ai
- design matrix generative ai
- design framework generative ai

Scopus search

The follow-up literature study was performed using Scopus and Google Scholar. The following search queries were used:

- TITLE-ABS-KEY ((morphological AND (chart OR matrix OR table OR analysis)) AND ("generative AI" OR "generative artificial intelligence") AND (applications OR systems OR programs OR tools OR methods))
  Findings: just 4 results, 2 interesting.
- TITLE-ABS-KEY (("ai tool design" OR "ai system design") AND ("framework" OR method OR methodology OR approach OR practice OR taxonomy))
  Filteris: limited to Computer Science and Engineering subject areas, published since 2019 (last 5 years), and at least 1 citation.
  Findings: 17 findings, 3 relevant based on title and abstract, 2 relevant after reading intro, results and conclusions.

Google Scholar search

Due to the limited number of findings obtained from search literature with the Scopus search engine, a switch was made to Google Scholar. This resulted in a big increase in the number of findings. The first search query in the list below was identical to a previous Scopus search, but now yielded 8810 results instead of just 4.

- (morphological AND (chart OR matrix OR table OR analysis)) AND ("generative AI" OR "generative artificial intelligence") AND (applications OR systems OR programs OR tools OR methods)
  Filtering: since 2019 (6050 results), sorted by relevance, assessed first 50 results.
  Findings: initially resulted in 3 interesting finds. 2 discarded after reading more. Majority of findings not relevant because they were A) articles where GenAI was used to generate a morphological chart, B) focused on non-relevant domains, or C) less than 5 citations.
- "design" AND (specification OR dimensions OR principles OR features) AND ("framework" OR method OR methodology) AND (generative AND (ai OR "artificial intelligence") AND (applications OR systems OR tools OR methods))
- ((design OR development) AND (framework OR method OR methodology OR approach) AND ("generative AI" OR "generative artificial intelligence" OR "GenAI") AND (applications OR systems OR tools OR methods))
  Filters: since 2019, sorted by relevance, assessed first 50 results.

14.2. Conclusions about existing communication tools

The following results were drawn based on results from the literature study:

- A significant portion of identified research is about attempts to use GenAI to perform a morphological analysis in order to (partially) automate the design process, instead of applying morphological analysis to GenAI systems themselves.
  Notable examples: Zhang and Yin [79],
- Other existing communication tools and frameworks typically contain a lot of AI-specific jargon and technical information. They seem to be created by AI experts, for AI experts. These tools may be be useful to facilitate communication during the collaboration between AI and domain experts but they are not suited as stand-alone communication artifacts that domain experts can use or understand by themselves.
  Notable examples: Sengar *et al.* [80], Bandi *et al.* [81],
- The remaining communication tools and frameworks that were identified are not geared specifically towards *generative* AI nor engineering contexts.
  Notable examples: Rittelmeyer and Sandkuhl [82, 83]
- A gap was identified as a combination of characteristics: 1) morphological analysis of *generative* AI systems; (2) geared towards engineering contexts; and (3) understandable by non-AI experts.

Another **design requirement** for the communication tool was formulated based on these literature study conclusions:

> *The morphological chart should be understandable as a **stand-alone communication artifact** to domain experts without in-depth GenAI knowledge.*

14.3. Selection of related work for design process

In addition to identifying and verifying the gap in existing work, the second objective of this literature review was to compose a selection of key sources that will be used to provide input and inspiration for the subsequent design phase. Here, the target was to select between 5 and 10 sources used for in-depth analysis and comparison. The selection was based on the criteria outlined below. In the rest of this section, these key literature findings are presented. Their application to the design of the communication tool is presented in the next chapter (section 15.3). In the remainder of this section, each of the selected sources is presented.

- **Maturity:** Sources were selected for their maturity, assessed using various characteristics. Examples include the comprehensiveness of the framework (i.e., breadth and level of detail), extent of practical application or validation, number of citations, and size or reputation of the associated organization(s).

Note that these characteristics varied depending on the source type. For instance, citation count is relevant to scientific sources, whereas organizational size is more applicable to non-scientific sources.

- **Scope coverage:** Recognizing that different sources emphasize different aspects of AI system design, the selection ensured adequate coverage across all relevant dimensions.
- **Diversity:** The selection aimed to include a diverse range of sources to incorporate input from multiple perspectives. This is reflected in the wide array of backgrounds from the authors of the selected works:
  - Weisz *et al.* [2] - The authors are researchers affiliated with IBM Research, specializing in human-computer interaction and AI design. This work was presented at the CHI Conference on Human Factors in Computing Systems.
  - Yildirim *et al.* [84] - This interdisciplinary team comprises researchers from Carnegie Mellon University, with expertise spanning HCI, design, and AI. Publication: The study was published in the Proceedings of the 2023 ACM Designing Interactive Systems Conference
  - Shi *et al.* [85] - The main authors come from the Department of Electrical and Computer Engineering, Purdue University, USA, and their expertise is on GenAI and LLMs. Article published only on arXiv.
  - Rittelmeyer and Sandkuhl [82] - Authors come from Institute of Computer Science, University of Rostock, Germany & School of Engineering, Jönköping University, Sweden. Paper was published in Perspectives in Business Informatics Research, Lecture Notes in Business Information Processing.
  - Yüksel *et al.* [4] - Authors come from Department of Industrial Design Engineering, Gazi University, Turkey. Published in journal of Engineering Applications of Artificial Intelligence.
  - Schmid *et al.* [3] - Main authors from University of Leipzig, Germany, or IBM, Germany. Published in KI - Kunstliche Intelligenz journal.

14.3.1. Design Principles for Generative AI Applications

Weisz *et al.* [2] Defined six design principles for Generative AI applications, formulated as “Design for ...” statements. This recent framework (from 2024) claims to differentiate itself from existing work by focusing specifically on *Generative* AI applications: “Existing human-AI design guidelines fail to address the unique design challenges of generative AI because they do not cover generative use cases or new considerations stemming from generative variability, and they do not cover new or amplified ethical issues stemming from the models’ generative nature.”

The six design principles and corresponding design strategies for how to implement that principle are presented in Table 14.1 below, which was obtained from [2].

Table 14.1: Design Principles for Generative AI Applications, obtained from [2]

Design Responsibly Ensure the AI system solves real user issues and minimizes user harms. • Design for the user by understanding their needs and pain points, and not for the technology or its capabilities. • Consider and balance different values across people involved in the creation, adoption, and usage of the AI system. • Determine whether generative capabilities beyond the intended use case should be surfaced to the user or restricted. • Identify relevant user harms (e.g. bias, toxic content, misinformation) and include mechanisms that test and monitor for them.	**Design for Generative Variability** Help the user manage the ability of generative models to produce multiple outputs that are distinct and varied. • Generate multiple outputs that are either hidden or visible to the user in order to increase the chance of producing one that fits their need. • Show the user the outputs they have created and guide them to new output possibilities. • Design user-driven or automated mechanisms for organizing, labeling, filtering, and/or sorting outputs. • Help the user identify how outputs generated from the same prompt differ from each other.

Design for Mental Models Communicate how to work effectively with the AI system, considering the user's background and goals.

- Help the user understand the AI system's behavior and that it may produce multiple, varied outputs for the same input.
- Help the user learn how to effectively use the AI system by providing explanations of features and examples through in-context mechanisms and documentation.
- Build upon the user's existing mental models and evaluate how they think about your application: its capabilities, limitations, and how to work with it effectively.
- Capture the user's expectations, behaviors, and preferences to improve the AI system's interactions with them.

Design for Co-Creation Enable the user to influence the generative process and work collaboratively with the AI system.

- Assist the user in prompting effectively to produce outputs that fit their needs.
- Let the user control generic aspects of the generative process such as the number of outputs and the random seed used to produce those outputs.
- Let the user control parameters specific to their use case, domain, or the generative AI's model architecture.
- Allow both the user and the AI system to improve generated outputs.

Design for Appropriate Trust & Reliance Help the user determine when they should or should not rely on the AI system's outputs by teaching them to be skeptical of quality issues, inaccuracies, biases, underrepresentation, and other issues.

- Be clear and upfront about how well the AI system performs different tasks by explaining its capabilities and limitations.
- Show the user why a particular output was generated by identifying the source materials used to generate it.
- Encourage the user to review and think critically about outputs by designing mechanisms that slow them down at key decision-making points.
- Determine the role the AI system will take within the user's workflow.

Design for Imperfection Help the user understand and work with outputs that may not align with their expectations.

- Caution the user that outputs may not align with their expectations and identify detectable uncertainties or flaws.
- Help the user identify outputs that satisfy measurable quality criteria.
- Provide ways for the user to fix flaws and improve output quality, such as editing, regenerating, or providing alternatives.
- Collect user feedback to improve the training of the AI system.

14.3.2. CREATING DESIGN RESOURCES TO SCAFFOLD THE IDEATION OF AI CONCEPTS.

Yildirim *et al.* [84] performed three design experiments:

Design Experiment 1: identify AI capabilities in ways that are useful for designers. This experiment resulted in a resource of 8 high-level capabilities, collection of 40 AI examples with granular capabilities, and a grammar for capturing and extending this resource with new capabilities and examples.

Design Experiment 2: Can designers use AI capability abstractions and examples to improve their ideation process? How can we assess whether ideation is better? We detail a failed pilot study involving ideation sessions with HCI students. This experiment revealed the importance of AI model performance, and resulted in a Task Expertise-AI Performance matrix. The analysis of AI examples on the matrix suggested the need to search for situations where moderate performance creates value. The experiment also surfaced tensions around the user-centered design process when designers have predetermined that AI is the solution.

Design Experiment 3: Can designers sensitize innovation teams to look for opportunities where moderate model performance might be valuable? We assembled an interdisciplinary team of data scientists and critical care clinicians, and we brainstormed AI concepts for the intensive care unit (ICU). We found that starting with examples of AI systems that create value with moderate model performance helped the team generate concepts that were valuable and low-risk. The experiment revealed that an innovation process blending user-centered and technology-centered approaches leads to better ideation.

14.3.3. AN HCI-CENTRIC SURVEY AND TAXONOMY OF HUMAN-GENERATIVE-AI INTERACTIONS

The AI design specification framework presented in the article Shi *et al.* [85] focuses on systematically analyzing and organizing human-Generative AI interactions through a detailed taxonomy. This framework identifies six key dimensions for designing effective and user-centered GenAI systems. An overview of these dimensions is presented in Figure 14.1,

DIMENSIONS OF THE FRAMEWORK:

1. **Purposes of Using GenAI**: Outlines various goals for leveraging GenAI, such as refining outcomes, exploring alternatives, automating processes, enhancing experiences, augmenting sample data, understanding subjects, and getting answers to inquiries. Each purpose represents a distinct use case.
2. **Feedback from Models to Users**: Categorizes how GenAI systems communicate their outputs to users. This dimension includes:
   - **Output Modalities**: Text, images, audio, 3D models, etc.
   - **Functions of GenAI Models**: Generation from scratch, transformation, inter-modal conversion, and aggregation.
   - **Output Synchronization**: Timing of feedback—real-time, delayed, or preliminary.
3. **Control from Users to Models**: Examines how users influence GenAI outputs. Key aspects include:
   - **Methods to Improve Outputs**: Option selection, parameter tuning, natural language commands, re-initialization, etc.
   - **Objects Controlled**: Input prompts, latent space, parameters, and hyper-parameters.
   - **Mediums of Control**: GUIs, voice commands, gestures, brain signals, etc.
4. **Levels of Engagement**: Defines the degree of user involvement in creating outputs, ranging from passive (receiving information) to collaborative (co-creating content with GenAI).
5. **Application Domains**: Identifies diverse use cases, such as art and creativity, science and research, programming, education, robotics, and quality of life.
6. **Evaluation Strategies**: Describes methods to assess GenAI systems, including technical evaluations, user studies, and demonstrations.

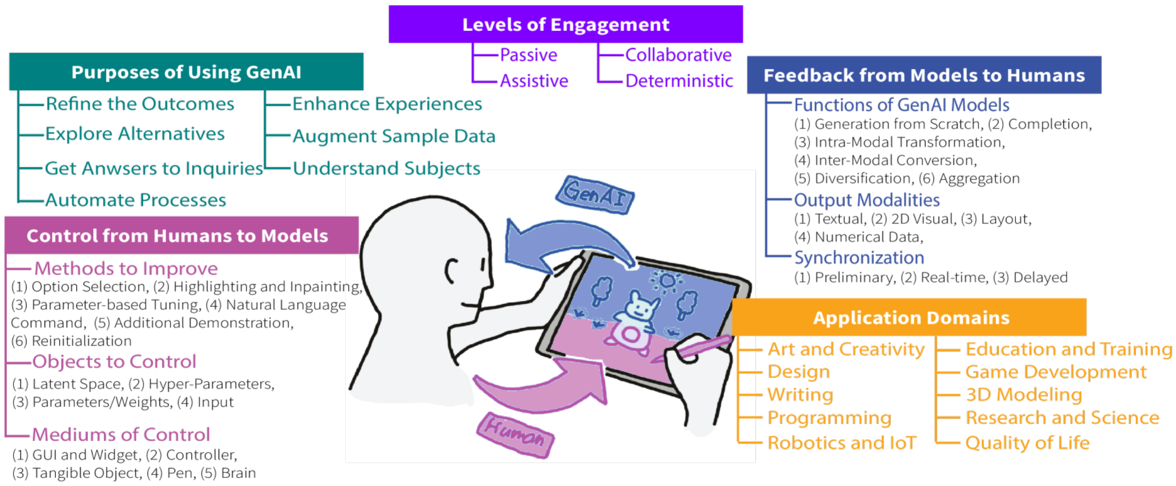


Figure 14.1: Visualization of the Human-GenAI interaction taxonomy developed by Shi *et al.* [85].

14.3.4. Morphological Box for AI Solutions: Evaluation and Refinement with a Taxonomy Development Method

Rittelmeyer and Sandkuhl [82] developed a structured framework for evaluating and refining AI solutions using a morphological box approach, enhanced by a taxonomy development method. The morphological box is a product development tool to break down a complex problem (in this case, AI solution design and implementation) into distinct features, each representing a critical dimension of the solution. For each feature, a range of possible values is defined, enabling organizations to evaluate different configurations systematically. The box is presented in Figure 14.2.

Feature	Values						
AI Focus	Processing input	Generating output	Computing task				
End-User	IT-Expert	IT-Savvy	No IT-Knowledge				
Computing Source	Cloud	Local computing center	End-Device	Hybrid			
Time to Decision	Real-time	Near Real-Time	Several Hours	Later			
Special Hardware Required	Computing	Data Capture	Data Visualization	Data Output			
Reliability and Precision of Results	~ 99,9 % Required	Defined by Enterprise	Defined by Domain	Defined by Competitors			
Point in Time of AI Use in Solution Development	Design-Time	Runtime	Accompanying runtime	Hybrid			
Primary Purpose	Assistance	Decision Making	Forecasting	Classification	Anomaly Detection		
Data Source	Own Data	Augmented Data	Open Data	Commercial Collection	Synthetic Data		
Data Quantity	Very High	High	Moderate	Low	Very Low		
Maturity	COTS	Commercial Components	Open Source Components	Prototype	Individual		
Data and Model Update Frequency	Continuously	In Case of Changes in Regulation	In Case of New Documents/Data	In case of Changes in Customer Behavior	In Case of Quality Problems		
Extent of Effect on Enterprise	Isolated Solution	Single Process	Workflow	Work System	Business Model		
Communication	Frequent & detailed, active collaboration	Regular, some collaboration	Minimal, min. collaboration	Specific moments	None		
Primary Data Type	Audio	Video	Raster Image	Vector Image	Transaction Records	Time Series Data	
Data Quality	Inconsistent	Duplicate	Incomplete	Outdated	Biased	Noisy	Corrupted
Data Security	Compliance	Data Encryption	Access Control	Data Integrity	Data Privacy	Incident Management	Audit & Monitoring

Figure 14.2: Morphological box to support AI tool development. Obtained from [83]

14.3.5. Review of artificial intelligence applications in engineering design

The framework presented in the article by Yüksel *et al.* [4] provides a comprehensive overview of how AI methods can be applied to engineering design. They include both classical AI methods (e.g., expert systems, genetic algorithms, fuzzy logic) and modern AI techniques (e.g., machine learning, deep learning) to address the different needs of the engineering design process. It emphasizes tailoring AI methods to specific stages and challenges within the design lifecycle. This framework serves as a roadmap for systematically integrating AI into engineering design workflows, enabling more efficient, innovative, and adaptable outcomes.

14.3.6. The AI Methods, Capabilities and Criticality Grid

Schmid *et al.* [3] developed a three-dimensional classification scheme for AI applications. The three dimensions they included were AI methods, capabilities and criticality. From these, the capabilities dimension is most relevant for this research, as the others are too technical or classification-focused. The capabilities overview, however, was found to be very elaborate and detailed. It is presented in Figure 14.3.

14.3.7. Other considered works

Some notable mentions other interesting works were identified that did not make the final selection: Strobel *et al.* [86], Lupetti and Murray-Rust [54], Furtado *et al.* [87], Hughes *et al.* [88], Brereton *et al.* [89], Sahu [90], Pradas Gomez *et al.* [91]. Informative, non-scientific sources include documentation from Microsoft[1], Google[2], and Apple[3] (last accessed: 1-10-2025).

[1] https://learn.microsoft.com/en-us/ai/playbook/technology-guidance/generative-ai/getting-started/llmops-checklist
[2] https://pair.withgoogle.com/guidebook/patterns
[3] https://developer.apple.com/design/human-interface-guidelines/machine-learning

CLASSIFICATION OF CAPABILITIES			EXAMPLES
SENSE	External	See	Optical Character Recognition
			Object Recognition
			Gesture Recognition
		Hear	Speech Recognition
			Sound Recognition
		Smell	Fragrant Detection
			Acid Detection
			Burnt Detection
			Caprylic Detection
		Taste	Sweetness Detection
			Sourness Detection
			Saltiness Detection
			Bitterness Detection
			Umami Detection
		Touch	Temperature Detection
			Pressure Detection
			Pain Detection
	Internal	Body Awareness	Detection of Self Movement
			Detection of Body Position
		Balance	Balance Detection
PROCESS AND UNDERSTAND	Factual	List	Context-specific Terminology
		Summarize	Automated Report Generation
		Respond	Data Association
		Select	Semantic Search
		Check	Parsing (Syntactic Analysis)
		Generate	Deductive Knowledge Extraction
	Conceptual	Recognize	Named Entities Recognition
		Classify	Semantic Domain Recognition
		Provide	Explanation
		Differentiate	Disambiguation
		Determine	Semantic Interpretation
		Assemble	Interlingual Translation
	Procedural	Recall	Process Memory
		Clarify	Environmental Modeling
		Carry Out	Discourse Modeling
		Integrate	Sensory Data Fusion
		Judge	User Modeling
		Design	Human Processor Models
	Metacognitive	Identify	Strategy Selection
		Predict	State Estimation
		Use	Transfer Approach
		Deconstruct	Code-Switching Strategies
		Reflect	Self-optimzation Methods
		Create	Generation of Narratives
ACT	Physical	Motion Planning	Motion Planning with Uncertainty
		Sensors and Manipulators	Passive und Active Sensors
		Kinematics and Dynamics	Dynamical Movement
		Human-robot Interaction	Multimodal Teleoperation
	Non-physical	Software Agents	Process Automation
			Transactional Systems
			Chatbots or Customer Service
COMMUNICATE	Natural Language Processing	Text Generation	Paraphrasing Tools
		Machine Translation	Interlingual Translation
		Text Analysis	Parsing (Syntactic Analysis)
		Information and Knowledge Extraction	Named Entities Recognition
		Information Retrieval	Semantic Search
		Document Analysis	Semantic Domain Recognition
		Spoken Dialog Systems	Clarification Dialogue
			Generation of Narratives
	Human-Machine Interaction	Cognitive Systems	User Modelling
		Interaction Paradigms and Modalities	Multimodal Interaction

Figure 14.3: Comprehensive overview of AI capabilities obtained from [3].

14.4. CONCLUSIONS FROM FOLLOW-UP LITERATURE STUDY

A number of conclusions were drawn based on the follow-up literature study that was performed.

NO EXISTING MORPHOLOGICAL CHARTS FOR GENAI SYSTEMS

A morphological chart for generative AI systems could not be found, despite extensive searches in Scopus, Google Scholar, and with "regular" Google image searches. It is concluded that such a chart has not been developed before.

EXISTING FRAMEWORKS: BY AI EXPERTS, FOR AI EXPERTS

Many design frameworks for GenAI systems were found, with varying levels of comprehensiveness and maturity. However, an overarching similarity is that all seem to be created by AI experts, for AI experts. This important conclusion is based on the two main findings.

Firstly, the existing frameworks represent the design problem as a set of design aspects (i.e. dimensions) or questions that have to be addressed. However, notably absent in the existing frameworks is an overview of the main design *options* for each of those aspects/dimensions. For example, Weisz *et al.* [2] directly address the AI system developers with things that they should consider, such as "Ensure the AI system solves real user issues and minimizes user harms.". However, they do not provide an overview of the ways *how* to achieve this. Conversely, when existing works do include specific options, they focus on a small subset of design aspects. For instance, Hughes *et al.* [88] and PricewaterhouseCoopers [92]. A framework covering all system-level design aspects and the corresponding design options for each of the design aspects was not discovered.

The second aspect indicating that the existing frameworks are geared towards AI experts is their wording. The GenAI frameworks typically use a lot of jargon and technical AI terms in their descriptions. For example, Schmid *et al.* [3], Strobel *et al.* [86], or Shi *et al.* [85]. Therefore, they are very difficult to understand and use by people who lack in-depth AI expertise, like (engineering) domain experts. In combination with the previous characteristic, this led to the conclusion that existing frameworks were created *"by AI developers, for AI developers"*.

OPPORTUNITY FOR NOVEL FRAMEWORK

Based on the previous conclusions, a gap is identified in the form of an AI design framework that is geared specifically towards (engineering) domain experts. This gap could be addressed by designing a novel framework that **avoids AI jargon**, and presents an **overview of the options** and possibilities for each of the AI system design dimensions, instead of merely listing questions and factors that should be accounted for by the AI system designers. Correspondingly, the following **design requirement** was formulated and added to the list presented previously in section 15.1:

> *The morphological chart should be understandable as a **stand-alone communication artifact** to domain experts without in-depth GenAI knowledge.*

15

Main design iterations

This chapter presents the method and findings used to iteratively develop the communication tool. In the figure below the main iterations steps (blue) and activities used to generate are outlined from left to right. The setup of this chapter is based on the main iterations steps outlined in this figure. The methods and their findings related to each iteration are discussed respectively in dedicated sections.

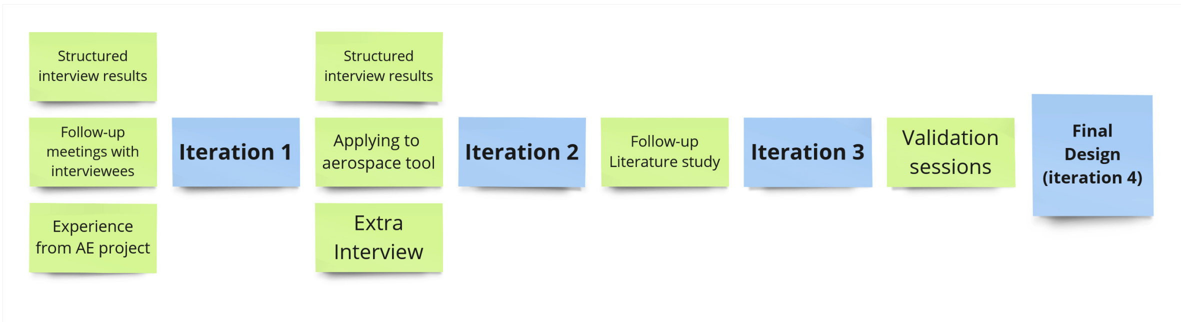


Figure 15.1: The iterative design process used to design and implement the morphological chart. The sequence of methods (green) that provided input to the next iteration (blue) are ordered from left to right.

High-level design breakdown and approach

The design of the morphological chart was approached by breaking it down into three main aspects. Firstly, the *usage*, i.e. the method by which AI and domain experts should use the tool. Are there any preceding steps? And when (in the timeline of the project) to use it? Secondly, the *content* which is basically the answer to the question which design dimensions and, for each of these dimensions, which specific design options should be included. Finally, there is the *visual* aspect, so what will it look like? This includes, for example, determining how to visually organize design dimensions and options, and using icons to represent them.

The focus of the first iterations (1-3) was on the content. The visual design was initially completely disregarded but in iterations 2 and 3, it was gradually given some more attention in preparation for the validation sessions. In these validation sessions, the usage was determined. For the last design iteration, all information was combined and all three aspects (usage + content + visual design) were finalized. The final design is presented in the next chapter.

15.1. Design iteration 1: creating the initial prototype

As previously discussed in section 12.4 and , the final part of the last two interviews (with AI-1 and AI-3) focused specifically on the morphological chart idea instead of discussing all solution ideas. This allowed gathering input from these experts regarding the potential implementation of this idea. Additionally, a follow-up meeting with AI-2 was organized to obtain similar input from them. The interview findings and meeting notes from these sessions are included in Appendix H. A summary of the most important findings from these sessions, typically mentioned by multiple experts, is presented in subsection 15.1.1. The input from these experts was one of the main ingredients used to create the initial prototype. Additionally, the knowledge and experience from the previous aerospace part of this thesis was also used to brainstorm a list of possible design dimensions and corresponding options. This list is provided in Appendix I. An early version of this list was also used as a basis for discussion during the follow-up meeting with interviewee AI-2. The initial prototype that was created, based on all collected information is presented in subsection 15.1.2.

15.1.1. SUMMARY OF DESIGN INPUT FROM INTERVIEWED AI EXPERTS

- **Understanding AI Capabilities and Limitations:** A recurring theme across interviews was the need for domain experts to gain a clearer understanding of the capabilities and limitations of generative AI. Participants highlighted that misconceptions about what AI can achieve often hinder effective collaboration. For instance, while AI can generate conceptual designs or basic sketches, it is not yet capable of producing fully functional Computer Aided Design (CAD) models or manufacturable outputs. Similarly, understanding the GenAI development lifecycle, including training data requirements and intermediate steps in the GenAI pipeline, was deemed essential for fostering mutual understanding between AI experts and domain experts.
- **Transparency and Human-in-the-Loop Systems:** Transparency in AI processes was emphasized as a critical feature for the communication tool. Interviewees noted the importance of intermediate outputs being accessible to domain experts, allowing them to verify and approve these steps before final outputs are generated. This approach supports the inclusion of human-in-the-loop workflows, where domain experts actively validate AI-generated results to ensure trust and alignment with project goals.
- **Role of GenAI within the tool:** The interviews underscored the importance of defining the tool's purpose and level of automation. The communication tool must clarify GenAI intended role: whether it serves as a supportive agent offering suggestions, an autonomous decision-maker, or a system under the complete control of the user. The tool should also allow to define the level of integration with existing workflows and engineering tools, such as KBE or Finite Element Modeling.
- **Preconditions for Success:** Several preconditions were identified as critical for the successful implementation of AI-driven systems. It would be beneficial for the communication tool to make this explicit. Examples of these preconditions are:
  - Well-documented workflows and structured data.
  - Effective risk management practices, such as the four-eyes principle, to support human-in-the-loop systems.
  - A clear division of responsibilities between designers, other software tools, and the new GenAI-powered system to avoid confusion and inefficiencies.

15.1.2. INITIAL PROTOTYPE

The full list of ideas for possible design dimensions and options (Appendix I) This list was created based on the input from follow-up meetings, combined with experience and previous interview results. Subsequently,

To create the first prototype of the tool the design input from experts was used as a guide to refine and extend the complete list of possible design dimensions and options from Appendix I. Thereby the first prototype was obtained, which is presented in Figure 15.2.

15.2. ITERATION 2: ADDITIONAL INPUT FROM EXPERTS

15.2.1. INTERVIEW ABOUT FIRST PROTOTYPE

The first prototype of the communication tool was put in front of a new interviewee (AI-4) who was not part of the first round of interviews. This interviewee was recruited based on recommendation by one of the interviewees from the first round. They were highly experienced in the field of (Gen)AI, having contributed and overseen more than 300 AI projects.

The interview consisted of two main blocks. During the first block, the research was introduced and the findings from the problem exploration phase were introduced. The interviewee was asked to reflect on the

Design dimension	Design options (textual)	Option 1	Option 2	Option 3	Option 4
GenAI integration with existing systems	1. Stand-alone AI system 2. Add-on to existing system 3. Fully integrated into existing workflow 4. Existing system(s) integrated into central GenAI	Stand-alone system	Add-on to existing system	Fully integrated	Existing systems centralized into AI
Control & autonomy of GenAI	**1. Intermittent AI, human in control:** human initiates AI (either iterative interaction or batch processing) **2. Continuous AI, human in the loop**: human directs or accepts AI actions **3. Continuous AI, human on the loop**: human monitors AI working on tasks **4. Proactive AI, human out the loop**: AI initiates tasks and has siginificant autonomy				
GenAI purpose	1. Intelligent search engine 2. Creative generation 3. Data extraction/transformation (bridging function) 4. Summarization, interpretation (analysis) 5. Decision making (support)				
Type of input(s)	1. Unstructured text (eg natural language) 2. Semi-structured text (eg source code) 3. Structured text: datafiles (eg .csv) 4. Images 5. Audio 6. Video 7. Others? (eg sensors, binary, ...)				
Context data collection	1. User provides all input data 2. Non-AI system provides (additional) input data 3. AI searches for (additional) input data, online or in custom database				
AI information access	1. Internet 2. Company intranet/wiki 3. Engineering database 4. Personal communication (eg email) 5. ... others?				
Explainability & Transparency	1. Stepwise traceability + Deterministic 2. Stepwise traceability + Non-Deterministic 3. Black-box approach + Deterministic 4. Black-box approach + Non-Deterministic				
Type of output(s)	1. Unstructured text (eg natural language) 2. Semi-structured text (eg source code) 3. Structured text: datafiles (eg .csv) 4. Images 5. Audio 6. Video 7. Others? (eg sensors, binary, ...)				
Viable product launch maturity	1. Prototype (TRL 4/5) 2. Minimum-Viable Product (TRL 6) 3. V1.0 spec (TRL 7) 4. Mature and extensively tested (TRL 9)				

Figure 15.2: First prototype of the morphological chart.

identified challenges. Thereby, additional input for the problem definition was obtained. The second part of the interview was focused on gathering input on the developed prototype.

During the meeting, notes were kept. These were processed after the meeting into a 2-page summary which was sent to the interview to verify and approve that everything was understood correctly. They approved the content.

DO YOU RECOGNIZE THESE FINDINGS? ANY ADDITIONS OR COMMENTS?

Yes, in addition the following comments:

1. Often the requirements of the potential solution are not clear. In particular with regard to the user experience and the training data. This applies to both sides (AI developers + end users/clients). Too often in a project they continue based on assumptions.
2. Clients/end users can have unrealistic expectations, for example that a model will have 100% accuracy. Some even view AI as something magical. When subsequently using the tool, they are easily disappointed and have a tendency to give up (after 1 try). Therefore, expectation management is crucial. Both in terms of realistic performance expectations, as well as usage of the tool (e.g. prompting). They need to learn how to use the final tool in order to get the expected performance out of it.
3. These issues can be mitigated by proper team composition. End-users should be actively engaged as part of the development team throughout the project.

WHAT IS YOUR CURRENT APPROACH TO ADDRESS THESE CHALLENGES?

We use CRISP-DM (CRoss Industry Standard Process for Data Mining) as project life cycle approach. Formalized processes, including end user input/feedback/engagement, are defined for the steps defined in this process.

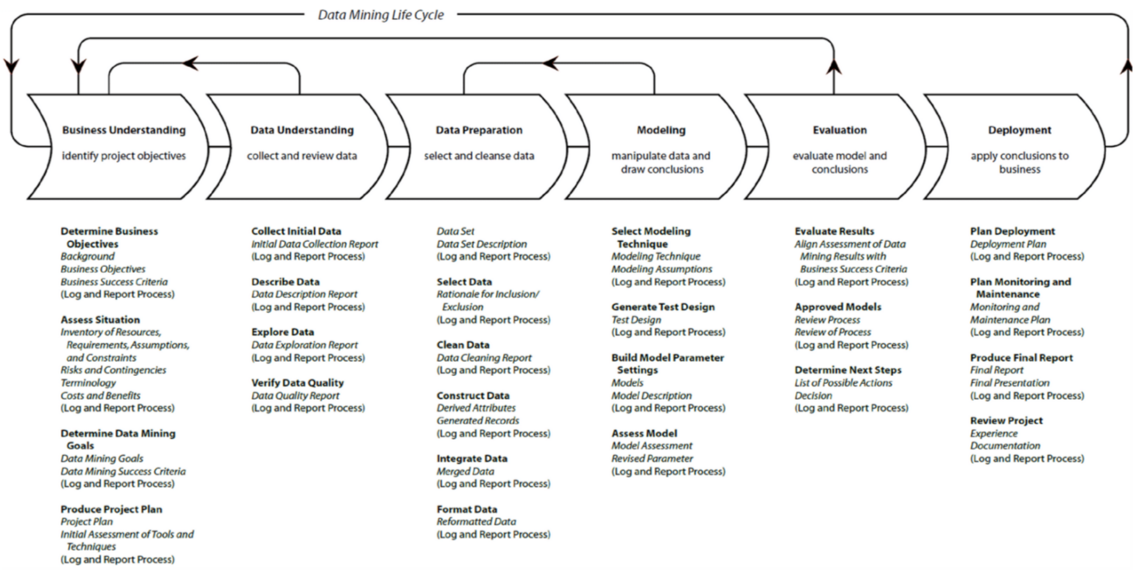


Figure 15.3: CRoss Industry Standard Process for Data Mining

WHAT ARE YOUR INITIAL THOUGHTS ABOUT THE MORPHOLOGICAL CHART CONCEPT + USAGE METHOD?

Very important to start with how things are currently done. High-level usage scenario should look something like this:

1. Sketch current engineering workflow
2. Specify desired improvement à define this using KPI's (current + target)
3. Perform impact analysis: evaluate all points in the current engineering workflow where GenAI provides potential for improvement. For every point, the 4 aspects below should be evaluated.
   (a) Development cost
   (b) Ease of development
   (c) Ease/cost of integration (adoption, learning, training)
   (d) Ease/cost of maintaining
4. Use morphological chart to characterize envisioned GenAI system

Additional notes on above:
The impact analysis + corresponding advice (step 3) is performed because clients generally have a poor understanding of suitable applications of GenAI technology within their workflows. In 99% of cases, clients approach us with the wrong question. They ask to automate step X with GenAI, but then the impact analysis shows that improving step Y is much more cost-effective. This is true despite the technical knowledge of clients (engineering organizations). In summary: Engineering clients are not good at identifying their own problem cases where GenAI is a viable solution.

Using the morphological chart in step 4 might not be necessary anymore, because different design options were already conceptualized and assessed by the AI team during impact analysis (step 3).

WHICH KEY DESIGN ASPECTS COULD BE ADDED TO MORPHOLOGICAL CHART?

It would be valuable to incorporate the following design aspects/questions:

- **Security & Privacy:** align with organizational data policy. For example, should all data be kept internal? Is it okay to send to a server outside EU?

- **User experience:** Understand/define the potential prompts. Which questions will the end user pose to the model? How will they specifically formulate these?
- **Life cycle management**: How will the GenAI system be evaluated and improved over time?

ANY FINAL COMMENTS OR INSIGHTS?

Make use of existing tools/capabilities, and incorporate them into my tool and methodology. For example, OpenUML or www.websequencediagram.com. The latter is particularly interesting. A key feature is its user-friendly textual language to quickly define and iterate on sequence diagrams. Additionally, many templates (created by user community) exist for a variety of interaction patterns. It would be useful to create additional templates specifically for GenAI user/system interactions.

15.2.2. APPLYING FIRST PROTOTYPE TO REPROCESS TOOL FROM AEROSPACE THESIS

In this exercise, the morphological chart was “filled out” for the REProcess prototype that was developed for the aerospace part of this thesis. This constituted selecting the design option(s) for every design dimension in the morphological chart prototype according to the design decisions made for the REProcess prototype. This lead to the following result:

1. **GenAI integration with existing systems:** 3. Integral component of envisioned system or workflow
2. **Control & autonomy of GenAI:** 1. Intermittent AI + human in control: human initiates AI (batch processing)
3. **GenAI purpose:** 3. Data extraction/transformation (bridging function) + 4. Summarization, interpretation (analysis)
4. **Type of input(s):** 2. Semi-structured text (eg source code)
5. **Context data collection:** 2. Non-AI system provides (additional) input data
6. **AI information access:** 5. Other: None
7. **Explainability & Transparency:** 1. Stepwise traceability + Deterministic
8. **Type of output(s):** 1. Unstructured text (eg natural language)
9. **Viable product launch maturity / Targeted product maturity for launch:** 1. Prototype (TRL 4/5)

CONCLUSIONS FROM THIS EXPERIMENT

A review of these results led to a number of insights and conclusions. First and foremost, the representation of the REProcess prototype by the morphological chart is still extremely high-level and abstract. Based on this information alone, an outsider who does not know the REProcess tool would still have no idea what the actual software system would look like. Adding additional contextual information, like a sketch of the current engineering workflow and the highlighting the step where the envisioned GenAI system should intervene, will be crucial. Furthermore, the existing definitions of the dimensions and design options included in the morphological chart should be refined and additional design dimensions should be added to make the resulting specification more fine-grained and informative.

15.2.3. SECOND PASS THROUGH ORIGINAL INTERVIEW RESULTS

During this iteration, the original interview results presented in chapter 12 were also reviewed again for inspiration. Specific statements were linked to related aspects of the morphological chart design in a dedicated column of the morphological chart, which is shown in Figure 15.4.

15.2.4. SECOND MAJOR ITERATION

Based on the findings from the design activities discussed in this section, a second major iteration step was performed. The result is presented in Figure 15.4. For every design dimension, the related input is listed (column 4).

Category	Design dimension	Design options (textual)	Input from experts:
Scope & Purpose	Integration with existing systems	1. Add-on to existing system 2. Integral component of new workflow 3. Stand-alone AI system 4. AI becomes central component, integrating other (existing) systems	AI-2: LLM Integration Approaches: – Embedding a tool into an LLM. – Embedding an LLM into an existing tool (e.g., Copilot). – Using an LLM alongside existing tools
	Role of GenAI	1. Intelligent search engine 2. Creative generation 3. Data extraction/transformation (bridging function) 4. Summarization, interpretation (analysis) 5. Decision making (support)	AI-1 Clarify purpose and approach (DIY vs agentic)
	Targeted PLM phase	1. Conceptualization 2. Development 3. Production 4. Service & Support 5. Retirement	AI-3: "one key AI concept that is misunderstood by domain experts is the capabilities of generative AI models. What it can do currently, what it cannot, and what outputs it can produce. So, for example, what kind of representation a generated CAD
Limitations & Prerequisites	Output quality assessment	1. Quality **quantification** methods **exist** 2. Quality **quantification** methods **can be defined** 3. Robust **qualitative** assessment methods **exist** 4. Robust **qualitative** assessment methods **can be defined** 5. **Unfeasible** to assess output quality **robustly**	AI-2: Proving Effectiveness: – What data proves the system works? – How critical is it to validate this, depending on the use case?
	Training data	1. High-quality, labelled training dataset **available** 2. Training data unavailable, but **can be retrieved** 3. **Manual generation** of training data required 4. Obtain training data **retrospectively** (from users of MVP implementation) 5. **Unfeasible** to obtain training data -> project not possible	AI-2: precondition for success, Is the data well-organized and properly meta-tagged? AI-3: Most important knowledge to share: The ML workflow of AI developers and required training data, AI-4: Often the requirements of the potential solution are not clear. In particular with regard to the user experience and the training data
	Level of work process formalization	1. Formalized, well-documented work processes, incl risk management 2. Formalized, well-documented work processes 3. Undocumented but standardized work processes 4. Undocumented, flexible work processes	AI-2: Are proper risk management and control practices, like a four-eyes principle, already used and/or defined in formal processes? This enables easy human-in-the-loop implementation AI-4:
	Data Security & Privacy*	1. No issue to send data to servers outside EU 2. Only EU servers allowed 3. Private online servers are allowed 4. Only internal servers (intranet) are allowed 5. Only on-device compute is allowed	AI-4: Security & Privacy: align with organizational data policy. For example, should all data be kept internal? Is it okay to send to a server outside EU?
User Experience	User control & system autonomy	1. Intermittent AI: human in control, initiates AI (either iterative interaction or batch processing) 2. Continuous AI: human in/on the loop, directs, accepts, monotors AI actions (background or foreground) 3. Proactive AI: human out the loop, autonomous AI initiates actions	<- taken from AI-2 + Tsiakis 2024 HAI interaction
	Workflow Flexibility & Automation	1. Static workflow (includes customizable user input settings) 2. Human-in-the-loop decision making & feedback 3. Neuro-symbolic AI system (rule-based, fuzzy logic) 4. AI agent controls workflow	AI-1: Workflow flexibility: "How much freedom does the generative AI have in in answering your questions? Is it like what we said before? Is it really fixed flow of things to do? Or is it depending on the output to do one thing or the other? Is it a
	Iteration & approval of GenAI output	1. No iteration or approval 2. Manual correction by user 2. Output approval (+ recompute if rejected?) 3. Choosing from options 4. Prompt-based iteration 5. Multimodal iteration (eg highlight part of image + specific	AI-2: Iterating on Output: Input into another system (e.g., code), Iteration within the chat interface (e.g., editing the second sentence), Copy-pasting elsewhere for further refinement without returning to the original system. AI-4: Often the requirements of the potential
	Required user training	1. Presentation 2. Documentation ('getting started') 3. Tutorial (interactive, online, video) 4. Training sessions	AI-4: They need to learn how to use the final tool in order to get the expected performance out of it.
GenAI Model Input	Input modality (context data)	1. Unstructured text (eg natural language) 2. Semi-structured text (eg source code, legal contracts, technical documentation) 3. Structured data files (eg spreadsheet, 3D model) 4. Images 5. Graphs 6. Other: ... (eg audio, video, sensors, binary, ...)	AI-1: how much knowledge is taken from the Generative AI and how much is done by traditional engineering tools / systems (like KBE app, CFD, FEM, etc.). And how much still by designer themselves? AI-2: LLM Capabilities: LLM = knowledge base + logic skills -> advisable approach is to rely more on
	Context augmentation approach	1. None, rely on knowledge embedded in AI model itself (not advised) 2. **User provides** additional input data 3. Non-AI system **retrieves** (additional) input data (incl context storage) 4. Non-AI system **computes** (additional) input data 5. **AI searches** for (additional) input data, online or in custom	AI-1: how much knowledge is taken from the Generative AI and how much is done by traditional engineering tools / systems (like KBE app, CFD, FEM, etc.). And how much still by designer themselves? AI-2: LLM Capabilities: LLM = knowledge base +
Output	Transparency	1. Black-box approach (not advised) 2. Stepwise traceability (process logs) 3. Piecewise traceability (input-output pairs)	AI-3: Provide maximum transparency how the AI model is operating or is being run
	GenAI Determinism	1. Deterministic (recomputations are identical) 2. Non-deterministic (recomputations are unique)	Personal experience
	Output modality	1. Unstructured text (eg natural language) 2. Semi-structured text (eg source code, legal contracts, technical documentation) 3. Structured data files (eg spreadsheet, 3D model) 4. Images 5. Graphs 6. Other: ... (eg audio, video, sensors, binary, ...)	AI-2: humand-readable or machine-readable? AI-3: Not only type of output but also its **maturity**
	Target level of output maturity	1. Proof of concept prototype (TRL 3/4) 2. Minimum-Viable Product (TRL 6) 3. Mature and qualification tested (TRL 8)	AI-2: Proving Effectiveness: How critical is it to validate and prove the system works depending on the use case?
Outlook	Future developments steps	1. Improve output quality and maturity 2. Develop additional features to expand capabilities 3. Integrate into larger system 4. Connect to other tools/systems/platforms 5. Other: ...	AI-4: Life cycle management: How will the GenAI system be evaluated and improved over time?

Figure 15.4: Second major communication tool iteration

15.3. ITERATION 3: FOLLOW-UP LITERATURE STUDY

The third iteration was created by taking each of the frameworks presented in section 14.3 and using it as a lens to reflect on iteration 2 of the framework. Every individual item of each literature frameworks was reviewed and categorized as: 1) directly related to design dimension(s) already included in the morphological chart, or 2) not related to any design dimension of the morphological chart. In case of the former, the literature item was added as a note to the corresponding design dimension in the morphological chart. After all literature frameworks were reviewed, these notes were used to refine the corresponding definitions. This resulted in the next iteration of the morphological chart. The following design dimensions were modified:

- Role of GenAI
- Targeted PLM phase
- User control & system autonomy
- Workflow Flexibility & Automation
- Iteration & approval of GenAI output
- Input modality (context data)
- Context augmentation approach
- Transparency
- Future developments steps

In case items were found to not be related to any specific aspects of the current morphological chart, they were evaluated based on relevance and potential added value. If they were deemed potentially interesting to incorporate, they were stored in a separate list of applicable findings. After reviewing every individual component from the literature frameworks, some general comments about the applicability and insights were captured as well. These individual analysis findings per framework are provided in Appendix J. Potential design dimensions to be added on the basis of this analysis step are described below:

- **Add dimension about risk (mitigation) features:** Many frameworks explicitly included features related explicitly to risk mitigation, such as highlighting high levels of uncertainty in GenAI output, and notifications to the user about potential to hallucination, potential bias, etc.
- **Add (even more) user-related design dimensions and options:** Additional characterizations of users and user interaction were found. For example, user output annotation and curation features, and characterization of the end-users by expertise (IT-expert, IT savvy, or layman) or their domain (writing, programming, robotics, etc.).

After some deliberation, it was decided to not include additional design dimensions in the third design iteration. The main reason for this design decisions was made based on the already large size of the morphological chart, which already consisted of 85 different design options spread out over 18 design dimensions. Instead, these additional findings will be revisited after testing the tool in the next iteration step.

15.3.1. LAYOUT UPDATE

In preparation of the testing and validation sessions a visual improvement to the communication tool was made. Instead of simply listing the design options as a textual list, they were now organized in columns. This made the tool much more recognizable as a morphological chart and facilitated the side-by-side comparison of design options participants needed to perform during the testing and validation sessions.

The version of the morphological chart used during validation is presented in Appendix K.

15.4. ITERATION 4: VALIDATION SESSIONS

The testing and validation sessions were set up to assess to what extent the developed morphological chart is successful in educating domain experts about GenAI systems (regarding capabilities, prerequisites, required investment) and facilitating accurate communication and expectations regarding envisioned GenAI systems. By adopting this dual focus, the validation sessions will also help to determine the most suitable application for the developed tool (i.e. use case 1 vs. use case 2) and thereby provide an answer to Research Question 4. During the validation sessions, the morphological chart was assessed on the following aspects:

1. **Clarity and understandability:** Do participants understand how to work with the morphological chart? Can they understand the design dimension descriptions and their corresponding options?
2. **Coverage:** Does the set of design dimensions and options cover all important areas? Should certain dimensions or options be added or removed?
3. **Educational benefits:** To what extent does the morphological chart improve the understanding of domain experts about capabilities and the possible design space of GenAI-powered tools?
4. **Ideation support:** To what extent does the morphological chart support ideation of novel GenAI-powered systems?
5. **Communication benefits:** To what extent does the morphological chart facilitate clear and accurate communication in order to help to align on the vision and expectations regarding novel GenAI-powered systems?

15.4.1. VALIDATION SESSION PARTICIPANTS

Two different validation sessions were set up. Both sessions were conducted with domain experts who had not been involved previously involved in the research. The first was conducted with a single domain expert (DE-4) who fulfills a chief engineer position in a large research and technology organization (+/- 500 engineers). Given that this session was conducted with only one expert, the emphasis of this session was on the first four aspects and somewhat less on communication because the communicative qualities could not be directly assessed.

In the second session, all 5 aspects were investigated but the emphasis of the second session, which was performed with 2 participants, was more on the communication benefits. Ideally, this second session would have been performed with a domain expert and AI expert to really test the communication benefit in the target scenario. However, a suitable setting for such an experiment could not be found. Therefore, the next best option was chosen, which was a setting with two domain experts (DE-5 + DE-6) who would discuss a novel GenAI-powered system. These domain experts were colleagues working in an academic environment.

15.4.2. VALIDATION SESSION SETUP

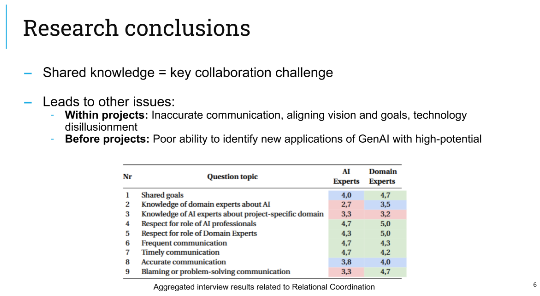
In terms of the overall process, a similar setup was used as for the initial round of interviews. MS Teams was used to record and automatically transcribe the sessions. Additionally, notes were made during the observation phase. Participants were informed of this setup and data capturing methods in the initial invitation, and a similar consent form as for the structured interviews (Appendix G) was signed by all participants.

The validation sessions consisted of three main phases. **The first phase** was the introduction. Here, a brief introduction to the research and background of the communication tool was provided so participants could understand the context and the problems that the communication tool is intended to address. **The second phase** revolved around capturing the first impressions and reflection of the participants about the morphological chart. Before introducing the chart itself, the participants were instructed about this intent. They were asked to think and reflect out loud while reading through the morphological chart from top to bottom. The idea behind this method was to capture their real and honest responses, rather than socially desirable answers, which is essential for proper validation. For this same reason, an additional explicit instruction was provided that the goal of the session was to *learn*, so participants were asked to avoid socially desirable answers and be honest. The main slides used to introduce the research and outline this exercise for phase 2 are presented in Figure 15.5.

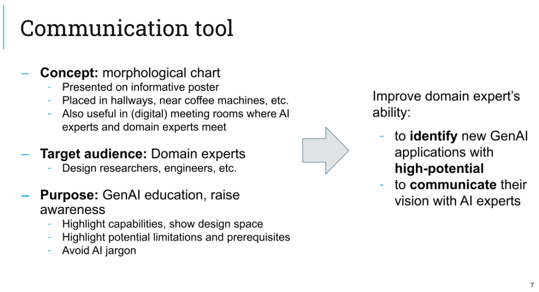
Note: at the particular moment in time that the validation sessions were conducted, the expected design direction was more towards the educational use case (through design of an education poster) than the communication use case. This explains why the communication tool introduction (slide c) is somewhat biased towards this.

In session 2, the second phase included another exercise after capturing the first impressions from the participants. This second exercise consisted of using the morphological chart to discuss a new GenAI-powered engineering tool that was previously conceptualized by one of the participants (DE-5). The other participant (DE-6) was not yet aware of this idea. Hence, during this exercise participant DE-5 initially took the lead to

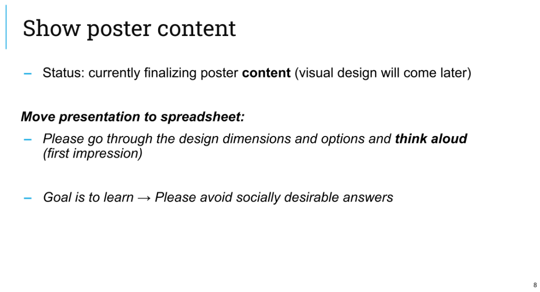
(a) Slide: Science communication introduction



(b) Slide: Research conclusions (presents Table 12.1)



(c) Slide: Communication tool introduction



(d) Slide: instruction for phase 2 exercise

Figure 15.5: Four slides used during the validation sessions to introduce the research and outline the test assignment

explain the idea. Subsequently, after the intended context and concept of the tool was broadly explained, the instruction was to use the morphological chart to further discuss and refine the idea together.

During both sessions, the researcher was present in an observer role. When encountering items that really could not be resolved without intervention, the researcher stepped in to clarify the intent and meaning of a particular concept so it could be used as intended. This was only done after a waiting period. When such clarifying interventions were required, this was noted as a point of improvement.

The third phase of both validation sessions consisted of reflection and evaluation questions to assess to what extent the communication tool could achieve the intended benefits. The four main evaluation questions are outlined below. Note that each question consists of a quantitative and qualitative part. This allows for optimal comparison of the answers while also obtaining detailed insights. During the second session, both participants were asked to write their answers down rather than discussing them out loud as to prevent them from influencing each other.

1. To what extent does my tool help DE to better understand the GenAI **capabilities**?
   - scale of 1 - 5
   - suggestions?
2. To what extent does my tool help to better understand the **possible design space** of GenAI systems?
   - scale of 1 - 5
   - suggestions?
3. To what extent do you think it can help with **ideation of GenAI** systems?
   - scale of 1 - 5
   - suggestions?
4. To what extent do you think it can help **communication about GenAI** systems with AI experts?
   - scale of 1 - 5
   - suggestions?
5. Any final comments or suggestions?

15.4.3. VALIDATION SESSION RESULTS

The quantitative results corresponding to evaluation questions 1-4 from the list above are presented in Table 15.1. Based on this table, a number of very clear conclusions can be drawn. Namely, that the morphological chart is much better suited as a communication tool (use case 1) rather than a stand-alone educational tool (use case 2). When looking at the additional comments made regarding the positive score for ideation, it became clear that a key reason why participants provided this score was due to both providing options as well as highlighting prerequisites and limitations. This, and several other qualitative findings are discussed in the remainder of subsection. Thereafter, the results regarding clarity and understandability are presented.

Table 15.1: Quantitive results from validation sessions (scoring range: 1-5).

Evaluation questions	**Average score** (n = 3)	**Spread [lowest - highest]** (n = 3)
1. Understanding GenAI **capabilities**	2.3	2 - 3
2. Understanding GenAI **design space**	3.3	2.5 - 4.5
3. Help with **ideation**	4.2	4 - 4.5
4. Help with **communication**	4.3	4 - 4.5

SUITED APPLICATION CONTEXT AND USAGE

The differentiation between the lower scores obtained for understanding GenAI capabilities and design space, and the higher scores for ideation and communication can be well-understood based on the following comments made by participants about what they envision is the best application of the morphological chart:

- **AI-4:** "So the way I think that tool is good. For me, you would use that tool at the requirements capture stage of a new project where you're developing a new AI model. Before you showed it to me, I'd have thought that it might be something that you'd have up in a breakout area or on a wall somewhere that people could look at and get more awareness of what AI is capable of, but I don't think it explains capabilities that well. I think it's good at specifying requirements of systems."
- **AI-4:** "I think it does that [ideation] a little better than the other two [understanding capabilities & design space]. So, I'd put that at a four because if you're giving people options, they can kind of fill in the gaps as to what's possible."
- **DE-5** "By going through this, you as AI expert are inviting me to give you detailed information that is relevant for you to decide if it's doable with AI, or with this technology instead of the other. So me as a knowledge expert, a domain expert, I don't learn about AI. It's not that I become more literate about this."
- **AI-5:** "I see it more as giving you all the information that apparently is relevant for AI experts to understand how to match the tools and technology they have to the domain experts needs."
- **DE-6:** "This is a communication tool that helps the domain expert to tell the AI people what they would like to have, and that would then require the AI people to have a look, have a chuckle, and come back to the domain experts and say, we may need to talk."
- **AI-6:** Good preparation for a meeting between AI expert and domain expert to consider items that are relevant: "What I think it does really well is it prompts me to think about the things that the AI expert is likely to ask about".
- **AI-6:** "Helped notice a few possibilities, options"
- **AI-6:** "[The tool] Serves as reminder about training data and validation."

POSITIVE REACTIONS SHOWING LEARNING AND EXPANDING THOUGHT PROCESSES

Multiple design dimensions induced expressions that indicated they really expanded the participants thinking. For examples, utterances of "Aha", or "Hmmm, interesting...". Provoking such reactions is one of the main goals of the morphological chart so these reactions are deemed highly positive. An overview is provided below:

- **Interaction synchronization:** *Up front* option -> "Aah, interesting" - DE-4

- **User approval & iteration (of GenAI output):** “"Aah, not thought about it before but actually useful to think about. Last 4 options all useful, only thought about approval or manual correction. [...] interesting to discuss with AI expert” - DE-6

  “This a category I didn’t think about before that’s actually useful. So I would have gone with this one [option 3] first to generate the thing and then you’d see whether you like it or maybe you’ll correct the code manually. But this actually highlights there are that there are other ways to do this.” - DE-6

- **Method of collecting (additional) input data:** “I understand the reason to put it in though” - DE-4
- **Target level of output maturity:** “Phoe... ideally top level but what is feasible?” - DE-6
- **GenAI determinism:** “Hmm [long pause]. Yeah.” DE-4
- **Output quality assessment:** “Companies might want to apply existing quality processes to AI output. Good prompt to make the user think about. That’s good” - DE-4
  DE-5 and DE-6 both ask question out loud: “Can you actually quantify this?”
- **Training data:** “No idea how much data is needed [...] Hmm, that is actually very hard to find.” - DE-6
  “Forces domain experts to think about data as important requirements for the project” - DE-6
- **Current work process, level of formalization:** “That one really makes sense” - DE-4
- **Data Security & Privacy:** “Probably same answer for an entire company but good that its in there” - DE-4

Provide more information about feasibility and implementation effort of composed solution

The domain experts expressed a need for more information about the feasibility and implementation effort associated with developed a composed solution. An explaining factor is that an AI expert was not present during the validation sessions, who would normally provide this input.

- **DE-5:** “However, if I as a domain expert, go through this and say this is what I want, there is nothing guaranteeing that that is actually possible. [...] if I just put a wish list together there is a high probability that it’s not going to be feasible. ”
- **DE-6** “Not sure what types/sizes of models would be required, or what’s feasible.”
- **DE-6:** “I’d really like some feedback on what the choices **mean** for the task at hand.”
- **DE-6:** “It would have been really interesting if there’s like some complexity scale that indicates ’Yeah, what you’ve just put together is really hard to do’ or ’Yeah, this is easy’.”

Additional design dimensions

Some additional design dimensions were suggested:

- **Commercialization dimension** suggested by **DE-4:** “Businesses are driven by time, cost and quality. So, [I’d suggest] putting somewhere some reference in to what the expected business output is. Is this to increase throughput time, so making a process quicker, or is it improving the quality?”
- **Scope for model improvement** dimension suggested by **DE-4:** “daily model update or only one revision per year?”,
- **Internal dissemination dimension** suggested by **DE-4:** “Used by one person or entire business?”

Improving clarity and understandability

Overall, the tool was found to be mostly clear and understandable based on the results from phase 2 of the validation sessions (first impressions, thinking out loud, discussing new idea for GenAI-powered system). However, as noted by DE-5, *“Tool answers can still be ambiguous”*. Specifically, the following dimensions need to be modified to improve clarity:

- **Targeted product life cycle phase:** Design and Development currently in separate options, but they should be together.
- **Interaction synchronization:** The dimension itself was not immediately clear, but it became clear after reading the options. Some ambiguity remained about the boundary between real-time and delayed (Seconds? Minutes?)

- **Workflow Flexibility & Automation:** Not immediately clear, but icons helped participants to finally figure it out (mostly). The specific focus of this dimension on the internal processing workflow needs to be clarified.
- **User approval & iteration (of GenAI output):** Dimension really clear but difference options 1 and 2 not distinctive enough. Option 3 needs to be clarified more specifically.
- **Method of collecting (additional) input data:** More tricky to interpret. "Collection" term makes it unclear. It provoked thinking from project-level perspective instead of software system implementation.
- **Transparency:** redefine red text for option 1 to *generally not advised*
- **Training data:** MVP terminology not clear. Also redefine dimension to *Availability of training data.*

15.4.4. DESIGN CHANGES COMPARED TO ITERATION 3

An overview of the changed and added design aspects compared to the morphological chart used during the validation sessions (iteration 3) are presented below, along with associated rationale. These modifications aimed to improve usability, clarity, and applicability while ensuring the tool remains an effective communication aid for both AI experts and domain experts.

ADDED TITLE: *Ideation Matrix for Generative AI-powered Automation Systems*

The new title was chosen to better reflect the intended **purpose** of the tool. The term **Ideation Matrix** was preferred over **Morphological Chart** as it more clearly conveys the matrix's role in structured brainstorming and early-stage AI system design. Additionally, the title explicitly highlights **Generative AI-powered systems** with a focus on **automation**, ensuring that the scope is well-defined. The term "engineering" was deliberately omitted, as the final design turned out to be broadly applicable across different domains, with minimal engineering-specific content.

ADDED ICONS FOR IMPROVED CLARITY

Icons were integrated into the matrix to visually support certain design dimensions. However, not all dimensions were assigned icons to prevent excessive visual clutter. The inclusion of icons followed three key criteria:

1. Emphasizing dimensions that are less understood or commonly overlooked by domain experts. This decision was informed by insights from the aerospace project, findings from previous design iterations, and validation session feedback.
2. Enhancing the clarity of design dimensions and options that were identified as difficult to interpret.
3. Providing a visual complement to textual descriptions, improving comprehension where beneficial. Notably, for some dimensions (e.g., training data), attempts to introduce icons led to reduced clarity rather than improvement, and were therefore omitted.

All icons were sourced from the Flaticon online database and were customized to match the color scheme of their respective categories. The curated collection of icons is available online[1].

ENHANCED TEXTUAL DESCRIPTIONS FOR CLARITY

Text descriptions of various design dimensions and options were revised to improve clarity, particularly those that were flagged as ambiguous during the validation sessions.

VISUAL REFINEMENTS

The overall layout of the matrix was improved for better readability and usability. This included the introduction of **"danger triangle"** icons to indicate generally non-advisable design options. These warnings replaced textual descriptions to streamline the design while maintaining essential guidance. Additional clarifications and details were included as comments in the Excel version of the matrix[2]. Furthermore, references to the online version, authorship, and attribution were added at the bottom of the matrix.

[1] https://www.flaticon.com/collections/NTY1NjA3NzY=

[2] https://docs.google.com/spreadsheets/d/1DVCikZT1rKJ4ci35XkS2aqfTi0JoNHLD/edit?usp=sharing&ouid=102117738994161547080&rtpof=true&sd=true

ADDED A COMMENTS SECTION

A dedicated section for comments was incorporated into the matrix. This allows users to take notes during design discussions, capturing relevant considerations for each design dimension.

INCORPORATION OF ADDITIONAL DESIGN DIMENSIONS AND OPTIONS

Based on feedback, additional design dimensions were integrated, including **Target Users** and **Targeted Improvement**. These align with suggestions from participant DE-4, who referred to them as the **internal dissemination dimension** and **commercialization dimension**, respectively. Additionally, several previously identified dimensions that were set aside during iteration 3 were revisited and included to provide a more comprehensive representation of user-related design considerations.

REMOVAL OF THE FUTURE OUTLOOK DIMENSION

The **Future Outlook** dimension was removed as it contained only a single item and did not contribute significantly to the core objectives of the matrix. Efforts to expand this category led to distractions from the intended application context. To maintain conciseness and relevance, this dimension was excluded from the final design.

DEVELOPMENT OF A USER MANUAL

A structured user manual was developed to guide AI and domain experts in effectively utilizing the matrix. The finalized manual is presented in subsection 16.1.1.

16

Final communication tool design

This chapter presents the final design of the communication tool developed during this research, called the **Ideation Matrix for Generative AI-powered Automation Systems**. The final design and intended use (including user manual) are presented in section 16.1. Subsequently, the design rationale, outlining the changes with respect to iteration 3 validated in the previous chapter and alignment with the design requirements, is provided in section 16.2.

16.1. Final design: The Ideation Matrix

The **Ideation Matrix** serves as a structured communication tool designed to facilitate discussions between domain experts and AI experts in the early stages of GenAI-powered automation system development. It enables users to systematically explore design possibilities, align expectations, and assess prerequisites and limitations. The finalized design is presented on the next page. The matrix is organized into five main categories of design dimensions: Scope & Purpose, User Experience, AI Input, AI Output, and Prerequisites & Limitations. Descriptions of these categories and their corresponding design dimension and options are provided in subsection 16.1.2.

16.1.1. User manual for the Ideation Matrix

Purpose and Intended Use

The **Ideation Matrix** is a structured communication tool designed to facilitate the **initial design meetings** between domain experts and AI experts when exploring the development of a GenAI-powered automation system. The primary purpose of the matrix is to:

- Help domain experts and AI experts collaboratively **define and refine** project ideas.
- Provide a structured approach to making **abstract AI-related ideas more concrete**.
- Identify **key design dimensions, dependencies, and feasibility constraints**.
- Highlight **prerequisites and limitations** associated with implementing GenAI solutions.
- Serve as an **intake form** that domain experts can fill out in preparation for a subsequent meeting with AI experts (optional).

How to Use the Ideation Matrix

The Ideation Matrix is designed for use during the early-stage **scoping and ideation** phase of GenAI-powered system development. It can be used in two main ways:

1. **As a discussion guide during meetings** The matrix is used to facilitate a structured discussion to systematically explore the domain expert's needs, assumptions, and expectations. The role of the AI expert is to clarify technical implications and feasibility related to each design decision. The discussion progresses as follows:

(a) The domain expert describes their **initial idea** for an AI-powered automation system. They are recommended to visualize the envisioned novel working process to support their story, for example by drawing a flowchart.

(b) Subsequently, the AI expert and domain expert walks through the **design dimensions** in the matrix together. Using a physical copy or large format screen is recommended. For each design dimension, the AI expert and domain expert collaboratively discuss the design options corresponding to the envisioned concept. For each dimension, they either select one or more specific options to implement the envisioned system, or define it as an open question that requires further investigation.

Note: the dimensions are structured in a logical order, but adhering to the top-to-bottom order is not mandatory. A more flexible approach can be also be used.

(c) At the end of the session, an initial assessment of the system feasibility, implementation effort and added-value can be made based on the completed matrix. If this assessment is positive, the matrix also provides basis to define the follow-up steps.

(d) *Note: a final recommended step is to provide a physical copy of the filled-out matrix to the domain experts. They should be encouraged to take this back to their own team and show it to others, aiding the dissemination of the tool and contributing to improving AI literacy.*

2. **As an intake form (optional)** Before the meeting, the AI expert may send the matrix to the domain expert and request them to fill it out to the best of their ability. The matrix serves as a **preparatory tool** that structures the domain expert's input:

   - The domain expert selects one or multiple options per design dimension where they have a clear preference or requirement.
   - They may also cross out options that they believe are irrelevant or unsuitable.
   - If certain design dimensions are unclear, they may leave them blank. These will be discussed during the meeting.
   - The AI expert reviews the filled-out matrix before the meeting, allowing for a more focused discussion.

Ideation Matrix for Generative AI-powered Automation Systems

Category	Design dimension	← Design options →						Comments
Scope & Purpose	Integration with existing software (PLM, CAD, …)	Add-on to existing system	Stand-alone AI system	Integrating component	Centralizing AI system			
	Target users	Small group of specialists (internal)	Department or business unit (internal)	Entire organization (internal)	Multiple (partner) organizations (external)	Commercial product (external)		
	Role of GenAI within system	User interaction and communication	Generate (new or refined)	Diversify, augment, explore alternatives	Analyze, interpret, summarize	Decision making (support)	Transform (bridging function)	
	Targeted product life cycle phase (implementation context)	Conceptualization and Requirements	Design and Development	Verification and Validation	Production and Deployment	Operations and Maintenance		
	Targeted improvement	Do things better (quality)	Do things cheaper (cost)	Do things quicker (time)	Do more things (scope)			
User Experience	User-AI synchronization	AI runs up-front (e.g. suggestion pop-up)	Real-time, concurrent AI (e.g. chatbot)	Delayed AI output (e.g. batch processing)				
	Internal workflow complexity & user control	Static, fixed workflow (no user control)	Static workflow, user controls inputs	Dynamic workflow, human-in-the-loop	Dynamic workflow, logic-based control	Dynamic workflow, controlled by AI agent		
	Output iteration & approval	No iteration or user approval ⚠	Manual correction & approval	User chooses from options	Prompt-based iteration	Multimodal iteration (e.g. highlight + instruct)		
	Required user training	Instructional presentation	Training sessions	Coaching on the job	Instruction manual or videos	Interactive tutorial within tool		
AI Input	Type of additional input (besides instruction prompt)	Unstructured text (e.g. documents, web publications)	Semi-structured text (e.g. source code, legal contracts, req. specifications)	Structured data files (e.g. spreadsheets, 3D models)	Images	Graph data	Other: … (e.g. audio, video, sensors, robot control)	
	Method of retrieving additional AI input (context injection)	None, rely on GenAI model knowledge ⚠	User provides additional context data	Logic-based context injection	External computation of additional context	AI agent context search and retrieval		
AI Output	Type of output	Unstructured text (e.g. documents, web publications)	Semi-structured text (e.g. source code, legal contracts, req. specifications)	Structured data files (e.g. spreadsheets, 3D models)	Images	Graph data	Other: … (e.g. audio, video, sensors, robot control)	
	Transparency	Black-box approach ⚠	Stepwise traceability (process logs)	Piecewise traceability (input-output pairs)	Provide output rationale			
	GenAI Determinism	Deterministic (identical re-runs)	Non-deterministic (variation in re-runs)					
	Target level of output maturity	Proof-of-concept prototype (TRL 3/4)	Minimum-Viable Product (TRL 6)	Mature & qualification tested (TRL 8)				
Prerequisites & Limitations	Output quality assessment methods	Quality quantification methods exist	Quality quantification can be defined	Robust qualitative assessment exists	Robust qualitative assessment definable	Unfeasible to robustly assess quality ⚠		
	Training data	High-quality, labelled training dataset available	Training data unavailable, but can be retrieved	Manual generation of training data required	Obtain training data retrospectively (from users of implemented prototype)	Unfeasbile to obtain training data ⚠		
	Formalization of current work process	Formalized, well-documented process	Undocumented, standardized process	Undocumented, flexible processes ⚠	Novel process, undocumented ⚠			
	Data Security & Privacy	No issue to send data to servers outside EU	Only EU servers allowed	Private online servers allowed	Compute on internal servers (intranet) ⚠	Compute on local device ⚠		

Design options marked with ⚠ are generally non-advisable, specifics provided in comments in **digital version** available at: https://docs.google.com/spreadsheets/d/1DVCikZT1rKJ4ci35XkS2aqfTi0JoNHLD/edit?usp=sharing&ouid=102117738994161547080&rtpof=true&sd=true
Author: Joris Koopman, www.linkedin.com/in/jpkoopman
Icons: Distributed under creative common license, authors listed here: www.flaticon.com/collections/NTY1NjA3NzY=

16.1.2. DESCRIPTION OF DESIGN DIMENSIONS AND OPTIONS

The Ideation Matrix is organized into five main categories of design dimensions: Scope & Purpose, User Experience, AI Input, AI Output, and Prerequisites & Limitations. Descriptions of these categories and their corresponding design dimension and options are provided below.

SCOPE & PURPOSE

The **Scope & Purpose** category captures the fundamental role of the GenAI-powered system within an organization and defines the overarching goals of its implementation.

- **Integration with existing software**: How the AI system will be incorporated into existing workflows, whether as an independent tool, an enhancement to an existing system, or a fully integrated solution.
- **Target Users**: Who will interact with the GenAI-powered system, ranging from small expert teams to entire organizations or external stakeholders.
- **Role of GenAI within system:** The role of the GenAI component within the broader envisioned automation system. Highlights the development GenAI-**powered** systems, rather than focusing solely on the GenAI component itself.
- **Targeted product lifecycle phase**: The stage(s) in the engineering or operational process where GenAI-powered system will provide value, such as early conceptualization, development, production, or post-deployment optimization.
- **Targeted improvement**: The primary objectives of introducing the GenAI-powered system, such as increasing efficiency, reducing costs, improving quality, or expanding system capabilities.

USER EXPERIENCE

This category addresses how users interact with the GenAI-powered system, the level of control they have, and how the system integrates into existing workflows.

- **User-AI Synchronization**: Defines whether the GenAI-powered system runs before interacting with the user (e.g. automatically generated recommendations), operates concurrently in real-time (e.g., interactive assistants) or provides results with a delay (e.g., batch processing).
- **Internal workflow complexity & user control**: Determines whether the workflow is predefined, user-guided, or dynamically managed by AI-driven decision-making.
- **Output iteration & approval**: Establishes how users engage with GenAI-generated outputs, whether through direct oversight, iterative refinement, or fully autonomous operation.
- **Required User Training**: Considers the extent of training or onboarding required for users to effectively interact with the AI system, ensuring successful adoption.

AI INPUT

The **AI Input** category defines the nature of the data provided to the GenAI-powered system and how additional context is gathered to enhance performance. This ensures that AI operates with the necessary information to deliver meaningful results.

- **Type of additional input**: Specifies the format of input data the GenAI-powered system requires beyond basic prompts, such as structured datasets, unstructured text, or images.
- **Method of retrieving additional AI input**: Outlines how the GenAI-powered system obtains additional context information (a process called context injection), either through user-provided data, automated retrieval mechanisms, or computation by external software systems (e.g. a KBE application from the aerospace thesis project).

AI OUTPUT

The **AI Input** category defines the nature of the output data obtained from the GenAI model.

- **Type of output**: Specifies the format of output data from the GenAI model, such as structured datasets, unstructured text, or images.

- **Transparency**: Defines mechanisms to enhance interpretability, such as process logs, explainability tools, or traceability of AI-generated outputs.
- **GenAI determinism**: Establishes whether the AI system should produce consistent (deterministic) or variable (non-deterministic) outputs for the same input conditions.
- **Target level of output maturity**: Defines the required level of refinement and usability of AI-generated outputs. Some systems may generate preliminary drafts that require human oversight, while others produce high-fidelity results ready for direct implementation.

Prerequisites & Limitations

This category captures the constraints and requirements necessary for ensuring that AI-generated outputs are reliable, interpretable, and aligned with user expectations.

- **Output quality assessment methods**: Describes how AI-generated content is evaluated for accuracy and reliability, including traceability, quality metrics, or human validation.
- **Training data availability**: Considers whether high-quality training data is readily available, must be collected, or is infeasible to obtain, impacting AI system viability.
- **Formalization of current work process**: Evaluates whether the current workflow is already well-documented and structured. A highly formalized process can ease the adoption of AI-driven automation, whereas loosely defined workflows may require additional standardization efforts before AI integration.
- **Data security & privacy**: Examines the impact of AI implementation on **data protection, regulatory compliance, and intellectual property security**. This dimension triggers domain experts to considers the feasibility of their envisioned idea from the perspective of adhering to legal and ethical standards and dealing with sensitive or proprietary information.

16.2. Verification & Validation

The design and use of the finalized Ideation Matrix was resulted from a rigorous, iterative design approach that integrated information from multiple key sources - including expert input and feedback, scientific & non-scientific literature, and personal experience - leading to a very well-considered design. An overview how the final design aligns with the design requirements is provided below, along with a brief discussion of how they were implemented through various design aspects of the final design.

1. **The morphological chart should facilitate the development of GenAI-powered systems used to (partially) automate engineering workflows within the context of complex, critical systems.**

   Implementation: The user manual outlines how the matrix provides a means to support design discussions and clarifies how AI can automate engineering workflows in complex, critical systems.

2. **The morphological chart should help domain experts and AI experts to align their vision and expectations about GenAI-powered systems.**

   Implementation: The ideation matrix makes the design dimensions and options explicit in a structured and comprehensive overview, facilitating design discussions, the alignment of expectations and building of a shared vision between AI experts and domain experts.

3. **The morphological chart should focus on GenAI features and design options that domain experts often misunderstand or are unaware of.**

   Implementation: This requirement guided the selection of design dimensions and options, incorporating experience from the aerospace thesis project as well as results from the follow-up literature study and design iterations.

4. **The morphological chart should help domain experts to understand the prerequisites and investment required to develop GenAI-powered systems.**

   Implementation: A dedicated prerequisites and limitations category, with four key design dimension was included. Attention symbols (!) are also added to highlight options that are particularly challenging to implemented.

5. **The morphological chart should be understandable as a stand-alone communication artifact to domain experts without in-depth GenAI knowledge.**

   Implementation: Special care was taken during the design of the tool to avoid AI jargon and define dimensions and options in a clear and intuitively understandable way. Furthermore, icons were added to provide visual clarification of design options that are more difficult to understand based on brief textual descriptions alone.

VALIDATION

The tool concept, specifically the use case of serving as a **communication tool** and its content, were validated based on the validation session findings section 15.4. These validation session also concluded that the Ideation Matrix is less suited for the education use case (e.g presenting the matrix on an educational poster placed in common areas like near the coffee machines of domain experts). To also make it effective for this use case, participants remarked the Ideation Matrix would need to provide more explanations of technical GenAI concepts and/or present informative examples. The next chapter suggests some design ideas how this could be implemented, along with more general conclusions and recommendations related to the communication thesis research.

17

CONCLUSIONS OF SCIENCE COMMUNICATION RESEARCH

The rapid advancements in GenAI technology have prompted widespread discussions about its potential applications, particularly in knowledge-intensive fields like engineering. While GenAI has demonstrated remarkable capabilities in processing unstructured data and performing complex tasks, its adoption in engineering — especially in the development of **complex and critical systems** — remains limited. A primary reason for this reluctance is the inherently **black-box nature** of GenAI models, which hinders explainability, trust, and accountability in engineering workflows.

The risks associated with GenAI are particularly pronounced in industries where safety, reliability, and precision are paramount - like aerospace engineering. In such contexts, adopting GenAI without sufficient oversight could lead to errors with severe consequences. This concern was reinforced by findings from the aerospace thesis component of this research, which explored the development of a GenAI-powered Reverse Engineer Process model (REProcess) tool. The design of the **REProcess** tool revealed the significant complexity of developing Explainable AI systems suited for the engineering of **complex and critical systems**. Within these contexts, the development of GenAI-powered engineering systems requires close **interdisciplinary collaboration** between AI experts and domain experts, yet these groups often struggle to communicate effectively. This science communication research bridges the gap between AI experts and domain experts by developing a structured communication tool called the **Ideation Matrix**. This tool was designed to facilitate early-stage collaboration in GenAI development projects by providing a structured framework for defining system requirements, aligning expectations, and ensuring a shared understanding of GenAI capabilities and constraints.

The following section summarizes the key findings of this research, addressing the main research questions and discussing broader insights gained through the development and validation of the communication tool. An overview of research limitations is provided next. The final section outlines directions for future work.

17.1. OVERVIEW OF KEY FINDINGS

This thesis project was to contribute to the development of GenAI-powered engineering systems by improving the interdisciplinary collaboration between AI experts and domain experts through the design of a communication tool. Answers to the research questions are presented first, followed by an overview of additional key findings.

RQ-1: IDENTIFY KEY CHALLENGES

What are the key collaboration and communication challenges between AI experts and domain experts in interdisciplinary AI projects, and how do these relate to each other?

The central challenge in interdisciplinary AI projects was identified as the lack of **shared knowledge** between AI experts and domain experts. This gap leads to misunderstandings, misaligned expectations, and ineffective collaboration. The research found that communication barriers stem from multiple factors, including differences in technical language, lack of common mental models, and varying levels of intrinsic motivation among domain experts to engage with AI concepts. The reluctance of some domain experts to ask questions due to fear of appearing uninformed further exacerbates the issue. These challenges were visualized in Figure 17.1.

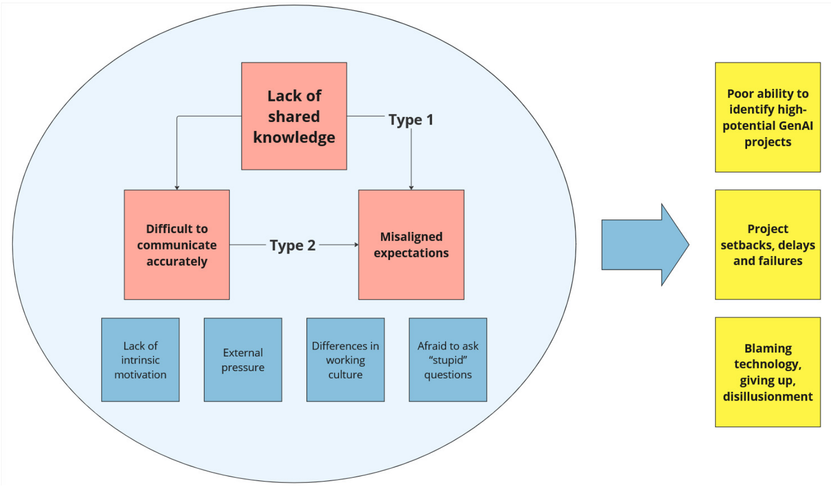


Figure 17.1: Key collaboration and communication challenges between AI experts and domain experts.

RQ-2: Determine Solution Direction

Which communication tool idea has the most potential to facilitate collaboration between AI experts and domain experts?

Several conceptual solutions were identified from which a **morphological chart** was selected as the most promising communication tool based on expert feedback, practical feasibility, and its alignment with findings from the aerospace thesis project. Its structured format allows for clear articulation of design decisions, making it particularly useful in early project discussions.

RQ-3: Communication Tool Design

How to implement the selected communication tool concept such that it provides real added value?

The design of the communication tool was broken down into three key elements: **content, methodology, and visual design**. An iterative design process was adopted, initially focusing on the design of the first two aspects. Over the course of three main iteration cycles, the content and methodology were refined using input and feedback collected from interviews (both the initial interviews and additional follow-up meetings and interviews), literature insights, and feedback from validation sessions. This input was formalized in the list of key design requirements outlined further below. The final design was structured to balance accessibility for domain experts with technical depth for AI experts, making it an effective means of bridging communication gaps. chapter 16 presents the finalized tool and its accompanying methodology.

RQ-4: Determine Design Focus

Which specific use case should the morphological chart be geared towards?

Two main use cases were identified focusing on education or communication, respectively. Based on the validation sessions, the morphological chart was found to be most effective when used **in early project phases**, specifically for ideation and requirements specification. By providing domain experts with a structured way to express their needs and expectations, the tool facilitates meaningful discussions with AI experts. Additionally, because the chart is understandable as a standalone document, it can also be used as a **pre-meeting intake form** to help domain experts prepare for discussions.

RQ-5: Identify Similar Existing Tools

To what extent do similar tools exist, both in scientific and non-scientific contexts?

While some existing tools share similarities with the developed morphological chart, they generally lack specificity in terms of GenAI-powered system development, especially in engineering contexts. Tools identified in literature (e.g., Shi *et al.* [85], Weisz *et al.* [93], and Rittelmeyer and Sandkuhl [82]) tend to be either too technical for engaging domain experts or lack a structured and comprehensive approach. This gap in existing tools highlights the added value of the communication tool developed in this research.

Key requirements & contributions

Based on the findings outlined above, a series of design requirements were formulated that together highlight the **key contributions** of this research. These are outlined below. For each requirement, the aspects corresponding to their implementation into the design communication tool are also provided.

1. The morphological chart should facilitate the development of GenAI-powered systems used to (partially) automate engineering workflows within the context of complex, critical systems.
   The Ideation Matrix structures discussions to clarify how AI can automate engineering workflows in complex, critical systems.
2. The morphological chart should help domain experts and AI experts to align their vision and expectations about GenAI-powered systems.
   The Ideation Matrix makes key design decisions and trade-offs explicit by presenting a structured overview of, thereby facilitating a common understanding between AI experts and domain experts of the system's scope.
3. The morphological chart should focus on GenAI features and design options that domain experts often misunderstand or are unaware of.
   This requirement guided the selection of design dimensions and options. The process incorporated experience from the aerospace thesis project as well as findings from the follow-up literature study and design iterations.
4. The morphological chart should help domain experts to understand the prerequisites and investment required to develop GenAI-powered systems.
   A dedicated "Prerequisites & Limitations" category with four key design dimension was included.
5. The morphological chart should be understandable as a stand-alone communication artifact to domain experts without in-depth GenAI knowledge.
   Special care was taken during the design of the tool to avoid AI jargon and define dimensions and options in a clear and intuitively understandable way. Furthermore, icons were added to provide visual clarification of design options that are more difficult to understand based on brief textual descriptions alone.

17.2. Research limitations and open issues

A number of factors the limit the generalizability of the results from the conducted research activities:

- **Small sample size:** The qualitative nature of the interviews placed practical restrictions on the number of interviews that could be conducted. Only a limited number of people participated in the initial series of structured interviews (3 AI experts and 3 domain experts). Using these participants to represent two different groups instead of one further reduced the sample size used to represent a group by half. Two measures adopted to mitigate this issue were to 1) first investigate similar collaborations and AI development challenges within literature, and 2) include different participants in the follow-up interview and validation sessions. However, even when including the 4 additional participants from these follow-up sessions, the overall sample size (10) still remains small which detracts from the generalizability of the results.

- **Participant collection:** In the end, most participants were recruited through networking (both personal network and those of already involved participants). The fact that this was not a random process, means it is another potential source of bias affecting the results.
- **Personal bias:** The results were processed and interpreted by a single researcher, leaving room for subjectivity and bias. An attempt to mitigate this was made through the inclusion of quantitative questions which do not leave room for interpretation. The important role that the personal experience from the previous aerospace thesis project played in defining this research and design project also contributes to an increased susceptibility to confirmation bias. From this perspective, it could be argued that the previous experience is both a blessing and a curse.
- **No validation in real-life setting:** An important limitation of the validation sessions was that, due to practical limitations, the communication tool could not yet be tested in the exact real-life setting that it was designed for, namely facilitating the communication between AI expert and domain experts. Given that design intent of the communication tool is to facilitate communication coming out from domain experts, however, these tests were still considered to give solid evidence towards the validity of the tool. In addition, the strong theoretical foundations, inclusion of input from many experts and the ability to use experience from the aerospace thesis project are all factors that also mitigate the impact of the validation limitations.

17.3. RECOMMENDATIONS FOR FUTURE WORK

The fundamental benefits of the developed Ideation Matrix already provide benefits, which was confirmed by the results from the validation sessions. However, many ideas were conceptualized that can be implemented in future work to improve the effectiveness of the tool, as well as extending it with additional capabilities (e.g. making it more viable for educational purposes). An overview of improvement ideas is provided below.

1. For every design dimension, order the design options (left-to-right) based on **implementation effort**, so the time and money required to do machine learning, develop user experience features, perform testing & validation, etc. Then the filled-out chart gives a global indication of the required development effort when viewed from afar. If, overall, a lot of right-hand side options are selected, the implementation effort will be high, and vice-versa. An alternative idea is to add 'complexity tokens' which represent the implementation effort related to each design options. Follow-up research is required to determine the implementation effort associated to each design option. Moreover, certain options will also be much challenging in particular combinations with others. Further research is required to determine the implementation effort and these specific interactions. A suggested method is to sent out a survey sent out to AI experts who start making use of the communication tool.
2. Create an **interactive user interface** for the morphological chart. Users could be this interactive version with a QR code (see Figure 17.2). A number of dynamic features were conceptualized that could add a lot of value to the communication tool:
    - Collapsed presentation mode: feature to visualize a collapsed version of the morphological matrix that only displays the selected design options for each dimension. By removing clutter, this alternate view can be used to present the selected design decisions to others in a much clearer way.
    - Alternate presentation mode: feature to automatically generate alternate visualization aimed at presenting the selected design options. For example, a circular arrangement of the selected design options, or a grid pattern. This feature is provides similar benefits to the collapsed presentation mode, but then more advanced (and more complex to implement).
    - Add feature to let the user cluster or highlight design dimensions linked to a particular concept, for example: *Design dimensions related to Trustworthiness and Explainable AI*, or *Design dimensions relevant from Ethical perspective.*

- An improvement to the ordering of design options from left-to-right could be to make it dynamic, based on a user-selected criteria (e.g. required computation power, development cost, complexity, etc)
- Display a tooltip with additional information when hovering on design dimensions and options. For instance, these tooltips could provide references to real-world examples showcasing particular design options, or show technical concepts and technologies that are associated to a design dimension. Some of this information is already attached in a hidden column of the digital version of the Ideation Matrix. An example of this is provided in Figure 17.3.
- Highlight or automatically lock (e.g. “graying out”) mutually exclusive design choices after one of these options is selected. For example, a *Mature & qualification tested* application can not be created for *Undocumented, flexible processes*. When one is selected, the interactive version could make it impossible for the user to select the other option. Currently, the responsibility of making this assessment lies with the AI expert involved in the process.

3. **Automated post-processing** of the selected design. For example, a software repository for a new GenAI system could be automatically generated that is filled with folders code files corresponding to the relevant software components. This is similar to the skeleton code code generation implemented for the MDKBE approach (step 2), which was discussed in the aerospace thesis (chapter 3).
4. **Categorize existing tools** (ChatGPT, GitHub CoPilot, etc.) with the morphological chart. These could be presented as example use cases. This can achieve two things at once. Firstly, it shows an example of how to use the tool so it helps people to use it. Secondly, it could help to engage and educate people. Some design decisions embedded in existing tools might be different from what they expected, which provokes them to rethink what they know and how to design similar tools. For example, many people will think they are interacting with a single LLM when ineracting with ChatGPT. In reality, there are dozens of distinct model architectures and configurations operating together internally, with key models fine-tuned for specific functions. For instance, it uses specific models for sentiment analysis to adapt tone and formality to the user’s input style and intent, and search decision model to decide whether a query requires internal knowledge or an online search. All these models works in tandem in a so-called pipeline to ensure accuracy, relevance, and fluency in responses. Thus, ChatGPT is defined as a **Dynamic workflow, controlled by AI agent**.
5. Include a design dimension that lists an array of **alternative technological options** that are *not* generative AI to implement the desired functionality. Different but similar ways to make users to consider this are also possible, like including an explicity question in the methodology and user manual. The question that’s important is whether generative AI is indeed the best method, or if there was a fixation on this technology (due to hype, unrealistic expectations, etc).

FURTHER TESTING

In addition to the extensive list of identified design improvements and additional features, another interesting line of work is to further test and validate the existing design. It would be highly interesting to deploy the communication tool in real-world interdisciplinary projects to assess its impact in practice. When further testing the tool, a key improvement opportunity consists of the set of quantitative validation questions that were used in this research. These should be focused more specifically on the design requirements of the tool. A quantitative assessment to what extent the morphological chart helped to clarify GenAI project prerequisites and limitations would be valuable.

This thesis report and the finalized Ideation Matrix design were shared with all interviewed participants to make a start with the dissemination of the tool.

DESIGN TRADE-OFF CHALLENGE

One of the most challenging aspects of the design was the trade-off between coverage and specificity. This became particularly noticeable after conducting the design experiment where the first prototype of the morphological chart was applied to the REProcess tool developed for the aerospace thesis. Due to the extremely wide range of capabilities and potential applications of GenAI, achieving good coverage while preventing to overload the user with too much information - by including too many design dimensions and options - results in a very high-level tool that does not offer much added value. On the other hand, focusing too much

Figure 17.2: Design idea to guide users to an interactive version of the morphological chart, with additional dynamic features.

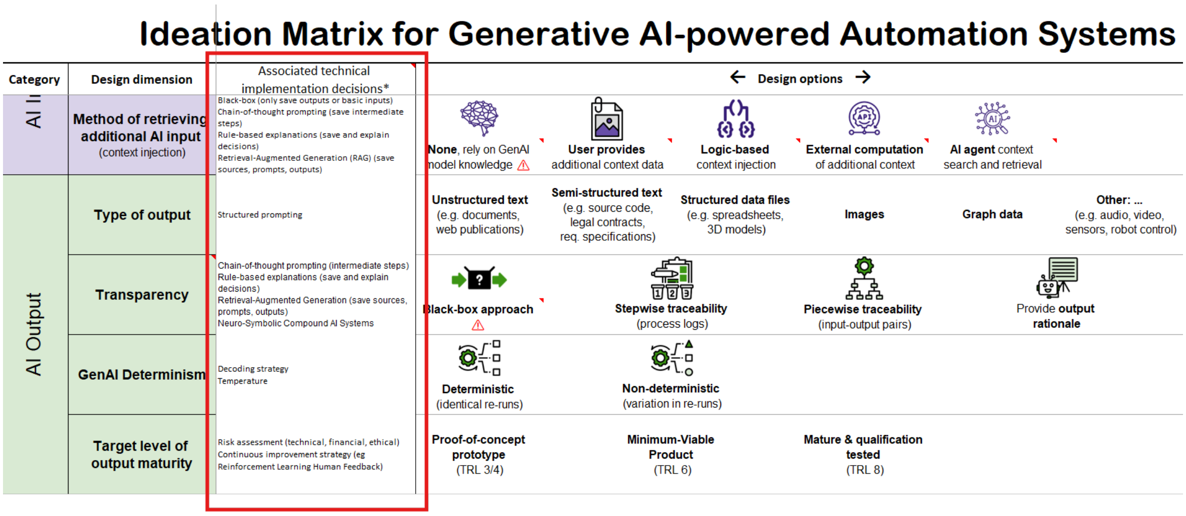


Figure 17.3: Example highlighting the idea to include additional information highlighting technical implementation associated to particular design options.

on a particular set of GenAI capabilities detracts from the tool's objective to raise awareness and increase understanding about novel ways to use GenAI technology to improve existing practices and processes. In reality, the optimal balance between these aspects will depend on the specific application context and it will evolve over time as the AI literacy of domain experts improves. A lot of interesting future work could focus on investigating and optimizing this further.

ADDITIONAL APPLICATION AREAS

The communication tool design scope was focused specifically on the context of engineering complex and critical systems. However, the large majority of design dimensions and options that were incorporated into the final design of the communication tool are not specific to this particular context. The main reason for the limited scope was the *need* for well-functioning interdisciplinary collaboration which is particularly high in the context of engineering complex and critical systems. However, aside from some minor things like the inclusion of *3D models* as one of the options for the *Types of additional input* dimension, the final design of the morphological chart is applicable to other contexts as well. Thus, the tool could be easily be adopted in other many other contexts to reap the communication and ideation benefits. This represent another interesting area for future work.

FINAL TAKEAWAYS

When adopting GenAI, it is crucial to do it according to a responsible and context-aware approach. This means there will be an increased need for more interdisciplinary GenAI projects in the coming years, which will grealy benefit from improved communication and AI literacy of domain experts. A greater emphasis on setting up and facilitating interdisciplinary collaborations is required. The research presented in this thesis contributes to the growing field of interdisciplinary collaboration in GenAI-powered system development. By addressing a critical communication gap between AI experts and domain experts, the developed communication tool provides a structured and accessible means to facilitate meaningful discussions. While there are still opportunities for further refinement and validation, the insights gained from this study serve as a foundation for future work aimed at improving the effectiveness of interdisciplinary GenAI projects.

An overarching discussion that reflects on this combined thesis project and some final remarks are provided in chapter 18.

PART III

COMBINED THESIS DISCUSSION

18

OVERARCHING DISCUSSION ON INTEGRATED THESIS

This integrated thesis project provided a unique opportunity to explore both the technical and communicative dimensions of Generative AI (GenAI) in engineering. By approaching the field from two complementary perspectives—developing a GenAI-powered engineering tool and designing a communication tool for interdisciplinary collaboration—this research has yielded valuable insights into both the potential and challenges of GenAI adoption in complex, high-stakes domains.

One of the key takeaways from this dual perspective is that the successful integration of GenAI in engineering is not solely a technical challenge, but also a human and organizational one. While technological capabilities continue to advance at an unprecedented rate, the ability of AI experts and domain experts to collaborate effectively remains a fundamental limiting factor. This thesis contributes to bridging that gap by providing both a concrete engineering demonstration and a structured framework for interdisciplinary communication.

18.1. KEY REFLECTIONS ON THE INTEGRATED RESEARCH APPROACH

THE INTERSECTION OF TECHNOLOGY AND COMMUNICATION

Developing an actual Generative Artificial Intelligence (GenAI)-powered engineering tool (as done in the aerospace engineering part of this thesis) provided deep insights into the practical challenges of implementation—not just in terms of coding and system integration, but also in defining clear design requirements, managing expectations, and ensuring explainability. At the same time, studying the collaborative processes behind AI development (in the communication part of this thesis) highlighted how misaligned assumptions and knowledge gaps between engineers and AI specialists can hinder the success of such projects. This interdisciplinary approach reinforced a critical finding: **AI systems are not just technical artifacts; they are socio-technical systems that require careful collaboration between multiple stakeholders to be successful.**

BRIDGING THE GAP BETWEEN AI EXPERTS AND DOMAIN EXPERTS

The interdisciplinary nature of GenAI development demands a shared understanding between AI experts and domain experts. However, as identified through interviews and literature, this shared understanding is often lacking. The two groups approach problems from different perspectives, use different terminologies, and have different priorities.

The morphological chart-based communication tool developed in this thesis was specifically designed to facilitate structured discussions between these groups. It helps domain experts articulate their needs without requiring extensive AI knowledge, while also providing AI experts with a systematic way to explain technical constraints. The validation sessions confirmed that this structured approach improves clarity, helps align expectations, and reduces the risk of miscommunication—a key barrier to effective AI adoption in engineering.

RETHINKING THE ROLE OF DATA IN GENAI-POWERED SYSTEMS

The aerospace engineering part of this thesis demonstrated how the latest generation of Large Language Models (LLMs) can operate effectively without requiring extensive task-specific datasets. This challenges a deeply ingrained assumption in the AI community: that large amounts of training data are always necessary.

Many AI experts — due to their background in machine learning and data science — place a strong emphasis on data availability when evaluating potential projects. However, as this research has shown, GenAI's capabilities extend beyond traditional AI paradigms. Engineers and AI specialists must update their assumptions about what is feasible with modern GenAI, and remain open to new methodologies that do not strictly rely on vast amounts of domain-specific training data.

Conversely, domain experts — especially those working in highly regulated industries like aerospace — often overestimate the limitations of AI, assuming that GenAI is not reliable enough for their field. This skepticism, while valid to some extent, can lead to missed opportunities. A cultural shift on both sides is needed to strike a balance between caution and innovation when exploring AI-driven solutions.

18.1.1. EXPANDING THE IMPACT: USING THE COMMUNICATION TOOL BEYOND AI PROJECTS

Beyond its intended application in interdisciplinary AI projects, the ideation matrix developed in this thesis could serve a broader purpose. It has the potential to help future students and researchers in defining their own thesis projects and navigating the vast design space of GenAI-powered systems. By providing a structured approach to breaking down complex AI design decisions, the tool can assist in:

- Scoping AI research projects more effectively
- Clarifying design trade-offs early in the process
- Identifying key challenges and limitations before implementation begins

This potential for broader application reinforces the idea that structured communication is a critical enabler of progress — whether in academia, industry, or interdisciplinary research initiatives.

18.1.2. THE ROLE OF VISUALIZATION IN AI-DRIVEN ENGINEERING AND COMMUNICATION

Throughout this integrated thesis, a recurring challenge emerged: the complexity of visualizing AI-powered engineering processes and effectively communicating these processes to diverse stakeholders. While the initial focus of the science communication thesis was on interdisciplinary collaboration, an initial research direction explored during this work was the role of visualization in making complex AI-driven engineering workflows more comprehensible (see Appendix D).

Engineering workflows, particularly those involving Generative Artificial Intelligence (GenAI), inherently involve high-dimensional, abstract data and complex decision-making processes. Traditional visual modeling languages such as Unified Modeling Language (UML) and Systems Modeling Language (SysML) are widely used in engineering but have notable limitations when applied to AI-based systems. As identified in prior literature, these notations often lack sufficient cognitive effectiveness for conveying abstract AI-related concepts [94**?**]. Additionally, layout algorithms used in existing modeling tools do not always follow established visualization principles, leading to cluttered and inefficient representations [95]. Research into visualization guidelines has shown that effective visual representations must optimize **perceptual discriminability, visual expressiveness, and semiotic clarity** [96]. However, these principles are not yet well integrated into mainstream tools used in engineering. This gap presents an opportunity to improve both AI system design workflows and communication practices between AI experts and domain experts.

Although this research direction was ultimately not pursued in-depth within this thesis, it remains a significant area of interest for improving GenAI adoption in engineering. The challenges observed during the aerospace engineering thesis—such as defining clear AI-related system requirements and explaining AI decision-making processes—demonstrate the need for better visualization techniques. Further research in this area could provide **a crucial link between AI system design, engineering workflows, and effective interdisciplinary communication**. Future work could explore integrating cognitive visualization principles into AI engineering methodologies, ensuring that **engineers, AI experts, and decision-makers can collaboratively interpret and refine AI-driven designs** in a more structured and intuitive manner.

18.2. FINAL REMARKS

This thesis was conducted during a period of unprecedented acceleration in artificial intelligence. Over the past few years, the transition of Artificial General Intelligence from a distant possibility to an imminent reality has reshaped industries, workplaces, and society at large. Now, in 2025, the widespread integration of AI into various aspects of life seems inevitable — a matter of years, not decades. Even those with a highly skeptical stance toward the technology cannot deny the momentum driving its adoption.

However, rapid technological progress presents a crucial challenge: ensuring that the integration of GenAI into high-stakes domains is conducted responsibly, effectively, and transparently. Nowhere is this more critical than in the engineering of complex and critical systems, where careful implementation is essential. The potential benefits of GenAI in these domains—ranging from knowledge augmentation to workflow automation—are vast, but so too are the risks. Issues of trust, explainability, and interdisciplinary misalignment pose serious barriers to successful implementation. Without deliberate efforts to address these challenges, the introduction of GenAI could lead to unintended complications rather than meaningful improvements.

Beyond these technical challenges, there is a broader, more fundamental concern: the potential impact of AI on human interaction and collaboration. As artificial intelligence takes on more roles in communication, decision-making, and negotiation, we must carefully consider its effects on how people engage and connect. Emily Bender articulates this concern succinctly:

> *"We've learned to make machines that can mindlessly generate text. But we haven't learned how to stop imagining the mind behind it."*

As AI systems take on greater responsibilities in facilitating communication, there is a risk that human interaction could become increasingly mediated by technology. If we allow AI to act as an intermediary in our discussions, negotiations, and collaborations, we may gradually rely less on direct human-to-human engagement. While AI can enhance communication in many ways, overreliance on it could lead to unintended barriers, making interactions more transactional and reducing opportunities for deeper mutual understanding. This concern extends beyond the field of engineering; it is a societal challenge that underscores the importance of ensuring that AI tools foster, rather than diminish, meaningful connections.

This thesis has demonstrated that structured communication and collaborative design processes can help bridge the gap between engineers, researchers, and domain experts in interdisciplinary projects. However, the work is far from complete. The increasing complexity of AI models, the evolving regulatory landscape, and the growing role of AI in industry ensure that interdisciplinary communication will remain an ongoing challenge. The research presented in this thesis provides a foundation, but further work is needed to refine, expand, and adapt communication strategies as the role of GenAI continues to evolve.

Encouragingly, the European Union's vision of Industry 5.0 provides a framework for addressing these challenges. Unlike its predecessor, which prioritized automation and efficiency, Industry 5.0 reintroduces the human element. It envisions a future where technology augments rather than replaces human capabilities, emphasizing resilience, sustainability, and human-centric industry practices. The development of GenAI-powered systems should align with this vision, ensuring that AI serves as a collaborative partner that enhances human expertise rather than an autonomous system that isolates or replaces it.

As we move forward into this new era of artificial intelligence, one thing is clear: the success of GenAI in engineering and beyond will not be determined by technological breakthroughs alone. Instead, it will depend on our ability to collaborate, communicate, and ensure that AI is developed with purpose, transparency, and responsibility. This thesis contributes to that effort, and it is my hope that its insights will help shape a future where GenAI is not only powerful but also trusted, understood, and thoughtfully integrated for the betterment of engineering and society as a whole.

PART IV

APPENDICES & BIBLIOGRAPHY

A

Background information about KBE and MBSE

A.1. Knowledge Based Engineering

Knowledge Based Engineering (KBE) is a technology based on the use of dedicated software tools, called KBE systems or KBE platforms, which can be viewed as the merger of traditional CAD systems with Expert Systems AI (see Figure 2.1). The CAD part of a KBE system allows for the generation and manipulation of product geometries, while the Expert Systems AI functionalities allow embedding engineering knowledge and reasoning (typically using some customized programming language). The relation between KBE, CAD and AI is visualized in Figure A.1. Through this combination of functionalities, KBE systems can be used to capture and systematically reuse product and process engineering knowledge, thereby providing a means to automate repetitive and non-creative design tasks and support Multi-disciplinary Design Optimization (MDO) activities in all the phases of the design process[34].

Some examples of KBE systems are Siemens Knowledge Fusion[1], the Adaptive Modeling Framework from Technosoft[2], and ParaPy[3]. In practice, these KBE systems are used by KBE developers to create so-called KBE applications. While the KBE *system* is a general purpose platform, a KBE *application* is a particular implementation focused on a specific use case. KBE applications can be developed for any engineering

[1] https://plm.sw.siemens.com/en-US/nx/cad-online *(note: integral component of Siemens NX CAD software)*
[2] https://www.technosoft.com/
[3] https://parapy.nl

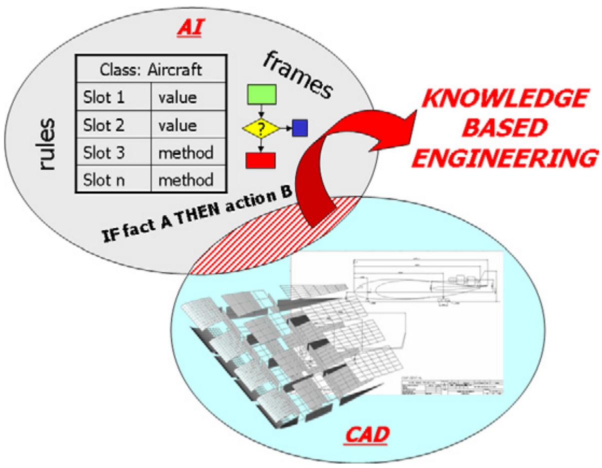


Figure A.1: Relation between Knowledge Based Engineering, (Expert Systems) Artificial Intelligence and Computer Aided Design. Obtained from [34].

domain ranging from the design of bridges, to ships or aircraft. To create these applications, KBE developers typically collaborate with *domain experts*, who provide the domain-specific engineering knowledge and expertise to be embedded into the application.

Within DEFAINE the ParaPy KBE system was used. As suggested by the name, ParaPy is a Python-based KBE language. When greatly simplifying, the development process of a ParaPy KBE application consists of opening a new Python project, importing the ParaPy package, and then using its custom programming syntax and features to define the desired engineering product (e.g. an aircraft) and embed a set of associated engineering rules (e.g. a mathematical relation between aircraft mass and surface area of the wings). Subsequently, once the desired product features and engineering rules have been incorporated in the code, the Python program can be executed to instantiate the KBE application and launch its GUI. A screenshot of the GUI from a ParaPy KBE application is provided in Figure A.2.

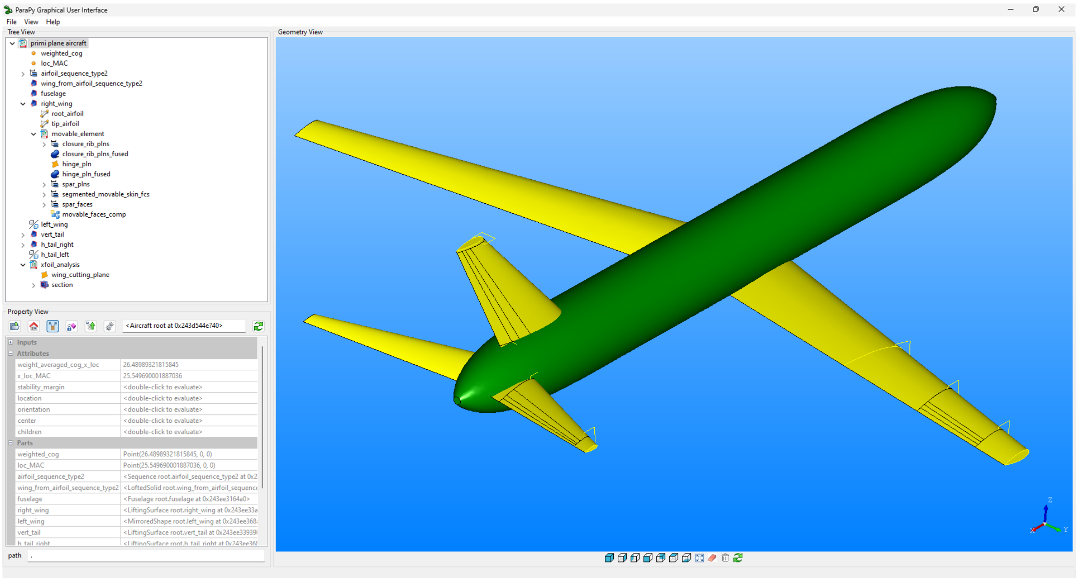


Figure A.2: Screenshot of the GUI from a ParaPy KBE application called Primiplane. Application source code obtained from [41].

Within this GUI, users of the KBE application can inspect the product's geometry. Moreover, they can modify its governing parameters (e.g. aircraft mass) which automatically triggers an update of any related parameters and parts (e.g. wing surface area). This allow the users to see and better understand the effects of their changes. Furthermore, depending on the functionalities embedded by the KBE developers, the KBE application can also have buttons to trigger certain high-level actions, like running simulations using external software packages or automatically generating a PDF report of the current design.

By simplifying the generation and modification of product geometries, and by embedding and automating entire sequences of steps required to perform high-level tasks like running simulations, KBE applications can be used to improve the efficiency and productivity of engineers. Another benefit is that KBE applications are a way to capture and retain valuable engineering expertise, even when engineers leave the organization [97].

A.1.1. KBE DEVELOPMENT CHALLENGES

Besides the benefits, there are also a number of challenges associated with KBE applications: 1) their development is often case-based and unstructured, **lacking a standardized development methodology**; 2) they tend to become **black-box applications** due to having large code bases, as well as software development being prioritized over writing and updating documentation; and (3) they suffer from **limited reusability and project-to-project knowledge transfer**, as high-level design knowledge is often not captured, and there is no standard exchange format for transferring knowledge between different KBE tools [7, 8].

In response to these issues, a number of methodologies have been developed that aimed to improve and formalize the KBE application development of which the Methodology and tools Oriented to Knowledge-based engineering Applications (MOKA) approach has been the most accepted[4] [8]. One of the main aspects of the MOKA methodology is the role of the so-called knowledge engineer, who forms a bridge between engineering domain experts and KBE developers. More specifically, their job consists of talking with engineers in order to capture their domain knowledge and formalize it into a knowledge model. This knowledge model is then provided to KBE developers as a basis for implementing the KBE application [98, 99].

Despite its promise, MOKA adoption remains low, as it primarily supports knowledge engineers in creating knowledge models rather than directly benefiting domain experts and KBE developers. Also hindering MOKA's adoption is the reliance on the availability of specialized knowledge engineers - who are scarce - along with a knowledge modeling approach that is not intuitive to engineering domain experts [8]. Due to these considerations, the novel KBE application development approach proposed and developed within the DEFAINE project is based on MBSE and leverages SysML, a widely adopted standard in systems engineering practice.

A.2. MODEL-BASED SYSTEMS ENGINEERING

The International Council on Systems Engineering (INCOSE) defines Model-Based Systems Engineering (MBSE) as *"The formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases"* [100]. Central to the practice of MBSE is the integrated modeling platform, which is the software used to generate, govern, and visualize the models. Some notable examples are CATIA Magic[5], Enterprise Architect[6], and Papyrus[7]. The DEFAINE project selected CATIA Magic (i.e. Magic Systems of Systems Architect) as the modeling platform (rationale provided in [9]).

A.2.1. SYSTEM MODELING LANGUAGE

The Systems Modeling Language (SysML) is the most widely adopted modeling language within the domain of MBSE. It is an extension of the Unified Modeling Language (UML), which focuses specifically on the domain of software engineering. SysML supports modeling of requirements, product structure, behavior (i.e. process information), and parametrics in order to provide a complete system description. All elements of a model are contained and captured in a model repository. The different aspects of the model can be described and represented using 9 types of diagrams. A particular model element may appear on zero, one or multiple diagrams. In addition, a model element often has relationships to other model elements that may appear on the same diagram or other diagrams. For an extensive overview of SysML, including detailed descriptions of every SysML diagram and its usage, the reader is referred to Friedenthal *et al.* [11].

[4]Others notable examples are Common Knowledge Acquisition and Documentation Structuring (CommonKADS), and Knowledge-Oriented Management Advisory and Design (KNOMAD)

[5]`https://www.3ds.com/products/catia/catia-magic` (note: also known as Magic Systems of Systems Architect, previously Cameo Systems Modeler)

[6]`https://sparxsystems.com/products/ea/`

[7]`https://eclipse.dev/papyrus/` (note: open-source)

B

USER MANUAL FOR REPROCESS PROTOTYPE

This appendix provides a user manual for the REProcess prototype. First, the user commands and associated input parameters are described in section B.1. section B.2 presents which high-level function calls invoke which steps of the algorithm.

B.1. USER COMMANDS TO INITIATE THE REVERSE ENGINEERING STEPS

A source code snippet presenting each of the required user actions depicted in Figure 7.1 is provided below. In the prototype implementation, this code is located within the `if __name__ == "main"` block at the bottom of the `c2m.reverse_process_module()`. All of the the input parameters passed into the respective steps are also declared in this block, directly above these lines. Regarding the three steps, note that only the first user command explicitly contains a class instantiation. The DataflowAnalyzer and LLMPrompter classes are initialized indirectly by calling the `analyze_kbe_dataflow` and `perform_LLM_abstraction` class methods from the ReverseProcessModel class.

```python

####################    Run analysis    ####################
## Step 1: instantiate REProcess tool
prcs_model = ReverseProcessModel(target_kbe_app_dir,
                                 target_class_qualified_name,
                                 init_kwargs_dict,  # -> optional
                                 )

## Step 2: Run dataflow analysis on target slot
prcs_model.analyze_kbe_dataflow(target_slot_qualified_name)

## Step 3: Perform LLM-based source code abstraction
selected_prompting_mode = str(input("\nWhich prompt generation context mode should be used "
                                    "for the LLM-based abstraction? \n -> Choose 'm' for "
                                    "method, 'c' for class, 'b' for both, or press <enter>"
                                    " to skip LLM abstraction step:"))

if selected_prompting_mode:
    if 'm' in selected_prompting_mode.lower() or 'b' in selected_prompting_mode.lower():
        ## A.) METHOD prompting_context_mode
        prcs_model.perform_LLM_abstraction(system_prompt=system_prompt_method,
                                           user_instruction=user_instruction_method,
                                           llm_checkpoint=hf_llm_checkpoint,
                                           keyword_response_start=True,
                                           prompting_context_mode='method')

```

```
    if 'c' in selected_prompting_mode.lower() or 'b' in selected_prompting_mode.lower():
        ## B.) CLASS prompting_context_mode
        prcs_model.perform_LLM_abstraction(system_prompt=system_prompt_class,
                                           user_instruction=user_instruction_class,
                                           llm_checkpoint=hf_llm_checkpoint,
                                           keyword_response_start=True,
                                           prompting_context_mode='class')

```

B.1.1. DESCRIPTION OF MAIN INPUT PARAMETERS

The input parameters used in the source code snippet are described in the list below. Only the input parameters related to step 1 and 2 are described. For the general definitions of the `system_prompt` and `user_instruction`, the reader is referred to subsection 5.2.4. The specific variations of these parameters mentioned in the source code, denoted by `_method` and `_class`, as well as the other LLM-specific parameters are presented in section 7.4.

- **target_kbe_app_dir:** A string value specifying the directory name of the KBE application that should be analyzed. These directories are located within the reference folder. Example values are: "Modular_UAV_app", or "primiplane_tut".
- **target_class_qualified_name:** A string specifying the fully qualified name of the class to be analyzed within the KBE application. The KBE app will be instantiated using the class specified in this parameter. Some example values are: "application.Propeller", "application.Blade", "primiplane.Aircraft", or "asystor.aircraft.Aircraft".
- **init_kwargs_dict :** A dictionary containing initialization parameters for the target class. These parameters provide specific values used for instantiation of the KBE object. Depending on the specific KBE application, this parameter can be optional. Namely, when default input values are specified for all input slots of the target class. An example of an init_kwargs_dict for the "application.Propeller" class of the "Modular_UAV_app" is provided below:

```
init\_kwargs\_dict = {
        "d_prop": 0.508,
        "num_blades": 2,
        "blade_root_airfoil": "NACA_2414",
        "blade_tip_airfoil": "Clark_Y",
        "root_airfoil_thickness_factor": 1,
        "tip_airfoil_thickness_factor": 0.5,
        "rpm": 6500,
        "v_0": 5,
        "h": 300,
        "num_elements": 10
    }
```

- **target_slot_qualified_name:** A string specifying the fully qualified name of the target slot (input, attribute or part) within the class. This parameter identifies the specific slot for which the corresponding workflow will be reverse engineered. Some example values include: `prop_symmetry_point`, `main_blade.blade_profiles[5]`, `xfoil_analysis`, `vert_tail.movable_element.movable_faces`. Note how for part sequences, a specific sequence is specified between square brackets.

B.2. ALGORITHM STEPS

The list below provides a detailed description of all of the processing steps triggered by the three user commands referred to in Figure 7.1.

1. **Instantiate ReverseProcessModel**
   (a) Initialize class parameters
   (b) Instantiate KBE app
2. **Initialize DataflowAnalyzer** (perform activity graph analysis)
   (a) Instantiate Analyzer
      i. Generate precedents tree
      ii. Transform to NetworkX graph
      iii. Visualize 'raw' precedents graph (optional)
   (b) Transform to activity graph
      i. Merge sequence instances
         A. Sequences of type 1 parts: merge all sequence instance nodes into their part node.
         B. Sequences of type 2 & 3 parts: remove all sequence instance nodes, except for first instance (child_0) and, if applicable, user-defined target sequence instance.
      ii. Merge input slot nodes into their part node
         A. Before merging, add relevant inputs as metadata to the part node and downstream edges.
         B. Execute input merging loop:
         C. If corresponding part node not found, try to add part node to the graph.
         D. If that also fails, remove the node (with path preserve) instead of merging.
      iii. Remove parapy nodes
         A. Before removing, identify type 2 parts and find their geometry nodes. Save this as metadata
         B. Execute remove parapy loop:
         C. For (sequences of) Type 1 parts, merge geometry nodes into part node
         D. Skip removing geometry nodes of type 2 parts
         E. Remove all nodes of which the SLOT is from parapy/kbeutils unless they are defaulting inputs slots (this also removes 'mystery nodes' from kbeutils functions)
         F. Also remove node if it belongs to an OBJECT generated with a parapy geom class, even for defaulting input slots.
      iv. Introduce initialization nodes
         A. Identify the nodes that belong to root-level objects (legacy implementation using cluster_nodes_by_node_object method + excluding if-statement in subsequent for-loop)
         B. Add 'init node' for the root-level object(s)
         C. Merge remaining input nodes into their corresponding init node
         D. For special case where the init node remains disconnected from the rest of the graph due to lacking data dependencies, add artificial edges to try representing the control flow.
         E. In case the analysis target was a type 2 part (sequence), merge the stand-in target node (i.e. the geometry node)
   (c) Add lots of additional metadata to the activity graph, and its nodes and edges. For example:

i. Mapping of type 2 part objects to their corresponding geometry nodes

ii. Source code definition corresponding to each slot

iii. Data flowing across each edge

(d) Determine color coding, based on node objects

(e) Visualize intermediate graphs (optional)

(f) Relabel activity graph nodes, from cache objects to qualified identifiers (QIDs)

(g) Visualize QID graph

(h) Generate DOT file for QID graph (optional)

3. **Initialize LLMPrompter** (Perform LLM-based source code abstraction)

(a) Instantiate LLMPrompter

i. Initialize specified LLM and load onto GPU

(b) Loop through all nodes:

i. Skip over special nodes not suited for LLM abstraction (init nodes, geometry nodes)

ii. Format input prompt, based on provided inputs (system prompt, user instruction, prompt context mode, keyworded response start) and LLM settings (checkpoint, quantization, decoding strategy, etc.)

iii. Run LLM inference

iv. Save LLM output to dict and add as node metadata to QID graph

(c) Add LLM outputs to QID graph

(d) Add node description text as node metadata

i. Use LLM output, if available

ii. Else, for special nodes, format a description based on boilerplate text .

(e) Write LLM outputs and responses to .txt file (optional)

(f) Visualize abstracted graph (optional)

(g) Update QID graph visualization and its DOT file with abstraction results (optional)

C

VERIFICATION TESTING

TWEAKS TO THE ORIGINAL KBE APPLICATIONS

Below, the tweaks and additions made to the respective KBE applications are discussed. As will become apparent, all tweaks and additions were marked with the initials ("JK") so they can be easily identified in the source code of the KBE application.

Primiplane app:

1. In the *aircraft.py* module several slots were added to represent the computation of a stability margin. These computations referenced lower-level Parapy slots (e.g. the underlined part is this example: *self.fuselage.cog.x*) nested within parts of the aircraft object, in order to evaluate this aspect of PR-03. Moreover, a "regular" class method (i.e. not stereotyped as Attribute or Part) was defined that performed part of this computation, called `compute_stability_margin(self)`, to cover this aspect of PR-04.

   Finally, a `JK_xfoil_results` attribute was added to this file, which simply brings the nested attribute value `self.xfoil_analysis.xfoil_analysis` up to the root level. This was used to check if there were any other differences in output between root-level and lower-level target attributes, besides the final node.

2. In the *liftingsurface.py* module, an Input slot was added ("color") to evaluate differences between Input and Input slots. Furthermore, a sequence of type 2 parts (airfoil_seq_type2) was introduced because this specific type was not present in either of the two KBE applications. Correspondingly, the `wing_from_airfoil_seq_type2` part was added as well to assess how the prototype handled parts that used this type 2 sequence part as an input.

3. In the *fuselage.py* module, the `fu_length` slot was added as a lower-level test case of the Input slot type.

4. A separate *liftingsurface_DynamicType.py* module was created. It is a copy of the fuselage.py module, except that the `profiles` part is redefined as a DynamicType that uses either the `Circle` or `Octagon` class to generate the cross-sectional fuselage profiles. It determines this based on the value of the `fu_shape` Input, which was added as well.

5. The *main.py*, *movable.py*, and *xfoilanalysis.py* modules were left unchanged, as well as the data files in the airfoil_library.

Modular UAV app:

1. For comparison of LLM performance w.r.t. prompting context mode, all code comments placed above attribute and part definitions were moved to within the method definition so the information was also available for method prompt context mode.

2. In *drone.py* a test case for a sequence of type 3 parts was added in the form of the `arm_main_st3b` part.

3. Some minor tweaks were made to the Modular UAV app to fix minor issues. For example, the *airfoil.py* module contained a regular class parameter (`degree`), instead of defining it as a Parapy slot. This resulted in a breaking compilation error. The statement was changed to an input slot to solve this issue.

   Another examples from *propeller.py* is position definition of hub part (line 100). Here, `self.hub.height` was changed to `self.hub_height` because this attribute already existed in but it was not used in this statement. This was considered a typo, and therefore corrected. While the original definition did not cause any errors, the dependency of the Part's position on one of its own attributes led to a loop in the precedents graph that was confusing to interpret.

4. In *drone.py*, The `arm_main_st3b` part was added as a test case for type 3 part sequences. Therefore, the `arm_offset_JK` attribute was also added. This attribute is used as a substitute for the original `arm_offset` attribute to define the position of the `arm_main_st3b` part. Otherwise, this dependency would mean the `arm_main_st3b` part still depends indirectly on `arm_main`, via the `arm_offset` attribute. This dependency results in both `arm_main` variants being included in the output, which was not necessary and only found to be confusing.

5. In *propeller.py*, the following changes were made:

   - Two attributes were added to represent lower-level (Parapy) slots as root-level application slots, namely `JK_propeller_cog_attribute` and `JK_eta_from_bet_analysis`. Note that only the latter was used in for the final verification test set.
   - The `main_blades_st3b` part was introduced, which is very similar to the definition of the original main_blade part. However, the key difference is that main_blade is a singular (type 3) part, whereas this attribute is a sequence of (type 3) parts. This part was used in test case L-2.1. It was also used to study the behavior of the Parapy KBE system[1].

6. In *prop_blade.py*, the `JK_prop_blade_cog_attribute` was introduced. It was only used during development. This method was not evaluated in any of the final set of verification test cases.

7. The remaining Python modules were left unchanged: main, arm, battery, bet_calculations, frame, motor, prop_hub, results_pdf, slicer, utilities, xfoil_analysis, xfoil_section, propeller_strip. The data files in the inputs sub-directory were also left unchanged.

Overview of test cases

An overview of the test cases is presented below. The values between parentheses at the end of some test cases indicate the additional input that was supplied to further specify the target slot, after being prompted by the REProcess prototype that this was required. The "-> TopoDS_Shape" indications refer to cases where the algorithm found a `TopoDS_Shape` slot for the specified part, and determined by itself that this should be used as the specific target. Note that this occurs based on the initially specified target slot, as well as after additional specification are provided by the user through the prompts.

1. **Root level slots:**
   (a) R-1.1 (Inp) -> primiplane.Aircraft.fuselage_weight_frac
   (b) R-1.2 (@Inp) -> primiplane.aircraft.wing_fuel_weight_frac
   (c) R-2.1 (Attr) -> primiplane.Aircraft.stability_margin
   (d) R-2.2 (Attr) -> modular_UAV.Propeller.JK_eta_from_bet_analysis
   (e) R-3.1 (Part type 1) -> primiplane.Aircraft.left_wing
   (f) R-3.2 (Part type 2) -> primiplane.Aircraft.right_wing
   (g) R-3.3 (Part type 3) -> modular_UAV.Propeller.bet_analysis (delta_r)
   (h) R-3.4 (Part type 3) -> primiplane.Aircraft.xfoil_analysis (section -> TopoDS_Shape)
   (i) R-4.1 (Part type 1 sequence) -> modular_UAV.Blade.blade_profiles[1] (-> TopoDS_Shape)

[1] Specifically, it was checked whether the lower-level xfoil_analysis slot, which incorporates a kbeutils function, would be re-evaluated for every sequence instance. This was indeed found to be the case, verifying that each sequence instances is truly a standalone object

(j) R-4.2 (Part type 2 sequence) -> primiplane.LiftingSurface.airfoil_seq_type_2 ([1] -> TopoDS_Shape)

(k) R-4.3 (Part type 3 sequence) -> modular_UAV.Drone.arm_main_st3b[0].my_arm (-> TopoDS_Shape)

2. **lower level slots:**

   (a) L-1.1 (Inp) -> primiplane.Aircraft.vert_tail.airfoil_tip

   (b) L-1.2 (@Inp) -> primiplane.Aircraft.vert_tail.color

   (c) L-2.1 (Attr) -> modular_UAV.Drone.arm_main_st3b[0].arm_width

   (d) L-2.2 (Attr) -> modular_UAV.Propeller.bet_analysis.eta

   (e) L-3.1 (Part type 1) -> primiplane.Aircraft.right_wing.lofted_solid (-> TopoDS_Shape)

   (f) L-3.2 (Part type 1) -> modular_UAV.Drone.propeller_main.scaled_propeller (-> TopoDS_Shape)

   (g) L-3.3 (Part type 2) -> primiplane.Aircraft.right_wing.root_airfoil (-> TopoDS_Shape)

   (h) L-3.4 (Part type 3) -> modular_UAV.Drone.propeller_main.bet_analysis (eta)

   (i) L-3.5 (Part type 3) -> primiplane.Aircraft.vert_tail.movable_element.movable_faces_comp (TopoDS_Shape)

   (j) L-4.1 (Part type 1 sequence) -> modular_UAV.Propeller.main_blade.blade_profiles[5] (TopoDS_Shape)

   (k) L-4.2 (Part type 2 sequence) -> primiplane.Aircraft.right_wing.airfoil_seq_type_2[0] (-> TopoDS_Shape)

   (l) L-4.3 (Part type 3 sequence) -> modular_UAV.Propeller.bet_analysis.sections[0] (beta_n)

   (m) L-4.4 (Part type 3 sequence) -> modular_UAV.Propeller.scaled_propeller.bet_analysis.sections ([0], blade -> TopoDS_Shape)

3. **Special test cases:**

   (a) The SplitSolid() class was identified as an edge case. The get_precedents_tree() method generates a deviating pattern, including unique nodes for this class. A small, additional KBE app was created to generate a test case for it called *ref_KBE_app*. Here, the Omelet.omelet_fused was used as the target slot.

   (b) R-2.2 was executed twice: once with, and once without the keyworded response start feature.

   (c) L-3.2 was executed twice, using both the method and class prompting context modes for comparison purposes. Also the LLM output files were generated for these test cases to compare performance, both in terms of output and processing time.

   (d) L-2.1 showed also attributes nested within sequences could be targeted.

   (e) R-4.1, R-4.2, and L-4.1 all used non-zero sequence instances to demonstrate the capability to target those as well besides the “standard” option to use instance 0 as the representative sequence instance.

   (f) An additional KBE app was introduced, called Asystor, which was used to verify the prototype is able to handle larger, more complex applications that are defined using a hierarchical directory structure.

C.1. REPROCESS PROTOTYPE REQUIREMENT VERIFICATION

The overview below presents how each of the other prototype requirements were verified. The full dataset was provided to separately, and can be requested by contacting the thesis supervisor.

- **PR-01:** Verified by the entire set of test cases presented above.
- **PR-02:** Verified by the entire set of test cases presented above.
- **PR-03:** Verified by the entire set of test cases. The usage of kbeutils functionalities specifically is included in test cases R-2.2, L-2.2 (both KBE applications used for testing used the xfoil module from kbeutils). The Blade class from the Modular UAV app (*prop_blade.py*) included the `y_coords` attribute, which does not have any precedents. This slot is appears in R-2.2, R-3.3, R-4.1, and R-4.3, amongst others.

- **PR-04:** The full set of test cases was designed specifically to cover all features from PR-03. The DynamicType class was introduced through the *fuselage_DynamicType.py* variant used in test case R-2.1 to verify the implemented prototype could handle DynamicType occurences. Validators are used extensively on the input slots of the *drone.py* module, so this feature was verified by test cases R-4.3, L-2.1, L-3.2, L-3.4. The special Asystor app test case was used to verify the system could handle KBE modules located in different, hierarchically structured folders.
- **PR-05:** This input requirement was fully met for the Primiplane app. For the Modular UAV app, the aspect of providing additional info regarding a slot as docstrings (instead of comments on the line above the part slot) was not met. However, the impact of this was considered small. At most, it would have slightly reduced the LLM performance for the method prompt context mode.
- **PR-06:** Verified by the entire set of test cases presented above, except for input slots. For these, the prototype was found to not be working as intended. This will be discussed in the next section.
- **PR-07:** Verified by the entire set of test cases presented above, except for input slots. For these, the prototype was found to not be working as intended. This will be discussed in the next section.
- **PR-08:** This requirement was implemented through the design decision to use the `get_precedents_tree()` as the main (dynamic) analysis method.
- **PR-09:** This requirement was implemented through the design decisions related to the diagram visualization:
  - Coloring is used as a substitute for swimlanes.
  - Tooltips are shown when hovering the cursor on specific nodes or edges to present additional, more specific information to the user. For example, the node type, object class, and corresponding class method source code are shown when hovering on nodes. Edge tooltips present the data flow by listing all of the slots flowing through a connector. For connectors flowing to/from Parapy parts, this includes all individual input slots. These tooltips are a substitute for the specification window that can be opened when using modeling platforms such as CATIA Magic Systems of Systems Architect to inspect the metadata associated to nodes and node connectors.
  - The implemented partial visualization method provides a functionality similar to that of interactive SysML diagrams in CATIA Magic where a separate, lower-level diagram can be used to provide a detailed specification of an action that is represented by a single node in a higher-level diagram.
- **PR-10:** This requirement was implemented through the addition of the partial visualization method (see subsection 7.5.4).
- **PR-11:** Overlap between various test cases, e.g. R-4.2 and L-4.2, was used to verify the method was deterministic.
- **PR-12:** This hardware-related requirement was implemented through the design decision to use a high-performance workstation for the LLM inference.
- **PR-13:** The computation time was measured for all test cases. The longest LLM inference run was 10 minutes, 7 seconds for all nodes in the graph (average of 3.8 seconds per node). This occurred for test case R-2.2 (modular_UAV.Propeller.JK_eta_from_bet_analysis). Surprisingly, the evaluation of the `get_precedents_tree()` method itself took longer than the LLM inference for the most complex test cases(roughly 20 minutes). Still, this meant all computations were completed within the pre-defined performance requirement of 1 hour.

D

INITIAL COMMUNICATION RESEARCH DIRECTION

Before settling on the development of a communication tool for AI-domain expert collaboration, an alternative research direction explored in for the communication thesis project was improving **software visualization techniques to enhance program comprehension**. This exploration was motivated by the recognition that visual representations play a crucial role in making complex engineering and software models more accessible and understandable.

However, the literature review revealed that while extensive research has been conducted on effective visual communication principles, particularly in the domain of software engineering, the gap between research findings and practical implementation remains significant. Many existing modeling languages and tools—such as UML and SysML—do not fully incorporate well-established visualization principles. Despite the existence of guidelines for aesthetically and cognitively effective visual notations, these guidelines are rarely implemented in industrial tools.

D.1. VISUALIZATION GUIDELINES FOR SOFTWARE DIAGRAMS

The reviewed literature identified several key criteria and frameworks for designing effective software visualizations, ranging from low-level layout aesthetics to high-level cognitive effectiveness principles. A summary of notable contributions is provided in Table D.1.

Table D.1: Summary of key visualization guidelines in software engineering

Authors	Year	Title	Citations	Key Guidelines	Theoretical Foundations	Comments
Purchase *et al.* [101]	2002	Layout Aesthetics in UML: User Preferences	99x	User preference-based aesthetics: bends, edge crossings, orthogonality, layout width, text direction, font type, inheritance style, directional indicators	Based on literature (Colman, Stott Parker 1996; Petre 1995)	Layout-specific guidelines

Table D.1: Summary of key visualization guidelines in software engineering

Authors	Year	Title	Citations	Key Guidelines	Theoretical Foundations	Comments
Moody and Hillegersberg [96]	2008	Evaluating the Visual Syntax of UML: An Analysis of the Cognitive Effectiveness of the UML Family of Diagrams	58x	Cognitive effectiveness principles: semiotic clarity, perceptual discriminability, visual expressiveness, graphic parsimony	Based on Moody's framework (2009)	Defines criteria for evaluating UML diagrams
Wong and Sun [94]	2006	On Evaluating the Layout of UML Diagrams for Program Comprehension	23x	Guidelines based on perceptual theories: Gestalt Laws, Marr's theory, Theory of Notation	Perceptual organization and segregation theories	Covers not only layout but also color, proximity, and interpretation
Agrawala *et al.* [102]	2011	Design Principles for Visual Communication	145x	General principles for effective visualization in communication	Broad synthesis of perceptual and cognitive theories	
Siebenhaller and Kaufman [95]	2006	Drawing Activity Diagrams	19x	Algorithm-based approach for optimizing diagram layout	No theoretical framework	Focus on automated layout generation
Munzner [103]	2014	Visualization Analysis and Design	636x	High-level visualization framework; strong emphasis on interactivity	Synthesis of various visualization theories	Nested model for research design; interactive visualizations emphasized
Pacione *et al.* [104]	2004	A Novel Software Visualization Model to Support Software Comprehension	75x	Notation-based comprehension framework	No specific theoretical foundation	
Moody and van Hillegersberg [35]	2009	The "Physics" of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering	773x	Defines "cognitive effectiveness" as a measurable property of visual notations	Synthesis of multiple theories, including semiotics and communication theories	Foundational paper for defining cognitive effectiveness in visual notation design

D.2. Key Findings in Program Comprehension Research

To complement the study of software visualization, a review of program comprehension research was conducted. This section summarizes key studies that investigated how visual representations impact the understanding of software systems.

Intuitive Understandability of Process Diagrams (Jost et al., 2016)

Jost *et al.* [105] conducted an empirical study comparing the intuitive understandability of process diagrams created with UML Activity Diagrams (AD), BPMN, and EPC. Their research introduced a computational method for determining control-flow complexity and evaluated how quickly and accurately non-experienced users could interpret these diagrams. The study found that higher control-flow complexity negatively impacted interpretability.

Summarizing Large Execution Traces for Comprehension (Hamou-Lhadj & Lethbridge, 2006)

Hamou-Lhadj and Lethbridge [106] introduced a trace summarization algorithm designed to filter large execution traces and improve software comprehension. Their study focused on distinguishing between "utility" and "implementation detail" components within a trace. The method was evaluated using a case study where programmers assessed the quality of generated sequence diagrams using a Likert-scale questionnaire.

Execution Trace Abstraction Using Meta Patterns (Noda, 2012)

Noda [107] proposed a novel trace summarization approach that integrates static and dynamic analysis to detect meta patterns in Java source code. The study introduced metrics for program complexity, such as the number of source files, runtime objects, and detected patterns. While the work provided a framework for abstraction, it lacked empirical validation through comprehension experiments.

Controlled Experiment on Trace Visualization (Cornelissen et al., 2011)

Cornelissen *et al.* [49] conducted one of the first controlled experiments to quantitatively measure the benefits of trace visualization in program comprehension. Their study compared participants using the Eclipse IDE with those using the Extravis visualization tool. Performance was assessed based on task completion time and correctness, revealing that visualization tools significantly aided comprehension.

Comparing Trace Visualization Tools (Fittkau et al., 2015)

Fittkau *et al.* [108] extended prior research by comparing the effectiveness of two trace visualization tools—Extravis and ExplorViz—using controlled experiments. Their study applied the Goal-Question-Method approach and found that different tools yielded varying levels of comprehension benefits, highlighting the importance of visualization design.

Evaluating UML Sequence Diagrams for Thread Interaction (Xie et al., 2007)

Xie *et al.* [109] assessed the effectiveness of an extended UML sequence diagram notation in helping students understand multi-threaded concurrency. Using Bloom's taxonomy, the study formulated questions at different abstraction levels to evaluate comprehension. A pre-test/post-test experiment revealed that participants using the extended notation performed better in understanding thread interactions.

Controlled Experiments on Synchronization-Adorned UML Diagrams (Xie et al., 2008)

Xie *et al.* [110] conducted follow-up research with an improved experimental design, comparing standard UML sequence diagrams with their extended notation. The study employed a larger sample size and introduced two experiments—one with a simple program and another with a more complex system. Their findings demonstrated that synchronization-adorned UML diagrams significantly improved comprehension in multi-threaded environments.

D.3. Conclusion

Improving visualization-based communication remains an important challenge, particularly in engineering disciplines where complex system models are frequently used. However, the literature study results presented above indicated such a the widespread gap between visualization research and practical modeling tools that it was deemed infeasible given the limited scope of this thesis project to dive further into this field and use it as the topic for communication thesis. Future work may explore how the principles summarized in this appendix can be incorporated into modern software reverse engineering tools like the REProcess tool to enhance the clarity and usability of model-based design and software engineering diagrams.

E

Detailed steps and results from exploratory research conducted with SciSpace ChatGPT

This appendix presents a detailed account of the exploratory research activities that were conducted with SciSpace ChatGPT during to development of the conceptual framework (chapter 11). Note that this method was only used for filtering and exploration to identify potentially useful theories and models. These findings provided a starting point for further research in which the findings were checked and verified based on original sources.

E.0.1. Prompt context introduction

During the aerospace part of this thesis, the importance of providing GenAI models with significant contextual information was discovered. Therefore, an elaborate introduction was written and provided at the start of the SciSpace ChatGPT conversation. This introduction is presented below.

> I'm currently working working out my research plan for my science communication thesis research.
>
> Research context: I focus on the development of new ways-of-working and corresponding software tools. I simplify the situation by categorizing people into two groups. Group 1 are generative AI researchers/developers and staff who are designing and implementing the future ways of working to be used in engineering practice. Group 2 are engineers who are actually working in everyday practice on the development of complex, technological, safety-critical systems and will need to approve of and adopt these tools.
>
> Main problem/hypothesis: Researchers/developers of generative AI systems (group 1) are quickly developing new tools with new capabilities, but they currently don't focus enough on the specific needs and demands of engineers who are working on complex, safety-critical technologies and systems (group 2). These two groups need to communicate better and collaborate more closely moving forward. My focus is on improving the communication and collaboration *process* between these two groups.
>
> Research goal: Develop a communication tool/methodology/framework that can improve collaborative development of AI devs & engineers on engineering tools powered by Generative AI.
>
> Your assignment: My next step is to build a theoretical framework to further strengthen my interview protocol. My theoretical framework should help to further specify my interview questions. In particular on the topics of:
>
> 1.) Defining/characterizing collaboration: I need to more precisely define the collaboration in my specific context between group 1 (AI researchers/developers), and group 2 (engineers). For example their relationship and the (perceived) role they fulfill in this. The goal of this literature

research is to define statements regarding specific collaboration barriers/enablers/factors that I can use in the interview to get more specific answers. Otherwise I see a risk where my interviewees will give too generic answers to the questions about Challenges and Goals.

2.) The second aspect I want my theoretical framework to focus on are the different phases that the collaboration between the two groups could be in, and the consequences related to this. When thinking about the GenAI tools that would be developed, I realized that it will make a big difference for the collaboration characteristics if the GenAI tools are still in the conceptual phase (defining vision, needs, requirements), or in active development (iterative, adjusting specific things, very concrete questions) or already implemented and in active use (collaboration focused on reflection and evaluation). My goal here is to support/extend this notion. What does literature say about these different phases? Are there more phases? What are the collaboration characteristics related to each phase?

E.1. MOST USEFUL THEORIES FROM UT TWENTE LIST

After the extensive context introduction, the following question was posed:

From the following list of communication theories, what are the top 5 theories that could be useful for my theoretical framework? Also describe why and how they would be relevant.

[list of 48 theories from [61] inserted here]

E.1.1. GENERATED OUTPUT

Based on this question the following output was generated:

DIFFUSION OF INNOVATIONS THEORY

- **Relevance**: This theory explains how, why, and at what rate new technologies or innovations spread through cultures or groups. In your context, it can help analyze the adoption of generative AI tools by engineers (Group 2) and how the collaboration between developers and engineers can facilitate or hinder this process.
- **How it applies**: Diffusion theory can guide questions about the **adoption barriers** of AI tools, helping you identify the factors that promote or impede the acceptance of these new ways of working within engineering teams. Understanding the stages of innovation diffusion (knowledge, persuasion, decision, implementation, confirmation) will also help frame interview questions about current challenges in collaboration.

MEDIA RICHNESS THEORY

- **Relevance**: Media Richness Theory (MRT) deals with the effectiveness of different communication media based on their ability to convey rich information. For example, face-to-face communication is considered "richer" than emails because of the availability of verbal and non-verbal cues.
- **How it applies**: In improving communication between AI developers and engineers, this theory can help assess which **communication tools** are most effective at conveying complex technical details, reducing misunderstandings, and ensuring that the right information is transmitted in the right way. This will help in framing questions about how communication tools are used and where they fall short.

MENTAL MODELS

- **Relevance**: Mental Models refer to the cognitive frameworks or structures that people use to understand and interact with the world. AI developers and engineers likely have different mental models due to their distinct expertise and experiences.
- **How it applies**: Investigating how the different mental models of engineers and AI developers affect their communication and collaboration can help identify **gaps in understanding** or alignment. Questions could explore how each group conceptualizes the role of AI in safety-critical systems and where their mental models diverge, causing barriers to collaboration.

SOCIAL PRESENCE THEORY

- **Relevance**: Social Presence Theory focuses on the degree to which a communication medium allows users to perceive others as being physically present, even in digital or remote interactions. It's important in understanding how collaboration happens in remote or asynchronous environments.

- **How it applies**: For teams working remotely or using digital collaboration tools, Social Presence Theory can help frame how the perceived "presence" or **engagement** of team members affects the quality of collaboration. This can be used to ask about the impact of digital tools on team cohesion and how engineers and developers feel connected—or disconnected—during collaborative development efforts.

ADAPTIVE STRUCTURATION THEORY

- **Relevance**: This theory examines how people interact with technology and each other within organizational settings, especially focusing on how structures (such as tools or systems) evolve based on human actions and feedback.

- **How it applies**: This theory is relevant when investigating how **collaboration tools** or communication systems (e.g., shared documents, design platforms) are used by developers and engineers and how they might need to be adapted over time. The theory can guide questions about how current systems either enable or hinder collaboration and where adaptability is needed to improve communication and teamwork.

E.2. ADDITIONAL THEORIES TO CHARACTERIZING TYPES OF COLLABORATIONS

After obtaining the result presented above, the following question was posed:

> What theories/models exist that I can use to characterize types of collaborations? For instance on characteristics like short-term vs long-term collaboration, intensive vs sporadic, etc.

E.2.1. GENERATED OUTPUT

Based on this question the following output was generated:

RELATIONAL COORDINATION THEORY

- **Overview**: Relational Coordination Theory, developed by Jody Hoffer Gittell, emphasizes the role of high-quality communication and strong relationships in coordinating work across teams. It focuses on relationships and their impact on performance in complex and interdependent tasks.

- **How it applies**: This theory can be used to characterize **intensive** vs. **sporadic** collaboration by looking at how teams communicate and relate across different types of tasks. Intensive collaborations are marked by frequent, high-quality communication and shared goals, whereas sporadic collaborations might involve more transactional and infrequent interactions.

- **Key characteristics**: Frequency of communication, shared goals, mutual respect, and task interdependence.

TIME-SPACE MATRIX (JOHANSEN, 1988)

- **Overview**: The **Time-Space Matrix** characterizes collaboration based on whether the interactions are synchronous or asynchronous and whether they happen in the same physical location or remotely.

- **How it applies**: This model can help distinguish between **short-term** vs. **long-term** collaborations and **intensive** vs. **sporadic** interactions by considering how the collaboration is coordinated over time (e.g., in real-time or over a longer period) and whether participants are co-located or remote.

- **Key characteristics**: Time (synchronous/asynchronous) and location (same place/different places).

COLLABORATIVE NETWORK THEORY

- **Overview**: This theory explores how organizations or teams collaborate in networks to achieve shared objectives. It is particularly useful for understanding long-term, sustained collaborations vs. short-term or project-based collaborations.

- **How it applies**: This theory helps to characterize **long-term** vs. **short-term** collaboration by analyzing how formal or informal networks of collaboration are structured, and how they adapt over time.

Long-term collaborations are more likely to involve strong, trust-based networks, while short-term collaborations may be more transactional.

- **Key characteristics**: Network structure (formal/informal), duration (temporary/ongoing), purpose, and resource sharing.

PROJECT LIFECYCLE MODELS

- **Overview**: Project lifecycle models, such as the **PMI's Project Lifecycle** or **Agile frameworks**, provide structures for understanding collaborations based on their stages (initiation, planning, execution, monitoring, and closure). These models also highlight how collaborations may evolve from intense (development phase) to sporadic (maintenance phase).
- **How it applies**: These models can help characterize the **intensity** of collaboration at different phases of a project. Early stages (planning, design) tend to be more **intensive**, while later stages (implementation, maintenance) may be more **sporadic** or less involved unless issues arise.
- **Key characteristics**: Collaboration intensity, focus on deliverables, project stages, feedback loops.

TRANSACTIONAL VS. TRANSFORMATIONAL COLLABORATION

- **Overview**: This framework differentiates between **transactional collaboration**, where parties engage in exchanges of resources or information for immediate benefit, and **transformational collaboration**, which focuses on deeper, long-term partnerships aimed at mutual growth and innovation.
- **How it applies**: You can use this to assess the nature of collaboration: short-term, goal-focused engagements (transactional) vs. long-term, deeply interwoven collaborations that seek transformative change (transformational).
- **Key characteristics**: Depth of engagement, duration, mutual benefit, and innovation potential.

COLLABORATIVE CONTINUUM (THOMSON & PERRY, 2006)

- **Overview**: The collaborative continuum model suggests that collaboration exists on a spectrum from **cooperation** to **coordination** to **collaboration**, depending on the depth and formality of the interaction.
- **How it applies**: This model is useful for characterizing collaborations based on their formality and intensity. **Cooperation** represents low-intensity, short-term, or sporadic collaboration, while **collaboration** involves higher-intensity, more formal, and long-term relationships.
- **Key characteristics**: Degree of interdependence, intensity of collaboration, shared authority, and trust.

STRENGTH OF WEAK TIES (GRANOVETTER, 1973)

- **Overview**: Granovetter's theory of the **Strength of Weak Ties** explains that weak, sporadic connections between individuals or groups (e.g., occasional collaborators) can be just as important as strong, intense relationships for sharing novel information and forming new opportunities.
- **How it applies**: You can use this to explore **sporadic** vs. **intensive** collaboration. Weak ties are valuable in short-term or sporadic collaborations, whereas strong ties are critical for long-term, intensive partnerships.
- **Key characteristics**: Frequency of interaction, strength of relationship, and information exchange.

ADAPTIVE STRUCTURATION THEORY

- **Overview**: Adaptive Structuration Theory (AST) explains how teams and organizations adapt communication and decision-making structures based on available tools and technologies.
- **How it applies**: This theory can be used to characterize the evolution of collaboration over time, from **initial** (short-term, sporadic) to **mature** (long-term, intensive) collaborations. It helps assess how teams adjust collaboration styles depending on the phase of a project and the communication technologies in place.
- **Key characteristics**: Adaptability, technology use, evolving team structures, and duration of collaboration.

TUCKMAN'S STAGES OF GROUP DEVELOPMENT (1965)

- **Overview**: Tuckman's model identifies five stages of group development: **forming**, **storming**, **norming**, **performing**, and **adjourning**.
- **How it applies**: This model can be used to assess how collaborations evolve over time, from initial, **short-term** stages (forming) to more **long-term** engagements (performing), where the group operates effectively and efficiently. The model also highlights how collaboration intensity shifts through these stages.
- **Key characteristics**: Group development stages, conflict management, team cohesion, and performance.

TASK INTERDEPENDENCE THEORY

- **Overview**: Task Interdependence Theory classifies collaboration based on how much one member's work relies on the actions of others. There are three main types: pooled, sequential, and reciprocal interdependence.
- **How it applies**: **Task interdependence** is a way to characterize collaboration intensity. **High interdependence** (reciprocal) requires close, frequent interaction (intensive collaboration), whereas **low interdependence** (pooled) allows for more independent work and **sporadic** collaboration.
- **Key characteristics**: Degree of interdependence, collaboration frequency, complexity, and coordination needs.

F

INTERVIEW DESCRIPTION

This one-page interview description was attached to the email invitations sent out to potential interviewees:

Introduction
My name is Joris Koopman (email: j.p.koopman@student.tudelft.nl) and I'm working on my MSc thesis for a double degree in Aerospace Engineering and Science Communication at TU Delft. My thesis focuses on the development of novel engineering tools & methods powered by (generative) AI.

Science Communication research:
For the Science Communication part of my thesis, I'm investigating the communication between different stakeholders involved in the development process of AI-powered tools & methods. In particular, I'm investigating the interaction between two groups:

- **Group 1:** Domain experts who work on or interact with complex, critical systems. (e.g. aerospace, nuclear, medical technology, etc). In addition, these experts are involved in AI-driven projects that aim to improve current work practices in terms of efficiency, quality, speed, safety, etc.
- **Group 2:** (Generative) AI researchers and developers working on novel tools and methods in collaboration with domain experts from group 1.

The nature of the complex, critical domains imposes strict requirements on any new tools or methods adopted. As a result, successful tool development relies on effective interdisciplinary collaboration between these two groups. However, there are many (potential) challenges: different backgrounds, different organizations, unrealistic expectations, etc.

Research Objective and Plan
My main objective is to enhance the collaboration between the two groups outlined above by increasing the effectiveness of their communication. To achieve this, I first need to learn more about current practices and identify key collaboration and communication challenges. I am therefore reaching out to interview professionals from both groups.

Indicative questions
A few example questions are provided below to give you an idea of the topics we'll discuss:

- How do you think your domain-specific work practices will be transformed by (generative) AI-powered tools and methods, and why?
- Based on your experience, what are the 3 most relevant challenges related to AI tool development? [select from list of options]
- To what extent is the shared knowledge about each other's domain sufficient to communicate effectively? [scale 1- 5]
- From the following ideas for communication tools/methods, which one(s) do you like best, and why? [review list of options]

Confidentiality
The interviews are part of a TU Delft thesis research project. Transcripts and recordings are kept confidential and results will be anonymized before presentation/publication. Participants may refuse to answer, or withdraw from the study at any time.

G

INTERVIEW CONSENT FORM

Interview consent form
You are being invited to participate in a research study titled *Improving interdisciplinary communication in advanced AI projects*. The study is carried out by Joris Koopman (j.p.koopman@student.tudelft.nl) for a Master thesis project at TU Delft, under supervision of Caroline Wehrmann (c.wehrmann@tudelft.nl).

The purpose of this study is to better understand the interdisciplinary collaboration between AI researchers/developers and experts from other domains, and identify key collaboration and communication challenges. To this end, a series of interviews will be conducted with both groups (AI researchers/developers and domain experts). These will serve as input to inform the design of a tool or method that improves interdisciplinary communication.

The collected data is used for the MSc thesis project of the researcher and will be anonymized in any publications (e.g. thesis report). In addition, publications will mainly present aggregated results. Specific, anonymized quotes may be included but full transcripts or recordings will not be published.

As with any online activity the risk of a breach is always possible. To the best of our ability your answers in this study will remain confidential. We will minimize any risks through the following measures:

- Only MS Teams and its built-in functionalities will be used to capture data.
- Recordings and transcripts will only be processed locally.
- Recordings and unprocessed (i.e. identifiable) transcripts will only be shared with supervising researchers. They will not be shared with any other third-parties. This includes uploading to web and cloud services.
- Recordings and unprocessed (i.e. identifiable) transcripts will be removed at the end of this research project (Feb 2025).

Your participation in this interview is entirely voluntary and you can withdraw at any time. You are free to omit any questions. Furthermore, until 17 Jan 2025, you may: 1) access your data in order to rectify it or erase it; 2) submit a written request to be excluded from the study, upon which all data will be removed. After this date, preliminary results may have already been submitted and exclusion from the study cannot be guaranteed.

Please sign below to consent to the terms of this study:

Name of participant	Date	Signature

H

DESIGN INPUT FROM INTERVIEWEES ABOUT MORPHOLOGICAL CHART

H.1. INPUT FROM AI-3

Conclusions/takeaways:

- Most important **cross-domain knowledge to share:**
  - Make DE understand the capabilities and limitations of GenAI technology, including different levels of maturity of an output. For example, it can generate images of concepts, but not their technical drawings. Or for auto-generated part design: AI can generate simple 2D sketches/drawings to be used during the conceptual design phase, but not (yet) a complete CAD model.

    AI-3: "one key AI concept that is misunderstood by domain experts is the capabilities of generative AI models. What it can do currently, what it cannot, and what outputs it can produce. So, for example, what kind of representation a generated CAD model is, and whether that is usable for manufacturing purposes or whether it can be only used for conceptualization and during the planning phase of the product development life cycle. The key point here is understanding the generative AI model technology itself, and the capabilities and the limitations of the model."
  - The current engineering workflow of Domain Experts + The ML workflow of AI developers and required training data.

    AI-3: "Of course, it depends on different use cases, but in the collaboration between domain experts and AI engineers, it's very important to emphasize understanding first of all the existing workflows of domain experts. And at the same time, it's also very important for domain experts to understand how the AI development life cycle, or the machine learning development life cycle works. So what the requirements are for better data, training data, et cetera. These are the main points that I think are very important to facilitate the collaboration."
- Most important AI features to educate DE on:
  - Provide transparency, Human-in-the-loop system, with intermediate checking/approval steps
    AI-3: "At the same time, providing maximum transparency how the AI model is operating or is being run. This also a very important key concept that domain experts should be aware of. Transparency in terms of, for example, if you have an intermediate step that is performed before the final output in the AI pipeline, the output of that intermediate step should go back to the user or to the domain expert in order to verify the output. So they can be the human-in-the-loop factor here, in order to verify how the AI model was able to generate the output, based on the intermediate outputs or the intermediate steps. Providing maximum transparency of the AI pipeline to the domain experts is very useful, and actually might be required by domain experts so they can verify and approve the output of the intermediate step before the AI pipeline generates the final output."

H.2. INPUT FROM AI-1

Three key aspects that need to be included:

- **Purpose:** What exactly are you delegating to the LLM? Are you merely using it to perform a certain step (eg connect/transforming inputs and outputs from other analysis steps in the engineering process)? Or is it agentic, and does it get the responsibility to make certain decisions? These could be design decisions (e.g. select a certain part, use certain architecture, etc.) or design process decisions (based on analysis results X, the next step is to perform analysis Z).
  Related to above, two types of errors can occur: 1) it can do things wrong, or 2) it can do the wrong thing. When doing things wrong (option 1), its usually because LLM is doing activity that should be done by tool or human (eg calculations)
- **Level of automation / user interaction:**
  - Do you want it to be a chat that says: "hey, here's my suggestion for your new design. You could do this or that".
  - Do you simply want it taken care of by the AI?
  - Or do you want the engineers in complete command and control over it?
- **Computational support**, how much knowledge is taken from the Generative AI and how much is done by traditional engineering tools / systems (like KBE app, CFD, FEM, etc.). And how much still by designer themselves?

Triangle of ownership basically defines the design space as division of **responsibility** in the automation and interaction between:

- **Designer – AI:**
  - How much control over the workflow?
  - How much knowledge from Designer vs AI?
- **Designer – Tool:**
  - Through what interface (chatting)?
  - How much manual input vs (AI-powered) automation?
- **LLM – Tool:**
  - How much design knowledge used from LLM itself (so embedded in NN) vs traditional computations/analyses?

Quotes from AI-1 related to conclusions:

- **Issue of design feature dependencies** "Core assumption of morphological matrices in general is that design dimensions are independent, which does not hold for LLMs" -> Challenging to create a matrix because many design choices/aspects depend on others and certain design choices are mutually exclusive.
- "I think that you first need to decide what it [the AI] is gonna be doing. What is the intention? Like what is the purpose of the AI? Why do you want it there? Why did you want to do this? And then once you have it, you have a subset of questions. So if it's to analyze things, then you ask the question if it is using tools or not using tools. Is it supposed to do that? But if it's just summarizing, for example, then you don't have to ask all those questions and the according levels [options]. So that's why I said maybe a morphological matrix is not ideal."
- **Clarify purpose and approach:** "I think what you should add is to try to understand what is the purpose of the AI tool, the generative AI tool. Similar to synthesis and so on, but I don't think it's clear enough what the model can do. Especially because you're talking about generative AI and due to the ChatGPT legacy and experience. Basically, an **agentic LLM** can do two things: it can choose what steps to take, and it can execute steps as well. That's a distinction."

- **AI capabilities:** “the problem when you are developing a tool is that you can get it wrong in two ways. You can get it wrong because the Large Language Model is doing the wrong thing, or because it’s doing the thing wrong. So, doing the thing wrong typically is not the problem. The problem is that you are giving the Large Language Model an activity that should be done by another tool or by a human, so we’re not using the Large Language Model to its best. For example, ‘add 72 point something plus something’, you should use a calculator for doing that. Just don’t ask the model.”
- **Level of user control vs automation (division of responsibilities/ownership):** “what is the automation that you’re expecting from the tool? Do you want it to be a kind of chat suggesting new design, like ‘Hey, a suggestion. You could do this or that.’ Or can I just ignore it and have the engineers in command and control?”
- **Level of user control vs automation:** “How much do you want this to be automated? Do you want it to be taken care of, or do you want to do it in an interactive way? To what extent is someone in command?”
- **Computational support:** “And then within the computational support: how much is done by the LLM, or generative AI, and how much is done by traditional tools?”
- **Workflow flexibility:** “How much freedom does the generative AI have in in answering your questions? Is it like what we said before? Is it really fixed flow of things to do? Or is it depending on the output to do one thing or the other? Is it a free script React architecture?”
- **Purpose:** “I think that you first need to decide what it [the AI] is gonna be doing. What is the intention? Like what is the purpose of the AI? Why do you want it there?”

H.3. INPUT FROM FOLLOW-UP MEETING WITH AI-2

- **User-Friendliness: Customization Control** The ability to create a system prompt yourself versus only having access to a simple text box. This raises the question: how much do you want to do yourself?
- **Types of Domain experts who approach AI team**
  - **Dimension 1: Level of AI understanding**
    - Those who simply view AI as an abstract entity that can provide a solution to their problem.
    - “Find and replace” users: for instance, “if it can easily find IBANs, it should also handle names or locations.”
    - Users who understand the broad family of AI solutions but only need knowledge about specific applications.
  - **Dimension 2: Source of motivation to start project with AI team:**
    - "The desperate" who are seeking a solution for acute problems. Eg being overloaded with more work than they/their team can handle.
    - “The status seekers” who are incentivized by prospect of gaining professional status associated with being involved in successful ai project. It makes them appear competent
    - “The balanced” who have a pretty good understanding of the tech and required investment from their side. Generally less urgent/stressed than the desparate and more intrinsically motivated than the status seekers.
- **Preconditions for Success:**
  - How well is your workflow documented?
  - Is the data well-organized and properly meta-tagged?
  - Are proper risk management and control practices, like a four-eyes principle, already used and/or defined in formal processes? This enables easy human-in-the-loop implementation
- **Levels of AI Autonomy:**
  - **Human Out of the Loop:** Fully autonomous operation.

  - **Human on the Loop:** Monitoring with occasional intervention (e.g., wildlife cameras).
  - **Human in the Loop:** Active involvement in decision-making.
  - **Human in Control:** Full oversight and command.
- **Neuro-Symbolic Compound AI Systems:** Combining symbolic reasoning with neural networks.
- **Response Latency:**
  - Do you prefer well-checked and thought-out answers, or snappy responses?
  - Implementation could involve an agent, a controller, or just an LLM (e.g., GPT-4).
  - Consider batch processing versus real-time chat systems.
- **Interpreting Results:** Should outputs be human-readable (e.g., text) or machine-readable (e.g., JSON)?
- **Iterating on Output:**
  - Input into another system (e.g., code).
  - Iteration within the chat interface (e.g., editing the second sentence).
  - Copy-pasting elsewhere for further refinement without returning to the original system.

Managing Temporary Context:

- Do you build a chat history?
- Consider the trade-off: more compute and time versus better continuity.
- **Criticality of Downtime:** Batch processing use cases versus real-time requirements like chatbots.
- **LLM Capabilities:** LLM = knowledge base + logic skills -> advisable approach is to rely more on logic skills than on stored knowledge.
- **AI Scope:** How broad or narrow is the task spectrum? Narrow implementations can involve either prompts or fine-tuning.
- **Control Over Behavior:** Fine-tuning is essential for controlling AI skills and behavior.
- **Context Provision:** Do users need to supply context manually, or is this automated?
- **LLM Integration Approaches:**
  - Embedding a tool into an LLM.
  - Embedding an LLM into an existing tool (e.g., Copilot).
  - Using an LLM alongside existing tools.
- **Proving Effectiveness:**
  - What data proves the system works?
  - How critical is it to validate this, depending on the use case?

Input and Response Taxonomy: Structured prompting, such as JSON-based “fill in the gap” tasks.

Ethical Considerations: Categories of risk: Low, Medium, and High.

I

LIST OF DESIGN DIMENSION IDEAS

The list below contains thoughts and ideas for design dimensions collected over time. Some were ideated based on personal knowledge and experience, others inspired by literature or mentioned by interviewees during the initial interview series and follow-up conversation conducted with some of them. This list formed the basis for the first prototype (iteration 1) that was created.

- **Target users:** Only specific roles; All engineers in organization; All staff in organization; All engineers from partnership (inside and outside organization); Open to community; Etc.
- **Output (what does the GenAI tool provide?):** Analysis; Recommendations; Synthesis; (autonomous) Decisions
- **AI Role: Analyze, Synthesize, Agent**
- **GenAI modality:** Text (natural language); Source code; Datasets; Images; Video; Etc.
- **Explainability & traceability:** Black-box (only save output or prompt+output); Chain-of-thought prompting (save all steps); Rule-based breakdown + prompting (save all steps); RAG (save prompts, retrieved sources, outputs); Etc.
- **User control:** Running permanently in background; Only on request of user itself; Also on request of other users; Etc. Inspired by Tsiakas and Murray-Rust [111]: "Human-AI interaction is defined as 'the completion of a user's task with the help of AI support, which may manifest itself in non-intermittent scenarios' [71]. Following this definition, there are three main Human-AI (HAI) interaction paradigms, *intermittent, and continuous, and proactive, taking into consideration 'how differences in initiation and control result in diverging user needs'*".
- **GenAI scope:** Provide entire functionality; Only provide specific, partial functionalities (eg when combined with heuristic algorithm).
- **Human-AI Team structure:** 1 human & 1 AI; many humans & 1 AI; 1 human & many AI's; many humans, many AI's
- **Collaboration hierarchy:** Human above AI; human equal to AI; AI above human.
- **AI information access:** Company intranet/wiki; Engineering database; Personal communication (e.g. email); Access to public internet; Etc.
- **Relations to other tools (+ integration?):** Stand-alone system; Add-on to existing tool; Synergy with existing tool; Replacement of existing tool;
- **Future developments:** Expand capabilities of this tool; Integrate into larger system; Connect to other tools/systems/platforms; Etc.
- **Model performance metrics:** Look into Design at Scale AI Lab from TU Delft.

- **Cardinality / Traceability:** At what level are questions posed to the LLM? Ie what is the size of the information chunk that is provided in one prompt, and how comprehensive is the output that is requested? Are we mapping entire classes to a single description; or a sequence of code lines to a listed summary; or…?
- **Neural Network Datatype:** Text tokens, Graphs (Graph Neural Networks), Pixels, …
- **Targeted project lifecycle phase:** Conceptualization; Development; Production/Manufacturing; Service&Support; Retire
- **Model training effort:** Pre-trained model (off-the-shelf); Finetune model; Training your own model
- **Model input collection (prompt engineering):** Which information sources are collected and combined into the input? → None (only use embedded knowledge); Additional textual information; Example Question-Answer pairs; Additional non-textual information (i.e. multimodal, like image retrieval); etc.
- **Desired response structure:** None; keyworded response start; structured prompting.
- **Required maturity: Prototype (Technology Readiness Level 4/5); Minimum-Viable Product (TRL 6); Launch V1.0 spec (TRL 7); Mature and extensively tested (TRL 9)**
  Note: This corresponds to different approaches in the Machine Learning (ML) workflow. If domain expert agrees to target a lower maturity (prototype or Minimum Viable Product), more Reinforcement Learning can be used instead of extensive Finetuning, testing and verification up front. This directly translates into more iterative approach. However, on the flipside more effort should be invested into the pipeline of collecting user feedback and quickly using this to improve the model after launch.

J

Iteration 3: Additional results from comparing morphological chart with existing literature frameworks

Design Principles for GenAI applications

- **Authors:** Weisz *et al.* [2]
- **Reviewed components:** Table 1 (Table 14.1 in this report)
- **Relevant for which aspects of morphological chart:** All
- **Applicable findings:**
  - Add risk dimension with list of potential system harms? Options: address value tensions between users, manage emergent behaviors, potential bias, hallucination
  - Add customizable user experience dimension? Options: No differentiation, user can tune specific things, different user profiles, ...?
  - Add output annotation and curation? Options: user-driven or automated systems to organize, label, filter and/or sort outputs. Could also include: output highlighting to draw attention to differences or variations across outputs + ways to make uncertainty visible in output.
- **Comments:** Only model that related well to transparency, determinism and user training dimensions. This also indicates a gap in existing related work. Also really underlines the importance of the limitations and prerequisites category of the morphological chart

Design resources to scaffold AI concept ideation

- **Authors:** Yildirim *et al.* [84]
- **Reviewed components:** Table 3, page 7 (2332)
- **Relevant for which aspects of morphological chart:** Role of GenAI
- **Applicable findings:**
  - Provide lots of synonyms to clarify different Role of GenAI within envisioned system
- **Comments:** Aside from the roles of GenAI, the approach of this was more interesting than the content and findings of the article itself

An HCI-centric Taxonomy of Human-GenAI interactions

- **Authors:** Shi *et al.* [85]
- **Reviewed components:** Sections 4-8 -> all Categories and Options

- **Relevant for which aspects of morphological chart:** All
- **Applicable findings:**
  - Output Synchronization: Improvement over previous Human-AI interaction mode dimension
  - Add Application domains dimension? Options: Research & Science, Writing, Programming, Robotics & IoT, Training, 3D modeling, Design
- **Comments:** Very useful overall. Cardinality could be used for GenAI function categorization: Interesting but not yet determined how to use or include. Objects to control and Mediums of control: this raised the question whether the current morphological chart has got enough user interaction aspects.

Morphological Box for AI solutions

- **Authors:** Rittelmeyer and Sandkuhl [82]
- **Reviewed components:** Figure 1, page 2
- **Relevant for which aspects of morphological chart:** All
- **Applicable findings:**
  - Add end-user characterization? Options: IT-expert, -savvy, or layman
- **Comments:** In terms of the form, this was the most similar tool to the morphological chart of this research which made it interesting to use for comparison. However, the communicative quality of this tool was rated to be rather low. It adopts a more technical perspective, has several unclear dimensions and options with lacking explanation, and the ordering is rather chaotic. It was interesting to have such a similar tool to compare with but due to its quality, the yield in terms of applicable findings was rather low.

Review of AI applications in engineering design

- **Authors:** Yüksel *et al.* [4]
- **Reviewed components:** Table 2 + Fig 20
- **Relevant for which aspects of morphological chart:** Role of GenAI
- **Applicable findings:**
  - Design practices category used for Role of GenAI within system
- **Comments:** Quite technical, good understanding application of different AI methods in engineering context. However, besides the design practices category, it was not suitable for direct application to the morphological chart to inform specific design dimensions and options.

The AI methods, capabilities, criticality grid

- **Authors:** Schmid *et al.* [3]
- **Reviewed components:** Table 3 (capabilities) + Fig 2 (criticality)
- **Relevant for which aspects of morphological chart:** Role of GenAI + Risk/Criticality
- **Applicable findings:**
  - Add risk dimension? Options: Based on AI Criticality (Fig 2) which directly relates to risk assessment -> adopt these levels of risk + mitigation through test & regulation
- **Comments:** The AI capabilities overview (Figure 14.3): turned out to not be so useful. They used a different perspective for the medium level categorization (e.g. factual, conceptual, procedural, etc.), which is too technical. One level higher the terms are too broad (e.g. "process and understand" category). Lowest level is too fine-grained, (e.g. terms like recognize, classify, provide, differentiate, determine, assemle). The risk dimension was found to actually be more interesting. Aligns with risk classification from European AI act [1].

K

Morphological chart iteration used for validation

Category	Design dimension	Option 1	Option 2	Option 3	Option 4	Option 5	Option 6
Scope & Purpose	**Integration with existing systems**	**Add-on to existing system**	**Stand-alone system**	**Integrating component**	**Centralizing component**		
	Role of GenAI within system	**User interaction** and communication	**Generate** (new or refined)	**Diversify**, augment, explore (alternatives)	**Interpret, summarize, sense-making**	**(Support) decision making**	**Transform** (bridging function)
	Targeted product life cycle phase	**Conceptualization** and **Design**	**Development** and **Testing**	**Manufacturing** and **Production**	**Distribution** and **Marketing**	**Monitoring** and **Support**	
User Experience	**Interaction synchronization**	**Up-front AI output** (before user interaction)	**Real-time, concurrent AI output**	**Delayed AI output** (after user command)			
	Workflow Flexibility & Automation	**Pre-defined, static**	**User-controlled, static** (tuning inputs)	**Flexible, human-in-the-loop**	**Flexible, non-AI control** (rule-based, fuzzy logic)	**Flexible**, controlled by **AI agent**	
	User approval & iteration (of GenAI output)	**No iteration or approval**	**Simple re-runs**	**Manual output correction** by user	User **approval** or choosing from **options**	**Prompt-based iteration**	**Multimodal iteration** (e.g. highlight + instruct)
	Required user training	Instructional **presentation**	Instruction **manuals or videos**	**Training sessions**	**Interactive tutorial** within tool		
GenAI Model Input	**Type(s) of (additional) input data**	**Unstructured text** (natural language)	**Semi-structured text** (source code, legal contracts, technical documentation)	**Structured data files** (spreadsheets, 3D models)	**Images**	**Graphs**	**Other: ...** (audio, video, sensors, robot control, etc)
	Method of collecting (additional) input data	**None**, rely on GenAI model knowledge (not advised)	**User provides** (additional) input data	**Non-AI system retrieves** (additional) input data	**Non-AI system computes** (additional) input data	**AI searches** for (additional) input data	

Output	**Output data type(s)**	**Unstructured text** (natural language)	**Semi-structured text** (source code, legal contracts, technical documentation)	**Structured data files** (spreadsheets, 3D models)	**Images**	**Graphs**	**Other: ...** (audio, video, sensors, robot control, etc)
	Target level of output maturity	**Proof-of-concept prototype** (TRL 3/4)	**Minimum-Viable Product** (TRL 6)	**Mature & qualification tested** (TRL 8)			
	Transparency	**Black-box approach** (not advised)	**Stepwise traceability** (process logs)	**Piecewise traceability** (input-output pairs)	Provide **output rationale**		
	GenAI Determinism	**Deterministic** (identical re-runs)	**Non-deterministic** (variation in re-runs)				
Limitations & Prerequisites	**Output quality assessment**	Quality **quantification** methods **exist**	Quality **quantification** methods **can be defined**	Robust **qualitative** assessment methods **exist**	Robust **qualitative** assessment methods **can be defined**	**Unfeasible to robustly assess** output quality (project is unfeasible)	
	Training data	**High-quality, labelled** training dataset **available**	Training data **unavailable**, but **can be retrieved**	**Manual generation** of training data required	Obtain training data **retrospectively** (from users of MVP implementation)	**Unfeasbile** to obtain training data (project is unfeasible)	
	Current work process: level of formalization	**Formalized, well-documented** including **risk management**	**Formalized, well-documented** work processes	**Undocumented** but **standardized** work processes	**Undocumented, flexible** work processes	**Novel, undocumented** work process	
	Data Security & Privacy*	No issue to send data to servers **outside EU**	Only **EU servers** allowed	**Private online servers** are allowed	Only **internal servers** (intranet) are allowed	Only **on-device compute** is allowed	
Outlook	**Future developments steps**	Collect data for **training or finetuning custom AI**	Improve **output quality** and maturity	Improve **user experience** and control	**Expand capabilities**, additional features	**Connect to other systems**, tools, platforms	**Other: ...**

Bibliography

[1] Directorate-General for Communications Networks, Content and Technology, *AI Act | Shaping Europe's digital future,* (2024).

[2] J. D. Weisz, J. He, M. Muller, G. Hoefer, R. Miles, and W. Geyer, *Design Principles for Generative AI Applications,* in *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, CHI '24 (Association for Computing Machinery, New York, NY, USA, 2024) pp. 1–22.

[3] T. Schmid, W. Hildesheim, T. Holoyad, and K. Schumacher, *The AI Methods, Capabilities and Criticality Grid: A Three-Dimensional Classification Scheme for Artificial Intelligence Applications,* KI - Kunstliche Intelligenz **35**, 425 (2021).

[4] N. Yüksel, H. R. Börklü, H. K. Sezer, and O. E. Canyurt, *Review of artificial intelligence applications in engineering design perspective,* Engineering Applications of Artificial Intelligence **118**, 105697 (2023).

[5] A. for Education, *Generative AI Explainer,* .

[6] M. Baan, *Design Exploration Framework based on AI for froNt-loaded Engineering (DEFAINE) Project,* (2020), last accessed: 15-2-2025.

[7] W. J. Verhagen, P. Bermell-Garcia, R. E. van Dijk, and R. Curran, *A critical review of Knowledge-Based Engineering: An identification of research challenges,* **26**, 5.

[8] A. R. Kulkarni, *DEFAINE Report 3.1.1. Requirement-product-process Ontology* (2022).

[9] D. Bansal, A. R. Kulkarni, and G. La Rocca, *DEFAINE Report 3.2.2. ParaPy language ontology and mapping to Knowledge Model* (2024), confidential.

[10] A. Raju Kulkarni, D. Bansal, G. la Rocca, F. Mendes Fernandes, R. Augustinus, and B. Timmer, *An MBSE approach to support Knowledge Based Engineering application development,* (2023), 10.13009/EUCASS2023-495.

[11] S. Friedenthal, A. Moore, and R. Steiner, *A Practical Guide to SysML: The Systems Modeling Language*, 2nd ed. (Morgan Kaufmann, 2012).

[12] F. M. Fernandes, *A Model-Based Systems Engineering Framework for developing Knowledge Based Engineering Applications,* (2023).

[13] E. Chikofsky and J. Cross, *Reverse engineering and design recovery: A taxonomy,* **7**, 13.

[14] G. Canfora, M. Di Penta, and L. Cerulo, *Achievements and challenges in software reverse engineering,* **54**, 142.

[15] B. Cornelissen, A. Zaidman, A. Deursen, L. Moonen, and R. Koschke, *A Systematic Survey of Program Comprehension through Dynamic Analysis,* **35**, 684.

[16] A. Kuhn, S. Ducasse, and T. Gîrba, *Semantic clustering: Identifying topics in source code,* 12th Working Conference on Reverse Engineering, **49**, 230.

[17] S. Thakare and A. W. Kiwelekar, *Discovery of Layered Software Architecture from Source Code Using Ego Networks,* arXiv:2106.03040 [cs] .

[18] D. Holten, B. Cornelissen, and J. Wijk, *Trace Visualization Using Hierarchical Edge Bundles and Massive Sequence Views* (2007) p. 54.

[19] T. Ziadi, M. A. A. da Silva, L. M. Hillah, and M. Ziane, *A Fully Dynamic Approach to the Reverse Engineering of UML Sequence Diagrams,* in *2011 16th IEEE International Conference on Engineering of Complex Computer Systems*, pp. 107–116.

[20] C.-H. Lung, M. Zaman, and A. Nandi, *Applications of clustering techniques to software partitioning, recovery and restructuring,* **73**, 227.

[21] D. Heuzeroth, T. Holl, G. Hogstrom, and W. Löwe, *Automatic Design Pattern Detection*, p. 103.

[22] F. Arcelli Fontana and M. Zanoni, *A tool for design pattern detection and software architecture reconstruction,* **181**, 1306.

[23] M. Pinzger and H. Gall, *Pattern-supported architecture recovery,* in *Proceedings 10th International Workshop on Program Comprehension* (IEEE Comput. Soc) pp. 53–61.

[24] W. D. Pauw, D. Lorenz, J. Vlissides, and M. Wegman, *Execution Patterns in Object-Oriented Visualization,* .

[25] H. Brunelière, J. Cabot, G. Dupé, and F. Madiot, *MoDisco: A model driven reverse engineering framework,* **56**, 1012.

[26] K. Lano, *Agile Model-Based Development Using UML-RSDS*, p. 373.

[27] L. Li, J. Wang, and H. Quan, *Scalpel: The python static analysis framework,* arXiv preprint arXiv:2202.11840 (2022).

[28] W. Hesse and T. Tilley, *Formal Concept Analysis Used for Software Analysis and Modelling,* in *Formal Concept Analysis: Foundations and Applications*, Lecture Notes in Computer Science, edited by B. Ganter, G. Stumme, and R. Wille (Springer) pp. 288–303.

[29] L. Floridi and M. Chiriatti, *GPT-3: Its Nature, Scope, Limits, and Consequences,* Minds and Machines **30**, 681 (2020).

[30] OMG, *XML Metadata Interchange (XMI) Specification,* (2015).

[31] A. Aleksandravičienė and A. Morkevičius, *MagicGrid® BOOK OF KNOWLEDGE* (Vitae Litera, 2021).

[32] D. Bansal and G. La Rocca, *DEFAINE Report 3.3.1. KBE visual editor* (2024), public.

[33] O. M. Group, (2015).

[34] G. La Rocca, *Knowledge based engineering: Between AI and CAD. Review of a language based technology to support engineering design,* Advanced Engineering Informatics **26**, 159 (2012).

[35] D. Moody and J. van Hillegersberg, *Evaluating the Visual Syntax of UML: An Analysis of the Cognitive Effectiveness of the UML Family of Diagrams,* in *Software Language Engineering*, Lecture Notes in Computer Science, edited by D. Gašević, R. Lämmel, and E. Van Wyk (Springer, Berlin, Heidelberg, 2009) pp. 16–34.

[36] J. H. Larkin and H. A. Simon, *Why a Diagram is (Sometimes) Worth Ten Thousand Words,* Cognitive Science **11**, 65 (1987).

[37] K. Lano, *AgileUML manual,* ().

[38] N. Anquetil, A. Etien, M. H. Houekpetodji, B. Verhaeghe, S. Ducasse, C. Toullec, F. Djareddir, J. Sudich, and M. Derras, *Modular Moose: A new generation software reverse engineering environment,* in *International Conference on Software Reuse* (2020).

[39] G. La Rocca, *Knowledge-based engineering to support aircraft multidisciplinary design and optimization,* Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering **224**, 1041 (2010).

[40] *ParaPy Documentation — ParaPy 1.10.0 Documentation* (2023).

[41] G. La Rocca, *Ae4204 knowledge based engineering,* Course material (2022).

[42] D. Eppstein, *Series–parallel graph,* (2023).

[43] D. Bansal, *Round trip engineering documentation,* (2023), internal DEFAINE project documentation of decisions and progress.

[44] G. van Rossum, B. Warsaw, and A. Coghlan, *PEP 8 – Style Guide for Python Code | peps.python.org,* (2001).

[45] D. Goodger and G. van Rossum, *PEP 257 – Docstring Conventions | peps.python.org,* .

[46] N. Developers, *NetworkX 3.1 documentation,* .

[47] B. Rozière, J. Gehring, F. Gloeckle, S. Sootla, I. Gat, X. E. Tan, Y. Adi, J. Liu, R. Sauvestre, T. Remez, J. Rapin, A. Kozhevnikov, I. Evtimov, J. Bitton, M. Bhatt, C. C. Ferrer, A. Grattafiori, W. Xiong, A. Défossez, J. Copet, F. Azhar, H. Touvron, L. Martin, N. Usunier, T. Scialom, and G. Synnaeve, *Code Llama: Open Foundation Models for Code,* (2024), arXiv:2308.12950 [cs] .

[48] Ingo Staack, Christopher Jouannet, Kristian Amadori, Reinier van Dijk, Max Baan, Robert Braun, and Petter krus, *Knowledge-based Integration of Aircraft System Simulation within Aircraft Conceptual System Design,* Aerospace Europe Conference 2023 , 9 pages (2023).

[49] B. Cornelissen, A. Zaidman, and A. van Deursen, *A controlled experiment for program comprehension through trace visualization,* IEEE Transactions on Software Engineering **37**, 341 (2011).

[50] A. C. Edmondson and I. M. Nembhard, *Product Development and Learning in Project Teams: The Challenges Are the Benefits,* Journal of Product Innovation Management **26**, 123 (2009).

[51] M. O'Rourke, S. Crowley, and C. Gonnerman, *On the nature of cross-disciplinary integration: A philosophical framework,* Studies in History and Philosophy of Science Part C: Studies in History and Philosophy of Biological and Biomedical Sciences **56**, 62 (2016).

[52] S. H. Frost and P. M. Jean, *Bridging the Disciplines: Interdisciplinary Discourse and Faculty Scholarship,* The Journal of Higher Education **74**, 119 (2003), 3648252 .

[53] Q. Yang, A. Steinfeld, C. Rosé, and J. Zimmerman, *Re-examining Whether, Why, and How Human-AI Interaction Is Uniquely Difficult to Design,* in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20 (Association for Computing Machinery, New York, NY, USA, 2020) pp. 1–13.

[54] M. L. Lupetti and D. Murray-Rust, *(Un)making AI Magic: A Design Taxonomy,* in *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, CHI '24 (Association for Computing Machinery, New York, NY, USA, 2024) pp. 1–21.

[55] Y. K. Dwivedi, L. Hughes, E. Ismagilova, G. Aarts, C. Coombs, T. Crick, Y. Duan, R. Dwivedi, J. Edwards, A. Eirug, V. Galanos, P. V. Ilavarasan, M. Janssen, P. Jones, A. K. Kar, H. Kizgin, B. Kronemann, B. Lal, B. Lucini, R. Medaglia, K. Le Meunier-FitzHugh, L. C. Le Meunier-FitzHugh, S. Misra, E. Mogaji, S. K. Sharma, J. B. Singh, V. Raghavan, R. Raman, N. P. Rana, S. Samothrakis, J. Spencer, K. Tamilmani, A. Tubadji, P. Walton, and M. D. Williams, *Artificial Intelligence (AI): Multidisciplinary perspectives on emerging challenges, opportunities, and agenda for research, practice and policy,* International Journal of Information Management **57**, 101994 (2021).

[56] D. Piorkowski, S. Park, A. Wang, D. Wang, M. Muller, and F. Portnoy, *How AI Developers Overcome Communication Challenges in a Multidisciplinary Team: A Case Study,* Proceedings of the ACM on Human-Computer Interaction **5** (2021), 10.1145/3449205.

[57] M. Sadek, M. Constantinides, D. Quercia, and C. Mougenot, *Guidelines for Integrating Value Sensitive Design in Responsible AI Toolkits,* in *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, CHI '24 (Association for Computing Machinery, New York, NY, USA, 2024) pp. 1–20.

[58] T. Miller, I. Durlik, A. Łobodzińska, L. Dorobczyński, and R. Jasionowski, *AI in Context: Harnessing Domain Knowledge for Smarter Machine Learning,* Applied Sciences **14**, 11612 (2024).

[59] D. Council, *The Double Diamond,* .

[60] M. Scheutz, S. A. DeLoach, and J. A. Adams, *A Framework for Developing and Using Shared Mental Models in Human-Agent Teams,* Journal of Cognitive Engineering and Decision Making **11**, 203 (2017).

[61] C. S. research department, *Communication Studies theories: Overview by category | University of Twente,* .

[62] H. Lederman, *A Theory of Common Ground*, Ph.D. thesis, Oxford University (2014).

[63] S. Mohammed, L. Ferzandi, and K. Hamilton, *Metaphor no more: A 15-year review of the team mental model construct,* Journal of Management **36**, 876 (2010).

[64] J. S. Eccles and A. Wigfield, *Motivational beliefs, values, and goals,* Annual Review of Psychology **53**, 109 (2002).

[65] C. R. Berger and R. J. Calabrese, *Some explorations in initial interaction and beyond: Toward a developmental theory of interpersonal communication,* Human Communication Research **1**, 99 (1975).

[66] R. L. Daft and R. H. Lengel, *Organizational information requirements, media richness and structural design,* Management Science **32**, 554 (1986).

[67] B. Alhaji, J. Beecken, R. Ehlers, J. Gertheiss, F. Merz, J. Müller, M. Prilla, A. Rausch, A. Reinhardt, D. Reinhardt, C. Rembe, N.-O. Rohweder, C. Schwindt, S. Westphal, and J. Zimmermann, *Engineering Human–Machine Teams for Trusted Collaboration,* Big Data and Cognitive Computing **4**, 35 (2020).

[68] B. H. Spitzberg, *Preliminary Development of a Model and Measure of Computer-Mediated Communication (Cmc) Competence,* Journal of Computer-Mediated Communication **11**, 629 (2006).

[69] K. Holtzblatt and H. Beyer, *Contextual Design*.

[70] D. Mashek and M. Nanfito, *People, tools, and processes that build collaborative capacity,* (2015), available at: collaborations@claremont.edu, michael@nanfitoandassociates.com.

[71] J. H. Gittell and H. N. Ali, *Relational Analytics: Guidelines for Analysis and Action* (Routledge, New York, NY, 2021).

[72] E. M. Rogers, *Diffusion of Innovations*, 5th ed. (Free Press, New York, NY, 2003).

[73] F. D. Davis, *Perceived usefulness, perceived ease of use, and user acceptance of information technology,* MIS Quarterly **13**, 319 (1989).

[74] S. Kulviwat, G. C. Bruner, and O. Al-Shuridah, *Consumer acceptance of technology model,* Journal of Consumer Marketing **24**, 360 (2007).

[75] N. Forsgren, J. Humble, and G. Kim, *Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations* (IT Revolution Press, Portland, OR, 2018).

[76] Project Management Institute, ed., *The Standard for Project Management and a Guide to the Project Management Body of Knowledge (PMBOK Guide),* seventh edition ed. (Project Management Institute, Inc, Newtown Square, Pennsylvania, 2021).

[77] *RC Survey - Resources,* (2024).

[78] A. T. Himmelman, *Collaboration for a change: Definitions, decision-making models, roles, and collaboration process guide,* Minneapolis: Himmelman Consulting (2002).

[79] Z. Zhang and H. Yin, *Research on design forms based on artificial intelligence collaboration model,* Cogent Engineering **11**, 2364051 (2024).

[80] S. S. Sengar, A. B. Hasan, S. Kumar, and F. Carroll, *Generative artificial intelligence: A systematic review and applications,* Multimedia Tools and Applications (2024), 10.1007/s11042-024-20016-1.

[81] A. Bandi, P. V. S. R. Adapa, and Y. E. V. P. K. Kuchi, *The Power of Generative AI: A Review of Requirements, Models, Input–Output Formats, Evaluation Metrics, and Challenges,* Future Internet **15**, 260 (2023).

[82] J. D. Rittelmeyer and K. Sandkuhl, *Morphological Box for AI Solutions: Evaluation and Refinement with a Taxonomy Development Method,* in *Perspectives in Business Informatics Research*, edited by K. Hinkelmann, F. J. López-Pellicer, and A. Polini (Springer Nature Switzerland, Cham, 2023) pp. 145–157.

[83] J. D. Rittelmeyer and K. Sandkuhl, *A Survey to Evaluate the Completeness and Correctness of a Morphological Box for AI Solutions,* in *Advanced Information Systems Engineering Workshops* (Springer, Cham, 2024) pp. 115–129.

[84] N. Yildirim, C. Oh, D. Sayar, K. Brand, S. Challa, V. Turri, N. Crosby Walton, A. E. Wong, J. Forlizzi, J. McCann, and J. Zimmerman, *Creating Design Resources to Scaffold the Ideation of AI Concepts,* in *Proceedings of the 2023 ACM Designing Interactive Systems Conference* (ACM, Pittsburgh PA USA, 2023) pp. 2326–2346.

[85] J. Shi, R. Jain, H. Doh, R. Suzuki, and K. Ramani, *An HCI-Centric Survey and Taxonomy of Human-Generative-AI Interactions,* (2024), arXiv:2310.07127 [cs] .

[86] G. Strobel, L. Banh, F. Möller, and T. Schoormann, *Exploring Generative Artificial Intelligence: A Taxonomy and Types* (2024).

[87] L. S. Furtado, J. B. Soares, and V. Furtado, *A task-oriented framework for generative AI in design,* Journal of Creativity **34**, 100086 (2024).

[88] R. T. Hughes, L. Zhu, and T. Bednarz, *Generative Adversarial Networks–Enabled Human–Artificial Intelligence Collaborative Applications for Creative and Design Industries: A Systematic Review of Current Approaches and Trends,* Frontiers in Artificial Intelligence **4** (2021), 10.3389/frai.2021.604234.

[89] M. Brereton, A. H. Ambe, D. Lovell, L. Sitbon, T. Capel, A. Soro, Y. Xu, C. Moreira, B. Favre, and A. Bradley, *Designing Interaction with AI for Human Learning: Towards Human-Machine Teaming in Radiology Training,* in *Proceedings of the 35th Australian Computer-Human Interaction Conference*, OzCHI '23 (Association for Computing Machinery, New York, NY, USA, 2024) pp. 639–647.

[90] S. Sahu, *UX Design For AI (Artificial Intelligence),* (2024).

[91] A. Pradas Gomez, M. Panarotto, and O. Isaksson, *Evaluation of Different Large Language Model Agent Frameworks for Design Engineering Tasks,* in *DS 130: Proceedings of NordDesign 2024, Reykjavik, Iceland, 12th - 14th August 2024* (2024) pp. 693–702.

[92] PricewaterhouseCoopers, *The path to generative AI value: Setting the flywheel in motion,* (2024).

[93] J. D. Weisz, M. Muller, J. He, and S. Houde, *Toward General Design Principles for Generative AI Applications,* (2023).

[94] K. Wong and J. Sun, *On evaluating the layout of uml diagrams for program comprehension,* Software Engineering Journal **22**, 412 (2006).

[95] M. Siebenhaller and A. Kaufman, *Drawing activity diagrams,* Software and Systems Modeling **5**, 213 (2006).

[96] D. Moody and J. Hillegersberg, *Evaluating the visual syntax of uml: An analysis of the cognitive effectiveness of the uml family of diagrams,* Software Engineering Journal **33**, 425 (2008).

[97] G. La Rocca, *Knowledge Based Engineering Techniques to Support Aircraft Design and Optimization,* Ph.D. thesis, TU Delft (2011).

[98] M. Stokes, *Managing Engineering Knowledge: MOKA: Methodology for Knowledge Based Engineering Applications* (Professional Engineering Publishing, 2001).

[99] S. Preston, C. Chapman, M. Pinfold, and G. Smith, *Knowledge acquisition for knowledge-based engineering systems,* International Journal of Information Technology and Management **4**, 1 (2005).

[100] INCOSE, *Systems engineering vision 2020 - v2.03,* (2007).

[101] H. Purchase, J. Allder, and D. Carrington, *Layout aesthetics in uml: User preferences,* Journal of Graph Algorithms and Applications **6**, 255 (2002).

[102] M. Agrawala, W. Li, and F. Berthouzoz, *Design principles for visual communication,* ACM Transactions on Graphics **30**, 1 (2011).

[103] T. Munzner, *Visualization Analysis and Design* (CRC Press, Boca Raton, FL, 2014).

[104] M. Pacione, M. Roper, and M. Wood, *A novel software visualization model to support software comprehension,* Software Engineering Journal **30**, 402 (2004).

[105] Jost, Huber, Hericko, and Polancic, *An empirical investigation of intuitive understandability of process diagrams,* Journal of Software Engineering Research and Development **4**, 87 (2016).

[106] A. Hamou-Lhadj and T. Lethbridge, *Summarizing the content of large traces to facilitate the understanding of the behaviour of a software system,* Empirical Software Engineering **11**, 447 (2006).

[107] T. Noda, *Execution trace abstraction based on meta patterns usage,* Software Engineering Journal **27**, 45 (2012).

[108] F. Fittkau, J. Finke, W. Hasselbring, and J. Waller, *Comparing trace visualizations for program comprehension through controlled experiments,* Journal of Software: Evolution and Process **27**, 265 (2015).

[109] T. Xie, E. Kraemer, and R. Stirewalt, *Empirical evaluation of a uml sequence diagram with adornments to support understanding of thread interactions,* Software Engineering Journal **30**, 55 (2007).

[110] T. Xie, E. Kraemer, R. Stirewalt, L. Dillon, and S. Fleming, *Assessing the benefits of synchronization-adorned sequence diagrams: Two controlled experiments,* Software Quality Journal **16**, 187 (2008).

[111] K. Tsiakas and D. Murray-Rust, *Unpacking Human-AI interactions: From interaction primitives to a design space,* (2024), arXiv:2401.05115 .