POLITECNICO DI TORINO

Master Degree Course
Data Science and Engineering

Graduation Thesis

Optimizing Product Development and Innovation Processes with Artificial Intelligence

Supervisors

Prof. Francesca Montagna
Prof. Gaetano Cascini

Candidate

Angelica Marrone

Academic Year 2022-2023
July 2023 Session

Index

Introduction

The rapid progress in artificial intelligence (AI) technologies have transformed different disciplines and industries, allowing new opportunities and capabilities in problem solving, decision making, and automation. Among these industries, product development appears to gain considerably from the incorporation of AI tools, to improve and help design and innovation processes. Product development teams work to create competitive and innovative products, so, understanding the potential applications and obstacles of AI is crucial to ensuring the responsible and effective utilization of these technologies.

The development of a product is a multidisciplinary and demanding undertaking that spans several phases and activities, starting with the basic brainstorming phase through final production and launch on the market. As the global marketplace grows more competitive, companies are under increasing pressure to minimize development times, enhance efficiency, and deliver innovative and top - quality products aligned with customer desires and expectations. AI technologies provide a promising path to enhancing and assisting the process, enabling experts to make considerably more informed decisions, streamline workflows and eventually produce more successful products.

This thesis aims to provide an extensive exploration of the use of AI tools to support design and innovation processes in product development. The discussion will focus on the different aspects of AI, new product development, and the potential benefits and challenges associated with the integration of AI technologies in the product development process.

The thesis will offer a thorough examination of the product development process in the first chapter, describing its phases including planning, concept development, system-level design, detailed design, testing and refinement, and production ramp-up. This chapter will also discuss the role of operation research in product development, covering different types of reasoning employed to approach problems and make decisions. Moreover, the associated cognitive activities will be examined, including identification, structuring, control, development, and communication.

The second chapter will examine the current state of the art in artificial intelligence, focusing on the various branches of AI such as machine learning, neural networks, evolutionary computation, computer vision, robotics, expert systems, speech processing, natural language processing, planning, and large language models.

In the third chapter specific AI technologies will be linked to the possible solutions provided for product development process specific challenges. The logical connections will be established considering the continuously evolving state of the art in AI, acknowledging that the tools discussed are merely examples instead of an extensive list of available options. The chapter will showcase commercial software applications of AI, showcasing its practical use in real-world applications.

The fourth chapter will address the challenges and risks associated with the use of AI, discussing ethical considerations, bias propagation, societal impact, workforce displacement, data security and storage, legal issues, and integration challenges. This chapter will also cover migration and edge computing as additional elements to take into consideration when implementing AI in product development processes.

By analysing the various dimensions of AI technologies and their application, this thesis seeks to provide an insightful and thorough evaluation of the opportunities and challenges related to the integration of AI tools in design and innovation activities. As AI continues to develop and mature, the potential applications and implications of these technologies in the product development industry will only become more pronounced, making it vital for professionals to stay up to date and engaged with the most recent developments and best practices in this rapidly changing field.

1. Product Development Process

The product development process plays a fundamental role in bringing new products to market, ensuring they are designed, produced, and developed to fulfill the requirements as well as preferences of consumers while simultaneously adhering to the production constraints of cost, time, and quality. This complex, multi-faceted process entails a diverse array of disciplines, and that includes engineering, design, marketing, and management, The integration of modern technologies, like artificial intelligence, may also support the different phases of product development. Enhanced awareness of the fundamental principles of product development and the connections between its phases can help professionals involved in the creation of new products improve their approach, leading to more successful outcomes and greater market impact.

The core idea of the product development is the carefully driven evolution of an initial product idea or concept into a concrete, market-ready offering. A series of interconnected stages leads to this transformation, each leading to refinement, optimization, and validation of the product. The product development process can be imagined as a continuous cycle, in which the outcomes of each phase inform and have an impact on the subsequent phases, ultimately culminating in a finalized product that's ready for mass production and market introduction. The innate cyclical nature encourages iterative learning and continuous improvement, allowing the professionals involved to adapt and respond to the ever-changing market conditions, customer feedback, and technological advancements.

The entire process can be seen from a mathematical perspective as a function of some variables, like the product features (F), target market segment (M), cost (C), time to market (T) and quality (Q). The objective is to find the optimal combination of these variables such that market potential and profitability is maximised, while fulfilling the constraints imposed by the organization and the competition.

Product Development Process (F, M, C, T, Q) → Optimal Product

To facilitate the analysis and optimization of the process, various mathematical models and techniques can be employed, such as linear programming, decision analysis, simulation, and optimization algorithms. These tools can assist the developments teams in the action of identifying trade-offs, alternatives, and making informed decisions regarding the design, manufacturing, and marketing of their products.
Another useful tool are the graphical representations that can be used to illustrate relationships and dependencies between the various stages and activities of the development process. Classic examples are flowcharts or process maps that can be utilized to visualize the sequence of events and decision points with the logical purpose of helping to identify potential bottlenecks, redundancies, or inefficiencies that may jeopardise the progress of the project.

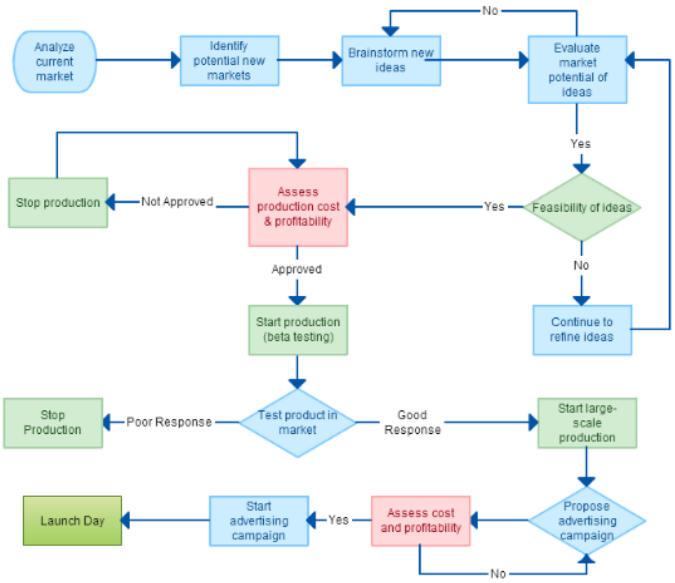

Figure 1: New Product Development schema

The product development process is organized and sequential, but it's also dynamic, collaborative, and iterative. Successful product development needs continuous communication and cooperation between different parties including the design team, the engineers, entrepreneurs, and suppliers, additionally an equally critical role is played by the constant reviewing and integration of feedback from clients and end users. This collaborative approach allows for a continuous improvement of products, ensuring they stay competitive and relevant in the face of rapidly evolving market trends and customer preferences.

While the science of product development advances, emerging technologies like artificial intelligence play an expanding role in boosting the efficiency, effectiveness, and innovation capabilities of the process. In the following chapters, we are going to dive deeper into the specific activities and phases of the product development process, along with the potential and present role of artificial intelligence in supporting mentioned processes with the aim of allowing a more innovative, agile, and competitive approach to the creation and development of new products.

1.1. Process' Phases

The product development process is articulated in several different phases that guide a product from an initial concept to final production. Each phase is characterized by specific goals, tasks, and milestones, contributing to a systematic approach to product design a creation. The core phases include planning, concept development, system-level design, detailed design, testing and refinement, and production ramp-up. By means of a scientific understanding of these phases and a rigorous approach, product development teams can optimize their workflow, manage resources efficiently, and increase the likelihood of successful product launches. A structured approach to product development also favours better communication and collaboration within teams, resulting in a direct improvement in terms of innovation and creativity.

1.1.1. Planning

The planning phase is the first and arguably one of the most critical stages in the product development process. During this phase, organizations identify market opportunities, establish project goals, and define the scope and requirements of the new product. The planning phase acts as the foundation for the following phases of the process, providing a clear direction and a framework for product development.

Before embarking on a new product development project, organizations must first identify market opportunities that have the potential to generate significant profit. This is the phase of market research aimed at gathering valuable information on customer needs, preferences, and pain points, as well as analysing trends, and uncovering potential market gaps.

Organizations can use various tools and techniques to identify market opportunities, few examples are SWOT analysis, PEST analysis, or Porter's Five Forces analysis.

Once a market opportunity has been identified, establishing clear project goals and objectives is necessary. These goals and objectives should be Specific, Measurable, Achievable, Relevant, and Time-bound (SMART) to ensure that they are well-defined and can be effectively tracked and evaluated throughout the project. Defining project goals and objectives helps aligning the efforts of the product development team and provides a shared vision of what the new product aims to achieve in terms of performance, functionality, market positioning, and customer satisfaction.

During the product development project boundaries, constraints, and requirements define the project's scope and guide the project's execution. Within such framework defining the scope involves identifying the key features and functionalities of the new product, as well as determining the resources, budget, and timeline for the project. A well-defined scope helps to manage stakeholder expectations, prevent scope creep, and ensure that the project remains focused and on track.

Requirements gathering is a critical activity during the planning phase as it provides data to determine the aforementioned constraints, its purpose is to ensure that the new product meets the needs of its target customers and aligns with the organization's strategic objectives. This activity is carried out by gathering input from various

stakeholders, such as customers, sales and marketing teams, engineering and design teams, and management, to identify different types of requirements for the new product. Requirements can be captured using various techniques, such as interviews, surveys, focus groups, or observation of user interactions with existing products. Another critical component of the planning phase is the Risk assessment. Assessing the potential risks helps organizations isolating factors that could negatively impact the success of the development project. Risks can arise from various sources, such as technological challenges, market uncertainties, or resource constraints, and can affect the project's cost, timeline, or quality. Companies relay on tools like Failure Modes and Effects Analysis (FMEA), risk matrices, or decision trees to systematically identify and rank based on priority risks and they consequences to statistically reduce the incidence or completely avoid disrupting inconveniences in the product development process.

With the goals, scope, requirements, and risks clearly defined, organizations can then proceed to develop a detailed project plan that outlines the tasks, resources, and timeline for the product development process. The project plan should include a work breakdown structure (WBS) that breaks down the integral workload into manageable tasks, as well as a Gantt chart or a Critical Path Method (CPM) diagram to visualize the project schedule and dependencies among tasks. Finally, an effective plan should define the roles and responsibilities of the team members and put in place efficient communication and organisational tools.

1.1.2. Concept Development

The concept development phase consists in the generation, analysis, and selection of ideas that will be the foundation for the final product. Concept development aims to convert abstract ideas into concrete concepts that can be refined and developed later on. This specific phase encompasses many critical activities, including idea generation, concept scoring, concept screening, and concept selection.

Idea generation is the process of creating a pool of potential concepts for a new product. Various methods, including brainstorming, mind mapping, SWOT analysis and customer feedback, might help accomplish this goal. The objective is to produce a diverse set of ideas that can be analysed and refined to determine the most promising ideas for further development.

Brainstorming is a well-known technique for idea generation, involving group discussions where participants are encouraged to think creatively and propose a wide variety of ideas with no criticism or judgment. Another valuable method is mind mapping, which involves displaying ideas and their relationships in a hierarchical way so that connections can be easily understood. SWOT analysis (Strengths, Weaknesses, Opportunities, and Threats) may be employed to evaluate the external and internal factors that could influence the success of a potential product concept.

Customer feedback can offer useful insights into pain points, preferences, and needs of the target market, aiding in the creation of product concepts that address real world problems and opportunities.

When a pool of ideas has been created, the next step is to screen the concepts to determine those with the most potential for further development. The process of concept screening involves assessing every idea against a set of pre-determined parameters, market potential, including feasibility, technical complexity, and alignment with the company's strategic objectives. Product development teams might dismiss concepts that are not likely to be successful and concentrate their efforts on the most promising possibilities by comparing and ranking the ideas based on these criteria.

Following the initial screening, the remaining concepts undergo a more comprehensive evaluation through concept scoring. The numerical scoring of each concept is determined by a set of weighted criteria that could include market size, competitive advantage, profit potential, technical feasibility, and manufacturing costs. The concepts are ranked based on their overall scores once the scores are added up.

This method enables teams to make better choices regarding which ideas to pursue by quantifying the weaknesses and strengths of each concept.

The selection of the most promising concept(s) for further development is the final stage. The decision is usually made based on the concept scoring results, with extra considerations including resource limitations, strategic priorities, and risk tolerance of the organization. The subsequent phases of product development will refine and develop the winning concept(s), ultimately resulting in a final product fitting the needs and expectations of the target market.

1.1.3. System Level Design

The system level design stage is a crucial element of the product development process, as it focuses on determining the general architecture and structure of the product. The idea generation phase's insights and strategies are expanded upon in this stage, turning into an extensive framework which will steer the subsequent detailed design phase.

Among the important objectives of the system level design stage is defining the product 's functional needs, which are the capabilities and performance attributes that the item needs to have to fulfil its intended purpose. Functional requirements could be recognized through a variety of methods several of them being consumer surveys, market analysis, evaluation of competing products, even sessions with subject matter experts. As soon as the practical needs are identified, they should be converted into technical specs that will outline the performance requirements, tolerances, along with additional measurable characteristics of the product.
Another essential facet of the system level design stage will be the decomposition of the item into its constituent subsystems. This procedure, known as system decomposition, involves breaking down the item into manageable and logically related components to facilitate their creation, management, and production. System decomposition helps with the allocation of functional requirements to precise subsystems or components while facing possible integration issues.
The process of the decomposition helps in the creation of a functional block diagram, a graphical representation of the product's design, illustrating the associations and interactions between the different subsystems, components, and interfaces. The functional block diagram helps to describe the flow of materials, energy, or information through the item, showcasing the contacts and dependencies between different elements. This diagram also can serve as a foundation for producing different system level design artifacts, like method specifications and interface management documents.
Another essential part of system level design is the evaluation and choice of design options. During this stage, multiple design concepts are sifted, each because of its unique range of advantages and drawbacks. To pick the best design choice, factors like specialized feasibility, reliability, cost, manufacturability, and market appeal are thoroughly examined and measured. This decision-making operation relies on different analytical methods, like decision matrices, trade-off analysis, or multi criteria decision analysis, all in the objective of weighing distinct elements and establish the optimal style choice.
Finally, risk assessment is the last major component of the system level design phase. Identifying possible risks and uncertainties connected with the item or its development process helps product development teams to create techniques for mitigating as well as handling those risks. Risk assessment entails using different tools and techniques, for example Failure Modes and Effects Analysis (FMEA), fault tree analysis, or chance registers, to systematically identify, prioritize, and tackle possible risks.

1.1.4. Detailed Design

The detailed design phase entails converting the system-level design into full specifications and documentation which can be utilized to direct the manufacturing, assembly, and tests of the service. This particular stage demands a full comprehension of the product's design constraints, performance requirements, and functionality, in addition to an extensive understanding of the materials, manufacturing processes, and engineering concepts associated with producing the item.
The first task for engineers is developing comprehensive specifications for every component of the item, including its dimensions, tolerances, materials, and floor finishes. These specifications are usually documented utilizing computer

aided design (CAD) program, which enables accurate modelling and representation of advanced geometries and assemblies. The selection of materials for every part plays a tremendous role in the product's general performance, cost, and manufacturability. Engineers should take into account things like availability, cost, weight, durability, stiffness, and strength when selecting resources, in addition to the specific manufacturing processes that will be utilized to produce the components.
The detailed design stage also involves determining the best manufacturing processes for every aspect of the product, considering things such as cost, material properties, production volume, along with geometric complexity. This might entail procedures including casting, forging, machining, injection moulding, or maybe additive manufacturing, among others. To make sure that the single parts of the product may be assembled and also integrated efficiently, engineers should develop comprehensive assembly instructions and information, and also recognize some required fixtures, jigs, or some other tooling necessary for the assembly operation, along with the style of standardized parts to streamline the assembly process and minimize costs.
A crucial element in this particular stage is tolerance analysis, as it requires identifying the acceptable variations in sizes, materials, and manufacturing processes to make sure that the item performs properly and may be produced uniformly. Engineers should think about the cumulative negative effects of tolerances on the product's overall performance, in addition to the implications for manufacturing costs and quality management.
To optimize the design, several strategies are usually applied. One of them is *Design for Manufacturability*, that is an engineering practice which seeks to enhance the style of a product to lessen manufacturing costs, improve quality, and also reduce production lead times. This might involve simplifying component geometries, decreasing the quantity of components, using standardized or common elements, or maybe designing features which facilitate easy assembly and disassembly.
Design for assembly is a complementary process to Design for Manufacturability, focusing especially on the simplicity and effectiveness of assembling the item. By reducing the quantity of fasteners, designing snap-fit connections or incorporating features that simplify alignment and orientation, engineers are able to decrease time and also work necessary to create the item, bringing about reduced manufacturing costs and also increased throughput.
With growing concerns about environmental impacts and resource usage, *Design for Sustainability* has additionally turned into a progressively relevant factor in the detailed design stage. Engineers should think about things including end-of-life disposal, recyclability, material usage, and energy efficiency when developing products, with the aim of reducing the product 's overall environmental impact.
Throughout the entire phase, engineers must constantly document and verify their work to make sure that the design specifications are consistent, complete, and accurate with the product's requirements and constraints. This might require the usage of design reviews, simulations, or even physical prototypes to verify the design and spot some possible issues or areas for improvement.

1.1.5. Testing and Refinement

The testing and refinement phase, focuses on analysing and optimizing the product to ensure it meets the preferred customer requirements, quality standards, and performance criteria. After the detailed design is done and a prototype or initial version of the product has been made, this phase usually begins. Testing and refinement are intended primarily to discover and correct any defects, shortcomings, or inconsistencies in the product before its production ramp-up stage.

A number of sub-steps are generally identified within testing and refinement, like verification, validation, and iterative improvements. Each one of these sub-steps plays a major role in the general success of the product and contributes to the optimization of the final design.

The process of verification involves confirming that the product was designed and developed in accordance with the specified requirements, design documents, and engineering standards. The systematic assessment of the product's components, features, and functions is part of this process to make sure they are in line with the design requirements. Verification may take numerous forms including inspection, analysis, simulation and testing. Engineers might perform stress analysis on a mechanical part to ensure that it is able to withstand the expected loads during

operation, or they may conduct functional tests to ensure that the software in the product functions remains as intended.

Validation examines if the product meets the demands and expectations of the intended users and stakeholders. This process entails evaluating the product's overall suitability, usability, and performance for its intended goal, marketplace, and operating environment. Validation usually consists of activities like user testing, field trials and market research. A product might go through usability tests to determine user satisfaction, intuitiveness, or its usability, or it might undergo environmental testing to measure its performance in several environments, like extreme temperatures or humidity levels.

After the verification and validation steps are completed, the product development team must address any problems, deficiencies, or opportunities for improvement. Iterative modifications to the product's design, features or functions are made during this stage to enhance its performance, reliability, or user satisfaction. The iterative improvement process might entail updating the design documents of the product, updating its software, or altering its hardware components.

The testing and refinement process usually involves several iterations because the product is continually evaluated as well as enhanced until it meets the established success criteria.

1.1.6. Production Ramp-Up

The production ramp-up phase is the final stage of the entire process, wherein the product is transitioned from a prototype or small-scale production to full scale manufacturing. The aim of this phase is to make sure that the manufacturing processes are capable, economical, and effective in producing high quality products in the needed quantities to satisfy market demand. The production ramp - up consists of different activities including process validation, supply chain management, quality control, and workforce training, all of which help ensure a successful introduction of the product into the market.

Process validation during this final stage ensures that manufacturing processes regularly produce products that meet determined quality standards and specifications. This involves analysing and implementing improvements to production equipment, assembly lines and manufacturing processes to reduce defects, waste and increase efficiency.

The success of the phase relies upon a strong supply chain able to provide the raw materials, components and resources required for the increased manufacturing volumes. The selection of suppliers, management of inventory, forecasting demand and logistics planning are necessary to ensure a smooth flow of materials and resources through the production cycle.

Quality control then guarantees that the products manufactured meet the desired quality standards and customer expectations. This comprises implementing quality assurance measures, such as statistical process control (SPC), inspection procedures, and testing protocols, to monitor and control product quality through the entire manufacturing process. The stability of the manufacturing process may be monitored using statistical process control charts like X-bar and R charts to identify deviations from ideal quality standards. These charts display the process mean (X-bar) and range (R) over time, allowing manufacturers to recognize trends, patterns, and potential issues that may affect product quality.

As production volumes grow during the ramp - up stage, it's indispensable to verify that the workforce is properly trained and skilled to deal with the increased workload and responsibilities. This might require providing training on new equipment, production techniques, quality control procedures, and safety protocols, ensuring that the workforce is equipped with the knowledge and skills required for effective and efficient production.

1.2. Operation Research Problem and Reasoning

Operation Research is an interdisciplinary field that employs mathematical and analytical methods to model, analyse, and optimize complex systems and decision-making processes. It plays a crucial role in product development, as it helps to identify and solve critical problems that may arise during the various phases of the process. By applying its techniques, product development teams can make more informed decisions, allocate resources efficiently, and improve the overall effectiveness of their projects.

In this context, problems are challenges or issues that arise during product development that can be addressed through the application of Operation Research methods. To solve operation research problems, various reasoning types can be employed to evaluate potential solutions, make decisions, and draw conclusions. Reasoning is a cognitive process that allows individuals or teams to logically analyse information, evaluate alternatives, and make informed choices. Reasoning types can be broadly classified into deductive, inductive, abductive, common sense, monotonic, non-monotonic, and probabilistic reasoning, each with its own unique characteristics and applications.

Operation research problems often involve multiple objectives, constraints, and uncertainties, making them particularly well-suited for analysis using specific techniques that include:

- Linear Programming, which is an optimization technique used to solve problems that involve linear objective functions and linear constraints and aims to find the optimal solution that maximizes or minimizes the objective function while satisfying all constraints. Deductive reasoning is often employed in Linear Programming, as it relies on established relationships between variables and constraints to derive an optimal solution.
- Integer Programming, which is a mathematical optimization technique that deals with problems where some or all the decision variables must take integer values. They can be seen as extensions of Linear Programming problems, incorporating additional constraints to ensure integer solutions. Similarly, deductive reasoning is often used to draw conclusions based on known premises or facts.
- Nonlinear Programming, which is an optimization method used for problems where the objective function or constraints are nonlinear. These techniques attempt to find the optimal solution by employing iterative algorithms that converge to the best possible outcome. Inductive reasoning is commonly used as it involves forming generalizations from specific observations or data points.
- Network Analysis is a technique for modelling and analysing complex systems that can be represented as networks or graphs. It is often used to study the relationships between entities, optimize resource allocation, or minimize costs. Common sense reasoning may be applied in network analysis, as it relies on intuitive understanding and everyday knowledge to make decisions.
- Simulation is a technique that involves creating digital or physical models of real-world systems to test and analyse their behaviour under various conditions. Simulation allows product development teams to evaluate potential solutions, optimize processes, and make informed decisions. Probabilistic reasoning is frequently employed in simulation, as it deals with analysing uncertain or incomplete information.
- Queuing Theory is the study of waiting lines or queues, focusing on the mathematical modelling and analysis of systems that involve service and waiting times. It is widely used in operations research to optimize the performance of queuing systems. Non-monotonic reasoning is often applied, as new information may change previously derived conclusions about the system's performance.
- Decision Analysis, which is a structured approach to making complex decisions, incorporating decision trees, influence diagrams, and utility theory to evaluate alternatives, assess risks, and make choices. Abductive reasoning is an example of commonly used reasoning type for this approach, as it involves inferring the most likely explanation or course of action based on available information.
- Multi-criteria Decision Making is a collection of techniques for solving decision-making problems that involve multiple conflicting criteria or objectives. These techniques, such as the Analytic Hierarchy Process or the Technique for Order of Preference by Similarity to Ideal Solution, allow decision-makers to evaluate and prioritize alternatives based on multiple factors. Monotonic reasoning is often associated with these techniques, as adding new information typically does not change previously derived conclusions about the relative importance of criteria.

Examples of operation research problems in product development include optimizing production schedules, managing supply chains, allocating resources, or selecting the most appropriate design alternative, among others. In the subsequent sections, we will delve into each of these reasoning types.

1.2.1. Deductive Reasoning

Deductive reasoning, also known as top-down reasoning or deduction, is a logical process that begins with a general premise or set of premises and derives specific conclusions based on those premises: if the premises are true and the reasoning process is valid, the conclusions drawn are guaranteed to be true as well. This type of reasoning is particularly useful in situations where the available information is structured and well-defined, allowing for a systematic approach to problem-solving and decision-making.
A classic example of deductive reasoning is the syllogism, a logical argument that consists of three parts: a major premise, a minor premise, and a conclusion. For instance:

Major premise: All humans are mortal. Minor premise: Socrates is a human. Conclusion: Therefore, Socrates is mortal.

In this example, the major and minor premises are both assumed to be true, and the conclusion is derived from the premises through a valid deductive reasoning process.

One of its primary strengths is its high level of certainty in the conclusions derived, as long as the premises are true, and the reasoning process is valid. However, it is limited by its reliance on the accuracy and completeness of the initial premises, which may not always be available or reliable. Moreover, it is less effective in dealing with semi-structured or unstructured input contexts, where the available information may be incomplete, uncertain, or ambiguous.

1.2.2. Inductive Reasoning

Inductive reasoning is a method of logical reasoning that involves drawing general conclusions based on a set of specific observations or instances. This approach is particularly useful when dealing with incomplete or uncertain information, as it allows decision-makers to infer patterns, trends, or relationships that may not be explicitly evident in the available data. Therefore, it can be applied to a wide range of situations, including the analysis of structured, semi-structured, and unstructured input contexts. Both in unstructured and semi-structured contexts, where there often is a mixture of structured and unstructured elements, inductive reasoning can be applied by identifying relevant information and drawing general conclusions from the available patterns, extracting meaningful information, and making generalizations based on the observed patterns.

Inductive reasoning operates on the principle of "bottom-up" logic, as it starts with specific instances and moves towards broader generalizations. While this method can be effective in generating insights and hypotheses, it is important to recognize that inductive reasoning may not always yield definitive or universally applicable conclusions. The reliability of inductive reasoning depends on the quality and representativeness of the available data, as well as the validity of the inferred patterns or relationships.

1.2.3. Abductive Reasoning

Abductive reasoning, also called inference to the best explanation, is a type of logical reasoning which aims to generate the most plausible explanation or hypothesis for a certain set of observations or data. Unlike deductive reasoning, which derives conclusions from known premises, and inductive reasoning, which generalizes from specific instances, abductive reasoning starts with an observation and tries to identify the underlying cause or explanation for that observation. In essence, it entails a process of hypothesis generation and evaluation, aiming to find the best satisfactory explanation given the available information.

This kind of reasoning is especially helpful when there's incomplete, uncertain, or ambiguous information - which is often the case in real-world problem-solving situations. The input context for abductive reasoning may vary, ranging from structured to unstructured, for which it is most conveniently utilized.

One of the main features is its focus on generating explanatory hypotheses that account for the observed data while remaining in line with background knowledge or assumptions. The process may be viewed as a search through a set of possible explanations, guided by criteria like simplicity, coherence, explanatory power, and alignment with current knowledge.

Abductive reasoning might combine heuristic search methods with pattern recognition and probabilistic reasoning techniques. Heuristic methods could produce a set of candidate hypotheses based on known patterns or relationships, and probabilistic reasoning would assess the likelihood of each hypothesis based on available data and background knowledge.

One of the main challenges in abductive reasoning is the potential for generating multiple plausible explanations for a given observation, which can lead to ambiguity and uncertainty in decision-making. Additionally, it is susceptible to errors and biases due to its subjective evaluation of the plausibility of different explanations.

1.2.4. Common Sense Reasoning

Human cognition relies on common sense reasoning to help people to make decisions and draw conclusions based on general knowledge, experience, and an understanding of the world. It is different from other kinds of reasoning in it relies on a broad range of background knowledge and it is frequently utilized to fill in information gaps or draw inferences when data is incomplete, contradictory, or ambiguous.

Common sense reasoning must enable individuals to extract relevant information, identify patterns, and connect seemingly unrelated pieces of data in order to effectively be applied to different contexts. Heuristics are mental shortcuts or guidelines that can simplify tough problems and aid in decision making during this process. If a photograph shows a group of people standing next to a cake with lit candles, a viewer may assume that the scene is a birthday party although no explicit information is given about the occasion.

1.2.5. Monotonic Reasoning

Monotonic reasoning is a fundamental approach in logic-based systems, where the addition of new information or premises does not lead to the retraction of previously inferred conclusions. In this type of reasoning, as more information becomes available, the set of inferred conclusions either remains the same or grows, but never shrinks. It is particularly useful when working with well-defined, structured contexts and deterministic relationships.

Monotonic logic systems, such as first-order predicate logic, provide a solid foundation for formalizing and analysing relationships between concepts, properties, and entities. One of the key aspects of monotonic reasoning is the use of logical inference rules, such as modus ponens, which can be applied systematically to derive conclusions from given premises. For example, consider the following premises:

All humans are mortal.

Socrates is a human.

Using monotonic reasoning, one can infer that "Socrates is mortal," and adding further premises will not change this conclusion.

However, monotonic reasoning may not be well-suited to situations involving incomplete, uncertain, or evolving information, as it lacks the flexibility to retract or revise conclusions when new information becomes available. In these cases, non-monotonic reasoning (discussed in 1.2.6) may provide a more suitable approach.

1.2.6. Non-monotonic Reasoning

A type of logic known as non - monotonic reasoning allows the modification or retraction of conclusions based on new information. This type of reasoning is especially helpful whenever the information is incomplete, uncertain, or subject to change. In contrast to monotonic reasoning, where conclusions once derived remain valid even if new information is added, non-monotonic reasoning allows much more flexible and adaptable decision - making tasks, better suited for unstructured real-world scenarios.

The use of default assumptions or rules is an important element of non-monotonic reasoning. Default rules allow conclusions to be drawn based on typical or anticipated conditions, while still accommodating exceptions when new information becomes available. For instance, in the lack of information to the contrary, one might assume that birds can fly (default assumption). Nevertheless, if it's later learned that a specific bird is a penguin, this new information would override the default assumption, leading to the revised conclusion that the particular bird can't fly.

This method permits conclusions to be drawn using incomplete or uncertain information but allows for revision of these conclusions as new information becomes available.

1.2.7. Probabilistic Reasoning

Probabilistic reasoning is an essential aspect of operation research, allowing decision-makers to handle uncertainty and make informed choices in complex, dynamic environments. By incorporating probability theory and statistical methods, probabilistic reasoning enables the estimation of uncertain outcomes and the assessment of risks associated with different alternatives. The decision-maker relies on probability distributions to represent the uncertainty associated with various input factors, such as demand, costs, or technological advancements. These inputs can be structured, semi-structured, or unstructured. depending on the context and the available information

A fundamental concept in probabilistic reasoning is the Bayesian framework, which allows for updating the probability estimates as new information becomes available. The Bayesian theorem is expressed mathematically as:

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(b)}$$

Where P(A|B) represents the conditional probability of event A occurring given that event B has occurred, P(B|A) is the conditional probability of event B occurring given that event A has occurred, and P(A) and P(B) are the marginal probabilities of events A and B, respectively.
This theorem can be applied in operation research to update decision-makers' assumptions about uncertain variables like the probability of a new product being successful or the likelihood of a competitor responding to a strategic move. For instance, imagine a company that wants to estimate the likelihood of success for a brand-new product launch based on historical data and market research. The company can make use of probabilistic reasoning to evaluate the likelihood of different sales levels, incorporating both structured data (such as previous sales figures) and unstructured data (such as client reviews or social media sentiment). The company can make better decisions concerning product development, marketing strategies and resource allocation by regularly updating these probability estimates as new data becomes available.

1.3. Cognitive Activities

Cognitive activities drive problem-solving, decision-making, and innovation. The activities encompass a wide range of mental processes that aid in the effective performance of tasks and the attainment of goals during product development. By analysing cognitive activities, we can better understand and simplify the complexity of the product development process, leading to more effective outcomes. In this context, cognitive activities can be classified in five major areas: identification, structuring, control, development, and communication. Each of these activities plays a role in the overall success of a project. Nevertheless, it's vital to remember that these groups aren't intended to be strict, they can often overlap or even be incorporated in even more phases, nor exhaustive but rather provide a useful framework for understanding the diverse cognitive processes involved in product development.

1.3.1. Identification

Identification involves recognizing and understanding the various elements, constraints, and requirements that are relevant to a given project. In the context of product development, identification can be viewed as the foundation upon which other cognitive activities are built, since it provides the necessary information for subsequent decision-making, problem-solving, and creativity.

Effective identification requires critical thinking and analytical skills, as professional figures must be able to discern relevant information from noise and prioritize key elements that will have the most significant impact on the project. The Pareto Principle, also known as the 80/20 rule, is often used, which posits that approximately 80% of the effects come from 20% of the causes.

In the context of the product development process phases, the identification activity is particularly prominent during the planning and concept development phases. During the planning phase, professionals must identify the project's scope, objectives, and constraints, while the concept development phase involves identifying potential product features, designs, and solutions that meet the identified requirements and constraints. Although identification is most prominent in these early stages, it remains an essential cognitive activity throughout the entire product development process, as new information and insights may emerge at any point, requiring adjustments and adaptations in response.

1.3.2. Structuring

Structuring enables the organization of information, ideas, and resources in a coherent and logical manner. This cognitive process involves the categorization, prioritization, and arrangement of elements, such as design requirements, constraints, or customer needs. It helps product development teams identify relationships, dependencies, and patterns among different aspects of the project, leading to a more comprehensive understanding of the design space.

Professionals employ various techniques to organize and classify the information, which can include creating hierarchical structures, using taxonomies or ontologies, developing matrices or graphs, and applying clustering or segmentation algorithms. An example of structuring activity in product development process is the use of affinity diagrams, which involves grouping related ideas or requirements into categories. This technique helps identify common themes and areas of focus, enabling to prioritize and address the most critical customer needs. Another example is the creation of a design structure matrix (DSM), a square matrix that represents the dependencies and interactions between components, tasks, or subsystems in a product. The DSM allows for the visualization and analysis of the complex relationships within a system, providing insights into potential issues, such as bottlenecks, feedback loops, or opportunities for modularization.

Throughout the product development process, the structuring activity is particularly relevant during the planning, concept development, and system-level design phases. In the planning phase, structuring helps in defining project objectives, scope, and resources, while in the concept development and system-level design phases, it supports the organization and synthesis of design alternatives and requirements.

1.3.3. Control

Control cognitive activity involves the management and regulation of tasks, resources, and decision-making to ensure that the project progresses effectively and efficiently. Control activities can be divided into subcategories, monitoring, and evaluation.

Monitoring involves tracking the progress of a project and comparing it to the established plan to identify any

deviations or discrepancies. Regular monitoring enables teams to detect potential issues early and make necessary adjustments to ensure that the project stays on track. Monitoring activities may include progress reports, status meetings, and performance evaluations.

Evaluation is the assessment of project performance, focusing on the effectiveness and efficiency of the process and the quality of the final product. Evaluating a project can help identify areas for improvement, guide future projects, and inform strategic decision-making. Some methods may include benchmarking against industry standards, customer feedback, and post-project reviews.

Control activities play a significant role in each phase of the product development process, though the extent and focus of control may vary depending on the specific phase. For example, during the planning and concept development phases, control activities may focus on evaluating potential concepts against established criteria. In the detailed design and testing phases, control activities may involve monitoring progress to ensure that designs meet technical specifications and performance requirements. Finally, during the production ramp-up phase, control activities may center on managing resource allocation and production schedules to ensure a successful product launch.

1.3.4. Development

Development entails creating and refining ideas, concepts, and designs, as well as evaluating and selecting the most promising solutions. This activity is crucial for driving innovation and ensuring that products meet the requirements and expectations of customers and stakeholders. Development involves a variety of mental skills and processes, such as creativity, problem-solving, critical thinking, and decision-making, which collectively play a role in the generation of competitive and innovative products.

Product teams are constantly revising and improving their concepts and designs based on new information, testing, and feedback, making development activities iterative. The iterative process ensures that the final product is both cutting edge and relevant to the target audience, while simultaneously being flexible to changing conditions and requirements. Several creativity techniques, including analogical thinking, lateral thinking, and morphological analysis, can help to stimulate the generation of novel and unconventional ideas, even further enhancing the development process.

The concept development and detailed design phases usually feature the most development activities. Concept development teams concentrate on producing a wide range of possible solutions and identifying the most promising concepts to carry forward. The detailed design phase is where the selected ideas are refined and elaborated upon, and teams work to finalize design specifications, manufacturing processes, materials, and other crucial aspects of the product.

1.3.5. Communication

Communication activity enables the sharing of information, ideas and feedback between team members, stakeholders, and customers. Effective communication is crucial in ensuring that all parties involved in the development process understand the goals, requirements, constraints, and progress of the project. Additionally, it plays an important part in encouraging collaboration, boosting creativity, and also promoting a shared understanding of the project's goals as well as expectations.

In the context of product development, communication can take different forms, which includes verbal exchanges, visual representations, written documentation, and digital interactions. Verbal communication, including face - to - face meetings, conference calls, or presentations, allows team members to discuss ideas, share insights, and offer feedback in real-time. Written communication, like reports, specifications, or meeting minutes, helps distribute and record essential information, ensuring that all stakeholders have access to the needed data and knowledge. Visual communication, like sketches, diagrams, or computer-aided design (CAD) models, allows product developers to

convey complex concepts and ideas more effectively, particularly when dealing with spatial, geometric, or structural aspects of the product. The utilization of digital communication tools such as email, instant messaging or collaboration platforms enables seamless information exchange and real time updates, therefore improving communication efficiency and reducing the chances of misunderstandings or misinterpretations. Strong communication skills - active listening, empathy, clarity, and succinctness - are crucial for team members. Clear communication channels, protocols, and guidelines can also facilitate the flow of information and reduce potential communication breakdowns or bottlenecks.

In the different phases of the product development process, communication plays distinct roles. It is essential during planning to gather customer requirements, define project objectives and establish expectations among stakeholders. During the concept development stage, communication encourages brainstorming, idea generation, and the evaluation of feasible solutions. Communication is vital throughout system - level design and detailed design phases for coordinating cross functional teams, discussing design alternatives, and also dealing with technical problems or constraints. Communication is essential throughout testing and refinement as well to discuss test results, identify areas for improvement and implement changes based on feedback. Effective communication during the production ramp - up stage helps make sure that all parties involved are informed about production schedules, quality standards and progress updates, leading to a successful product launch.

2. Artificial Intelligence – State of the Art

The emerging field of artificial intelligence (AI) includes various branches and subfields which aim to develop intelligent systems to process information, learn, reason, and interact with their surroundings. By boosting human capability and automating activities that were previously considered to be solely human intelligence, AI has got the ability to revolutionize different industries, including product development. This chapter examines the essential principles and concepts underlying the current state-of-the-art in Artificial Intelligence, providing a holistic understanding of this area and its potential applications. At its core, AI is built upon the premise of mimicking and simulating human intelligence to produce machines able to complete tasks independently or in collaboration with humans.

Among the foundational aspects of AI is the representation of knowledge, which implicates encoding information about the world in a way that computers can understand and process. Symbolic representations like graphs, rules and logic are generally used, along with probabilistic and statistical representations like probability distributions, Bayesian networks and Markov models.

Another critical factor is the capacity to reason and make decisions based on that represented knowledge, which requires the development of algorithms and techniques that permit machines to deduce new information, draw conclusions, and make choices that optimize specific objectives or goals. Such acumen might implicate the appliance of deductive, inductive, or abductive logic, along with other forms of reasoning, like analogical, case-based, or qualitative reasoning.

Learning is then an essential element to provide machines the ability to acquire new knowledge, adjust to changing circumstances, and enhance their effectiveness in the long run, similarly to human intellect. Different learning paradigms have been created during the final years, including supervised learning, in which machines learn from labeled examples, unsupervised learning, where they discover patterns and structure in unlabeled data, and reinforcement learning, where they learn by interacting with their environment and receiving feedback in the form of rewards or penalties. Optimization algorithms, including gradient descent or genetic algorithms, are frequently utilized in these learning paradigms to adjust the parameters of the learning models and also reduce the variance between the model's predictions and the desired outputs.

AI is strongly influenced by cognitive neuroscience and science insights, leading to models and algorithms based on the structure and function of the human brain. One of the more notable examples of this is artificial neural networks,

that are composed of interconnected nodes or neurons that process and transmit information in a manner analogous to biological neural networks. Neural networks have given rise to deep learning, an excellent subset of machine learning which makes use of multi - layered neural networks to learn hierarchical representations of data, enabling machines to immediately extract features and patterns from complex, high-dimensional data, like images, speech, and text. As artificial intelligence progresses, new approaches and techniques are being developed to handle a wide range of tasks and applications, including computer vision, natural language processing, robotics, and planning. This is due to the convergence of several factors, such as the availability of large datasets, improvements in computational power, and the development of new architectures and algorithms that can approach and learn from huge quantities of unstructured data.

In the context of product development, AI has the ability to support design and innovation processes by augmenting human capabilities, automating tasks, and providing valuable insights and predictions that can inform decision-making. Professionals can assess the potential of AI technologies to support their job and make educated decisions about how to integrate AI tools and techniques into their tasks by analysing the current state of the art in AI and its underlying principles.

2.1. Machine Learning

Machine learning is a subfield of artificial intelligence that focuses on the development of algorithms and models to enable computers to learn from and make predictions or decisions based on data. Fundamentally integral to machine learning is the pursuit of creating systems that can automatically enhance their performance over time by improving their underlying models through exposure to brand new data and experiences. Learning from data allows machine learning algorithms to discover patterns and relationships that can be used to make more precise predictions or solve more challenging problems.

There are several different approaches to machine learning, each one with their very own strengths and limitations: widely used techniques include supervised learning, unsupervised learning, reinforcement learning and deep learning, which will be explored in detail later on in this chapter. These methods use various mathematical and computational tools, which includes probability theory, optimization, and statistical analysis, to build models that can be extended from known examples to unknown data.

A six-phases process is the foundation of every machine learning technique:

- *Data collection and pre-processing:* First, gather a dataset containing input-output pairs, also known as labels and features. Pre-processing of the data may involve cleaning, normalization, and feature engineering to make it compatible with the algorithm chosen.
- *Model selection*: Next, an appropriate algorithm is picked based on the problem's nature, dataset size, and desired performance characteristics.
- *Model training*: The algorithm adjusts its parameters to minimize a predefined loss function depending on the training data to quantify the difference between the predicted outputs and the actual labels.
- *Model evaluation*: The performance of a model is assessed on a distinct dataset, commonly known as the validation or test set, to determine its ability to generalize to unseen data.
- *Model tuning and selection*: If necessary, hyperparameters, which are the settings that regulate the learning process, are modified, and different algorithms may be compared to select the best model.
- *Model deployment*: After a sufficient amount of training and evaluation, the model can be utilized to predict new, unlabeled data.

Hereafter, some fundamental concepts to take into consideration in any machine learning algorithm will be briefly explained.

Overfitting occurs when a model captures noise in the training data resulting in poor generalization on unseen data, while underfitting occurs when the model is too simplistic to capture the underlying patterns in the data. Techniques like regularization, early stopping and model selection can be used to address these problems.

Furthermore, bias and variance have to be considered too: bias refers to the error introduced by approximating a complex problem with a simplified model, while variance describes the error due to the model's sensitivity to small fluctuations in the training data. A model ought to ideally have low bias and low variance, but in practice there's usually a compromise between the two. Overfitting usually leads to low bias and high variance, whereas underfitting results in very low variance and high bias.

Cross-validation is a method for estimating the performance of a model on unseen data by splitting the dataset into k equally sized folds. The model is trained on k-1 folds and evaluated on the remaining fold. This process is repeated k times, with each fold serving as the test set once. The average performance across all k iterations is considered as the model's performance estimate.

Lastly, high-dimensional data can lead to computational challenges and increased risk of overfitting. The number of features can be reduced by utilizing feature selection and dimensionality reduction methods like Principal Component Analysis (PCA) and Recursive Feature Elimination (RFE) to preserve the most informative ones.

Machine learning has been applied to a wide range of tasks and domains, including image and speech recognition, natural language processing, game playing, medical diagnosis, financial modelling and much more. Researchers and practitioners have been able to create systems that can automate complex tasks, make data driven decisions, and respond to changing environments or requirements by leveraging its power.

Nevertheless, it is crucial to recognize that machine learning is not a one-size-fits-all solution, and the success of a particular approach depends on various factors, like the quality and quantity of the accessible data, the complexity of the problem, and the suitability of the selected algorithm.

2.1.1. Supervised Learning

Supervised learning is a subfield of machine learning where an algorithm learns from a set of labeled data to make predictions on new, unseen data. The goal is to construct a model that can accurately predict the output (or target) variable given a set of input features.
Supervised learning can be further divided into two main categories: classification and regression. Classification deals with predicting a discrete output, such as categorizing an email as spam or not spam, while regression deals with predicting a continuous output, such as estimating the price of a house based on its features. Several supervised learning algorithms have been developed, each with its strengths and weaknesses.
Linear regression is a simple, yet powerful supervised learning algorithm used for predicting a continuous target variable based on one or more input features. The goal of linear regression is to find the best-fitting straight line (in case of a single input feature) or hyperplane (in case of multiple input features) that minimizes the sum of squared differences between the actual target values and the predicted target values. The relationship between the input features and the target variable can be represented by the following equation:

$$y = b_0 + \ b_1 x_1 \ + \ b_2 x_2 + \cdots + \ b_n x_n + e$$

Where y is the target variable, $x_1, x_2, \ldots \ , x_n$ are the input features, $b_0, b_1, \ldots \ , b_n$ are the coefficients (weights) to be learned and e is the residual (error) term. To estimate the coefficients, linear regression can use methods such as ordinary least squares (OLS) or gradient descent.

Logistic regression is a variation of linear regression that is used for binary classification problems. In logistic regression, the goal is to find the best-fitting function (Sigmoid function) that models the probability of an instance belonging to the positive class given its input features. The Sigmoid function is given by:

$$P(y = 1 \mid x) \ = \frac{1}{1 + \ e^{-z}}$$

Where $P(y = 1 \mid x)$ is the probability of the instance belonging to the positive class and $z = b_0 + \ b_1 x_1 \ + \ b_2 x_2 + \cdots + \ b_n x_n$ similarly to the linear regression equation. The logistic regression model can be trained using methods such as maximum likelihood estimation or gradient descent.

Support Vector Machines (SVM) is a powerful supervised learning algorithm used for both classification and regression tasks. In the context of binary classification, the goal of SVM is to find the optimal hyperplane that separates the instances of two classes with the maximum margin. The margin is defined as the distance between the hyperplane and the nearest instances from each class, called support vectors. The hyperplane can be represented by the following equation:

$$wx + b = 0$$

Where w is the weight vector, x is the input feature vector and b is the bias term. SVM can be extended to non-linear classification problems by using kernel functions, such as the Radial Basis Function (RBF) kernel, which allows the algorithm to learn complex decision boundaries in higher-dimensional spaces.

Decision trees can be used for both classification and regression tasks. A decision tree recursively splits the input feature space into regions based on the feature values, resulting in a tree-like structure where each internal node represents a decision rule, and each leaf node represents an output value (class label or continuous value). The goal of the decision tree algorithm is to find the optimal decision rules that minimize a cost function, such as Gini impurity or entropy for classification tasks, and mean squared error for regression tasks.

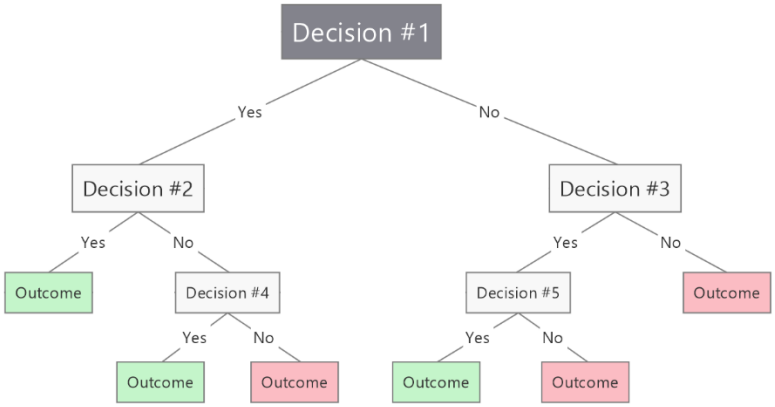

Figure 2: Decision Tree diagram

Random forests are an ensemble learning method that combines the predictions of multiple decision trees to improve the overall performance and reduce overfitting. The random forest algorithm constructs a set of decision trees by randomly selecting a subset of input features and instances (with replacement) for each tree. The final prediction of the random forest model is obtained by averaging the predictions of all decision trees for regression tasks or by taking a majority vote for classification tasks.

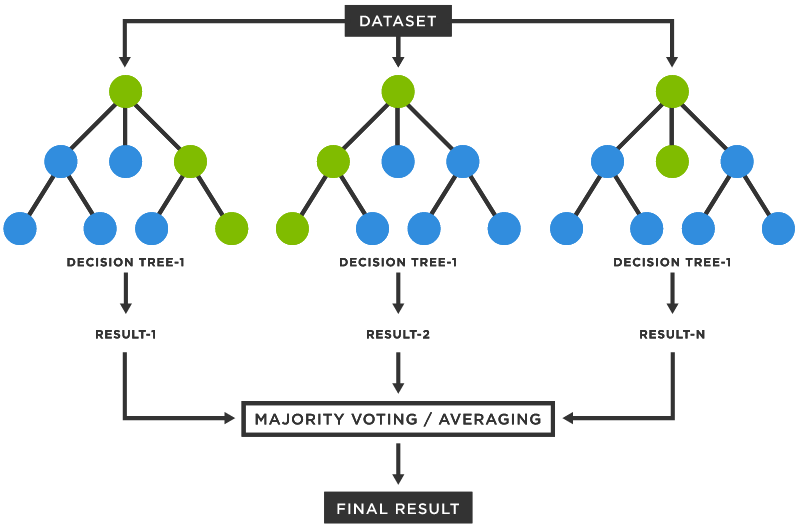

Figure 3: Random Forests diagram

k-Nearest Neighbours (k-NN) can also be used for both classification and regression tasks. Given a new data point, the k-NN algorithm finds the k training instances that are closest to the new point, according to a distance metric such as Euclidean distance. The prediction for the new data point is then obtained by taking the majority vote (in the case of classification) or averaging the target values (in the case of regression) of the k nearest neighbours. The k-NN algorithm can be summarized in the following steps:

- Calculate the distance between the new data point and each instance in the training dataset using a distance metric (e.g., Euclidean distance).
- Sort the distances in ascending order and select the k instances with the smallest distances.
- For classification tasks, determine the majority class among the k nearest neighbours. For regression tasks, calculate the average target value of the k nearest neighbours.
- Assign the majority class (for classification) or the average target value (for regression) as the prediction for the new data point.

One of the key advantages of the k-NN algorithm is its simplicity and ease of implementation. However, it can be computationally expensive, especially for large datasets, as it requires calculating distances between the new data point and all instances in the training dataset. Additionally, the choice of the hyperparameter k and the distance metric can significantly affect the performance of the algorithm.

By leveraging the power of supervised learning algorithms, companies can identify patterns, trends, and relationships that may not be apparent through traditional methods, ultimately resulting in more innovative and successful products.

2.1.2. Unsupervised Learning

Unsupervised learning is a branch of machine learning which focuses on discovering patterns and structures within data without the usage of explicit labels or pre-defined outcomes. These algorithms work with raw, unprocessed data instead of supervised learning which depends on labeled data to train models, aiming to discover hidden relationships and groupings which are not immediately apparent.

One of the primary objectives of unsupervised learning is to reduce the dimensionality of the data, which makes it simpler to analyse and visualize. High-dimensional data could be transformed into lower-dimensional representations using dimension reduction techniques including principal component analysis (PCA) and t-distributed stochastic neighbour embedding (t-SNE), while still keeping the essential structure and relationships within the data. PCA is a linear dimensionality reduction method that determines the directions or axes in the data with the largest variance using Principal Component Analysis (PCA). It works by calculating the eigenvectors and eigenvalues of the data covariance matrix. The eigenvectors correspond to the principal components, while the eigenvalues indicate the magnitude of variance captured by each component. The PCA transformation could be mathematically stated as follows:

$$X_{pca} = X_{centered} W$$

where $X_{centered}$ is the mean-centered data, and W is the matrix of eigenvectors sorted by their corresponding eigenvalues.

Another commonly used unsupervised learning technique is clustering, which seeks to partition the data into distinct groups or clusters based on similarity measures. Popular clustering algorithms include k-means, hierarchical clustering, and DBSCAN.

K-means is an iterative algorithm that assigns data points to clusters based on their proximity to cluster centroids. It begins by initializing k centroids randomly, and then iteratively updates the centroids and reassigns data points to clusters until convergence is reached. The objective function for k-means is:

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} ||x_i^{(j)} - c_j||^2$$

where J is the objective function, $x_i^{(j)}$ is a data point, c_j is the centroid of cluster j and $||\cdot||$ denotes the Euclidean distance.

Hierarchical clustering builds a tree-like structure to represent the nested relationships between data points and clusters. It can be performed using either an agglomerative (bottom-up) or divisive (top-down) approach. In agglomerative hierarchical clustering, each data point initially forms its own cluster, and clusters are successively merged based on their similarity. Conversely, in divisive hierarchical clustering, all data points start in a single cluster, which is recursively split into smaller clusters.

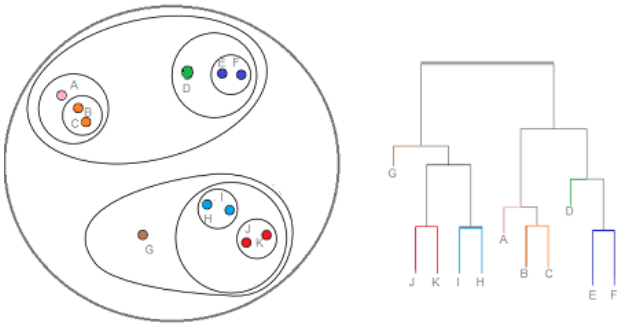

Figure 4: Dendogram representing nested clusters

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a density-based clustering algorithm that groups data points based on their density in the feature space. It defines a cluster as a dense region of points separated by areas of lower point density. DBSCAN is particularly useful for identifying clusters of arbitrary shapes and for handling noise in the data.

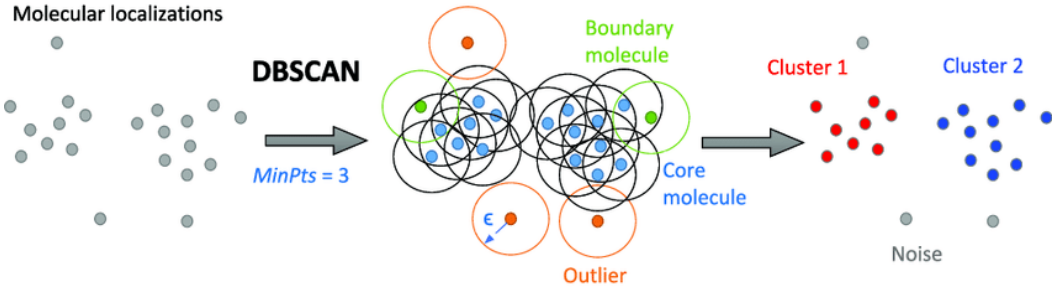

Figure 5: An example illustrating a DBSCAN method application

2.1.3. Reinforcement Learning

The study of reinforcement learning (RL) is a branch of machine learning which seeks to develop intelligent agents that can learn optimal decision-making strategies through interaction with their environment. RL is distinct from supervised and unsupervised learning in that it does not call for explicit supervision through labeled data. Instead, agents learn from feedback - both rewards or penalties - based on their actions in the environment. The goal of reinforcement learning is to learn a policy, which is a mapping from states to actions, that maximizes the expected cumulative reward over time.

The agent, the environment, the actions, the states, and the rewards are the essential components of reinforcement learning. The agent interacts with the environment by performing actions that change its state, and the agent is rewarded or punished for these actions. The agent aims to acquire a policy that maximizes the cumulative reward over time.

Among the key concepts in reinforcement learning is the notion of exploration versus exploitation. Exploration entails the agent taking uncertain actions to gain additional knowledge of the environment and enhance its understanding of state-action-reward dynamics. Exploitation, on the other hand, involves taking actions that are more likely to bring

the greatest rewards based on the agent's current knowledge. The success of reinforcement learning algorithms relies upon balancing exploration and exploitation.

The Markov Decision Process (MDP) is an essential concept of RL and it is a mathematical framework for modeling decision-making problems in an environment where the outcome is partially random and partly controlled by the decision maker. A tuple (S, A, P, R) is used to define an MDP, where S is a finite set of states, A is a finite set of actions, P is the transition probability function, $P(s'|s, a)$ represents the probability of transitioning from state s to state s' when action a is taken, R is the reward function, $R(s, a, s')$ represents the immediate reward received after transitioning from state s to state s' when action a is taken.

The objective of an agent in an MDP is to find a policy $\pi(s)$ that maps states to actions in such a way as to maximize the expected cumulative reward over time. This can be expressed mathematically using the concept of the value function, $V(s)$ which is the expected cumulative reward from following a policy π starting from state s:

$$V(s) = E\left[\sum_{t=0}^{\infty} \gamma^t R_{a_t}(s_t, s_{t+1}) \middle| s_0 = s, \pi\right]$$

where γ is a discount factor $(0 \leq \gamma < 1)$ that determines the relative importance of immediate versus future rewards, and E denotes the expectation.

Several algorithms have been developed to solve reinforcement learning problems, including value-based methods, policy-based methods, and actor-critic methods.

Value-based methods, such as Q-learning and SARSA, are focused on learning an optimal value function, which is then used to derive an optimal policy. In Q-learning, for example, the agent learns an action-value function, $Q(s, a)$, which represents the expected cumulative reward from taking action a in state s and following the optimal policy thereafter. The Q-learning update rule is given by:

$$Q(s, a) \leftarrow Q(s, a) + \alpha(R(s, a, s') + \gamma \max_{a'} Q(s', a') - Q(s, a))$$

where α is the learning rate $(0 \leq \alpha < 1)$.

Policy-based methods, such as REINFORCE and TRPO, are focused on directly learning a parameterized policy, $\pi(s; \theta)$, where θ represents the policy parameters. These methods typically use gradient-based optimization techniques to update the policy parameters in the direction that maximizes the expected cumulative reward. In the REINFORCE algorithm, for example, the policy gradient is computed as:

$$\nabla\theta E\left[\sum_{t=0}^{\infty} \gamma^t R_{a_t}(s_t, s_{t+1}) \middle| s_0 = s, \pi\right] = E\left[\sum_{t=0}^{\infty} \gamma^t R_{a_t}(s_t, s_{t+1}) \cdot \nabla\theta \log \pi(a_t | s_t ; \vartheta) \middle| s_0 = s, \pi\right]$$

The policy gradient can then be used to update the policy parameters θ using gradient ascent:

$$\theta \leftarrow \theta + \alpha \nabla\theta E\left[\sum_{t=0}^{\infty} \gamma^t R_{a_t}(s_t, s_{t+1}) \middle| s_0 = s, \pi\right]$$

Actor-critic methods, such as A2C and PPO, combine elements of both value-based and policy-based methods. In actor-critic algorithms, the agent learns a parameterized policy (the actor) and an associated value function (the critic) simultaneously. The critic is used to estimate the value function, which is then used to improve the policy. The policy gradient in actor-critic methods is computed as:

$$\nabla\theta E\left[\sum_{t=0}^{\infty} \gamma^t R_{a_t}(s_t, s_{t+1}) \middle| s_0 = s, \pi\right] = E\left[\sum_{t=0}^{\infty} \gamma^t (R_{a_t}(s_t, s_{t+1}) - V(s_t ; \psi)) \cdot \nabla\theta \log \pi(a_t | s_t ; \vartheta) \middle| s_0 = s, \pi\right]$$

where ψ represents the parameters of the value function. The policy parameters θ and value function parameters ψ are updated using gradient ascent and gradient descent, respectively:

$$\theta \leftarrow \theta + \alpha \nabla \theta E\left[\sum_{t=0}^{\infty} \gamma^t R_{a_t}(s_t, s_{t+1}) | \, s_0 = s, \pi\right]$$

$$\psi \leftarrow \psi - \alpha \nabla \psi E\left[\sum_{t=0}^{\infty} (R_{a_t}(s_t, s_{t+1}) - V(s_t\,; \psi))^2 \mid s_0 = s, \pi\right]$$

In recent years, deep learning has been integrated with reinforcement learning to develop deep reinforcement learning (DRL) algorithms, which can handle high-dimensional state and action spaces, complex environments, and large-scale problems. DRL algorithms, such as Deep Q-Networks (DQN) and Proximal Policy Optimization (PPO), use deep neural networks to represent the policy and/or value function, enabling the learning of complex and sophisticated decision-making strategies.

2.1.4. Deep Learning

Deep learning is a branch of machine learning which focuses on the creation and training of multilayer artificial neural networks (DNNs). The artificial neurons in these networks are designed to automatically extract and learn hierarchical features from raw data by passing it through several interconnected layers. Deep learning has a major benefit over traditional machine learning methods in that it is able to learn and represent complex patterns and representations directly from raw data without the need for feature engineering manually.

The architecture of a deep neural network consists of an input layer, multiple hidden layers, and an output layer. The layers consist of numerous artificial neurons or nodes that connect to the neurons in the neighbouring layers. The links between these neurons are associated with weights, which represent the strength of the connections. Additionally, each neuron has an associated bias term which regulates its output. The network is able to learn complex, non-linear patterns from the input data by mixing weights and biases.

The following are typical steps of a deep learning process:

- *Initialization*: Small random values will be used to initialize the weights and biases of the neural network.
- *Forward propagation*: The network then passes the input data through, and each neuron computes a weighted sum of its inputs, subtracts the bias term, and applies an activation function to create an output. The next layer of neurons gets this output and creates the overall output of the network until the last layer produces it.
- *Loss computation*: The output of the network is compared with the ground truth to determine a loss or error value. The loss measures the discrepancy between the predictions of the network and the actual target values.
- *Backward propagation*: The loss value is propagated back through the network, and the gradients of the loss with respect to each weight and bias are computed using a technique called backpropagation. The chain rule of calculus is used to compute the derivative of the loss function for each parameter in the network during this process.
- *Optimization*: The weights and biases of the network are updated using an optimization algorithm, including stochastic gradient descent (Adaptive moment or SGD) estimation (Adam), which adjusts the parameters to minimize the loss function. This update step is performed iteratively over multiple epochs or passes through the whole training dataset.

The forward propagation step is mathematically represented as follows:

Given an input vector x, the output of the first hidden layer h_1 can be computed as:

$$h_1 = f(W_1 x + b_1)$$

where W_1 is the weight matrix connecting the input layer to the first hidden layer, b_1 is the bias vector for the first hidden layer, and f is the activation function (e.g., ReLU, sigmoid, or tanh).

Similarly, the output of the second hidden layer h_2 can be computed as:

$$h_2 = f(W_2 h_1 + b_2)$$

This process continues until the output layer is reached, which computes the final output y:

$$y = f(W_L h_{L-1} + b_L)$$

where L is the number of layers in the network.

The backpropagation step involves computing the gradients of the loss function with respect to the weights and biases. For example, the gradient of the loss function J with respect to the weight matrix W_1 can be computed as:

$$\nabla W_1 J = \frac{\partial J}{\partial W_1}$$

These gradients are then used to update the weights and biases using the chosen optimization algorithm.

Deep learning's ability to learn complex, hierarchical representations from raw data has led to state-of-the-art performance in various tasks, surpassing traditional machine learning methods and, in some cases, even human performance.

One of its most popular architectures is the Convolutional Neural Network (CNN), which is specifically designed for processing grid-like data, such as images. CNNs consist of multiple convolutional layers, pooling layers, and fully connected layers. Convolutional layers apply convolution operations to the input data using a set of learnable filters or kernels. These filters enable the network to learn local patterns and features from the data while preserving the spatial relationships between the input elements. Pooling layers reduce the spatial dimensions of the data by performing down sampling operations, such as max-pooling or average pooling, which help reduce the computational complexity and control overfitting. Fully connected layers are used at the end of the network to produce the final output or predictions.

Another important deep learning architecture is the Recurrent Neural Network (RNN), which is designed to process sequences of data, such as time series or natural language text. RNNs consist of recurrent layers that maintain an internal hidden state, allowing them to capture and remember information from previous time steps in the input sequence. This ability to model temporal dependencies makes RNNs well-suited for tasks like language modelling, sentiment analysis, and time series prediction. However, RNNs suffer from challenges like vanishing and exploding gradients, which can impede learning in long sequences. To address these issues, more advanced RNN architectures, such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), have been developed.

2.1.5. Generative Models: Generative Adversarial Networks and Variational Autoencoders

Generative Adversarial Networks (GANs) are a type of AI model that can generate new data instances that resemble the input data. Two neural networks play a 'game' where the generative network creates new instances, and the discriminative network evaluates their authenticity, i.e., whether they are from the original dataset or were generated. GANs have seen extensive use in image creation and modification and even more complex applications like drug discovery. They represent an exciting frontier of AI that can enhance product development, creating innovative solutions for complex problems.

Variational Autoencoders (VAEs) are another type of generative model that uses the principles of autoencoders and introduces a probabilistic spin on the latent representation. They can generate synthetic data that are similar to the input data, proving valuable in a variety of applications, from anomaly detection to synthetic biomedical data generation. The ability of VAEs to exert a level of control over the generated instances brings a new level of sophistication to the AI landscape, offering exciting opportunities for product development and innovation.

2.1.6. Gradient-based Algorithms

Gradient-based algorithms form the backbone of many AI technologies, particularly those in machine learning. These algorithms use the principle of gradient descent to find the local minimum of a differentiable function. The inclusion of gradient information in the search space accelerates the learning process and improves the model's ability to find

optimal or near-optimal solutions. Techniques such as backpropagation in neural networks use gradient-based algorithms to fine-tune model parameters and minimize error. In the realm of product development, gradient-based algorithms can drive efficiency and accuracy, yielding better products in shorter timescales.

2.1.7. Hyperparameter Tuning Techniques and Sampling Methods

Hyperparameter tuning is a critical component of developing and refining neural network models, and techniques like grid search and random grid search are often employed. Grid search exhaustively tries every combination of the provided hyperparameter values to determine the best model, albeit at a higher computational cost. On the other hand, random grid search randomly samples the hyperparameter space and selects combinations to evaluate, significantly reducing the computational burden. These techniques contribute to refining AI models, which, when applied to product development, can lead to more precise and efficient solutions.

Latin Hypercube Sampling (LHS) is an advanced method for generating sample designs. It provides a more efficient and comprehensive coverage of the parameter space than simple random sampling methods. LHS can be used in conjunction with other optimization techniques for efficient model tuning or for generating synthetic data. By ensuring the entire space is explored with a fewer number of samples, LHS provides a balanced coverage, thus reducing the computational burden. In the context of product development, LHS can help generate comprehensive and representative datasets, supporting more robust and accurate model training and validation.

2.2. Evolutionary Computation

Evolutionary computation is a branch of artificial intelligence inspired by the process of natural selection and evolution. It employs a population of candidate solutions to optimize complex problems iteratively by applying principles of survival of the fittest, crossover, and mutation. These algorithms mimic the evolutionary process, searching for optimal solutions within a problem space.

One of the most common types of evolutionary computation techniques is Genetic Algorithms (GAs). These algorithms are based on the principles of genetics and natural selection, using a combination of selection, crossover (recombination), and mutation operators to evolve a population of candidate solutions towards an optimal or near-optimal solution.

2.2.1. Genetic Algorithms

Genetic algorithms consist of several key components and steps, as described below:

Encoding: The first step in a genetic algorithm is to represent the candidate solutions as a data structure, often called a chromosome. Chromosomes can be represented in various ways, such as binary strings, integer arrays, or real-valued vectors, depending on the problem at hand.

Initialization: A population of candidate solutions (chromosomes) is generated randomly or using a heuristic approach, considering the problem's constraints.

Fitness Evaluation: A fitness function is used to evaluate the quality of each candidate solution in the population. The fitness function is problem-specific and measures how well a solution satisfies the optimization objectives.

Selection: The selection operator is used to choose a subset of the population for reproduction, based on their fitness values. The higher a candidate solution's fitness value, the higher its probability of being selected. Common selection techniques include roulette wheel selection, tournament selection, and rank-based selection.

Crossover: The crossover operator combines the genetic material of two or more parent solutions to create offspring solutions. Crossover techniques vary depending on the chromosome representation. For binary strings, one-point, two-point, and uniform crossover are common methods. For real-valued vectors, techniques such as blend crossover

(BLX) and simulated binary crossover (SBX) can be employed. Mathematically, one-point crossover can be represented as follows:

$$parent_1 = [a_1, a_2, \dots, a_n]$$

$$parent_2 = [b_1, b_2, \dots, b_n]$$

$$crossoverPoint = random\ integer\ between\ 1\ and\ n-1$$

$$offspring_1 = [a_1, a_2, \dots, a_{crossoverPoint}, b_{crossoverPoint+1}, \dots, b_n]$$

$$offspring_2 = [b_1, b_2\ \dots, b_{crossoverPoint}, a_{crossoverPoint+1}, \dots, a_n]$$

Mutation: The mutation operator introduces small random perturbations to the offspring solutions, helping to maintain diversity within the population and prevent premature convergence. Mutation techniques depend on the chromosome representation. For binary strings, bit-flip mutation can be applied, while for real-valued vectors, Gaussian or polynomial mutation can be used. Mathematically, Gaussian mutation can be represented as follows:

$$offspring = [x_1, x_2, \dots, x_n]$$

$$mutationRate = probability\ of\ mutation\ for\ each\ gene$$

$$mutationStrenght = standard\ deviation\ of\ the\ Gaussian\ distribution$$

$$mutatedOffspring = [x_i + mutationStrenght\ \cdot randomGaussian()\ if\ random() < mutationRate\ else\ x_i\ for\ x_i\ in\ offspring$$

Replacement: The offspring solutions are integrated into the population, either by replacing the entire population (generational replacement) or by replacing a subset of the least fit individuals (steady-state replacement).

Termination: The algorithm iterates through steps 3 to 7 until a termination criterion is met, such as a maximum number of generations, a target fitness value, or a lack of improvement in the best solution.

By mimicking the principles of natural selection and evolution, genetic algorithms can efficiently search for high-quality solutions while maintaining diversity in the population, making them well-suited for various optimization problems, including those encountered in product development.
For example, in the optimization of design parameters, genetic algorithms can be employed to find the optimal combination of design variables that minimize cost, maximize performance, or satisfy multiple objectives simultaneously. By iteratively evolving the population of candidate solutions, genetic algorithms can explore a wide range of design alternatives and converge on a near-optimal solution that meets the desired objectives and constraints.

2.3. Computer Vision

Computer vision is a subfield of artificial intelligence that focuses on teaching computers to interpret and understand visual information from the surrounding world. This involves the development of algorithms and techniques that can process, analyse, and make sense of images and videos, enabling computers to perform tasks that typically require human vision. The main objective of computer vision is to replicate human visual perception, which can be achieved using mathematical models and computational techniques, such as convolutional neural networks and fully convolutional networks.
Some practical applications of computer vision in product development include automated quality control, visual inspection, and defect detection. For instance, computer vision algorithms can be used to identify manufacturing defects in products by comparing images of the produced items with reference images of defect-free products. This can help improve production efficiency, reduce waste, and ensure consistent product quality. Another application of computer vision in product development is in the field of augmented reality (AR), which involves overlaying digital information onto the real world. AR can be used to visualize and evaluate product designs in real-world settings,

enabling designers to make more informed decisions and identify potential design issues early in the development process. Computer vision techniques, such as object tracking and pose estimation, are essential for accurately aligning digital content with the physical environment in AR applications. In addition to the applications mentioned above, it can also be utilized in the analysis of consumer behaviour and market trends. By analysing visual data from social media platforms, product reviews, and online forums, computer vision algorithms can extract valuable insights into customer preferences, emerging trends, and potential areas for product innovation.

To further illustrate the capabilities of computer vision, let us consider the following example. Suppose a company is developing a new line of furniture and wants to evaluate the aesthetic appeal and functionality of their designs. By using computer vision techniques, such as object recognition and image understanding, the company can automatically analyse images of their products and identify key design elements, such as colour schemes, materials, and shapes. This information can then be used to refine the designs and optimize them for both aesthetics and functionality. Furthermore, it can also be used to simulate how the furniture would look in different environments, such as living rooms, bedrooms, or offices. This can help the company to better understand how their products would fit into various spaces and make adjustments to their designs accordingly.

2.3.1. Object recognition

Object recognition is a component of computer vision, which involves the identification and classification of objects inside a picture as well as video. The main purpose of object recognition is determining the category or maybe group of a certain item, like a vehicle, an individual, or even an animal, within the visual input.

A typical procedure for object recognition would be the usage of convolutional neural networks (CNNs), that are special deep-learning models meant to process grid-like details, like pictures. The structure of any CNN is composed of several layers, incorporating convolutional layers, pooling layers, and fully connected layers. The convolutional layers are accountable for extracting local features from the input image, while the pooling layers help minimize the spatial dimensions of the feature maps, making the product much more computationally efficient. The fully connected layers, on the other hand, are employed to classify the input according to the extracted features.

Mathematically, a convolutional layer performs a convolution operation between the input image I and a set of filters F to develop a feature map M**.** This can be represented as:

$$M(x,y) = (I > F)(x,y) = \sum_{i=1}^{m}\sum_{j=1}^{n} I(x_i, y_j) > F(i,j)$$

where $>$ denotes the convolution operation, and m and n would be the size of the air filter.

2.3.2. Image Understanding

Image-understanding goes beyond object recognition by emphasizing extracting higher level semantic info from images or videos. This includes tasks like scene-recognition, and object relationship understanding. The main goal of image-understanding is providing an extensive interpretation of the visual input, which could be utilized to support problem-solving and decision-making in numerous applications. A very common approach to image-understanding is the usage of semantic segmentation, which involves assigning a class label to each pixel in the image. This creates a pixel-wise classification map which offers a detailed understanding of the objects and their boundaries within the scene. A standard approach for semantic segmentation is the usage of fully convolutional networks (FCNs), that are CNN based models which substitute the fully connected layers with convolutional layers to generate a spatial output which retains the exact same dimensions as the input image. Mathematically, the result of an FCN can be represented as a probability map P with dimensions $H \times W \times C$, H and W are the height and width of the input image, and C is the number of class labels. The likelihood of any pixel (x, y) belonging to a class c can be denoted as:

$$P(x,y,c) = Softmax(Z(x,y,c))$$

where Z is the output of the final convolutional layer, along with $Softmax$ is an activation function which converts the raw output scores into probabilities.

2.4. Robotics

Robotics is a multidisciplinary field that combines engineering, computer science, and other related disciplines to design, develop, and operate robots. Robots are programmable machines or devices that can interact with the physical world, perform complex tasks, and execute actions autonomously or semi-autonomously. In the context of product development, robotics can be employed to support design and innovation processes, enhance manufacturing and assembly capabilities, and enable the creation of new and innovative products. By incorporating intelligent control and autonomous exploration capabilities, robots can perform complex tasks, adapt to changing environments, and contribute to the product development process in various ways. For example, robots can be used to automate repetitive tasks, such as assembly or inspection, enabling faster and more accurate production. Furthermore, robots equipped with advanced sensors and AI algorithms can assist in the design and prototyping stages, providing valuable insights and feedback.

2.4.1. Intelligent Control

Intelligent control refers to the application of advanced computational techniques and algorithms to control the behaviour and actions of robots, enabling them to adapt and respond to changing environments and conditions. This is achieved through the integration of various artificial intelligence techniques, such as machine learning, neural networks, and evolutionary computation, with traditional control methods, such as feedback control, feedforward control, and model predictive control.

A common approach to implementing intelligent control is through the use of reinforcement learning (RL), a type of machine learning where an agent learns to make decisions by interacting with an environment and receiving feedback in the form of rewards or penalties. The agent's goal is to maximize the cumulative reward over time by selecting the optimal actions in each state.

2.4.2. Autonomous Exploration

Autonomous exploration is a critical aspect of robotics, enabling robots to navigate and interact with their environments without human intervention. This capability is particularly important for robots operating in unstructured or unknown environments, such as search and rescue missions, space exploration, or product development tasks that require robots to adapt to changing conditions and requirements.

A key component of autonomous exploration is the robot's ability to perceive and understand its environment. This is achieved through the integration of various sensors, such as cameras, LIDAR, ultrasonic sensors, and accelerometers, that collect data about the robot's surroundings. The data is then processed using computer vision, machine learning, and other AI techniques to generate a representation of the environment, such as a map or a 3D model, which the robot can use to plan its actions and movements. One common approach to autonomous exploration is the use of the Simultaneous Localization and Mapping (SLAM) algorithm, which allows a robot to incrementally build a map of its environment while simultaneously estimating its position within that map. The SLAM problem can be formulated as a probabilistic estimation problem, where the goal is to compute the posterior distribution $P(x_t, m\,|z_{1:t}, u_{1:t})$ representing the robot's belief about its pose x_t and the map m at time t, given a sequence of sensor measurements $z_{1:t}$ and control inputs $u_{1:t}$ up to time t.

The SLAM algorithm typically involves two main steps: a motion update, which estimates the robot's pose based on its previous pose and control inputs, and a measurement update, which corrects the pose estimate and updates the map based on the new sensor data. Graph-based SLAM and Extended Kalman Filter (EKF) SLAM are two common implementations of the SLAM algorithm. Graph-based SLAM represents the map and robot poses as a graph, with nodes corresponding to poses and landmarks, and edges representing spatial constraints between them. The algorithm aims to find the configuration of nodes that best satisfies the spatial constraints, often by minimizing a cost

function representing the discrepancies between the estimated and measured constraints. In EKF SLAM, the robot's pose and map are represented by a state vector and an associated covariance matrix, and the algorithm iteratively updates the state vector and covariance matrix using an EKF-based estimation process. In addition to SLAM, other techniques, such as path planning and obstacle avoidance algorithms, are essential for ensuring the safe and efficient movement of robots within their environments. Path planning algorithms, such as A* and Rapidly-exploring Random Trees (RRT), are used to find the optimal path from the robot's initial position to a goal position, while obstacle avoidance algorithms, such as the Vector Field Histogram (VFH) and the Dynamic Window Approach (DWA), are employed to ensure the robot can navigate around obstacles in real-time.

2.5. Expert Systems

Expert systems are a branch of artificial intelligence that focuses on creating computer programs capable of emulating the decision-making abilities of human experts in specific domains. These systems rely on a knowledge base, consisting of facts and rules, to draw inferences and make decisions based on the input they receive. Expert systems have a wide range of applications, including decision support systems and teaching systems, which can be used in various industries, from healthcare and finance to engineering and product development.

2.5.1. Decision Support Systems

Decision support systems (DSS) are a type of expert system designed to assist users in making informed decisions by providing them with relevant information and analysis. In the context of product development, a DSS can help project managers, designers, and engineers evaluate potential design alternatives, assess risks, and optimize processes. A DSS typically consists of three main components: the knowledge base, the inference engine, and the user interface. The knowledge base contains domain-specific facts, rules, and relationships that the system uses to reason and make decisions. For example, in a product development DSS, the knowledge base might include information about materials, manufacturing processes, design principles, and market trends. The knowledge base can be represented using various techniques, such as production rules, frames, or semantic networks. A production rule is a simple if-then statement that represents the relationship between facts and actions. In mathematical notation, a production rule can be expressed as:

$$IF\ condition\ THEN\ action$$

For instance, a production rule in a product development DSS might be:

$$IF\ material =' steel' AND\ manufacturingProcess =' casting'\ THEN\ costPerUnit = 10$$

The inference engine is responsible for applying the knowledge base to specific problem instances, making deductions, and drawing conclusions. Inference engines typically use a combination of forward chaining and backward chaining algorithms to reason and generate solutions.
Forward chaining starts with the given facts and applies production rules to derive new facts until a goal is reached or no more new facts can be derived. In contrast, backward chaining starts with the goal and works backward, searching for facts and rules that can support the goal.
The user interface is the component through which users interact with the DSS, inputting data, and receiving results. User interfaces can range from simple text-based interfaces to more sophisticated graphical interfaces, depending on the complexity of the system and the needs of the user.

2.5.2. Teaching Systems

Intelligent tutoring systems or educational software are expert systems that provide personalized instruction and guidance to learners in a particular subject area, also called teaching systems. AI methods including natural language processing, machine learning, and knowledge representation are utilized by these systems to model the domain knowledge, the current understanding of the learner, and the instructional strategies most suitable to the learner's

requirements. A teaching system can help designers, engineers, and other professionals in boosting their knowledge and skills in different facets of the development process, like materials selection, design optimization, and manufacturing methods. Teaching systems generally consist of four primary components: the user interface, the student model, the tutoring model, and the domain model.

The domain model describes the subject matter knowledge that the teaching system is designed to impart to the student. The knowledge base of decision support systems may be represented using techniques such as production rules, frames, or semantic networks similarly to the domain model. The student model is a dynamic representation of current knowledge, skills, and understanding of the subject matter by the learner. The teaching system can adjust its instruction to the learner's requirements and progress by continuously updating this model based on the learner's interactions with it. The tutoring model delivers and selects appropriate instructional strategies and feedback depending on the domain model and the student model. This model incorporates pedagogical knowledge and strategies, including scaffolding, questioning, and elaboration, to facilitate successful learning experiences tailored to the individual learner. Like decision support systems, the user interface is the interface whereby learners interact with the teaching system, entering data, getting feedback, along with engaging with instructional materials. User interfaces for teaching systems can range from text-based interfaces to multimedia rich environments, depending on the subject matter and the learner's preferences.

2.6. Speech Processing

Speech processing is a subfield of artificial intelligence that focuses on the analysis, synthesis, and understanding of human speech. Speech processing aims to allow machines to interpret and generate spoken language, thus facilitating communication and interaction between human beings and AI systems. Speech processing is a difficult task as human speech is extremely variable and can easily be influenced by accents, dialects, feelings and speaking rate. There are three major elements to speech processing: speech recognition, speech synthesis, and speaker identification.

2.6.1. Speech Recognition

Speech recognition, also known as automatic speech recognition (ASR), refers to the process of converting spoken language into a textual representation. ASR systems use mathematical models and algorithms to analyse the acoustic properties of speech signals and identify the corresponding linguistic units, such as phonemes, words, and phrases. One of the most common techniques used in ASR is the Hidden Markov Model (HMM), a statistical model that represents the relationship between the acoustic properties of speech and the underlying linguistic units. HMMs are particularly well-suited for ASR due to their ability to model time-varying processes, such as speech signals, and to account for uncertainties and variability in the data. Given a sequence of acoustic observations, an HMM can estimate the most likely sequence of linguistic units that produced the observed speech signal.

The process of training an HMM-based ASR system typically involves the following steps:

Feature extraction: Convert the raw speech signal into a set of acoustic features, such as Mel-frequency cepstral coefficients (MFCCs), which capture the spectral properties of the signal.

Acoustic model training: Train an HMM for each linguistic unit (e.g., phoneme) using a set of labeled speech data. The trained HMMs can represent the statistical relationship between the acoustic features and the corresponding linguistic units.

Language model training: Develop a statistical model, such as an n-gram model, that captures the likelihood of different sequences of words or phrases occurring in the target language. This model can help the ASR system to disambiguate between acoustically similar words or phrases based on their contextual probabilities.

Decoding: Given a new speech signal, use the trained HMMs and the language model to estimate the most likely sequence of linguistic units that produced the observed acoustic features.

2.6.2. Speech Synthesis

Speech synthesis, also called text - to - speech (TTS), is the creation of spoken language from a textual input. TTS systems aim to make natural - sounding, understandable speech which very much resembles human speech patterns.

A very common procedure for speech synthesis is the *concatenative synthesis technique*, which requires assembling sections of recorded speech (e.g., diphones, phonemes, or syllables) to create the desired speech output. This method usually requires a large database of recorded speech units, that are indexed and aligned based on their prosodic and acoustic properties. The database is used to pick the speech segments that are required during synthesis, and they are concatenated using smoothing methods to remove discontinuities and artifacts at the segment boundaries.

The *parametric synthesis technique* is another approach to speech synthesis which utilizes mathematical models to produce speech signals based on a set of input parameters, including fundamental frequency, duration, and spectral envelope. The source-filter model is an example of a parametric synthesis model that separates the speech signal into a source component (e.g., the glottal excitation signal) and a filter component (e.g., the vocal tract transfer function). The model is able to create a broad range of speech sounds and prosodic variations by adjusting the input parameters.

Neural network-based TTS systems like Tacotron and WaveNet show promising results in generating top - quality, natural - sounding speech because of recent advances in deep learning. Deep learning architectures including convolutional neural networks (CNNs) and recurrent neural networks (RNNs) are utilized in these models to learn the intricate connections between the textual input and the speech signal.

WaveNet, for instance, is a generative model which utilizes a stack of dilated CNN layers to record the temporal dependencies in the speech signal. WaveNet is able to produce a raw speech waveform by predicting the amplitude of any speech signal at each time step given a sequence of linguistic features (e.g., pitch, phonemes, and duration) as input. The model is trained on a huge dataset of speech signals, learning the intricate patterns and structures of the target language.

Tacotron is an end - to - end TTS system which utilizes a sequence - to - sequence model and attention mechanisms to transform textual input directly into spectrograms which represent the frequency content of the speech signal over time. A vocoder like the Griffin-Lim algorithm or a neural vocoder like WaveNet can be used to convert these spectrograms into raw speech waveforms. Tacotron can produce speech signals that resemble human speech in both acoustic properties and prosody by training the model on a vast quantity of text and speech data.

2.6.3. Speaker Identification

Speaker identification is the task of determining the identity of a speaker based on their voice characteristics. This process can be useful in various applications, such as voice biometrics, personal assistants, and audio indexing. Speaker identification systems typically rely on extracting a set of distinctive features from the speech signal, which can be used to represent the speaker's unique vocal characteristics, such as pitch, formant frequencies, and spectral envelope.
A common approach to speaker identification is the Gaussian Mixture Model-Universal Background Model (GMM-UBM) framework. In this method, a GMM is trained on a large dataset of speech signals from multiple speakers, representing the general distribution of acoustic features in the population. This model, known as the Universal Background Model (UBM), serves as a baseline for comparing the acoustic features of an unknown speaker.
To identify a specific speaker, a new GMM is adapted from the UBM using a set of speech data from the target speaker, resulting in a speaker specific GMM. The speaker specific GMM captures the unique characteristics of the target speaker, while still retaining the general structure of the UBM. When presented with an unknown speech signal, the system compares the likelihood of the observed acoustic features under the speaker specific GMM and the UBM, making a decision based on the ratio of the likelihoods.

With the advent of deep learning, speaker identification systems have also started to employ neural network-based architectures, such as deep neural networks (DNNs) and long short-term memory (LSTM) networks, to learn more complex and discriminative representations of speaker-specific features. These deep learning models can be trained on large-scale datasets to capture the subtle variations in voice characteristics, leading to more accurate and robust speaker identification performance.

2.7. Natural Language Processing

Natural Language Processing (NLP) is a subfield of artificial intelligence which concentrates on the interaction between computer systems and human languages. NLP seeks to make computers capable of understanding, interpreting, generating, and responding to human language in a significant and contextually relevant way. NLP encompasses a wide range of tasks, including sentiment analysis, text summarization, language translation, named entity recognition, along with question-answering systems. NLP helps AI systems comprehend and interact with human users and data sources by producing algorithms and strategies for processing and analysing natural language.

The challenge of representing the intricate structure and meaning of human language in a form that can be quickly processed by computer algorithms is the core of NLP. It entails separating language into its component parts - words, phrases, sentences, and paragraphs - and coding their relationships and semantic properties in a structured manner. Various mathematical and computational models have been developed to represent and process natural language, some of which are outlined below.

Tokenization is the process of breaking a text into individual words or tokens, which serves as the foundation for most NLP tasks. For example, given the sentence "The cat is on the mat," tokenization could generate the following list of tokens: ["The", "cat", "is", "on", "the", "mat"] .

Individual tokens within a text are tagged with grammatical categories (nouns, adjectives, verbs, and adverbs) *using Part-of-Speech (POS) tagging*. This information may be utilized for various NLP tasks including parsing, named entity recognition and sentiment analysis. The POS tags within the example sentence above will be: ["The/DT", "cat/NN", "is/VBZ", "on/IN", "the/DT", "mat/NN"] .

Parsing entails analysing the grammatical structure of a sentence and developing a parse tree that mirrors its syntactic connections. This can be achieved using a variety of algorithms and formalisms, including context - free grammars (CFGs), dependency parsing, and probabilistic parsing models. Parse trees are able to offer helpful insights into the semantic connections between words and phrases, useful for tasks like information extraction, question-answering, and text summarization.

Word embeddings are mathematical representations of words that convey semantic properties and relationships in a high dimensional vector space. Unsupervised machine learning methods like Word2Vec, GloVe, or FastText usually generate these representations by training a neural network on big corpora of text data. Word embeddings offer a compact and effective way to represent and compare the meanings of words and also may be used for various NLP tasks like text classification, sentiment analysis and machine translation. One popular method for generating word embeddings is the Word2Vec algorithm, which learns vector representations of words according to their co-occurrence patterns in a corpus. Word2Vec is based on the distributional theory, which says that words that are used in similar circumstances have similar meanings. The algorithm utilizes a shallow neural network to predict the context words for a given target word, adjusting the vector representations of words throughout the process. The resulting embeddings represent semantic relationships between words, causing words with comparable meanings to have similar vector representations.

Sequence-to-Sequence Models are a kind of deep learning architecture used for tasks that involve mapping input sequences to output sequences, for example machine translation, text summarization, and speech recognition. Generally, these models consist of an encoder network that processes the input sequence and a decoder network which generates the output sequence. Usually, the encoder and decoder networks are implemented as recurrent neural networks (RNNs), long short-term memory (LSTM) networks, and gated recurrent units (GRUs) which can deal

with variable - length sequences and also capture long range dependencies. For instance, in a machine translation task, the encoder network would encode the meaning of the input sentence in the source language into a fixed-length vector representation. The decoder network would then generate the corresponding sentence in the target language word for word using this representation. An attention mechanism is usually involved in this process, enabling the decoder to concentrate on specific parts of the input sequence while producing the output sequence, therefore enhancing the model's ability to handle lengthy sentences and complicated structures.

The use of *transformer models* is a recent advancement of NLP and has led to breakthrough results in areas like machine translation, text summarization and sentiment analysis. The main development of transformer models is the self-attention mechanism, which allows them to efficiently process and analyse long-range dependencies in natural language data. The self-attention mechanism calculates a weighted sum of the input representations for each position in the sequence, the weights being determined by the similarity between the input positions. The model can dynamically focus on various parts of the input sequence while processing every position, allowing it to capture intricate relationships and dependencies between words and phrases. Usually, transformer models are implemented as deep, multi-layer architectures with separate encoder and decoder networks for sequence-to-sequence tasks. One important example of a transformer model is the BERT (Bidirectional Encoder Representations from Transformers) model, which has achieved cutting edge results on numerous NLP benchmarks. BERT is a transformer model which may be trained for specific tasks, like text classification, named entity recognition, or question-answering, utilizing little training data specific to each task. BERT along with other transformer models have made substantial progress in NLP by utilizing the effective representational abilities of the transformer architecture and the massive amounts of pre-training data.

2.8. Planning

Planning is an important element of artificial intelligence that entails creating a sequence of actions or decisions to achieve a goal, typically within the context of uncertain or dynamic environments. Algorithms in AI systems usually plan by examining possible action sequences to determine the best or near-optimal solution to a problem. The planning element of AI is explored in this chapter, which focuses on the fundamental principles, algorithms, and methods that AI systems use to create and execute effective plans.

In AI planning the state space is the foundation for every further step, which comprehends the set of all possible states that could be reached from a given initial state through a sequence of actions. The state space can be visualized as a graph, wherein every node corresponds to a state and the edges represent the actions that transition between states. AI planning is designed to investigate this state space to find out a sequence of actions or a plan that will take the user from the initial state to a goal state fitting specified objectives or constraints.

The A* search algorithm, a best-first search algorithm employing heuristics to guide the search process, is among the most well-known planning algorithms. The A* algorithm works by keeping a priority queue of nodes, where each node represents a state and also has an associated cost function, $f(n) = g(n) + h(n)$. Here, $g(n)$ denotes the cost of the path from the initial state to the current node n, and $h(n)$ is a heuristic estimation of the cost from node n to the objective state. The algorithm picks the node having the lowest $f(n)$ value, expands it by generating its successors and updates the priority queue accordingly. The search process continues until the goal state is reached or the priority queue is empty.

The strength of the A* algorithm mainly relies on the choice of the heuristic function $h(n)$. A well-constructed heuristic must be admissible (it never overestimates the true cost) and consistent (it meets the triangle inequality). Heuristics like these ensure that the A* algorithm will always find the best solution to a planning problem if a solution exists.

Some other planning algorithms are proposed in the literature, along with the A* algorithm, such as:

Dijkstra's algorithm: A graph search algorithm that discovers the shortest path between the initial state and the goal state, without considering any heuristic information. The search process of Dijkstra's algorithm is guaranteed to yield the optimal solution, although it might be slower than A* > due to the absence of heuristic guidance.

MCTS (Monte Carlo Tree Search): A randomized search algorithm that develops a search tree incrementally through a sequence of simulations starting from the initial state. The algorithm picks a node to expand based on a selection policy (e.g., Upper Confidence Bound for Trees, or UCT) at each step, generates its successors, and updates the value estimation for the node. MCTS is especially effective in large or continuous state spaces in which the use of heuristics is challenging.

Partially Observable Markov Decision Processes (POMDPs): A framework for planning in uncertainty, where the state of the environment is only partly observable. POMDPs handle the planning problem like a stochastic process with hidden states and utilize techniques like belief state updates and value iteration to determine best action sequences.

AI planning has lots of applications, including robotics, autonomous vehicles, natural language processing, and game playing, among others. Researchers and developers can make AI systems reason on their environment, make informed decisions, and perform challenging tasks goal-directedly by integrating planning algorithms and techniques. The ongoing development of planning algorithms and their integration into AI systems will lead to substantial enhancements in the overall capabilities and performance of technologies, ultimately resulting in more intelligent and adaptive systems.

Moreover, AI planning has already been combined with machine learning techniques to develop more robust and adaptive planning algorithms. For instance, reinforcement learning could be employed to learn optimal action sequences from trial and error, while supervised learning can be utilized to learn heuristic functions from labeled training data. These hybrid planning approaches have the potential to address some of the difficulties related to traditional planning algorithms, such as the curse of dimensionality and the reliance on handcrafted heuristics.

Handling constraints and preferences in real-life problems is core in planning. Planning algorithms are able to incorporate constraint satisfaction techniques to make sure that plans follow constraints specified, and preference-based planning algorithms can optimize plans based on user-defined preferences or utility functions. These advanced planning techniques enable AI systems to produce more flexible and customizable solutions, which are better suited to the dynamic and diverse nature of real-world problems.

2.8.1. Scheduling

Scheduling is an essential aspect of AI planning that deals with the allocation of resources and the sequencing of tasks to achieve specific goals within a given time frame. Effective scheduling enables the optimization of resource usage, timely completion of tasks, and ensures a well-structured workflow. In the context of product development, AI-based scheduling techniques can significantly enhance the process by optimizing resource allocation and streamlining project timelines.

Scheduling problems can be classified into various types, such as single-machine scheduling, parallel-machine scheduling, flow-shop scheduling, and job-shop scheduling. In general, scheduling problems involve determining the optimal order and timing of tasks while considering constraints such as precedence relations, resource availability, and deadlines. Formally, a scheduling problem can be represented as a tuple (T, R, C), where T denotes a set of tasks, R represents the set of resources, and C corresponds to the set of constraints.

Mathematically, scheduling can be approached using various optimization techniques, including linear programming, integer programming, and dynamic programming. For instance, one of the most common scheduling problems, the single-machine scheduling problem, can be modeled using integer linear programming (ILP). The objective function in this case aims to minimize the completion time of the last task, also known as makespan, subject to the constraints imposed by the problem.

Suppose there are n tasks $\{t_1, t_2, \dots, t_n\}$ with processing times $\{p_1, p_2, \dots, p_n\}$ to be scheduled on a single machine. The decision variables x_{ij} are binary and represent whether task i is scheduled immediately before task j ($x_{ij} = 1$) or not ($x_{ij} = 0$). The completion times for each task are denoted by C_i, where $i = 1, 2, \dots, n$. The ILP model for the single-machine scheduling problem can be formulated as follows:

$$(1)\ C_i = p_i + \sum_{j=1}^{n} x_{ij}(p_j + C_j)\ , for\ i = 1,2, \dots, n$$

$$(2)\ \sum_{i=1}^{n} x_{ij} = 1, for\ j = 1,2, \dots, n$$

$$(3)\ \sum_{j=1}^{n} x_{ij} = 1, for\ i = 1,2, \dots, n$$

$$(4)\ x_{ij} \in \{0,1\}, for\ i,j = 1,2, \dots, n$$

Constraint (1) enforces that the completion time of task i is equal to its processing time plus the sum of processing times and completion times of all tasks scheduled immediately after it. Constraints (2) and (3) ensure that each task is scheduled exactly once before and after another task. Constraint (4) imposes the binary nature of the decision variables.

In addition to traditional optimization techniques, AI-based methods such as genetic algorithms, simulated annealing, and particle swarm optimization have been employed to solve scheduling problems. These metaheuristic algorithms have shown great potential in solving complex scheduling problems by exploring the solution space more effectively and providing near-optimal solutions in reasonable computational time.

For instance, the genetic algorithm (GA) is a population-based optimization technique inspired by the principles of natural selection and genetics. In the context of scheduling, a GA starts with an initial population of candidate solutions (schedules) and iteratively evolves the population through operations such as selection, crossover, and mutation. The fitness of each candidate solution is evaluated based on the objective function (e.g., minimizing makespan), and the algorithm converges when a termination criterion is met, such as reaching a maximum number of iterations or finding an acceptable solution.

In the GA-based approach to scheduling, a chromosome represents a candidate solution (schedule) and is typically encoded as a permutation of task indices. The fitness function is derived from the scheduling objective, such as minimizing makespan or total tardiness. Simulated annealing (SA) is another metaheuristic algorithm inspired by the annealing process in metallurgy. In the context of scheduling, SA starts with an initial solution (schedule) and iteratively explores the solution space by applying local search moves (e.g., swapping tasks). At each iteration, the algorithm decides whether to accept a new solution based on its quality and a temperature parameter that controls the level of exploration. As the algorithm progresses, the temperature parameter decreases, gradually reducing the probability of accepting worse solutions. This cooling schedule allows SA to balance exploration and exploitation effectively, avoiding getting trapped in local optima.

Particle swarm optimization (PSO) is a population-based optimization technique inspired by the social behaviour of bird flocks or fish schools. In the context of scheduling, each particle (candidate schedule) in the swarm is initialized with a random position in the solution space and a random velocity. At each iteration, the particles update their positions and velocities based on their personal best solutions and the global best solution found by the swarm. This process allows the swarm to converge towards an optimal or near-optimal solution through collective learning.

2.8.2. Game Playing

Game playing has been a significant research area within artificial intelligence, as it provides a controlled environment for testing and developing AI algorithms, heuristics, and strategies. Game playing can be applied to the product development process (PDP) as a means of simulating competitive environments, generating innovative ideas, and enhancing decision-making processes. In this section, we discuss the techniques and concepts involved in AI-based game playing and their potential applications in the PDP. Games can be broadly categorized into two types: deterministic games, where the outcome of each move is known, and non-deterministic games, where the outcome

is uncertain. The techniques used for game playing in AI can be classified into search-based methods, knowledge-based methods, and machine learning-based methods.

One of the fundamental approaches to AI-based game playing is *search-based methods*, such as the minimax algorithm and its variants. The minimax algorithm is a decision-making algorithm for two-player, zero-sum games with perfect information. It works by constructing a game tree, where each node represents a game state, and the edges represent possible moves. The algorithm recursively searches the tree to find the optimal move for the current player by minimizing the maximum possible loss.

The minimax algorithm can be represented mathematically as follows:

$$minimax(node, depth):$$

$$utility(node), if\ terminalTest(node)\ or\ depth == 0$$

$$max(minimax(child, depth - 1)), if\ the\ node\ is\ a\ max\ node$$

$$min(minimax(child, depth - 1)), if\ the\ node\ is\ a\ min\ node$$

where $utility(node)$ is the utility value of the game state represented by the node, $terminal\_test(node)$ checks if the node represents a terminal state (i.e., the game has ended), and $depth$ is the maximum search depth. One limitation of the basic minimax algorithm is its computational complexity, which grows exponentially with the search depth. To address this issue, various pruning techniques, such as alpha-beta pruning, have been developed to reduce the search space without affecting the outcome of the search.

Knowledge-based methods incorporate domain-specific knowledge and heuristics to guide the search process and improve decision-making in game playing. For example, in chess, AI systems may use evaluation functions to estimate the value of a game state based on factors such as material balance, king safety, and pawn structure. These evaluation functions can be represented mathematically as linear combinations of weighted features:

$$E(state) = w_1 f_1(state) + w_2 f_2(state) + \cdots + w_n f_n(state)$$

where $E(state)$ is the evaluation value of a game state, $f_i(state)$ is the value of feature i in the game state, and w_i is the weight assigned to feature i. Knowledge-based methods can be particularly useful in the PDP, as they can leverage domain-specific knowledge and expertise to guide the search for optimal solutions and inform decision-making processes.

Machine learning-based methods, such as reinforcement learning and deep learning, have recently gained prominence in AI-based game playing. These methods involve training AI systems to learn optimal strategies and decision-making policies through interaction with the game environment. Reinforcement learning, for example, is an approach in which an AI agent learns to make decisions by receiving feedback in the form of rewards or penalties. The agent seeks to maximize the cumulative reward over time by exploring the environment and learning from its actions.

AI-based game playing techniques can be integrated into the PDP to support various activities, such as ideation, decision-making, and optimization. AI-based game playing can be used as a tool for stimulating creativity and innovation in the PDP. By exploring a vast search space of possible solutions, AI systems can generate novel and unconventional ideas that might not be immediately apparent to human designers. Additionally, game playing can foster collaboration and idea exchange among team members, leading to more innovative and successful products. Game playing techniques can also be used to create competitive simulations that model market dynamics, customer preferences, and competitor behaviour. These simulations can help product developers understand the competitive landscape, identify opportunities and threats, and devise effective strategies to gain a competitive edge. Moreover, by learning from simulation data, AI systems can iteratively refine design solutions and identify optimal configurations that satisfy the desired performance requirements.

2.9. Large Language Models

Large language models, such as GPT-4, have garnered considerable attention in recent years for their ability to generate human-like text, perform complex tasks, and even support creative endeavours. These models are based on deep learning techniques and are trained on vast amounts of textual data, enabling them to capture the statistical patterns and structures inherent in human language. As a result, large language models can be applied in various stages of the product development process (PDP), such as idea generation, concept development, and even marketing, to support innovation and streamline workflows.

The foundation of large language models, such as GPT-4, lies in the transformer architecture, which was introduced by *Vaswani et al.* in 2017. The transformer architecture is a type of neural network specifically designed for handling sequence-to-sequence tasks, such as machine translation or text generation. Its core components are the self-attention mechanism and the position-wise feed-forward networks.

The self-attention mechanism allows the model to weigh the importance of each word in a sequence relative to the others. Mathematically, the self-attention mechanism can be represented as:

$$Attention(Q, K, V) = Softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Where Q, K, and V are the query, key, and value matrices, respectively, and d_k is the dimension of the key vectors. The $Softmax$ function is applied to ensure that the attention weights sum to one. The position-wise feed-forward networks consist of two fully connected layers with a ReLU (rectified linear unit) activation function in between. These networks are applied to each position in the input sequence independently and identically, allowing the model to capture local interactions between words. Training a large language model like GPT-4 involves using unsupervised learning techniques, specifically a form of self-supervised learning called masked language modelling. In this approach, a portion of the input text is masked, and the model is tasked with predicting the masked words based on the context provided by the surrounding words. The model's parameters are updated using gradient descent, an optimization algorithm that minimizes the difference between the model's predictions and the actual words. The size of large language models is measured in terms of the number of parameters they contain, which directly influences their ability to capture complex linguistic patterns and relationships. GPT-4, for instance, has billions of parameters, allowing it to generate high-quality text that closely resembles human-written text. Large language models can be employed in various stages of the NPD to facilitate innovation and enhance efficiency. By leveraging the vast knowledge captured in their training data, large language models can generate novel ideas and concepts for new products, helping to inspire creativity and innovation among teams. In concept development, they can be used to draft product descriptions, user stories, or even design specifications, helping to refine and communicate the core concepts behind new products. By analysing large volumes of textual data, such as customer reviews or social media posts, large language models can identify trends, preferences, and pain points, providing valuable insights for product development teams to inform their decision-making. Models can also assist in crafting persuasive and engaging marketing materials, such as product descriptions, advertisements, or social media posts, helping to raise awareness and drive interest in new products.
Despite their impressive capabilities, large language models are not without limitations. They can sometimes generate text that is coherent but factually incorrect or inconsistent, and they may also propagate biases present in their training data. As such, it is essential for development teams to exercise caution and employ human oversight when incorporating large language models into their workflows. Evaluating the performance of large language models is crucial for ensuring their effectiveness in the PDP. For example, perplexity measures the average likelihood assigned by the model to each word in a given sequence, with lower perplexity values indicating better performance. Mathematically, perplexity can be expressed as:

$$Perplexity(W) = P(w_1, w_2, \dots, w_n)^{\frac{-1}{n}}$$

Where W is the sequence of words and $P(w_1, w_2, \dots, w_n)$ is the probability of the sequence according to the model.

The BLEU (Bilingual Evaluation Understudy) Score was originally developed for evaluating machine translation systems, and it measures the similarity between the model-generated text and human-written reference texts. It calculates the geometric mean of n-gram precision scores, with higher BLEU scores indicating better performance.

The ROUGE (Recall-Oriented Understudy for Gisting Evaluation) Score measures the overlap between the model-generated text and human-written reference texts in terms of n-grams, with higher ROUGE scores indicating better performance.

These evaluation metrics can help development teams assess the quality and reliability of large language models, ensuring that their incorporation into the PDP is both effective and responsible.

3. AI for Product Development Process

Artificial intelligence has the potential to revolutionise the way products are created, produced, and sold, especially in the field of product development. In this particular section we are going to explore how AI technologies can be utilized to enhance the product development process at different phases and the potential advantages they are able to offer.

In the planning phase, AI could be utilized to help in market research and customer needs identification by analysing huge amounts of customer feedback, product reviews and online discussions using natural language processing (NLP) algorithms to detect areas for improvement, preferences, and trends. Machine learning models also can predict market demand, allowing product development teams to make better choices regarding which product concepts to pursue.

AI may also be implemented to facilitate creativity and ideation throughout the concept development stage. Generative design algorithms make use of machine learning methods to look right into a vast design space and produce novel products based on specified constraints, like performance needs or manufacturing constraints: these algorithms enable product development teams to investigate a broader range of possible solutions, leading to far more creative and enhanced product designs. AI can also assist in the assessment of various design concepts by enabling teams to simulate performance and calculate their market potential, cost, and manufacturability.

In the system-level design phase, AI can assist in the improvement of product architecture and the selection of materials and components. Machine learning models can be trained on historical data to predict the performance, cost, and reliability of various system configurations and materials, enabling product development teams to make much more informed choices regarding their designs. Furthermore, AI may be utilized to automate the generation of system-level design documentation, for example functional block diagrams and interface specifications, streamlining the design process and lowering the chance of mistakes.

AI can also be utilized during the detailed design stage to optimize particular components and assemblies to ensure their manufacturability, cost, and performance meet specifications. Topology optimization algorithms, for instance, could be used to produce light and structurally effective component designs, while machine learning models are able to predict the manufacturability and cost of different manufacturing procedures including injection molding, machining, or additive manufacturing. AI-powered CAD tools also automate repetitive and time-consuming tasks like making 2D drawings or even checking design rules, freeing engineers up to concentrate on high value, creative jobs.

Throughout the testing and refinement stage, AI could play a crucial role in the analysis and interpretation of test data. The training of machine learning models enables them to recognize patterns and anomalies in test data, like performance degradation, component failures, or manufacturing defects. Product development teams can accurately and quickly determine issues, pinpoint root causes, and implement design changes by utilizing AI in this fashion.

Furthermore, AI may be employed to optimize the design of actual physical tests, like wind tunnel tests or crash testing, ensuring the most relevant and informative test scenarios are executed.

AI can also certainly streamline supply chain management, enhance manufacturing processes as well as quality control during the production ramp - up stage. Machine learning models are able to anticipate and minimize production flaws, and computer vision algorithms are able to automate the inspection of manufactured parts to make sure they meet quality standards. AI can even improve production schedules and inventory control, leading to shorter lead times, fewer stockouts and in general operational efficiency.

In summary, AI technologies could improve the product development cycle from the initial planning phase to the ramp - up stage. Development teams are able to boost their competitiveness as well as efficiency by utilizing AI tools as well as strategies and, considering that they continue to strengthen, the opportunities for designing and manufacturing products which are perfect for consumers and companies will go on to expand.

They may also help with product development project management by optimizing resource allocation, risk assessment and improvement tracking. For instance, machine learning models can examine historical project data to identify correlations and patterns between project parameters like budget, schedule and complexity, and project outcomes like financial success rate, price, and duration. Project managers can make better choices about resource allocation, prioritization, and scheduling making use of these models, which will inevitably lead to more effective product development projects.

Additionally, AI can facilitate collaboration and communication among development teams by determining possible misunderstandings and suggesting clarifications, all using natural language processing, therefore minimizing miscommunications, and also increasing staff productivity. AI may also help cross-functional collaboration by automatically producing design documentation that is customized to the necessities of different stakeholders which includes manufacturing engineers, suppliers, and regulatory authorities.

The incorporation of AI into the product development process will also help with more sustainable and eco-friendly products since AI-powered tools can help identify opportunities for reducing waste generation, energy consumption, and material usage, while simultaneously optimizing the usage of recyclable and renewable materials. Furthermore, machine learning models are able to predict the environmental impact of different product models and manufacturing processes, enabling development teams to make better choices concerning the sustainability of the products.

To exploit the AI potential in product development, organizations should invest in the appropriate infrastructure, programs as well as education, including the adoption of AI hardware and software, internal AI capabilities growth and a culture of ongoing learning and improvement. By doing so, organizations can position themselves to make the most of the transformative power of AI in product development.

3.1. Bridging AI and Cognitive Science

Cognitive science includes all areas of human cognition - perception, decision-making, reasoning, memory, problem-solving along with learning -making it an interdisciplinary field. On the other hand, artificial intelligence seeks to develop machines and algorithms capable of executing tasks which generally require human intelligence. By merging insights from cognitive science and AI capabilities, we can reach a deeper understanding of cognitive activities involved in product development and also start utilizing AI tools to facilitate and improve these activities.

Bridging AI and cognitive science in the context of product development is going to involve the creation of AI algorithms and strategies inspired by and modelled after human cognitive processes, as one of the crucial parts. Cognitive computing refers to the creation of AI systems that will learn from experiences, adjust to completely new information, and draw conclusions from data in ways that imitate human cognitive activities.

Take the cognitive action of identification as a good example from the product development process. Identification refers to understanding and determining key features, relationships, and elements within a problem space, such as identifying customer requirements, industry trends or technical constraints. AI algorithms might be created to analyse and process a huge amount of structured and unstructured data to facilitate cognitive activity, including textual info from consumer reviews or social media feeds, image information from product prototypes or business analysis, and also numerical data from sales reports or business metrics.

To accomplish this objective, algorithms or methods including natural language processing (NLP) for text analysis, computer vision for image analysis, and machine learning for pattern recognition and prediction, might be employed. For instance, an AI tool may utilize NLP algorithms to process customer feedback and find out recurring themes or sentiments, providing helpful insights into what buyer wants and personal preferences. The analysis of product images might be attained using computer vision algorithms to highlight important design characteristics or possible improvement areas.

Mathematically, the algorithms can be modelled as optimization problems, where the objective is to maximize or minimize a certain performance metric or cost function. For example, in the case of NLP, an AI algorithm can be designed to maximize the accuracy of sentiment classification, while minimizing the misclassification rate. This can be formulated as:

$$minimize: \quad C = w_1(1 - accuracy) + w_2\, misclassificationRate$$

Where C is the cost function to be minimized, w_1 and w_2 are weight parameters that determine the relative importance of the two terms, and $accuracy$ and $misclassificationRate$ are the performance metrics of the algorithm. By solving this optimization problem, the algorithm can be trained to find the optimal set of parameters that result in the best performance in terms of sentiment classification and misclassification rate.

Combining these two fields also entails the creation of AI tools to improve human cognition and structure and support decision-making. In the context of product development, this can include the design of systems that will help in activities like design optimization, concept evaluation, and idea generation. Algorithms could be utilized to develop novel design concepts by mixing design elements, features, and materials in novel ways according to historical statistics and design constraints. Human designers are then able to refine and evaluate these AI-generated design concepts based on their expertise and intuition.

In this case, the algorithms can be formulated as search and optimization problems, where the goal is to find the optimal design concept that meets certain criteria or constraints, such as cost, performance, or manufacturability. This can be represented mathematically as:

$$maximize: \quad F(x) = w_1 f_1(x) + w_2 f_2(x) + \cdots + w_n f_n(x)$$

$$subject\ to: \quad g_i(x) \leq 0, i = 1,2, \ldots, n$$

Where $F(x)$ is the objective function to be maximized, x represents the design variables, $f_i(x)$ are the individual performance metrics, w_i are the weight parameters that determine the relative importance of the performance metrics, and $g_i(x)$ are the constraints that the design must satisfy.

Development teams are able to explore a huge design space and discover innovative design concepts that meet criteria and constraints by using AI tools that use search and optimization methods including genetic algorithms and particle swarm optimization. The resulting AI-generated design concepts can be visualized and then provided to human designers that use graphical representations or interactive user interfaces, facilitating collaboration between human and AI components in the design process. Moreover, the application of cognitive science and AI to the product development process also requires understanding the psychological facets of human-AI interaction. This knowledge can improve the interaction between human designers and AI tools, leading to a far more successful and efficient product development process.

AI systems should be developed to be explainable, transparent, and trustworthy to allow seamless human-AI interaction. Explainability refers to the capability of a system to give understandable and clear explanations for its

actions, choices, and suggestions. Incorporating interpretable machine learning models or offering visualization tools that allow users to explore and comprehend the decision-making processes of an AI system, might be the solution to achieve this goal.

On the flip side, transparency refers to making the system's operations, goals, and limits clear as well as understandable to users. This may be accomplished by supplying documentation, user guides, and training materials which clearly outline the system's capabilities, constraints, and potential biases. Transparency allows professionals to fully grasp the AI system's capabilities and make better choices regarding its application in their jobs.

Trustworthiness is also an important part of human-AI interaction because it determines the users' will to trust and accept AI system recommendations and outputs. These systems should be robust, reliable, and fair to build trust. The robustness of a system is its capability to create consistent and accurate results despite noise, uncertainty, or adversarial inputs. The system's reliability is the term for its ability to perform consistently over time and across tasks, while fairness describes the equitable treatment of different user groups or stakeholders from the system, avoiding biased or discriminatory results. Additionally, AI tools may be developed to complement and boost human creativity and problem-solving abilities by including features like serendipitous discovery, analogical reasoning, and inspiration. These systems are able to produce and provide unexpected and diverse design concepts, materials, or technologies which can stimulate human designers and also spark creative thinking. Analogical reasoning could be facilitated by algorithms finding and retrieving design patterns, case studies, or relevant examples from different domains, allowing human designers to get insights and make connections across different fields. Serendipitous discovery could be supported by tools able to uncover patterns, trends, or hidden relationships in data, resulting in unpredicted insights and breakthroughs in the product development process.

3.2. Association between Cognitive Activities and AI branches

This section considers the correlations between cognitive activities involved in product development and different branches of artificial intelligence. By analysing these associations, we can determine which AI techniques and approaches are most effective in improving and supporting each cognitive activity.

3.2.1. Identification

Identification refers to recognizing and comprehending the fundamental features and relationships within a problem space, like customer needs, market trends, or technological constraints. AI branches including ML algorithms (supervised and unsupervised learning techniques) can be used to analyses unstructured and structured data to support cognitive activity by identifying patterns and relationships in customer feedback, sales reports and industry metrics. This information can offer valuable insights into customer preferences and market trends. NLP methods can process and analyze textual data from sources like customer reviews, social media feeds, and market research reports. These algorithms can assist in pinpointing customer needs and market opportunities by extracting relevant data and identifying recurring themes or sentiments. Image data from product prototypes, market analysis, or competitor analysis can be processed and analyzed using computer vision techniques to determine design features or potential areas for improvement. This information can help developers refine their designs and better deal with customer needs.

3.2.2. Structuring

Structuring entails organizing and categorizing the identified elements and relationships into a meaningful and coherent structure, such as defining product requirements, design specifications, or project plans. Expert systems, especially decision support systems, are included in a branch that can boost this cognitive activity by providing assistance, recommendations, and best practices based on domain-specific rules, heuristics, or knowledge bases to product developers. Neural networks are the foundation for deep learning algorithms, to process and analyse complex, high-dimensional data, like product feature sets or design parameters, and instantly produce meaningful

structures, like hierarchical representations or semantic embeddings, which may be used to inform design decisions and project planning.

3.2.3. Control

Control refers to the management and coordination of the various tasks, resources, and procedures involved in the product development process, such as project scheduling, resource allocation, or risk management. AI branches that could help support this cognitive activity include evolutionary computation techniques, like genetic algorithms, which can optimize project schedules, resource allocations, or design parameters, taking into consideration many objectives and constraints, and providing solutions that balance the trade-offs between competing goals. Other planning techniques, including scheduling algorithms and game - playing methods, can be utilized to create and evaluate alternative project plans taking into account time, cost, risk, and functionality, assisting developers in controlling different factors of the product development process.

3.2.4. Development

Development involves the creation and improvement of designs, concepts, and ideas, along with the testing and validation of prototypes and solutions. Branches that will support this activity include robotics techniques, like intelligent control and autonomous exploration algorithms, which can be employed to design and develop innovative products, e.g. robotic systems, autonomous vehicles, or smart devices, by providing advanced control strategies and learning abilities that can adapt to brand new information and environments. Large language models like GPT 4 can easily be implemented to generate novel ideas, concepts or design narratives based on historical data and contextual information, while also offering inspiration and creative suggestions to product developers during development. Machine learning methods, including reinforcement learning and deep learning algorithms, can be utilized to create and improve models that learn from information, feedback, or interactions with the environment to generate complex systems like recommendation engines, predictive models, or intelligent agents.

3.2.5. Communication

Communication is a vital element of the product development process, since it involves the exchange of feedback, ideas, and information among staff, stakeholders, and customers. AI types that can help support this cognitive activity include: NLP methods to analyse and process textual communication, like emails, reports, or meeting notes, and instantly produce summaries, highlights, or action items, helping to enhance the efficiency and effectiveness of communication inside the team; using speech processing techniques to create and implement voice-based interfaces like voice assistants, chatbots, or teleconferencing systems to improve the user experience and facilitate interaction among and with people involved in the project; in the end computer vision methods can be implemented to create and put into action visual communication tools, such as augmented reality (AR) systems, virtual prototyping environments, or interactive design platforms, that could assist developers to better communicate their ideas, strategies, and designs to team members, as well as to stakeholders.

The correlation between cognitive activities and AI branches throughout product development proves the potential of AI to enrich and complement human cognition in the different phases of product development.

The AI tools and methods mentioned in this chapter are not exhaustive, and brand-new solutions and techniques will certainly emerge as the state of the art in artificial intelligence continues to progress, allowing product development teams to take on the various complex challenges and opportunities they encounter.

3.3. Examples of Commercial Software

In this section, we discuss several examples of commercial software that leverage artificial intelligence to support the design and innovation processes in product development. These tools showcase the practical applications of AI

technologies, providing valuable insights into how AI can enhance efficiency, creativity, and competitiveness in the industry.

3.3.1. MarketSim

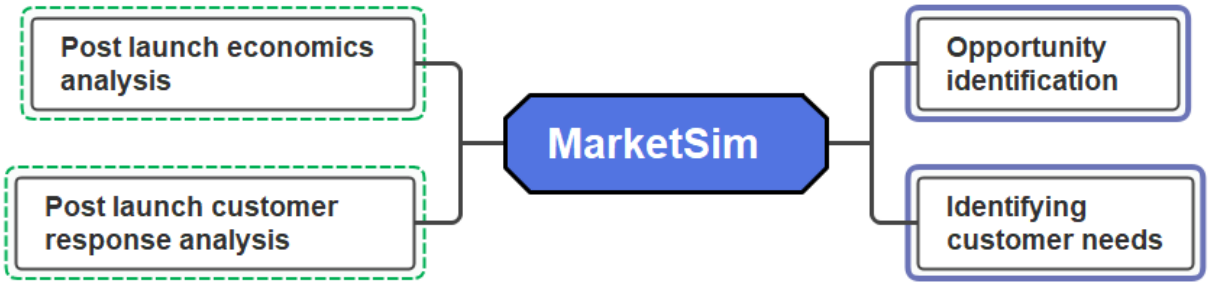

Figure 6: Schema depicting the link between NPD and the software

MarketSim is a sophisticated marketing simulation program that uses artificial intelligence to allow development teams to forecast the potential market success of their latest products accurately. MarketSim leverages machine learning methods such as clustering, regression analysis and time series forecasting to examine historical sales data, consumer preferences and competitive landscapes. Product developers are able to improve their market success by making data-driven decisions regarding product features, pricing strategies and market positioning. In this particular section, we examine the way MarketSim works, its different components as well as the additional value it offers to product development.

- **Data Collection and Pre-processing**: MarketSim needs a great amount of data regarding product sales, consumer behavior and market trends to make precise market predictions. It gathers information from a variety of sources such as point - of - sale systems, customer surveys, online reviews and social media. When the data is collected, it goes through pre - processing steps to handle missing values, eliminate outliers, and normalize the information, ensuring it is ideal for the subsequent machine learning algorithms.
- **Feature Engineering and Selection**: MarketSim makes use of feature engineering to develop new features that better reflect the underlying market patterns after pre-processing the information. This entails methods including data aggregation, transformation, and dimensionality reduction. MarketSim takes advantage of feature selection algorithms, like recursive feature elimination and feature importance ranking, to figure out the most crucial features for market success prediction. while minimizing the computational complexity of machine learning models.
- **Clustering and Segmentation**: The software utilizes clustering algorithms, including k - means and hierarchical clustering, to find patterns and segment the market data into distinct groups. These groups usually represent different consumer segments, product categories, or geographical regions, each with their preferences, needs, and purchasing habits. MarketSim enables targeted predictions and recommendations for each group by segmenting the current market, enabling development teams to adapt products and marketing strategies to specific consumer segments.
- **Regression Analysis and Time Series Forecasting**: It also uses regression analysis and time series forecasting techniques to forecast market demand and price sensitivity. The relationships between a variety of independent variables, like product features and marketing initiatives, along with dependent variables, like product sales and market share, are estimated using regression models like linear regression, logistic regression and support vector regression. The temporal patterns in sales data are analysed as well as predicted using time series forecasting models like autoregressive integrated moving average (ARIMA) and Long Short-Term Memory (LSTM) neural networks. By combining regression analysis and time series forecasting, it can produce precise predictions of market demand and price

sensitivity for different product configurations and marketing scenarios. These predictions enable product developers to make educated choices regarding marketing efforts, pricing strategies, and product features, ultimately maximizing the potential market success of the latest products.

- **Validation and Performance Evaluation**: The software employs many validation and performance evaluation methods to guarantee the validity and reliability of its predictions. Estimating the performance of machine learning models on unseen data is accomplished through cross-validation (k-fold cross-validation and leave-one-out cross-validation). On the other side, the accuracy, robustness and generalizability of the models are measured using performance metrics like mean squared error, R-squared and precision-recall curves. By continuously evaluating and fine tuning its models, MarketSim ensures that its predictions and recommendations stay reliable and relevant in the face of changing market conditions and consumer preferences.

MarketSim has indeed the capability to conduct scenario analysis and also provide decision support to teams - which is one of its primary features. By simulating different product configurations, pricing strategies, as well as marketing efforts, it is able to calculate their possible effect on market demand, revenue, and profitability. These insights could be utilized by development teams to compare and contrast various scenarios, determining the most promising strategies, and making better choices regarding their products and marketing initiatives. Its decision support capabilities go far beyond straightforward scenario evaluation. since it may also incorporate optimization algorithms, like linear programming and multi-objective optimization, to identify the best combination of product features, prices, and marketing efforts that maximize specific goals, like revenue, market share, or profit margin. Product developers are able to analyze the trade-offs between various goals and make choices in sync with their overall business objectives and priorities. In order to make its predictions and recommendations much more accessible and actionable for development teams, MarketSim utilizes advanced visualization and reporting methods. The analyses and simulations are provided via graphs, charts, and interactive dashboards which enable users to easily examine the data, quickly determine trends, in addition to gain insight into the market potential of their products or services. Customizable reports can be generated to summarize the key findings and recommendations, providing a concise and clear overview of the market landscape in addition to the possible effect of their choices.

The output of the software may eventually be used in conjunction with other development tools, like CAD software, product lifecycle management (PLM) systems, and project management tools, to simplify product development and enhance staff collaboration, leading to more productive and profitable products, by hooking market insights as well as predictions straight to the design, development, and control phases of the product development process.

3.3.2. IBM Watson Studio

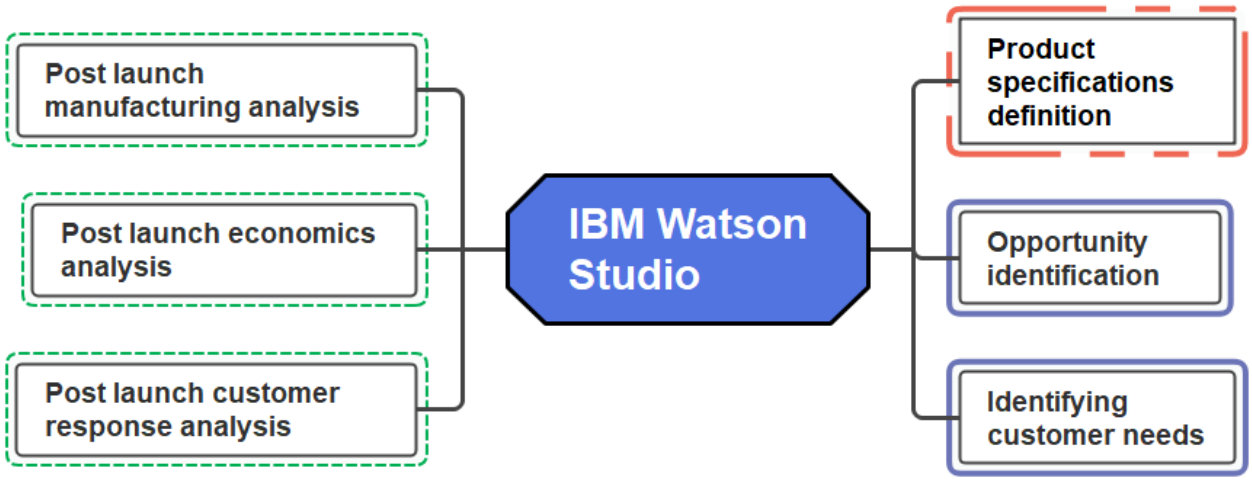

Figure 7: Schema depicting the link between NPD and the software

IBM Watson Studio represents a comprehensive, AI-driven data science and machine learning platform that empowers product development teams to efficiently create, train, and deploy machine learning models. This

platform boasts an array of pre-built machine learning algorithms, spanning deep learning, unsupervised learning, and reinforcement learning, which product developers can harness to embed AI-driven features into their products, such as predictive maintenance, demand forecasting, and personalized user experiences.

Watson Studio's collaborative environment facilitates seamless teamwork among data scientists, engineers, and other stakeholders involved in the product development process. This environment allows users to share data, models, and code, streamlining the workflow and fostering effective communication. Watson Studio also supports integration with popular development tools and platforms, such as Jupyter Notebooks, GitHub, and Apache Spark, ensuring compatibility with existing workflows.

Data preparation and *exploration* are crucial steps in developing machine learning models. Watson Studio provides robust tools for data cleaning, transformation, and visualization, enabling product development teams to efficiently prepare datasets for machine learning. Additionally, the platform's data exploration features support the *identification of patterns*, trends, and relationships within the data, which can inform feature engineering and model selection. The platform offers a wide range of pre-built machine learning algorithms, covering supervised learning (e.g., regression, classification), unsupervised learning (e.g., clustering, dimensionality reduction), and reinforcement learning. Users can leverage these algorithms to build models tailored to their specific product development needs. Watson Studio also supports deep learning frameworks, such as TensorFlow and PyTorch, facilitating the development of more advanced models, such as neural networks. Model training in Watson Studio is streamlined through its AutoAI feature, which automates the process of model selection, hyperparameter tuning, and feature engineering. AutoAI evaluates various model configurations and determines the optimal settings based on user-defined performance metrics. This automation not only accelerates the model training process but also enables non-expert users to harness the power of machine learning.

After training a machine learning model, it is essential to evaluate and validate its performance before deploying it in a product. Watson Studio provides comprehensive tools for model evaluation, such as confusion matrices, ROC curves, and precision-recall curves, which help users assess the model's accuracy, sensitivity, and specificity. The platform also supports cross-validation and train-test splitting techniques, ensuring that the model's performance is evaluated rigorously and reliably. Once a model is trained and validated, it can be deployed in a product or service. Watson Studio simplifies the deployment process by offering seamless integration with IBM Cloud services and other cloud platforms, such as AWS and Microsoft Azure. This enables product development teams to scale their models quickly and efficiently, meeting the demands of their target audience.

Moreover, Watson Studio supports real-time monitoring of deployed models, providing insights into their performance and facilitating ongoing optimization. Users can track various performance metrics, identify potential issues, and update models as needed, ensuring that their products remain responsive, accurate, and reliable.
As AI-driven features become more prevalent in products, it is increasingly important to provide transparency and explainability in machine learning models. Watson Studio incorporates explainable AI features, enabling users to understand the factors that influence a model's predictions and decisions. This can help product development teams build trust with their customers and comply with relevant regulations and guidelines.
Watson Studio is part of the broader IBM Watson ecosystem, which offers a suite of AI-driven services and APIs that product development teams can leverage to enhance their products. These services include natural language understanding, speech-to-text, text-to-speech, visual recognition, and language translation, among others. By integrating these services into their products, developers can create more advanced and intelligent features, improving the overall user experience.

IBM Watson Studio's capabilities can be applied to various aspects of the product development process, including:

- **Demand forecasting:** By leveraging machine learning algorithms, product development teams can create models to predict customer demand, allowing for more informed inventory management and production planning decisions.
- **Quality control:** Using computer vision and image recognition algorithms, teams can develop systems to automatically inspect products and identify defects, improving overall product quality and reducing manual inspection efforts.

- **Predictive maintenance**: Through the application of machine learning models, product development teams can predict the likelihood of equipment failure, enabling more efficient maintenance scheduling and reducing downtime.
- **Personalized user experiences**: By incorporating recommendation algorithms and natural language processing, teams can create products that adapt to individual user preferences and needs, fostering customer satisfaction and loyalty.

3.3.3. Hootsuite Insights

Figure 8: Schema depicting the link between NPD and the software

Hootsuite insights is an AI powered social media analytics application which allows development teams to obtain essential Insights into consumer sentiment, personal preferences, and behaviour. This comprehensive solution offers a significant resource for product designers and marketers, helping them to make educated choices regarding marketing strategies, design improvements and product features, ultimately leading to better market fit and consumer satisfaction.

It leverages artificial intelligence technology, including natural language processing (NLP) and machine learning algorithms, to analyse vast volumes of social media information. The platform is able to process and then interpret unstructured data, including text, pictures, and also videos, instantly, obtaining beneficial insights from conversations and interactions across social networking platforms.

NLP is a key element of Hootsuite Insights' analytics capabilities. By employing innovative NLP methods, the platform can accurately process, analyse, and comprehend the huge quantities of text information produced by social media users generated content, identifying phrases and keywords, detecting emotions and sentiment, and obtaining relevant preferences and opinions. Among the primary NLP methods employed there is sentiment analysis, which entails determining the sentiment expressed in a piece of text, such as positive, negative, or neutral. Sentiment analysis algorithms employ machine learning models trained on huge datasets of labelled text data, to recognize and classify sentiment patterns in new, unlabelled data. Hootsuite Insights can provide development teams with a snapshot of consumer sentiment toward their products, competitors, and industry trends.

The platform is able to offer accurate and pertinent information to developers by training these algorithms to recognize relationships, trends, and patterns in the data. For instance, Hootsuite Insights makes use of clustering algorithms, including K - means and hierarchical clustering, to group similar social networks content together. The identification of common themes and topics of conversation amongst consumers can offer meaningful insights to their personal preferences, desires, and pain points for development teams. Furthermore, social media audience members could be segmented based upon behaviour and interests using clustering algorithms, enabling much more accurate communication and marketing tactics.

By using the potential of AI-driven social media analytics, Hootsuite Insights provides several tools for product development teams, helping them to make much more informed choices and attain much better outcomes:

- **Product Feature and Design Decisions**: analysing consumer opinion and comments on social media can provide teams with invaluable insights to the strengths and weaknesses of the products. By identifying areas where consumers are encountering difficulties or expressing dissatisfaction, teams can prioritize

design enhancements and feature improvements which directly deal with these problems. Furthermore, teams can determine possible areas for differentiation and development by using discussions about competitors' products.

- **Market Research and Trend Analysis**: Hootsuite Insights can also be employed as a highly effective market analysis tool, allowing teams to keep a finger on the pulse of industry trends and consumer preferences. By monitoring conversations and analysing the feedback for emerging patterns and trends, teams can determine potential market opportunities and stay ahead of the curve which is particularly helpful in fast moving industries where preferences and trends are continuously changing.
- **Marketing Strategy and Campaign Optimization**: by utilizing AI - driven analytics, teams can gain priceless insights into the effectiveness of their promotional approaches and campaigns. Being able to examine customer engagement, perception, and reaction to advertising content it gets much easier to decide which messages resonate with their intended target audience and regulate their campaigns accordingly enhancing marketing effectiveness, improving conversion rates, and eventually increasing product sales.
- **Consumer Profiling and Segmentation**: Hootsuite Insights may also be employed to create comprehensive consumer profiles and segments according to their online behaviour and interests. The platform identifies customer segments with differing preferences, desires and purchasing habits by analysing the content they share and their interactions with different brands which has become essential for maximizing the effect of communication plans, marketing, and product design.

3.3.4. Crayon

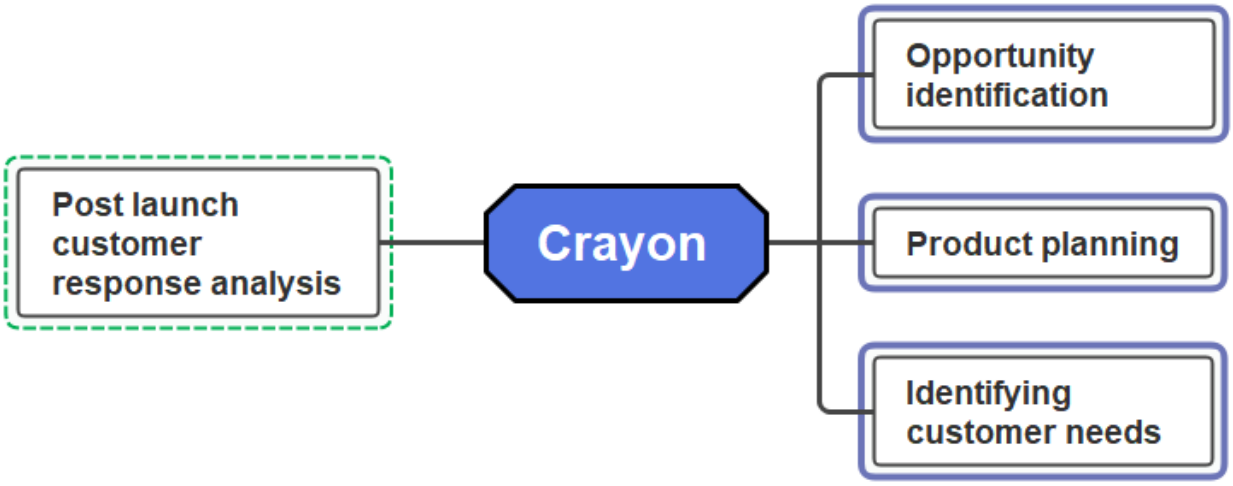

Figure 9: Schema depicting the link between NPD and the software

Crayon is an AI-driven competitive intelligence platform that enables product development teams to gain actionable insights into their competitive landscape. By employing a range of artificial intelligence techniques, including machine learning algorithms and natural language processing, Crayon gathers, analyses, and synthesizes data from a multitude of sources. This comprehensive approach provides product development teams with a wealth of information about their competitors, industries, and market trends, ultimately informing strategic decisions about product positioning, feature prioritization, and innovation efforts.

The Crayon platform leverages several branches of artificial intelligence from the outline to provide an effective competitive analysis. For instance, it employs machine learning (2.1) techniques to process vast amounts of data from various sources. This includes websites, news articles, social media, and product reviews. The machine learning models are trained to identify patterns, trends, and insights that can be valuable for product development teams. Furthermore, it utilizes natural language processing (2.8) to understand and interpret the textual data it gathers which involves breaking down complex sentences, identifying keywords, and understanding the context in which the information is presented. By doing so, Crayon can accurately analyse and categorize the content, providing

development teams with a structured and easy-to-understand view of the competitive landscape. Its AI capabilities are further enhanced by computer vision (2.4) techniques, which allow the platform to analyse visual content such as images, videos, and graphical elements. Computer vision algorithms also enable Crayon to recognize objects, logos, and other relevant visual features within the content, providing an additional layer of information that can be useful for product development teams.

The competitive intelligence gathered by Crayon can also be used to inform the application of other AI branches implemented in the product development process: for example, the insights gained from Crayon's analysis can be used to guide the development of expert systems (2.6) that assist in decision-making processes which involves creating decision support systems that provide recommendations based on the analysed competitive data, or designing teaching systems that help product development teams learn from the successes and failures of their competitors; similarly, the information obtained by Crayon can be leveraged to improve planning (2.9) activities within the product development process.

By understanding the competitive landscape, development teams can better anticipate market trends, adjust their strategies accordingly, and allocate resources more effectively adjusting the scheduling of product releases to coincide with market opportunities or optimizing game-playing strategies to gain a competitive advantage. Moreover, competitive insights can serve as valuable input for large language models (2.10) that can use the analysed data to generate creative ideas, draft product descriptions, and even automate documentation tasks, ensuring that their output is relevant, up-to-date, and informed by the latest market trends and competitor activities.

The integration of Crayon's AI-driven competitive intelligence into the development process offers numerous benefits, like making more informed decisions about product features, design improvements, and marketing strategies, leading to better market fit and consumer satisfaction; providing a comprehensive and structured view of the competitive landscape, allowing teams to identify opportunities and threats that may have otherwise gone unnoticed and eventually promoting a data-driven approach to product development, fostering a culture of innovation and continuous improvement.

3.3.5. Brandwatch

Figure 10: Schema depicting the link between NPD and the software

Brandwatch is a social listening and analytics platform that plays a crucial role in supporting product development teams as they monitor and analyse online conversations surrounding their products, brands, and competitors. The platform employs a combination of natural language processing (NLP) techniques, sentiment analysis algorithms, and machine learning models, which work together to offer valuable insights into consumer preferences and emerging market opportunities.
The application of NLP techniques allows for the efficient processing and understanding of vast amounts of unstructured text data derived from various online sources such as social media, forums, and blogs. NLP techniques are employed to perform tasks such as tokenization, part-of-speech tagging, and named entity recognition, which enable the platform to identify and categorize relevant information based on user-defined criteria. This is essential for understanding the context of online conversations and accurately capturing consumer sentiment.
Sentiment analysis is another key component that enables product developers to gauge consumer feelings: they are often based on supervised learning techniques, and they are trained on labelled datasets to classify textual data as

positive, negative, or neutral. These algorithms can identify and interpret various forms of sentiment expression, such as emojis, slang, and colloquial language, providing a more accurate and comprehensive representation. Machine learning models facilitates the job: for example, by applying clustering algorithms, like k-means and hierarchical clustering, the platform can group similar data points together, enabling for the identification of emerging trends and themes in online conversations. Additionally, anomaly detection algorithms can help identify unusual patterns or sudden changes in consumer sentiment, which may signal potential issues or opportunities that warrant further investigation.

The insights gleaned from AI-driven analytics can significantly inform the decision-making process in product development. For instance, the platform can highlight consumer preferences for specific product features or attributes, providing guidance on design improvements and feature prioritization. Furthermore, the platform can reveal consumers' unmet needs or pain points, which can inspire innovation and the development of new products that address these gaps in the market. Additionally, Brandwatch can be instrumental in informing marketing and communication strategies for product development teams, making them able to craft targeted and effective marketing campaigns that resonate with their target audience, also identifying key influencers and opinion leaders within specific industries or market segments, making it possible to forge strategic partnerships or collaborations that can amplify their message and improve their products' reach.
It is also possible to monitor the direct impact of their efforts and make data-driven adjustments as needed by tracking changes in consumer opinion and online conversations after the launch of a new product or marketing campaign, allowing for the assessment of the effectiveness of their strategies and for the necessary adjustments to fix and optimize their results. In terms of competitive intelligence, the tool can provide valuable insights into the strategies and tactics employed by competitors, helping teams to stay ahead in the marketplace by monitoring and analysing competitors' online conversations, product launches, and marketing campaigns and identifying potential threats and opportunities.

As AI technologies continue to advance, Brandwatch and similar platforms are expected to incorporate even more sophisticated techniques to enhance their capabilities. For example, the integration of deep learning algorithms for NLP tasks, such as named entity recognition, can improve the platform's ability to understand and interpret complex language structures and nuances. Additionally, reinforcement learning techniques can be employed to optimize the platform's data processing and analysis strategies, making it more efficient and accurate in generating actionable insights.

3.3.6. Tara AI

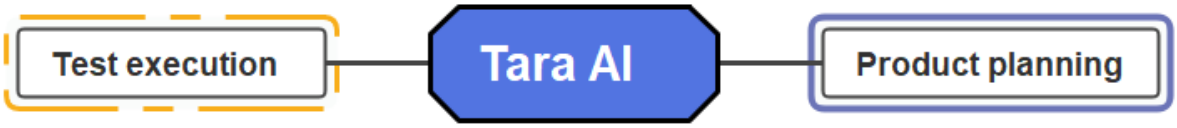

Figure 11: Schema depicting the link between NPD and the software

Tara AI is a platform designed to assist with project management and product development processes. It uses artificial intelligence to predict the time and resources required for a project, making it easier for teams to plan and execute their work.
It does this by analyzing past projects and the work patterns of the team. It takes into account factors like the complexity of tasks, the skills of team members, and the time taken for similar tasks in the past. This allows Tara AI to provide a more accurate estimate of the time and resources required for a project.

The uniqueness of team routines is not just understood but also leveraged by Tara AI. It uses this information to optimize project planning and execution. For instance, if a team works faster on certain types of tasks, Tara AI will take this into account when assigning tasks and predicting project timelines.

Based on this analysis of past performances, it can also suggest which team member is best suited for a particular task. This not only helps in assigning the right person to the right task but also in balancing the workload among team members.

Tara AI is also capable of predicting potential issues that might arise during the course of a project. It does this by analyzing historical project data and identifying patterns that have previously led to problems.
For example, if the AI notices that a particular type of task often takes longer than estimated, it might flag this as a potential risk for future projects. Similarly, if it identifies that certain combinations of tasks or team members often lead to bottlenecks, it can highlight these as potential issues.

Once potential issues are identified, Tara AI can suggest solutions. These might include adjusting the project timeline, reassigning tasks, or providing additional resources. The goal is to proactively address potential problems before they impact the project, thereby improving project outcomes and efficiency.

It uses machine learning algorithms to analyze past data and predict future outcomes. This includes predicting the time required for tasks, identifying potential bottlenecks in the project, and suggesting optimal task assignments. The AI is also capable of learning and improving over time, making the predictions more accurate as more data is collected

3.3.7. Tableau

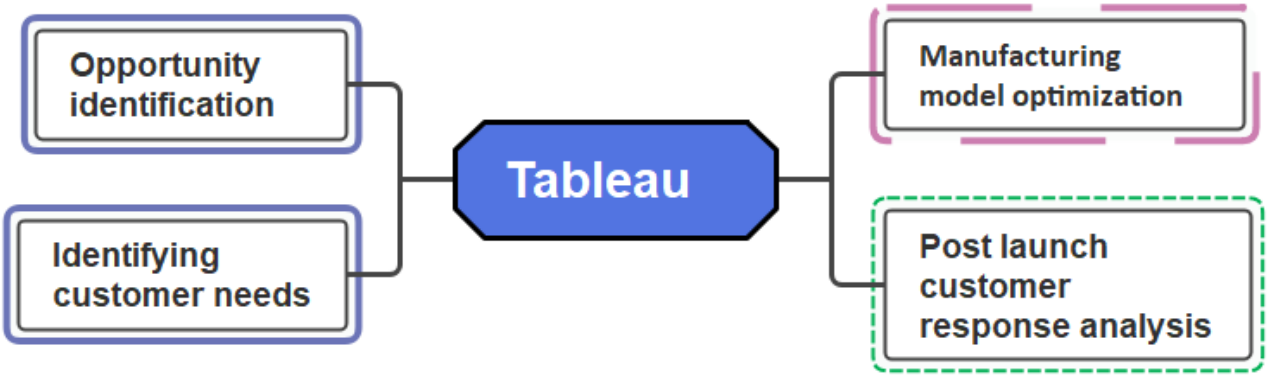

Figure 12: Schema depicting the link between NPD and the software

Tableau is a ground-breaking data visualization and analytics platform that has changed the way development teams process, evaluate, and draw insights from intricate datasets. It combines the power of different types of artificial intelligence (AI) to deal with the challenges of big data in product development, which includes figuring out market opportunities, optimizing product designs and streamlining the decision-making process. This section considers Tableau's capabilities and their impact on product development at different phases.

It is crucial to understand the key AI technologies that Tableau employs to augment its data visualization and analytics capabilities which include machine learning, pattern recognition, clustering algorithms, and anomaly detection. That allows to obtain valuable insights from data, leading to much more informed decisions and enhanced overall product outcomes. Machine learning, a subfield of AI, entails developing algorithms that can learn from and make predictions or decisions based on information. Its algorithms are essential in automating data analysis tasks in Tableau, like data segmentation, classification, and prediction allowing teams to focus on more strategic and creative aspects of the work, while the machine learning algorithms handle the tedious, time-consuming tasks. For instance, in the context of product development, a typical challenge is to predict the demand for a brand-new product based on past product sales information, market trends, and customer preferences. Product developers can make use of Tableau's machine learning capabilities to develop predictive models to examine these factors and produce accurate demand forecasts to make better choices regarding product attributes, pricing, and market positioning.

Another technique utilized in Tableau is clustering algorithms, which group related data points based on their features, uncovering trends, relationships and insights hidden in product development data taking into consideration

also customer feedback, product usage patterns and market research information. Clustering algorithms is also utilized to segment customers based on their personal preferences to recognize target customer segments and develop solutions that meet their customers 'needs and expectations. By understanding these customer preferences, development teams can deliver products that resonate better with their target audience, to reach higher customer satisfaction and improve market success.

Pattern recognition is another AI method used in Tableau and it consists of determining recurring patterns, relationships, or structures within data. Pattern recognition algorithms enable development teams to uncover hidden trends, dependencies, and correlations within their data to prioritize the most crucial features for their products by determining associations between product features and customer satisfaction. Additionally, it can be utilized to investigate the correlation between product usage patterns and customer retention, enabling teams to recognize areas for improvement and modify their products accordingly.

Tableau also makes use of anomaly detection, a feature that identifies data points which significantly deviate from expected patterns or trends. allowing teams to quickly spot and check out unexpected spikes or drops in sales, product performance issues or customer complaints. For instance, anomaly detection can be used to flag potential issues with a product's manufacturing process, like an increase in the amount of defective units making it possible to correct problems early on by modifying the manufacturing process or even conducting extra quality control checks to make sure the final product meets desired quality standards. Moreover, it can also determine unexpected trends in customer feedback, enabling to address any emerging concerns or cash in on new opportunities.

The incorporation of these techniques offers a comprehensive solution for dealing with and analysing complex datasets. Tableau combines machine learning, clustering algorithms, pattern recognition, and anomaly detection to produce a robust platform for obtaining useful insights from data, enabling product design and development optimization. Furthermore, its user-friendly interface along with its extensive visualization capabilities allows development teams to quickly examine and interact with their data, fostering a more collaborative and data-driven approach to product development. The data-driven approach not only enhances the overall efficiency of product development but also promotes innovation and creativity within the team.

3.3.8. Synera

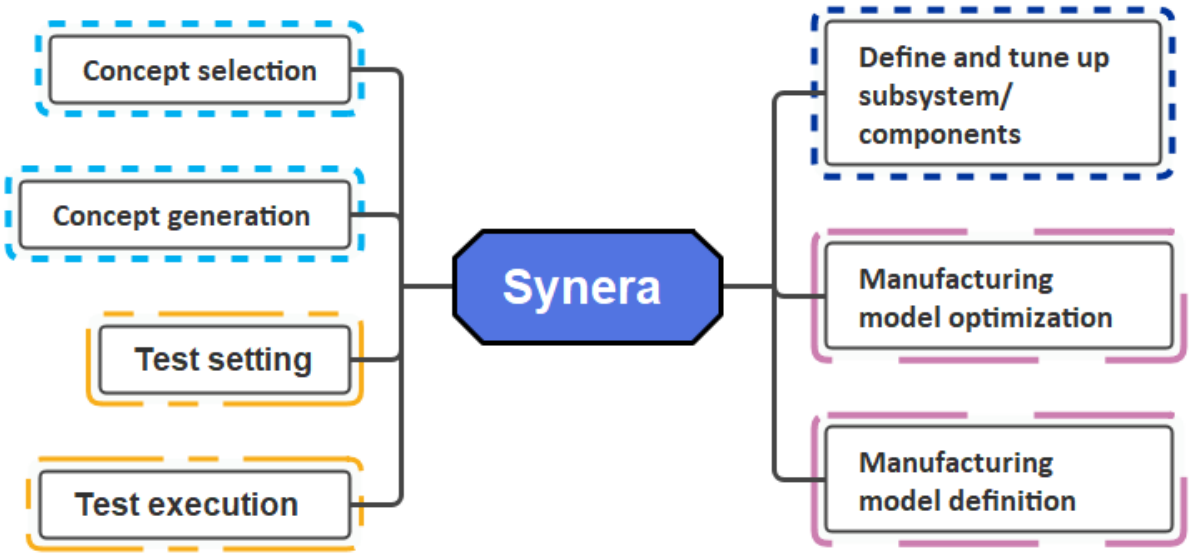

Figure 13: Schema depicting the link between NPD and the software

Synera, a state-of-the-art decision support tool, was created to help product development teams in optimizing their design options by analysing several criteria concurrently. Its AI-driven approach enables product designers to look into a wide range of design alternatives efficiently and rapidly, leading to well-informed decisions and better product quality. This section explores the internal workings of Synera, the tools it uses, as well as the advantages it brings to product development.

Among the primary components of Synera is its use of multi-objective optimization methods to find the best trade-offs between conflicting design goals, like cost, weight, and performance. Multi-objective optimization is a mathematical process which strives to identify the optimal solution to a problem with many contradictory goals. This optimal solution is obtained through a technique called Topology Optimization, which determines, through a series of iterations, the best solution to the structural problem faced. Mathematically a structural problem can be modelled considering the following elements:

- Volume constraints: they allow the definition of the points in the three-dimensional geometric space that should not be involved in the optimisation process.
- Boundary conditions: they specify the value or the derivate of optimisation variables, like displacement or stress.
- Manufacturing constraints they reflect the limitations and the possible outcomes associated with a particular manufacturing process.
- Equilibrium equations: A system is said to be in equilibrium when the sum of the forces and the sum of the moments acting on it are both zero, meaning the system is neither translating nor rotating.
- Objective functions: the objective refers to the function that is being maximized or minimized.

Problems of this type are therefore governed by a system of partial differential equations, characterised by numerous local solutions. This type of systems can be solved in many ways and one of them is the class of gradient-based optimization algorithms, where the sensitivity of the objective function to changes in the design variables is calculated and used to update the design. The process is typically iterative and continues until some stopping criterion is met.

The class of evolutionary algorithms, known as genetic algorithms (GAs), is another solution based upon the natural selection procedure. In Pareto-based genetic algorithms, a population of possible solutions is evolved over several generations to enhance their fitness concerning several objectives. The algorithm begins with an initial population of randomly generated solutions, and each solution is analysed based upon its fitness for each objective.

GAs use genetic operators as selection, crossover (recombination) and mutation to produce new offspring solutions throughout the evolutionary process. The selection operator favours solutions having much better fitness values, while crossover and mutation operators bring genetic diversity into the population by merging and altering the solutions, respectively.

For Pareto based GAs, the concept of Pareto dominance is employed to compare and rank solutions in the population. A solution is said to dominate another if it's better or equivalent in most objectives and purely better in more than one objective. The Pareto front is a set of non-dominated options that provide the most effective compromises between opposing goals.

The AI driven strategy of Synera is meant to help in different phases of product development, which includes concept development, detailed design, testing and refinement. Synera allows development teams to make better choices based on a holistic awareness of the trade-offs between several design goals by using the strength of multi-objective optimization methods.

The two main objectives are the minimisation of mass and average body tension (or maximization of stiffness) and are governed by three main parameters:

- Strut density: when set to high values, the final model is characterised by a robust structure.
- Global safety factor: It defines the average voltage level considered acceptable for the whole body (it is not a constraint on a maximum tension value relative to a particular surface).
- Shape quality: low values of this parameter lead to models with very discontinuous.

An additional parameter is then considered that defines the degree of complexity of the model as the number of nodes/fine elements that will compose it. This last one takes in consideration the hardware limits of the computer used to activate the optimisation process.

In summary, the entire basic process of producing a single topology optimized model can thus be reduced to the basic steps expressed by the following diagram:

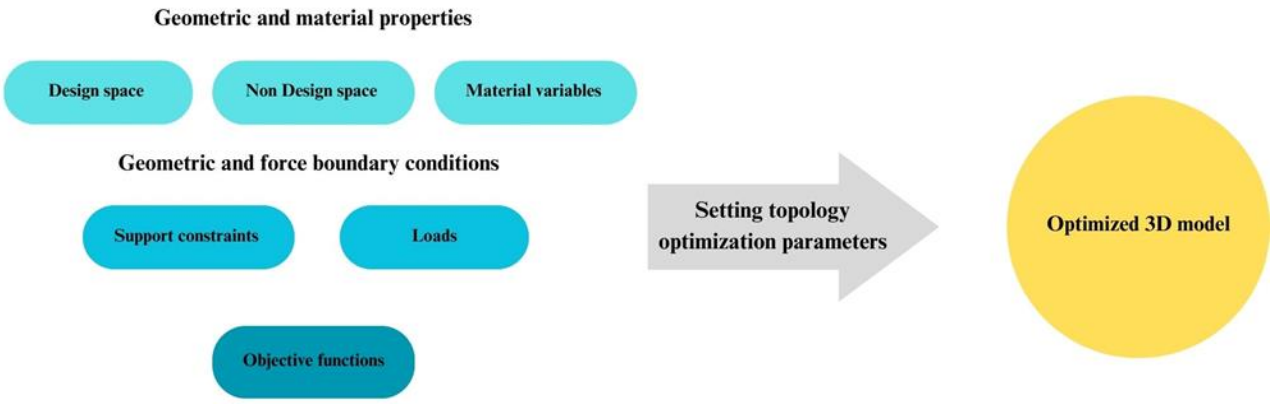

Figure 14: Topology optimized model basic steps

Product designers generate and then evaluate several design concepts during the concept development stage based upon feasibility, desirability, and viability. Synera aids in this process by offering a quantitative framework for comparing various design alternatives based on their overall performance concerning several objectives. Product designers are able to identify promising design concepts which achieve a sense of balance between objectives, like minimizing cost while maximizing performance and dependability.

Taking in consideration the minimization of the cost, the choice of the manufacturing process takes an important role in defining the planned cost of the product. Features implemented by Synera give the user the possibility of implement a dynamic cost model, based on the following inputs:

- 3D geometrical model: this model can be created using topology optimizations feature.
- Manufacturing process variables: they can be different for each of the three-manufacturing process available (milling, casting, LBM).
- Material and personnel cost: in case of topology optimized models, only the last type of costs must be considered.

The next step consists in enabling the manufacturing process optimization, and therefore tuning up the manufacturing relevant variables. General manufacturing optimization processes require the definition of a set process variables and at least one response parameter. Then it is possible to generate combinations of manufacturing variables upon which the response parameters are calculated. There are three main algorithms used to generate all the combinations: Latin Hypercube Sampling, Grid Search, Random Grid Search.
The choice of the generative algorithm affects the minimal number of combinations required to identify the optimal one and therefore the time necessary to fine tuning the manufacturing process.

Product development teams concentrate on defining the general architecture, taking in consideration the interaction (at least from a structural point of view) between multiple previously generated component models in order to create an optimal system. The process entails separating the product into its subsystems and evaluate how design choices can impact the general performance of the system itself. Synera enables system-level design by allowing designers to explore a broad range of architectural alternatives and evaluate their trade-offs in terms of pre,

performance, weight along with other related factors. which can help designers determine optimal system architectures that meet the desired product goals.

Product designers refine the design concepts selected during the detailed design stage and create detailed specifications for every component and subsystem. Synera facilitates this by offering a cost-effective technique for exploring the design space and identifying optimal solutions that meet multiple goals.

Development teams verify and also validate their designs throughout the testing and refinement stage through simulations, prototypes, and pilot production runs. The software is able to help in this stage by making it possible for engineers to do virtual testing and optimization of the designs, therefore determining possible issues and areas for improvement.

It can be utilized for simulation-based optimization studies, where the performance of a design is compared under various operating conditions and loading scenarios. Engineers can utilize the analysis of these simulations to obtain useful insights into the robustness and reliability of their designs, determine possible failure modes and make design modifications to deal with these concerns. The software can utilize machine learning algorithms to evaluate huge amounts of historical product data, identifying patterns and trends, and developing predictive models to guide design selections.
The results and accuracy of virtual testing is determinate also by the choice of the predictive/training model used to bring out the relationship between the design and the forecasting variables. These models include the use of Linear regression, Random Forest, OLS Simple Linear regression and Neural Networks.

All considered, product development departments could reap the benefits of integrating Synera's optimization abilities into the product development process for a variety of reasons, including:

- **Improved Decision-Making**: The quantitative approach to multi-objective optimization developed by Synera helps designers to obtain a comprehensive and clear understanding of the trade-offs among design objectives, allowing them to make educated choices which result in better products.
- **Accelerated Product Development**: The ability to quickly investigate and assess a great variety of design alternatives making use of optimization algorithms can considerably reduce the time spent on design iterations and actual physical tests, ultimately accelerating the product development process.
- **Discovering new design**: optimization methods can help product designers spot novel and revolutionary design solutions that may not be apparent through standard design methods. This can result in the creation of novel products that offer distinct value propositions and competitive advantages on the market.
- **Cost Reduction**: Synera allows product development teams to determine optimum design solutions that meet several goals, leading to lower material costs, manufacturing processes, and physical testing - ultimately leading to more affordable products.

The main disadvantage of Synera, especially considering the financial limits of small studios, is that it requires the access to other third-party software, but on the other hand this characteristic allows other already structured firms to implement the utilization of this software in the product realisation routines sticked to the team itself.

3.3.9. Fusion360

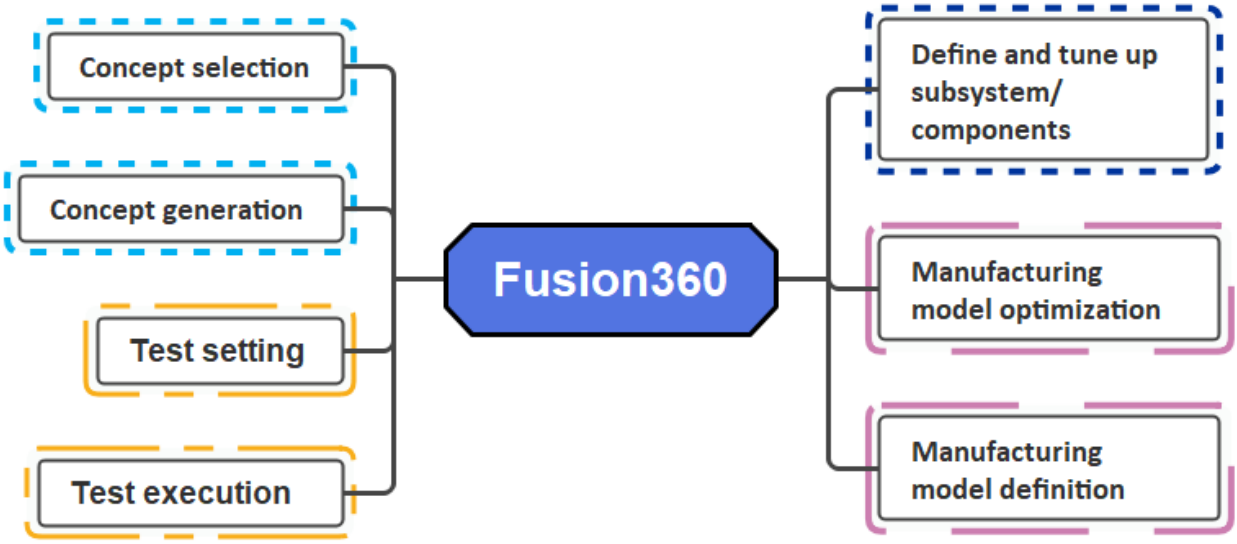

Figure 15: Schema depicting the link between NPD and the software

Fusion360 is a comprehensive cloud-based 3D computer-aided design (CAD), computer-aided manufacturing (CAM), and computer-aided engineering (CAE) platform developed by Autodesk. It offers a wide range of features and tools to support the product development process, including parametric and direct modelling, assembly design, rendering, and simulation. One of the most notable features of Fusion360 is its AI-driven generative design module, which has the potential to drastically reduce the effort associated to the concept exploration. Generative Design is an innovative approach to product design that employs artificial intelligence algorithms to explore and generate numerous design possibilities based on user-defined objectives and constraints. Connecting to what has been expressed to the case of Synera, Generative Design and Topology Optimization are the two main solution families operating in this direction. There is no well-defined dividing line separating them, so much so that they are often used as synonyms, and it seems that Generative Design exploits Topology Optimization itself. One of the possible distinctions between the two is proposed below. Topology Optimization relays on a process of progressive removal of portions of material from a rough model, with the aim of reaching or coming as close as possible to a predefined set of targets/objectives, given a set of constraints. During the execution of Generative Design, on the other hand, material is created from a discrete set of fundamental, well-defined components, such as housing holes for shafts or bolts, again depending on a sequence of constraints and the chosen objective functions.

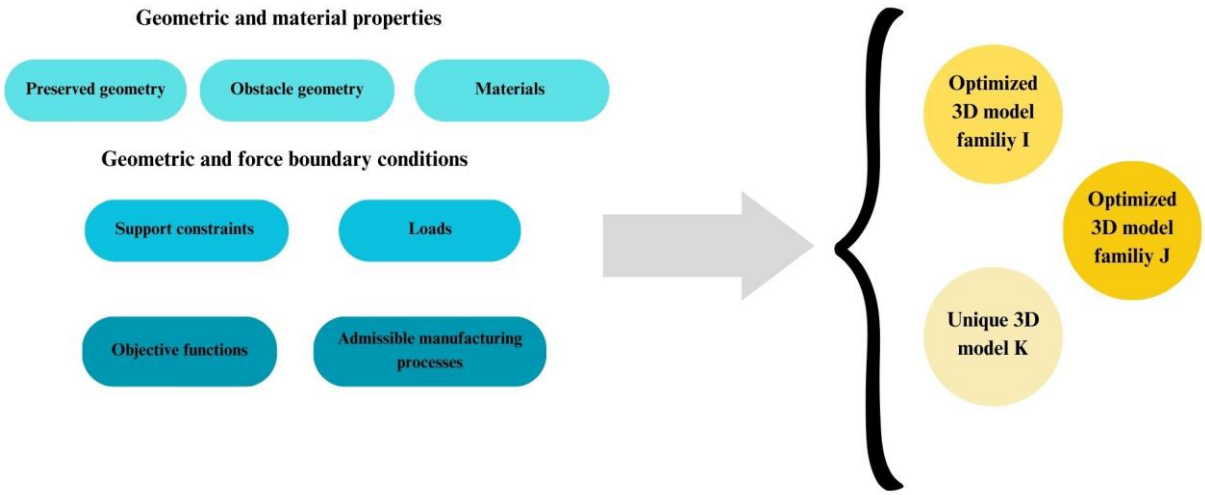

Figure 16: Example of GD and TO

It should also be noted that Fusion360 only allows customization of the target functions by importing a script in Python or C++ via the available API, therefore turns out to be a more complex and less immediate task in comparison to Synera.
A single run of Generative Design produces, unlike Topology Optimization, a multitude of designs already clustered by comparing the individual models generated, thus it can be more useful and immediate in contexts where there is not yet a strong structuring of the product idea.

All of this is possible only by virtue of using cloud computing, whereby the limitations in terms of graphics memory and RAM available to the workstation can be bypassed while requiring on the one hand the consumption of tokens on the other hand often quite long execution times are required.

Conceptually, the Generative Design process in Fusion360 typically involves the following steps:

- **Problem Definition:** in this initial stage, the designer defines the design problem by specifying the functional requirements, design constraints, and performance objectives for the product. This may include defining the boundaries of the design space, setting material, and manufacturing constraints, and establishing performance targets, such as minimizing weight or maximizing strength.

- **Design Space Exploration**: once the problem is defined, Fusion360's AI-driven algorithms begin to explore the design space by generating a vast number of design alternatives. These algorithms employ advanced optimization techniques, such as topology optimization, lattice optimization, and multi-objective optimization, to create diverse and innovative design solutions that satisfy the specified objectives and constraints.

- **Design Evaluation**: as design alternatives are generated, Fusion360's built-in simulation tools are used to evaluate their performance under various conditions, such as structural loads, thermal stresses, and fluid dynamics. This allows the designer to quickly identify the most promising design solutions based on their performance metrics, such as weight, strength, and deflection. The comparison between the output models is also simplified by the automatic creation of charts and diagrams and the possibility to export each single design and use the model for deep simulations.

- **Design Selection and Refinement**: after the performance evaluation, the designer can select one or more design alternatives for further refinement and optimization. Fusion360's parametric and direct modelling tools enable the designer to easily modify and fine-tune the generated designs, ensuring they meet the desired performance goals and adhere to manufacturing requirements.

- **Design Validation and Manufacturing**: Finally, once the design is refined and optimized, Fusion360's simulation tools can be used to validate its performance under real-world conditions, such as fatigue, vibration, and thermal stress. If the design meets the specified performance targets, it can be exported to various manufacturing processes, such as 3D printing, CNC machining, or injection molding, directly from the Fusion360 platform.

The design capabilities of Fusion360 offer several key benefits to development teams:

- **Vast exploration of different design solutions**: By automating the generation of design alternatives, Fusion360 allows designers to explore a much broader range of design possibilities than traditional design methods, leading to more innovative and unconventional design solutions that might not have been considered otherwise.
- **Improved Efficiency and Decision-Making:** algorithms can quickly generate and evaluate thousands of design alternatives, enabling product development teams to make more informed decisions and select the best design solutions based on their performance metrics. Fusion 360, moreover, arrange in-software visual and more parametric diagrams which can be useful in the data communication between teams from different areas involved in the product development. This can significantly reduce the time and effort required to identify optimal design solutions, ultimately accelerating the product development process.

- **Sustainability and Material Optimization**: AI can help create more sustainable products by optimizing material usage and minimizing waste, e.g. identifying design solutions that use less material while maintaining structural performance, leading to more eco -friendly and cost-effective products.
- **Customization and Personalization**: AI-driven generative design can enable development teams to create bespoke and personalized products tailored to specific user needs and preferences. By incorporating user input into the design process, customized solutions that cater to individual requirements, offering a higher level of personalization and differentiation in the market, can be generated.
- **Enhanced Collaboration**: Fusion360's cloud-based platform allows product development teams to collaborate seamlessly in real-time, enabling designers, engineers, and manufacturers to work together more effectively, which can help streamline the design process, reduce communication barriers, and foster a more collaborative and innovative design environment.
- **Integration with Manufacturing Processes**: being generative design capabilities tightly integrated with its manufacturing tools, seamless transition from design to production is enabled. This helps product designers to ensure that their designs are not only optimized for performance but also for manufacturability, ultimately reducing the time to market and ensuring a smoother product launch.

Despite the numerous benefits, Fusion360 also presents some challenges and considerations for product development teams:

- **Overcoming the Complexity Barrier**: The generative design process can sometimes generate complex and intricate design solutions that may be challenging to manufacture using traditional methods. Product development teams need to carefully consider the manufacturability of their designs and employ appropriate manufacturing techniques, such as additive manufacturing or advanced CNC machining, to bring these designs to life. Also, the interface is stiffer in comparison to the one in Synera.
- **Managing Intellectual Property and Data Security**: As algorithms generate design solutions based on user input and proprietary data, development teams must carefully manage their intellectual property and ensure that sensitive data is adequately protected which may involve implementing strict data access controls, encryption, and other security measures to safeguard valuable design information.
- **Balancing Performance and Aesthetics**: generative design often prioritizes performance optimization over aesthetics, which may result in designs that appear unconventional or unappealing. It is therefore important to strike a balance between optimizing for performance and maintaining a visually appealing design that resonates with end-users.

3.3.10. Midjourney

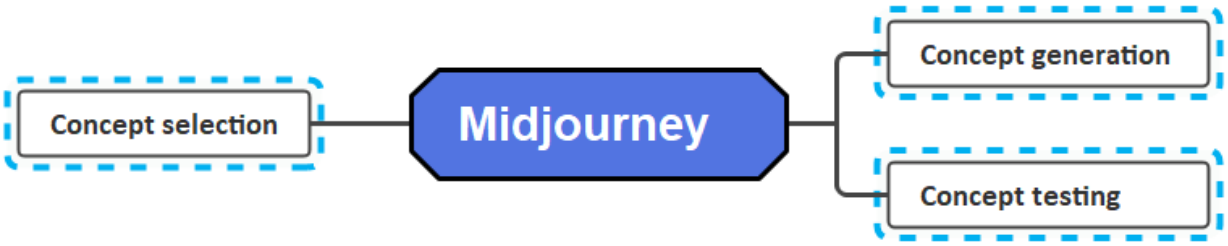

Figure 17: Schema depicting the link between NPD and the software

The applications of this software, but in general of all text-to-image, text-to-model and image to image software, fill the limits of more engineering-oriented tools in all those contexts in which specifications and in general fewer tangible variables play a more important role.

In the nascent stages of concept development, particularly in contexts where the product ideation lacks a clear definition, tools like Midjourney methodologies can serve a multitude of functions. These scenarios provide fertile

grounds for the application of such techniques, which can be instrumental in the brainstorming stages as a catalyst for image generation. Such methodologies can efficaciously guide the juxtaposition of diverse ideas, thereby fostering an environment conducive to creativity and innovation.

Midjourney methodologies and similar approaches can function as a support structure during the ideation process, facilitating the examination and assessment of the most valuable ideas. They can engender a variety of outputs that encapsulate the preferences expressed by a diverse range of respondents. This inclusivity ensures that the final product or concept is well-rounded and catered to a broad spectrum of potential users or consumers.
Incorporating these methodologies into the concept development process can lead to a more robust and comprehensive understanding of the product or idea at hand. It allows for a more informed decision-making process, better alignment with user or consumer preferences, and a higher likelihood of producing a product or concept that is both innovative and valuable.

For example in a scenario of a preference analysis conducted through a comprehensive survey in which a representative set of respondents, characterized by different categories of attributes, had to express their preference for significant combinations of semi-qualitative characteristics of a set of cars, Midjourney can on one hand be employed to assist respondents in providing more accurate and thoughtful responses by visualizing each combination of car attributes, on the other can be used to analyze the preferred combinations of car attributes for each cluster of users, thereby identifying patterns and trends in preferences across different demographic groups.
The first versions of this tool is based on GAN (Generative Aversarial Networks) type models, that can convert a text into an image and/or sequence of images (potentially low-frame-count video).
This model can be described as a dialogue between two parts, one of which is aimed at proposing and refining the image, while the other is focused on the criticism of the different images proposed in order to obtain an image that is as closely aligned as possible to the initial demands.
As the user base using Midjourney has grown, however, three fundamental problems related to the use of these algorithms have emerged:

- **Failure to converge**: the generator and discriminator fail to determinate an optimal image and therefore they remain in a continuous loop.
- **Mode collapse:** the discriminator can't discriminate between a real image and the generator's output, and for this reason the generator itself produces the same output regardless of its actual quality.
- **Vanishing gradients:** in this situation the generator can't learn by adjusting its parameters (weights and biases) through a process called backpropagation, since the discriminator passes insufficient information to It. As a company the learning process and the generation of output are blocked.

These limitations lead to development of the Diffusion model, which unlike GANs progressively create an image rather than all in one moment. On this type of model was built the foundations of the latest versions of Midjourney.

3.3.11. DALL-E

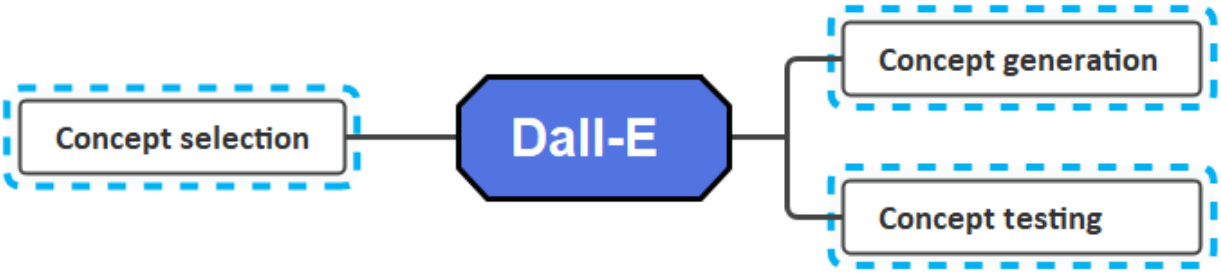

Figure 18: Schema depicting the link between NPD and the software

DALL-E is a specialized iteration of the GPT-3 language model that's been uniquely trained to generate images based on textual descriptions. This is achieved through a modified transformer architecture, a type of neural network

generally utilized in natural language processing tasks, but in this case, adapted for the generation of images and integrated with a Diffusion model.

A key feature of DALL-E is its enhanced capacity to interpret fewer specific prompts, thanks to the use of trained Latent Language Models (LLMs). As such, DALL-E could be an advantageous tool in supporting the development of diverse product ideas at the cost of producing less “artistic” image than Midjourney. Additionally, DALL-E presents a novel feature for expanding and modifying input images, essentially allowing for the creation of a complete environment surrounding the initial image.

Another fundamental difference linked to the user experience of Dall-e compared to Midjourney is the absence of all those features that want to modify the aesthetic style with which the images are generated.

Arguably the most ground-breaking aspect of DALL-E is the anticipated update to SHAP-E, transitioning the software from text-to-image to text-to-model, capable of generating Standard for the Exchange of Product Data (STEP) models. This progression holds considerable potential for application in more technical areas, such as engineering, since this model could be optimized and become a reliable reference for a variety of other fields such as the manufacturing process development.

Figure 19: Results of the prompt "futuristic rally vehicle exploring antarctic"

3.3.12. Dezgo

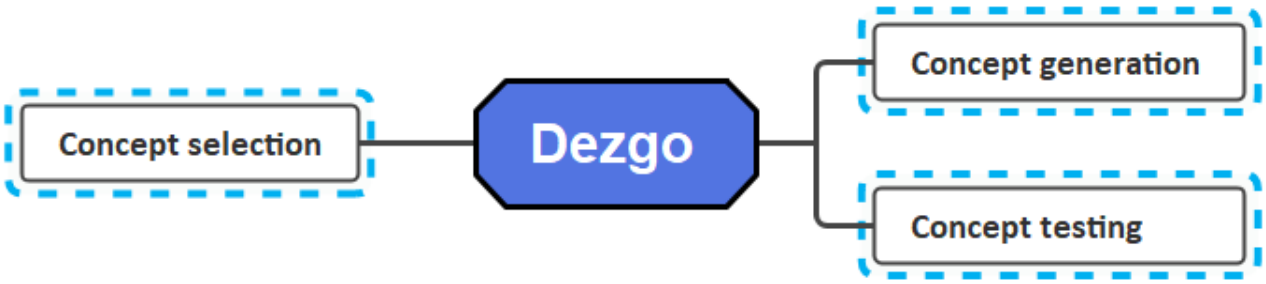

Figure 20: Schema depicting the link between NPD and the software

Dezgo is an AI-based image-to-image generator that uses Stable Diffusion AI to edit an existing image to fit a given text description. The user can provide a text prompt that describes how the final image should look like and adjust the strength of how strongly the original image should be altered, then the user can upload the original image to be modified and select the AI model used to generate the image.

By combining the user's input text with the selected original image, Dezgo generates outputs that reflect the user's creative intent while infusing new elements inspired by the text prompt. In situations where there is yet no codifiable data in written form about the preferences of one or more classes of potential customers, image-to-image tools of this kind can help explore possible alternatives visually.

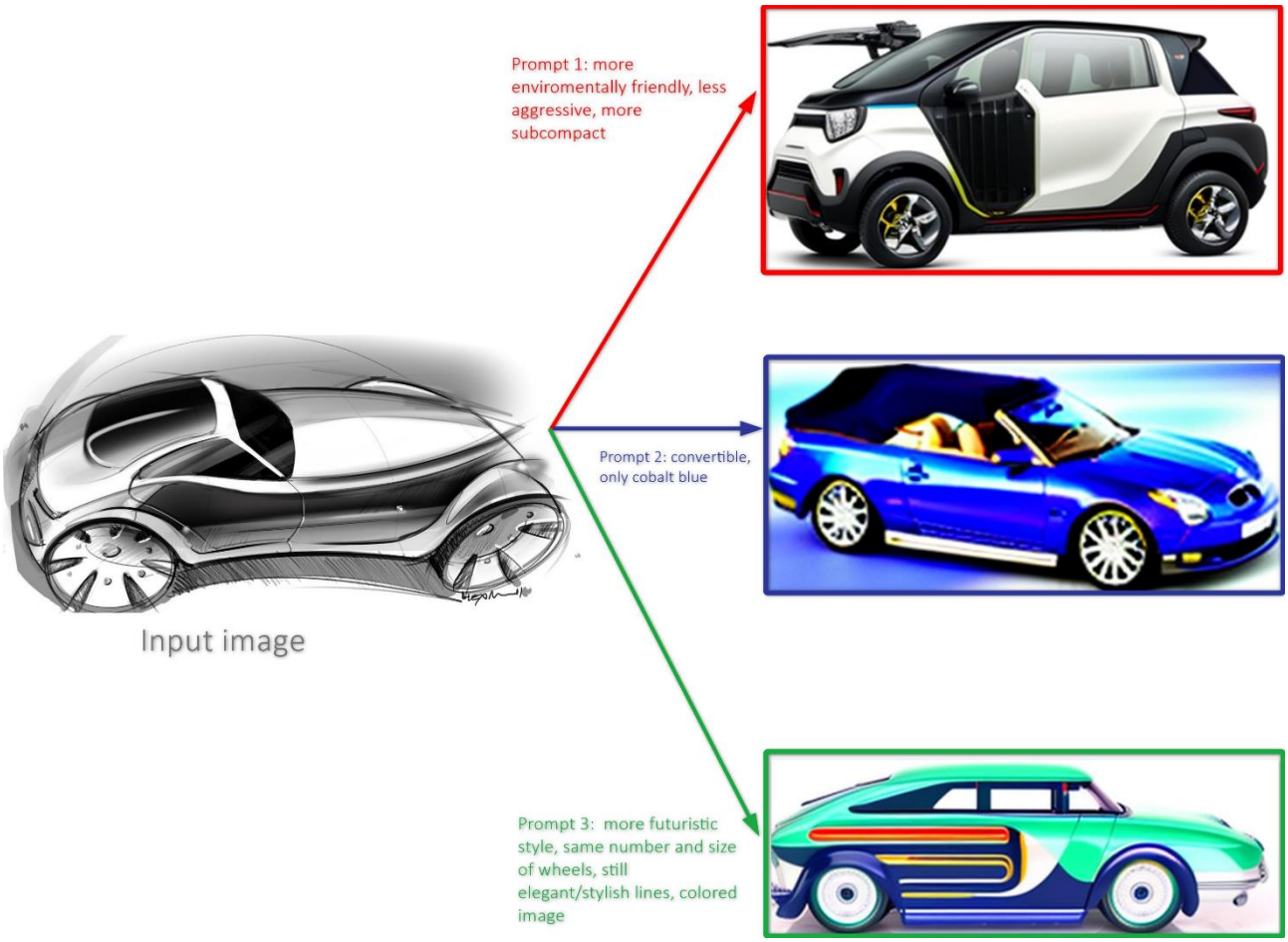

Figure 21: Example of image-to-image generation

3.3.13. ANSYS

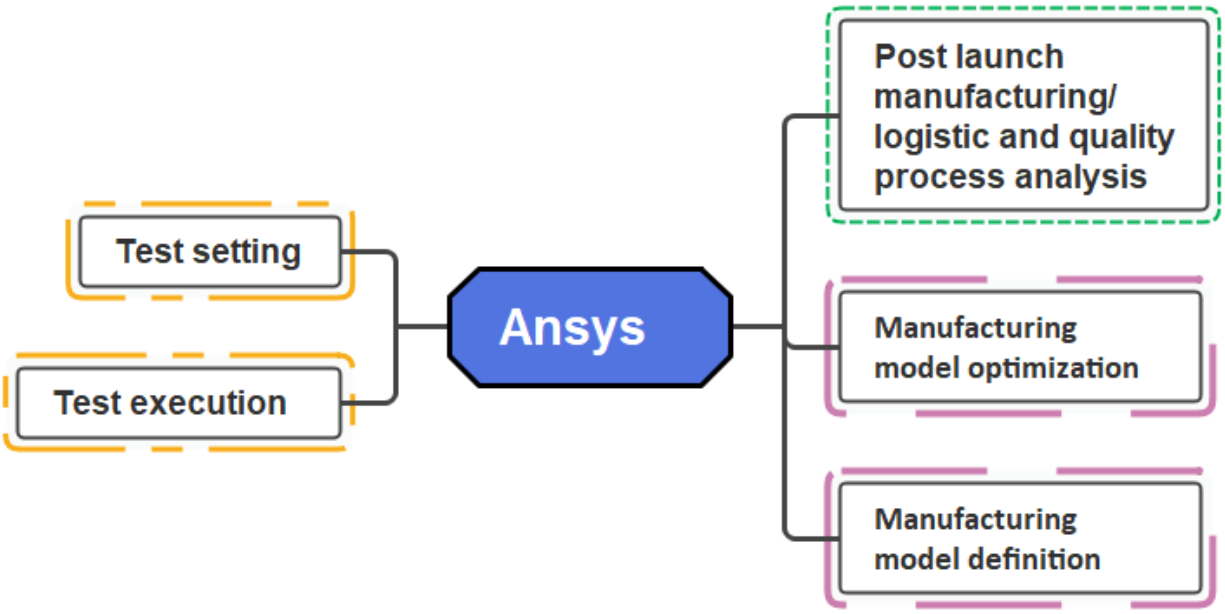

Figure 22: Schema depicting the link between NPD and the software

ANSYS is a leading engineering simulation software suite which brings together the most recent artificial intelligence (AI) and machine learning (ML) technologies to optimize product designs and reduce physical testing. It allows product developers to simulate and predict the performance of different design alternatives under different conditions, including stress, temperature, and fluid dynamics. This enables teams to make informed design decisions, hasten and lower the cost of physical testing, and boost efficiency and reliability. ANSYS exploits different techniques to learn from previous simulations and user inputs, allowing the software to continuously enhance its predictive abilities and provide more precise results. This never-ending learning process is made possible by the implementation of supervised learning, unsupervised learning, and reinforcement learning algorithms.

The software is trained using supervised learning algorithms on labelled data, enabling it to make predictions or classifications based on existing patterns and relationships. In terms of product design, supervised learning can be employed to generate predictive models for material properties, structural performance, along with various other crucial design attributes. It analyses historical data on similar structures and their respective failure points, being then able to predict the failure points of a structure under specific load conditions.
Unsupervised learning algorithms, on the other hand, enable ANSYS to identify hidden patterns, trends, and relationships within unlabelled data. The software can effectively evaluate huge datasets and also give valuable insights to teams with the aid of clustering and dimensionality reduction. Engineers can rapidly identify the most promising design alternatives based on their overall performance.
Error and trial can be implemented to optimize product designs with reinforcement learning algorithms within ANSYS to boost its decision making and general design by maximizing a predefined reward function via iterative learning processes. This is particularly helpful when traditional optimization techniques are unfeasible or computationally costly. In a real-life example, reinforcement learning can be used to optimize the shape of an aircraft wing to minimize drag while maintaining structural integrity.
The application of neural networks, especially along with deep learning techniques, is another important component of ANSYS. Deep learning methods, like convolutional neural networks (CNNs) and recurrent neural networks (RNNs) can efficiently process complex and high-dimensional data, allowing the software to handle much more intricate design problems. CNNs can be utilized in computer vision applications inside ANSYS to determine features, shapes, and patterns for product designs, and RNNs can be used to model time-dependent phenomena like fluid flow and heat transfer.
ANSYS implements evolutionary computation techniques such as genetic algorithms to tackle challenging optimization issues. Genetic algorithms can explore a vast search space of design options by modelling natural evolution, leading to optimal configurations which meet different goals, including manufacturability, performance, and cost. This can result in more creative and effective product designs that boost the competitiveness and market viability of the products.

3.3.14. GE Digital's Predix

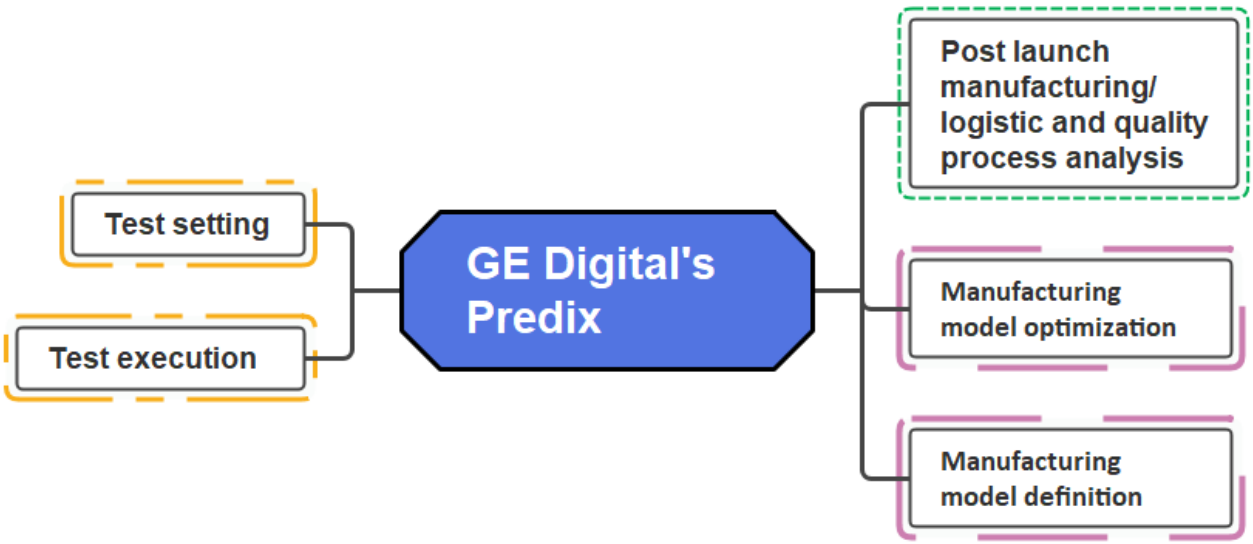

Figure 23: Schema depicting the link between NPD and the software

GE Digital's Predix platform is a remarkable example of how artificial intelligence (AI) can be effectively employed in the product development process, specifically in the realm of industrial equipment and processes. This industrial internet platform aims to optimize the performance, reliability, and efficiency of industrial systems focusing on predictive maintenance algorithms, anomaly detection models, and process optimization techniques: it allows product developers to actively monitor and enhance their products in real-time, which results in a significant reduction in costs, minimized downtime, and improved product quality and reliability.

To appreciate the impact of GE Digital's Predix on product development, it is necessary to delve into the specific AI techniques that are employed within the platform and how they contribute to the improvement of industrial systems:

- **Predictive maintenance:** it is a crucial aspect of optimizing the performance and reliability of industrial equipment. By employing supervised learning algorithms, Predix is capable of analysing historical equipment data, such as sensor readings and maintenance logs, to identify patterns that may indicate potential equipment failures. These patterns are used to build predictive models that estimate the remaining useful life (RUL) of components and determine the optimal maintenance schedule. It proactively identifies and address potential issues, reducing equipment downtime and associated costs while extending the overall lifespan of their products.
- **Anomaly detection**: it is another essential aspect of improving the reliability and efficiency of industrial systems. Predix leverages unsupervised learning algorithm like clustering and autoencoders to analyse large volumes of sensor data from equipment and processes, identifying deviations from normal behaviour patterns. These deviations, or anomalies, may be indicative of underlying issues, such as equipment malfunctions, process inefficiencies, or quality control problems. Detecting and addressing these anomalies in real-time is imperative to proactively mitigate potential issues before they escalate. Moreover, the insights gained from anomaly detection can inform future product design iterations, enabling teams to develop more robust and reliable products.
- **Optimizing industrial processes** is a complex and challenging task, often involving multiple conflicting objectives, such as minimizing costs, maximizing throughput, and ensuring product quality. Predix incorporates reinforcement learning and evolutionary computation techniques, such as genetic algorithms, to explore various process configurations and identify optimal solutions that balance these objectives. By modelling the industrial processes as Markov decision processes (MDPs) or multi-objective optimization problems, the software can learn optimal control policies or Pareto-optimal solutions that maximize the overall process efficiency and effectiveness.

Predix benefits from the integration of computer vision and natural language processing (NLP) technologies, which further enhance its capabilities in monitoring and optimizing industrial systems, for example to analyse visual data from cameras and sensors, enabling the platform to detect defects in products, identify potential hazards in the production environment, or monitor the progress of maintenance activities.
Similarly, NLP techniques can be used to analyse textual data from equipment manuals, maintenance logs, or operator notes, providing valuable insights into the operation and maintenance of industrial systems. By combining these insights with the information obtained from sensor data and predictive models, Predix can offer a comprehensive understanding of the state and performance of industrial systems.

3.3.15. Simcenter Amesim

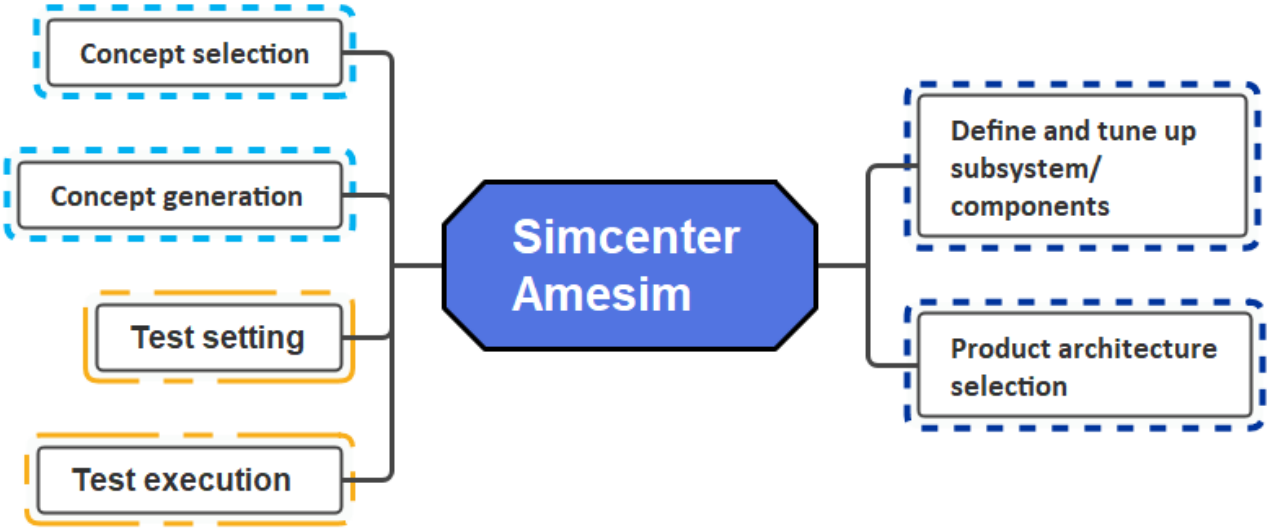

Figure 24: Schema depicting the link between NPD and the software

Siemens' Simcenter Amesim is a comprehensive simulation software platform designed to facilitate the modelling and analysis of complex multi-domain systems. It is extensively used in various industries, including automotive, aerospace, and industrial equipment, where intricate systems are required to work in harmony.
The AI-driven features of Simcenter Amesim can be categorized based on the AI branches outlined in this thesis, namely machine learning, neural networks, evolutionary computation, computer vision, robotics, expert systems, speech processing, natural language processing, planning, and large language models. In this section, we discuss how Simcenter Amesim leverages these AI branches to support design and innovation processes in product development.

Simcenter Amesim incorporates various *machine learning techniques*, such as supervised learning, unsupervised learning, and reinforcement learning, to improve the efficiency and accuracy of system simulations. The software can analyse historical data, learn the underlying patterns and relationships, and develop predictive models that can be used to forecast system behaviour under various operating conditions. These models can then be employed by developers to optimize system configurations, anticipate potential issues, and improve overall system performance.

Incorporating *neural networks*, Simcenter can develop sophisticated models able to simulate complex non-linear relationships between system components. By utilizing the inherent parallel processing capabilities of neural networks, the software can efficiently handle large-scale simulations and accurately predict system behaviour under a wide range of conditions, identifying optimal design solutions, validating system performance, and reducing the need for costly and time-consuming physical testing.

The tool also employs *evolutionary computation techniques* to optimize system configurations based on user-defined objectives and constraints. By simulating the process of natural selection, these algorithms can explore vast design spaces, identify trade-offs between conflicting objectives, and converge on optimal solutions that balance factors such as cost, performance, and manufacturability. Although not a primary focus, computer vision techniques can be integrated into the software to support advanced visualization and analysis of simulation results. By using image

processing algorithms, product development teams can analyze complex patterns and relationships within their simulation data, which can help inform design decisions, identify potential issues, and support the development of innovative products.
Simcenter Amesim can then be used to design and simulate *robotic systems*, incorporating AI-driven intelligent control algorithms and autonomous exploration techniques by modelling the dynamics, kinematics, and control systems of robots.
Expert systems can be integrated to guide through the simulation and analysis process. By incorporating knowledge-based reasoning and heuristic algorithms, these expert systems can provide valuable insights, recommendations, and best practices to help users optimize their system designs and improve product development outcomes. *Speech processing* techniques will also be added to the software in the close future to provide a more intuitive and engaging user experience, facilitating seamless communication between users and the simulation platform; along with *Natural language processing* (NLP) techniques, it will facilitate more efficient and easy documentation and reporting processes, automatically generating textual descriptions of simulation results, extracting relevant information from user inputs, and even responding to user queries in a conversational manner.
Simcenter Amesim already incorporates *planning algorithms* to support the efficient execution of simulation tasks and the optimization of system designs. With the employment of advanced optimization techniques and heuristic search strategies, the software can allocate resources, plan simulation tasks, and generate optimal design solutions that balance multiple objectives, such as cost, performance, and manufacturability.

With the recent developments in AI, the software could also be integrated with *large language models,* such as GPT-4, to enhance its natural language processing capabilities and support more advanced user interactions, exploiting the immense knowledge and understanding embedded in the aforementioned model to provide more context-aware and insightful recommendations, facilitate more effective communication with users, and even generate creative ideas and design suggestions that can drive innovation in the product development process.

3.3.16. Value Chain 2.0

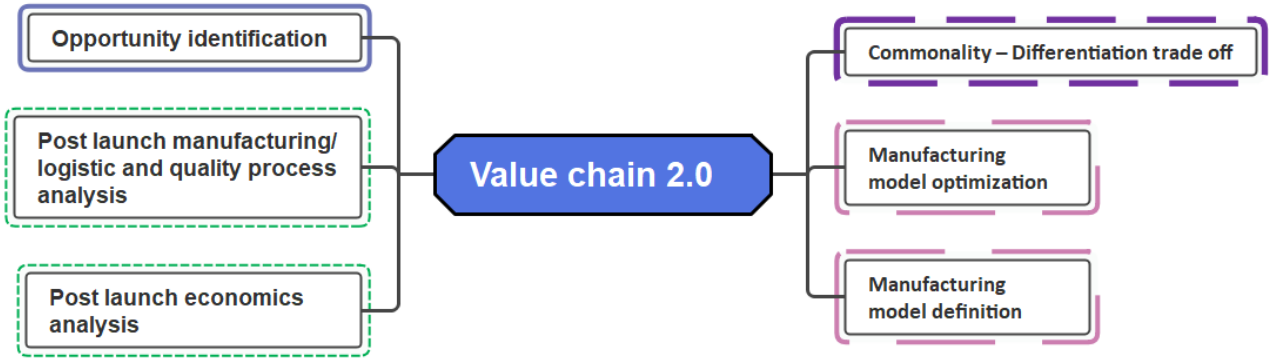

Figure 25: Schema depicting the link between NPD and the software

Value Chain 2.0 is an AI-driven supply chain optimization application to simplify and lower costs during product development. The software applies AI methods like machine learning, optimization algorithms and natural language processing to evaluate a vast quantity of supply chain data, pinpoint bottlenecks and recommend optimization strategies which eventually results in considerable improvements in the efficiency and cost-effectiveness of product development processes, leading to far more competitive and profitable products. Supply chain management is vital to the general success of a product; an effective one entails timely delivery of materials, efficient sourcing, and effective inventory control. Modern supply chains tend to be so complex that advanced AI technologies are required to optimize demand forecasting, supplier selection, transportation, and inventory control.
Among the primary challenges in supply chain management there is efficient forecasting demand to ensure an optimal balance between supply and demand. Value Chain 2.0 utilizes machine learning methods, including supervised learning and time series forecasting, to analyse past sales data and market trends, generating accurate demand forecasts. These predictions educate choices regarding resource allocation, production planning and inventory management, ultimately lowering costs and enhancing responsiveness to market fluctuations.

Supply chain management also relies heavily on selecting and determining the best suppliers. Using unsupervised learning algorithms like clustering and dimensionality reduction the software is able to segment and classify suppliers based on cost, quality and delivery performance, resulting in reduced expenses, better supplier performance, and enhanced product quality.
Efficient transportation is vital for minimizing costs and ensuring timely delivery of materials and finished products: Value Chain 2.0 utilizes optimization algorithms, including genetic algorithms and simulated annealing, to find the most efficient transportation routes and modes at the lowest cost. Product development teams can eventually decrease transport expenses, shorten lead times, and also enhance the overall performance of the supply chain operations through this approach.
An effective inventory management ensures the appropriate quantity of materials and finished goods is readily available at the right time and location: Value Chain tool utilizes reinforcement learning to balance the trade-offs between stock-outs, overstocking and carrying costs to attain optimum inventory levels and replenishment policies, enhancing service levels and general efficiency.
In today's globalized business environment, product developers frequently work with suppliers and partners across different countries and languages. Value Chain includes natural language processing features to facilitate seamless collaboration and communication between team members, suppliers, and partners, regardless of their native language. By automatically translating documents, emails, and other types of communication, it reduces misunderstandings and allows for better collaboration.

Value Chain 2.0 can easily integrate with other tools in the product development cycle, including design optimization software (such as GenOpt), simulation tools (such as ANSYS) and data visualization platforms (such as Tableau). This integration allows product development teams to take advantage of the full potential of AI technologies, driving efficiency improvements and fostering innovation across the whole product development lifecycle.

3.3.17. Jaggaer

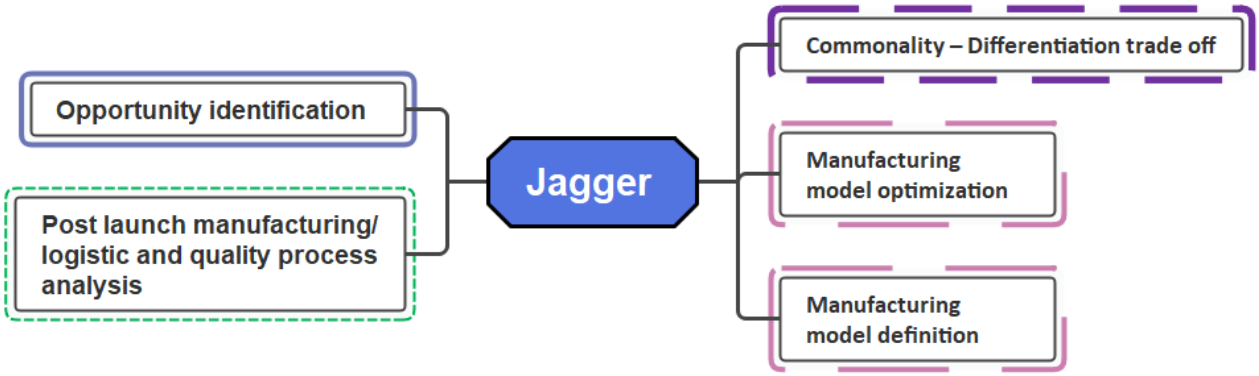

Figure 26: Schema depicting the link between NPD and the software

Jaggaer is an innovative AI-powered procurement and supplier management platform designed to assist in optimizing the sourcing, negotiation, and supplier management processes. In this comprehensive section, we will delve into the various technologies employed by Jaggaer and how they contribute to the efficiency and effectiveness of product development.

At the core of Jaggaer's AI capabilities lies machine learning, which plays a crucial role in analysing supplier performance data and identifying potential risks. Through the application of supervised learning algorithms, Jaggaer's platform can learn from historical data and build predictive models that enable development teams to make informed decisions regarding supplier relationships. Unsupervised learning techniques, such as clustering and dimensionality reduction, further enhance Jaggaer's capabilities by grouping similar suppliers and identifying patterns in the procurement data helping to quickly identify potential synergies, streamline supplier management processes, and uncover hidden opportunities for cost savings and efficiency improvements. Another important AI technique employed is reinforcement learning, which is particularly useful in automating and optimizing negotiation processes. By using reinforcement learning algorithms, the platform can simulate negotiation scenarios and learn

optimal strategies for achieving the best possible outcomes in terms of cost, delivery times, and quality. This allows for a more efficient negotiation with suppliers, reducing procurement costs and ensuring better alignment with project requirements. Jaggaer also leverages natural language processing (NLP) techniques to automate the analysis of procurement contracts and other legal documents: the platform can quickly identify key contract terms, potential risks, and areas for negotiation, not only reducing the time and effort required for contract review but also helping product development teams to better understand and manage their supplier relationships.
Expert systems are utilized to support the supplier evaluation process: by incorporating expert knowledge and predefined rules, it can assess supplier performance across various criteria, such as cost, quality, and delivery performance, prioritizing the efforts on managing the most critical supplier relationships.
In addition to the aforementioned AI techniques, Jaggaer also employs computer vision technology to streamline the supplier onboarding process: the platform can automatically extract relevant information from supplier documents, such as business licenses, certifications, and insurance documents. This not only speeds up the onboarding process but also reduces the risk of manual data entry errors and ensures that all supplier information is accurate and up to date.

Jaggaer's platform also integrates robotic process automation (RPA) capabilities to further improve the efficiency of procurement jobs: by automating repetitive tasks, such as data entry, invoice processing, and purchase order generation, RPA makes it possible to focus on higher-value activities, such as strategic sourcing and supplier relationship management which ultimately results in more efficient and cost-effective product development processes.

3.3.18. GPT-4

The OpenAI GPT 4, large language model, is a powerful tool able to support product development during different stages. The innovative technology employs its natural language processing (NLP) and understanding capabilities to aid and boost product development efforts.
The subsequent sections assess the role of GPT-4 in facilitating different areas of the design and innovation process, displaying its potential to change product development practices.

In the ideation and concept development stage, GPT-4 could play a critical role in generating creative ideas and inspiring novel solutions, using its enormous knowledge base to evaluate existing products, market trends and customer preferences and suggesting new ideas to meet unmet needs or even enhance existing designs. Moreover, it can help in brainstorming sessions by offering a diverse range of ideas, helping teams think outside of the box and explore new possibilities.
GPT-4's language generation capabilities can also considerably simplify the process of drafting product descriptions, user guides, along with various other technical documentation, being able to automatically produce precise, succinct, and engaging content which properly communicates the value proposition of the product by analysing its features, functions, and target market.
The natural language processing abilities of GPT-4 permit to effectively process and evaluate a lot of customer feedback from different sources, which includes product reviews, social media reviews, and customer support requests guiding choices about product design, feature prioritization and advertising strategies by identifying trends and recurring themes, and eventually leading to enhanced industry fit and consumer satisfaction.
Product development projects usually succeed through good communication and cooperation. and the LLM could help facilitate this by acting as a smart intermediary between team members, enabling real time translations, summarizing discussions, and also, when used in combination with text to image plugins, producing visual representations of complicated and detailed ideas to close communication gaps, reduce misunderstandings and also make increase the alignment between members of different teams and therefore allowing a better coordination toward a common objective.

GPT-4's potential in product development is even more amplified when incorporated with other AI complete different A.I. branches and algorithms discussed in this thesis. It can deliver valuable insights into product performance, customer preferences, and market dynamics when utilized along with machine learning methods being

supervised learning, unsupervised learning, or reinforcement learning. These insights can direct design decisions and innovation efforts, eventually resulting in far more competitive and successful products.
It could also complement computer vision solutions to analyse visual information including, prototypes and images from manufacturing processes. which could reveal design flaws, streamline production, and maintain product quality and consistency. Product development process may be further improved by the integration with expert systems making it able to make much more up to date recommendations, recognize potential risks and improve decision. GPT-4 and other large language models are likely to get far more sophisticated and capable of addressing a broader range of product development challenges as the field of AI evolves. Continual advancements in research might end up in stronger language models that can comprehend context, reason more effectively, and also generate more creative and innovative ideas.

As mentioned in the second paragraph nowadays chat gpt4 allows plus users to implement different types of plugins, lowering the effort necessary to reach the potential of generally complicated and not particular user friendly A.I. tools. The variety of these plugins guarantees the application of GPT 4 chats in a boundless range of fields, from legal information support to prompt optimisation. Some examples of promising plugins are listed below:

Code Interpreter: this is probably the most outstanding, flexible, and impacting add-on available today. It can basically automatically analyse, even when not guided, text of any kind and produce detailed graphs on any kinds. Moreover, its ability to autoanalyze immense database in a matter of minutes, without requiring specific indication on the method to use. It is worth to underline that, as happens with all prompts based A.I. tools, the greater the degree of detail of the input-indications, the closer will be the output to the user intention, hence in more skilled hands it can really surpass the capability in terms of time and effort management of other type of data analysis instruments mentioned in this chapter.

An example of a prompt session using Code Interpreter to analyse an xls dataset:

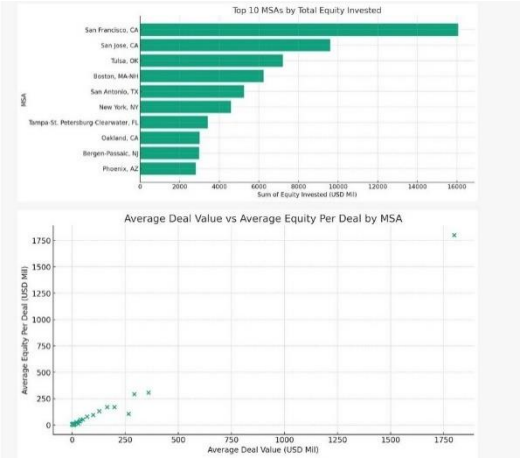

Figure 27: Results of Prompt 1 "Can you conduct whatever visualizations and descriptive analysis you think would help me understand the data?"

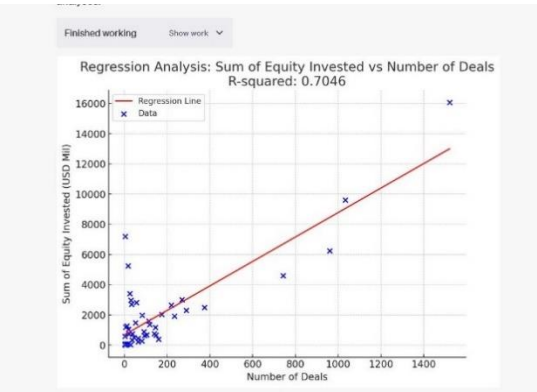

Figure 28: Results of Prompt 2 “Can you try a few regression analysis and look for interesting patterns?”

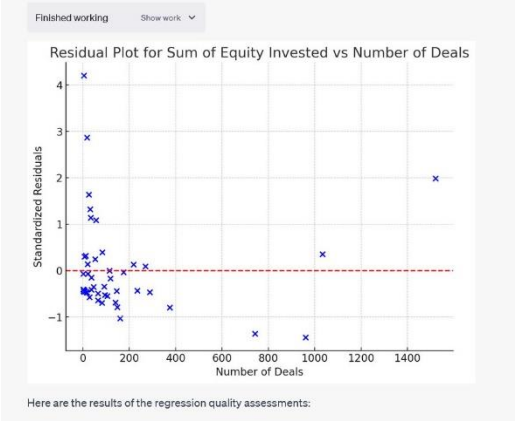

Figure 29: Results of Prompt 3 "Can you check to see the effect of any outliers on these regressions? And conduct any other regression quality?"

Browsing mode even thought this is not an actual add-in, it can enhance deeply the capabilities of GPT4 to gather data. In fact, by suggesting consciously links it is possible to concentrate all desired information in a single chat and, by interacting with it (and potentially use code interpreter functions) it is possible to quickly extract useful insights. However, the most immediate use of this new function is to concentrate in a single prompt what would have required specific searches using normal search engines. It is in fact the latter that are particularly exposed to the possible substitution if not integrated by LLMs.

Fiscal Note: it can be considered as a narrower version of the browsing mode that can enhance the velocity of discovering new changes in the social, economic and law areas that could lead to the rise of potential opportunities. It acts like a personal press review capable of adapting to the field of interest.

Although GPT4 is able to recognize and train to build around each user's way of communicating, at this date it is still possible to witness "hallucinations" in the form of:

- **Filling in gaps with incorrect information:** Sometimes, if the model does not have sufficient information to generate a response, it may fill in gaps with information that seems plausible based on its training but is actually incorrect. For example, if you ask it about a historical event that happened after its training cut-off, it might still generate a response, but this response could be completely fabricated. In practice, the answers can be conditioned by the LLM's perceived expectations of the questioner.
- **Generating overly creative responses:** Sometimes, the model might generate responses that are overly creative or imaginative, especially if the prompt encourages it to do so. This can lead to the generation of information that is not reality-based.
- **Confidently stating false information**: The model can sometimes make statements that are factually incorrect, yet it does so with confidence. This is because the model does not have a concept of "truth" or "falsehood"—it only knows how to mimic patterns in the data it was trained on. However, GPT4 can provide the textual/informative references used to answer the question appropriately, thus enabling verification of answers that appear less sensible and structured.

It is also possible to implement additional plugin, like Prompt Perfect, that can recommend prompt modification to simultaneously avoid this type of problem and obtain a mor effective prompt.

Considering the strong modularity of the architecture that characterises chat GPT and its almost universal applicability, it is a transversal tool for the entire product development process.

4. AI Challenges

This chapter explores the challenges and risks associated with the use of AI in product development, including bias propagation, societal impact, data security, data storage, legal issues, and integration challenges. By examining these challenges, product development teams can better understand the potential pitfalls of adopting AI technologies and develop strategies to mitigate risks and address potential issues. This chapter emphasizes the importance of striking a balance between the benefits of AI and the potential challenges and risks it presents, ensuring that AI is used responsibly and effectively in product development.

4.1. Bias Propagation

In the realm of artificial intelligence (AI) solutions, the propagation of bias is now a pressing problem, particularly in machine learning models. This problem has to be dealt with because AI technologies are increasingly being incorporated into different areas of products or services. Bias can manifest in several forms, which includes data bias, algorithmic bias, and cognitive bias, and may have substantial consequences on the performance, reliability, and fairness of AI systems. Comprehending the fundamental mechanisms, determining possible sources of bias, and developing strategies to prevent or even correct bias in AI models are crucial to preventing or correcting bias propagation and decreasing bias propagation. Thus, it's crucial to conduct comprehensive research as well as analysis to identify and address potential biases within AI systems. By doing so, we can make sure that AI technologies are developed and implemented in an impartial and fair way.

4.1.1. Data Bias

Data bias refers to the presence of systematic errors or inaccuracies in the data used to train or evaluate AI models. Data bias can result from various factors, such as sampling bias, measurement bias, and label bias.
Sampling bias occurs when the data used to train an AI model is not representative of the target population or distribution. This can lead to models that perform poorly on real-world data or that exhibit biased behaviour. For instance, if an AI system for product design is trained on a dataset consisting mainly of products from a specific geographic region, it may not generalize well to products from other regions. On the other hand, measurement bias arises from errors or inaccuracies in the data collection process, such as faulty sensors, human errors, or inconsistencies in data recording. This can result in AI models that are trained on erroneous or misleading data, leading to suboptimal or biased predictions. Lastly, label bias occurs when the labels or ground truth used to train supervised learning models are incorrect, incomplete, or inconsistent. This can result in AI models that learn to reproduce or even amplify the errors present in the training labels. To address data bias, the following strategies should be employed:

- Ensuring that data samples are representative of the target population or distribution by using stratified sampling or other techniques that account for potential sources of bias.
- Implementing robust data collection, processing, and validation procedures to minimize measurement errors and inconsistencies.
- Using active learning or other methods to improve the quality and consistency of training labels.

4.1.2. Algorithmic Bias

Algorithmic bias refers to systematic errors or inaccuracies in algorithm creation or implementation. This bias may be caused by model assumptions, optimization objectives or learning algorithms.

Machine learning models and AI models generally depend on simplifying assumptions about the data or relationships between variables. that can introduce bias if they don't effectively capture the true complexity or structure of the problem. For example, linear regression models presume a linear correlation between input features and output targets. If the true relationship is nonlinear, the linear regression model will be biased and could make inaccurate predictions. In order to overcome bias caused by model assumptions, it is recommended to use more flexible or

adaptive models which better reflect the true complexity of the problem, like nonparametric models, deep learning models or ensemble techniques.

AI models are usually trained by maximizing a defined objective function, such as minimizing the mean squared error or even boosting the likelihood of the data. These optimization objectives can generate bias if they do not align with the true goals or performance metrics of interest. If a model is trained to minimize the mean squared error of its predictions, it might lead to designs that are extremely conservative or even biased towards the mean, as it sets minimizing big mistakes ahead of accurately capturing the full range of results. In order to tackle bias resulting from optimization objectives, the objective functions used must be aligned with the true goals or performance metrics of interest, and also consider using methods like multi-objective optimization, regularization, or custom loss functions to better balance competing objectives and minimize bias.

AI models could also be biased due to the selection of the learning algorithm because different algorithms could be much more sensitive to noise, outliers, or other data irregularities. Additionally, certain learning algorithms are susceptible to overfitting, leading to models that excel on the training data but fail to adjust properly to new, unseen data. It is recommended to employ a variety of algorithms and techniques, such as regularization methods, ensemble methods, and cross-validation, to greatly reduce overfitting and enhance the generalization performance of the models to deal with bias resulting from algorithms learning.

4.1.3. Cognitive Bias

The term cognitive bias describes systematic inaccuracies or errors in human judgement, decision making or perception. The design and evaluation of AI systems, along with the interpretation and application of AI-generated insights, could be impacted by cognitive biases.

Some typical cognitive biases that might affect the usage of AI include:

Confirmation Bias: The tendency to seek, interpret or recall information in a way that supports existing hypotheses or beliefs. This can lead to biased decision making, as individuals could be much more likely to accept AI-generated insights that align with their expectations and discount those that oppose their viewpoints.

Anchoring Bias: The tendency to heavily depend upon an initial piece of information (the "anchor") when making choices. This can lead to biased decision making, because individuals might be influenced by preliminary AI-generated insights, even when subsequent information or analysis suggests that the original insights were incorrect or incomplete.

Availability Bias: The tendency to overestimate the probability of events depending on their availability in memory, which might be impacted by just how recent or emotionally charged the memories are. This can lead to biased decision making as individuals might be much more prone to relying on AI-generated insights that complement their most significant or memorable experiences.

In order to target cognitive biases, it's crucial to generate awareness and also promote critical thinking amongst team members. Education and training on typical cognitive biases can help accomplish this, in addition to structured decision-making processes which promote multiple perspectives, the usage of different data sources, as well as the analysis of alternative hypotheses.

4.1.4. Mitigating Bias Propagation in AI Systems

To summarize, to effectively mitigate bias propagation in AI systems, product development teams should adopt a holistic approach that combines strategies to address data bias, algorithmic bias, and cognitive bias. Some potential strategies include:

- Implementing robust data collection, processing, and validation procedures to minimize data bias.

- Ensuring the representativeness of training data by using stratified sampling or other techniques that account for potential sources of bias.
- Using diverse AI algorithms and techniques, such as nonparametric models, ensemble methods, or regularization methods, to minimize algorithmic bias.
- Aligning optimization objectives with the true goals or performance metrics of interest and considering multi-objective optimization, regularization, or custom loss functions to minimize bias in AI models.
- Creating awareness of cognitive biases and promoting critical thinking among team members through training, education, and structured decision-making processes.

By understanding the sources and mechanisms of bias propagation and implementing strategies to prevent or correct bias, product development teams can enhance the performance, reliability, and fairness of AI-supported design and innovation processes.

4.2. Societal Impact: Workforce Replacement

As technologies continue to advance and become more integrated into various aspects of development, concerns have been raised about the potential impact on the workforce, particularly in terms of job displacement and automation. This section will explore the possible effects of AI on the workforce, as well as potential strategies for mitigating these effects and ensuring a just transition for workers affected by AI-driven changes.

4.2.1. Job Displacement and Automation

Among the primary issues concerning the growing usage of AI is the possible displacement of human workers by automation. AI systems have already proven to outshine humans in tasks like data analysis, pattern recognition and optimization. As technologies continue to improve, chances are that a rising number of tasks will be automated, possibly leading to job displacement for workers in affected roles.

It's essential to understand that AI-driven automation is not an all or nothing phenomenon. AI systems may often complement human abilities rather than replacing them completely. AI can easily automate repetitive or time intensive tasks, freeing up human workers to concentrate on more innovative, complex, or strategic aspects of development. Workers that can adjust to these new technologies might experience better productivity, innovation, and overall work satisfaction because of this.

4.2.2. The Role of Education and Retraining

Investing in education and retraining programs to equip workers with the needed skills and expertise to adjust to the changing landscape is crucial to mitigate the damaging consequences of AI driven workforce displacement. The programs should aim to provide workers with a solid foundation of AI technologies and the capability to utilize these technologies in a real-world setting.

Along with formal education and retraining programs, companies can offer on-the-job training and support for employees moving to new positions within the organization. Mentorship programs, job shadowing opportunities or access to online resources and materials are all examples of this. By providing workers with all the tools as well as guidance they need to adapt to new technologies and roles, companies can make it possible to ensure a just transition for their workforce and also lessen the damaging effects of AI-driven automation.

4.2.3. The Importance of Social Safety Nets

It's essential to acknowledge that not all employees could effectively adjust to the changing landscape of product development, and that some job displacement could be inevitable. Strong social safety nets are required to aid workers that are impacted by AI-driven changes in this kind of instances.

Unemployment benefits, affordable healthcare, and housing, along with other kinds of social assistance can help workers in overcoming obstacles related to job loss as well as reskilling in new industries or professions. Society can lessen the damaging consequences of automation and foster a far more equitable and inclusive future by providing all workers with all the support and resources they require to adapt to the changing workplace landscape.

In conclusion, the increasing use of AI might have significant ramifications for the workforce, with employment displacement and automation concerns at the cutting edge. Companies and society can mitigate these potential adverse impacts by investing in education and retraining programs, offering on-the-job assistance for employees moving to new positions, and ensuring strong public safety nets to guarantee a good transition for those employees impacted by AI-driven changes.

4.3. Data Security, Data Storage and Legal Issues

The widespread use of AI technologies raises various concerns related to data security, data storage, and legal issues. This section will explore these concerns and discuss potential strategies for addressing them within the context of AI-supported development processes.

4.3.1. Data Security

Ensuring data security is a crucial aspect of integrating AI technologies. As AI systems process large volumes of sensitive information, protecting this data from unauthorized access, manipulation, or disclosure becomes increasingly important. The growing reliance on AI systems for data analysis and decision-making also means that security breaches can have a significant impact on the effectiveness and reliability of insights and recommendations based on those systems.

To address data security concerns, a multi-layered approach that includes encryption, access controls, and regular security audits should be implemented.

4.3.1.1. Encryption

A crucial safeguard for protecting sensitive data from unauthorized access is encryption. The process entails transforming plaintext data into a ciphertext format which can only be decoded using a particular key. Data kept in databases, file systems, and communication channels may be encrypted so that unauthorised people cannot read nor use it without having the decryption key.

There are two primary types of encryptions: symmetric key encryption and asymmetric key encryption. The exact same key is used for both decryption and encryption in symmetric key encryption, while in asymmetric key encryption, different keys are used for encryption and decryption (public and private keys, respectively). The selection of encryption technique is determined by the particular requirements and constraints of the process and the AI system utilized.

Mathematically, symmetric key encryption can be represented as:

$$Ciphertext = E(Plaintext, Key)\ Plaintext = D(Ciphertext, Key)$$

E is the encryption function, D is the decryption function, and Key is the shared key utilized for decryption and encryption.

Asymmetric key encryption can be represented as:

$$Ciphertext = E(Plaintext, Public\_Key)\ Plaintext = D(Ciphertext, Private\_Key)$$

Where E and D stand for the decryption and encryption functions, respectively, and $Public\_Key$ and $Private\_Key$ are the different keys used for encryption and decryption.

4.3.1.2. Access Controls

Access controls play a crucial role in data security by preventing systems and users from getting access to, altering, or deleting sensitive information. A comprehensive access control policy reduces the chance of unauthorized access to data and systems.

Access controls come in a variety of types, for example:

- *Discretionary Access Control (DAC)*: This method enables data owners to decide who can access their data and also the degree of access granted. DAC policies generally utilize Access Control Lists (ACLs) that link user or group identities to certain rights, like read, write, or execute.
- *Mandatory Access Control (MAC)*: Rules and security labels created by a central authority establish access to data as well as resources in MAC systems. Subjects (processes or users) and objects (data or resources) are given labels which are known as classifications or security levels. The object's security level has to match or even surpass the subject one before access is granted.
- *Role-Based Access Control (RBAC)*: RBAC policies assign permissions based on predefined roles which are connected to particular sets of privileges. Roles are then assigned to users either directly or through group memberships and they inherit the permissions associated with those roles.
- *Attribute-Based Access Control (ABAC)*: ABAC policies are more fine - grained because they consider multiple attributes like user identity, resource characteristics and environmental factors to determine access permissions. An access control engine evaluates ABAC policies, usually expressed as policies or rules, to decide if access should be given.

4.3.1.3. Security Audits

A comprehensive data protection strategy relies on regular security audits to find possible vulnerabilities, assess current security measures, and also offer feedback for the development of new security policies and procedures. Security audits usually involve a complete evaluation of processes, networks, and systems inside of a company, to find vulnerabilities that an adversary might exploit.

A security inspection consists of a number of key elements, such as:

- *Vulnerability Assessments*: the procedure for vulnerability assessments entails determining and evaluating the weak spots in systems, networks, and applications of a company. Automated manual testing or scanning tools may either accomplish this goal or a mix of both. Vulnerability assessments help to determine areas for improvement and guide targeted security methods development.
- *Penetration Testing*: making use of ethical hacking or penetration testing is a proactive technique for evaluating the security of computers and networks within a company. It entails emulating real world strikes to find out possible flaws and evaluate the effectiveness of current security measures. The testing may uncover hidden flaws, test security controls, and also help support the advancement of more effective security policies and procedures.
- *Security Policy and Procedure Review*: a security audit must have a comprehensive overview of the security policies and procedures of the business which entails evaluating the usefulness of existing policies as well as suggesting improvements, therefore ensuring the organization's safety measures are recent and aligned with industry standards.
- *Incident Response Planning*: to reduce security breaches and guarantee a quick recovery, an effective incident response strategy is needed. Throughout a security audit, companies must evaluate their incident response strategies, figure out their effectiveness and upgrade them as required to make sure they could efficiently control as well as react to security incidents.

Businesses could efficiently protect sensitive data and systems from unauthorised access, disclosure, and manipulation by employing an extensive data security approach which includes encryption, access controls and periodic security audits. This is essential to ensure the validity and reliability of insights and to safeguard the confidentiality and security of the data processed by these methods.

4.3.2. Data Storage

As AI-driven processes generate vast amounts of data, companies must consider efficient and secure strategies for data storage. This section will delve into specific techniques and technologies to manage large volumes of data effectively while maintaining data security and accessibility.

4.3.2.1. Cloud-Based Storage Services

The increasing data storage requirements of AI can be met with cloud-based storage services which are both affordable and scalable. These services offer internet - accessible remote storage infrastructure, enabling companies to store, retrieve and process data without maintaining their own storage hardware. Amazon Web Services (AWS), Microsoft Azure, and Google cloud Platform happen to be among the prominent Cloud storage providers.

Cloud storage solutions have several benefits, such as: scalability, which is the capacity of cloud storage solutions can be quickly increased or decreased according to the storage requirements of a company. This allows companies to pay only for the storage capacity they require, lowering costs and ensuring effective resource utilization; security, since nearly all cloud storage providers have built-in security features like data encryption, access controls and regular security audits to safeguard stored data from unauthorized access or manipulation; and finally flexibility based on needs and requirements, companies can in fact select from various storage options (e.g., object storage, file storage or block storage) and data redundancy levels.

4.3.2.2. Distributed File Systems

AI generates large amounts of data that could be handled using distributed file systems (DFS). The storage of data on several servers or nodes, often distributed geographically, can enhance availability and fault tolerance in such systems. Hadoop distributed file system (HDFS), GlusterFS, and Ceph happen to be the most commonly used distributed file systems. Companies can store data on several nodes making use of these file systems, so that the data remains accessible even when one node fails.

Distributed file systems offer several advantages, including fault tolerance, since the data is redundantly stored across a number of nodes, it remains still accessible even if one or more fail. This safeguards against data loss and minimizes downtime; DFS could also be easily scalable by adding or deleting nodes from the system, enabling businesses to change storage capacity as required; load balancing, since the distribution of data is among multiple nodes, the storage and retrieval workload can be balanced across the system, thereby boosting performance, and lowering the likelihood of bottlenecks.

4.3.2.3. Data Lakes

Data lakes are a relatively new data storage paradigm designed to accommodate the diverse data types and formats generated by AI processes. Unlike traditional databases, data lakes store raw data in its native format, allowing companies to ingest, store, and process large volumes of structured and unstructured data without the need for time-consuming data transformation processes. Data lakes can be implemented using various technologies, such as Hadoop, Apache Spark, or cloud-based storage services like Amazon S3. They provide a flexible and scalable solution for storing and processing AI-generated data, enabling companies to derive insights and drive innovation more effectively. They can store and manage a wide range of data types and formats, making them well-suited for the diverse data generated by AI-supported product development processes. Furthermore, like cloud-based storage

services and distributed file systems, data lakes can be easily scaled to accommodate growing data storage demands, ensuring efficient resource utilization and cost management. In the end, data lakes support advanced analytics, machine learning, and AI processing, allowing companies to derive valuable insights from their data more quickly and effectively.

4.3.3. Legal Issues

The growing use of AI technologies raises legal concerns, particularly in areas like intellectual property, security, and liability. AI generated designs or innovations can raise issues regarding intellectual property ownership, since conventional intellectual property laws are usually not able to handle the unique problems presented by AI-generated works.

AI systems use to process customer data, or any other sensitive information can also raise privacy concerns, particularly given the increasingly strict data protection regulations like the General Data Protection Regulation (GDPR) of the European Union. Businesses must ensure that their AI-enabled procedures comply with appropriate data protection laws and that proper measures are in place to safeguard the privacy of individuals whose information AI systems process.

AI technologies can also raise liability concerns, particularly if AI-driven decisions or suggestions result in flawed products, financial losses, or other negative outcomes. AI-driven procedures carry potential risks to which businesses should thoroughly consider and implement safeguards like insurance coverage, indemnification agreements or dispute resolution mechanisms to mitigate them.

4.3.3.1. Intellectual Property Rights in AI-Generated Works

AI technologies which can independently generate designs or innovations present legal challenges associated with intellectual property rights. Usually, intellectual property laws based on conventional concepts such as patent and copyright law grant rights to artistic creations to human creators and inventors. Nevertheless, these laws are rarely sufficiently suited to deal with the case of AI producing innovative works without significant human involvement.

A system able to create a brand-new design for a consumer product makes an example of the complexities of intellectual property rights, illustrative of the numerous potential drawbacks of such works. Under traditional patent laws, the owner of the system might argue that they ought to be granted a patent for the design, since they own the AI that created it. An avowed opponent might argue that the invention shouldn't be patentable as it had been created by an AI rather than a human. The controversy demonstrates the significance of legal frameworks able to deal with the specific challenges presented by AI generated works.

A possible resolution to this particular issue could be to update intellectual property laws to expressly declare AI-created works as eligible for protection when specific conditions are met. The law might specify that the AI output has to be novel, non-obvious, and must have industrial potential, just like human-made inventions. This kind of strategy will guarantee enough legal safeguards for AI created works while preserving the essential intellectual property principles.

4.3.3.2. Privacy Concerns in AI-Driven Product Development

AI technologies utilized in product development process raise privacy concerns among the already cited legal issues. As AI often rely on huge amounts of data to work efficiently, companies must ensure their AI enabled development processes comply with relevant data protection laws and regulations, like the General Data Protection Regulation (GDPR) of the European Union.

Businesses must adopt data protection by design and by default principles when developing AI-powered solutions to deal with privacy issues. Systems must be designed to enhance privacy and data processing activities must be limited to the minimum needed for the intended purpose only.

Anonymization methods are a great way to improve security by getting rid of personally identifiable information from datasets. A company could use k-anonymization, which entails separating records in a dataset into clusters ("k-groups") with the aim of making them indistinguishable based on a set of specified attributes. This is mathematically represented as:

$$P(A = a \mid B = b) = P(A = a \mid B = b, C = c)$$

A, B, and C refer to attributes of the dataset, and a, b, and c stand for specific values of those attributes. Based on this equation, the probability of attribute A obtaining the value a when attribute B contains the value b is equal to the likelihood of attribute A obtaining the value a when attributes B and C have the values b and c, respectively. K-anonymization ensures that people cannot be uniquely identified depending on the combination of their attributes by satisfying this condition.

4.3.3.3. Liability Issues in AI-Driven Product Development

AI-driven systems also raise liability concerns when decisions or recommendations result in faulty products or other undesirable outcomes. The idea of negligence is commonly found in conventional liability frameworks, and the notion will involve proving that a human actor violated a duty of care, leading to damage to another party. Nevertheless, these frameworks might not be enough for situations where an AI system makes a harmful choice instead of a human decision maker.

Companies should think about applying risk management strategies to deal with the specific issues posed by AI systems to solve liability concerns. For instance, in AI-driven decision-making processes, robust testing and validation of AI tools and systems might be necessary, together with clear human oversight and intervention.

Furthermore, legal frameworks might need to be rewritten to better reflect the actual application of AI systems. One possible strategy might be implementing a strict liability framework for AI driven product development, where businesses are responsible for any damage caused by their AI systems, regardless of negligence. This approach would supply companies with an enormous incentive to guarantee the safety and reliability of their AI driven product development processes, while simultaneously establishing a clear framework for addressing liability issues.

4.3.3.4. Regulatory Compliance in AI-Driven Product Development

The legal aspect of AI is essential due to the implementation of new regulations by governments around the globe to address the specific risks and problems related to AI solutions. All companies should ensure their AI driven product development processes meet applicable polices, like algorithmic transparency, explainability, and fairness.

The Artificial Intelligence Act (AIA) proposed by the European Union specifies specific requirements for AI systems, including transparency and accountability measures, and specific provisions for high-risk AI systems. In the EU, businesses will need to ensure their AI tools and solutions meet these standards, which might require new processes and procedures for system documentation, testing, and validation.

Companies should create dedicated teams to monitor and ensure compliance with relevant AI regulations to address regulatory compliance challenges and keep abreast of new regulatory developments. Moreover, companies should think about partnering with other stakeholders, academic institutions, and industry associations to push for clear, consistent, and evidence-based laws that deal with the various challenges and risks presented by AI technologies, while encouraging ongoing innovation.

4.4. Integration

Integrating AI technologies into existing product development processes and systems can be challenging due to factors such as retro-compatibility, upgradability, migration, and edge computing.

4.4.1. Retro-Compatibility

Among the primary hurdles of incorporating AI technologies into current development processes there is guaranteeing retro-compatibility with older systems. Many hardware and software tools are utilized by companies, some of which might be outdated or even incompatible with current AI solutions. To avoid disruptions in development processes and minimize costs related to system upgrades or replacements, it's crucial that new AI tools seamlessly integrate with existing systems.

AI technologies developed with backwards compatibility in mind must be considered by companies to deal with retro - compatibility problems, enabling them to work with a wide spectrum of legacy systems. Furthermore, companies must concentrate on purchasing or building AI tools that meet industry standards and best practices for a smoother integration. This section will go deeper into strategies that companies can adopt to ensure a smooth integration process, including the usage of adapters, middleware, and API based integration.

4.4.1.1. Adapters

One of the common strategies for integrating AI tools with legacy systems is the use of adapters. Adapters are software components designed to enable communication and data exchange between systems that might not be natively compatible. They act as a bridge between the legacy system and the new AI technology, translating data and commands between the two systems in a format that each can understand.

For example, consider a legacy product design software that stores data in a proprietary file format. An adapter can be developed to convert the data from this proprietary format to a more standard format, such as JSON or XML, which can then be easily processed by the AI tool.

Mathematically, an adapter can be represented as a function that maps the input from one system to the output required by another system. Let's denote the adapter function as $A(x)$, where x is the input from the legacy system, and $A(x)$ is the output required by the AI tool. If the legacy system produces an output L, the adapter will process this output and generate $A(L)$ for the AI tool to consume.

4.4.1.2. Middleware

Another strategy for ensuring retro compatibility is the use of middleware. Middleware is a software layer that sits between the AI tool and the legacy system, facilitating communication and data exchange between the two. Middleware can help manage the complexity of integrating AI tools with multiple legacy systems, providing a unified interface for data processing and communication.

Middleware can be designed to handle various tasks, such as data transformation, protocol conversion, and message routing. In the context of AI integration, middleware can be utilized to pre-process data from legacy systems, transforming it into a format suitable for AI processing, and then routing the AI-generated insights back to the legacy system for further action or analysis.

One common approach to implementing middleware in AI integration scenarios is the use of service-oriented architecture (SOA). SOA is a design paradigm that encourages the development of loosely coupled, modular software components that can be easily combined and reused across different systems. By adopting an SOA-based approach, companies can develop middleware components that are both flexible and scalable, enabling seamless integration of AI tools with a wide range of legacy systems.

4.4.1.3. API-based Integration

API-based integration is another strategy to ensuring retro compatibility between AI tools and older systems. The Application Programming Interfaces (APIs) are a collection of guidelines and protocols that govern the interaction and communication between software components. By developing APIs for both the AI tool and the legacy system, companies can facilitate data exchange and interaction between the two systems in a standardized and efficient way.

RESTful APIs and GraphQL APIs might be created for API based integration. RESTful APIs are built on the principles of Representational State Transfer (REST), a software architectural style which emphasizes standardized, stateless communication protocols over the web. GraphQL is a query language and runtime for APIs that enables clients to request only the information they require, lowering the amount of information transmitted between the server and the client.

By applying API based integration methods, companies can ensure that their AI tools can easily integrate with a number of legacy systems while not needing custom adapters or middleware. Making use of APIs, which can be quickly updated or extended to accommodate new functions or capabilities as AI technologies develop, may help ensure future proofing of AI integration efforts.

4.4.2. Upgradability

As AI technologies continue to develop at a fast pace, it's essential to ensure that AI-enabled processes are current and able to exploit the latest developments. Companies must ensure their AI tools and systems are easily upgradeable, so that new features, improvements, or bug fixes can be integrated quickly.

Companies must focus on using modular AI tools and solutions to enable easy updates and upgrades to address upgradability issues. The management and deployment of AI applications and tools in a variety of environments will be made easier by using containerization technologies like Kubernetes or Docker.

4.4.2.1. Modular Design and Architecture

One key aspect of ensuring upgradability in AI tools and systems is the adoption of a modular design and architecture. A modular approach involves designing AI systems in such a way that they are composed of multiple independent components, or modules, which can be easily updated or replaced without impacting the overall system functionality.

In the context of AI-supported product development, this could involve designing AI tools with a clear separation between core algorithms, data processing components, and user interfaces. This separation can facilitate the independent upgrade of individual components, allowing for the rapid incorporation of new AI techniques or improvements without necessitating a complete overhaul of the system.

For example, an AI system designed for generative design might have separate modules for the optimization algorithm, the geometry generation, and the visualization of the generated designs. When a new optimization algorithm is developed, it could be easily integrated into the system by replacing the corresponding module, without affecting the other modules or the overall system functionality.

4.4.2.2. Containerization Technologies

The upgradability of AI systems could be facilitated by containerization technologies like Docker and Kubernetes. AI tools and applications can be deployed, managed, and scaled properly across different environments using these technologies, while simultaneously being isolated from the underlying system and other tools or applications.

Companies can develop a standardized and portable environment that can be updated, replicated, or migrated between platforms or infrastructure by packaging tools and applications into containers. The process of upgrading systems could be made considerably easier by replacing the corresponding containers with newer versions without impacting the underlying system.

A demand forecasting company using a containerized tool might separate containers for its underlying machine learning model, pre-processing scripts, and data pipelines. When an updated version of the model is developed, it may be deployed by replacing the appropriate container without affecting other containers or system functionality.

4.4.2.3. Version Control and Continuous Integration

Another essential aspect of maintaining upgradability in AI systems is the implementation of robust version control and continuous integration practices. Version control systems, such as Git, enable the tracking of changes, making it easier to identify and address potential issues, rollback to previous versions, or integrate new features or improvements.

Continuous integration practices involve the automatic testing, building, and deployment of AI tools and applications whenever changes are made to the underlying source code. By implementing continuous integration pipelines, companies can ensure that their AI systems are always up-to-date and that any issues or inconsistencies introduced by updates or changes are quickly identified and resolved.

To illustrate, consider an AI system used for natural language processing. The system might include components for text pre-processing, feature extraction, and machine learning model training. By using version control and continuous integration practices, the development team can easily manage the evolution of the system, ensuring that each component is up-to-date, and that new features or improvements are quickly incorporated and tested.

4.4.3. Migration

Migration is a crucial aspect when integrating AI technologies into today's processes, as it involves transitioning data, applications, or systems from one environment to another. One example is migrating from on-premises infrastructure to cloud-based platforms, which offer more scalability and flexibility for AI workloads. Successful migration is essential for ensuring seamless integration of these tools into existing workflows and minimizing disruptions to ongoing operations.

To understand the migration process, it is helpful to break it down into several key stages:

- *Assessment*: The first stage in the migration process involves evaluating the current state of the data, applications, or systems that need to be migrated. This includes identifying dependencies, constraints, and potential risks, as well as determining the desired end state of the migration. In this stage, development teams should conduct a thorough analysis of their existing infrastructure, data formats, and workflows, considering factors such as security, performance, and compatibility.
- *Planning*: Once the assessment is complete, the next stage involves creating a detailed migration plan that outlines the specific steps, timelines, and resources required for the migration. This plan should address key aspects such as data migration, application migration, system migration, and user migration, as well as any necessary changes to infrastructure, hardware, or software. The planning stage may involve the use of mathematical models and optimization techniques to determine the most efficient migration strategy.
- *Preparation*: In the preparation stage, the development team must establish the necessary environment, tools, and resources for the migration. This may include setting up new infrastructure, configuring software and hardware, and developing custom scripts or tools to facilitate the migration process. Additionally, teams should establish a robust testing and validation framework to ensure the quality and integrity of the migrated data, applications, or systems.
- *Execution*: With the environment and resources in place, the migration process can be executed, following the steps outlined in the migration plan. This may involve transferring data between storage systems or databases, re-platforming applications, or moving systems to new hardware or virtual environments. During the execution stage, it is crucial to monitor the migration process and address any issues or bottlenecks that arise promptly. This may require adjusting the migration strategy or making real-time decisions based on observed performance or resource utilization.
- *Validation*: After the migration has been executed, the validation stage involves verifying that the migrated data, applications, or systems are functioning correctly and meet the desired end state. This may involve conducting rigorous testing, comparing the migrated data or system outputs with the original environment, and validating that all dependencies and constraints have been addressed. In some cases, this stage may

require the use of statistical analysis or machine learning algorithms to compare the performance of the migrated system with the original system and ensure that it meets the desired quality criteria.

- *Optimization*: Once the migrated data, applications, or systems have been validated, the final stage in the migration process is optimization. This involves identifying opportunities to improve performance, streamline workflows, or reduce costs in the new environment. This may include refining data structures, tuning algorithms, or adjusting infrastructure settings to optimize resource utilization and performance.

By understanding the migration process and its key stages, teams can successfully integrate AI technologies into their existing workflows and systems. This can help them unlock the full potential of AI in supporting design and innovation processes while minimizing disruptions and ensuring the continued success of their projects.

4.4.4. Edge Computing

The emerging technology of edge computing is a promising answer to the increasing demand for analysing and processing real time data from connected devices and systems, like in the IoT. Edge computing speeds up response times, lowers network congestion, and helps to boost efficiency, especially for applications involving real time analysis, autonomous decision making, or control of physical systems.
To fully understand the idea of edge computing one must contrast it with the traditional cloud computing model. In cloud computing, data is moved from the source products to centralized data centers for storage and processing. Even though this particular approach provides many benefits, like scaling and management simplicity, it can lead to higher latency and inefficiencies as a lot of information has to be transmitted over the network. Nevertheless, edge computing decentralizes data processing by executing computations on devices situated close to the data source (sensors, embedded systems, or edge servers). Consequently, data must be transmitted less frequently through the network leading to lower latency, decreased network congestion and much better utilization of resources.

The following equation for latency offers a visual illustration of the concept of edge computing:

$$Latency = Propagation\ Delay + Transmission\ Delay + Processing\ Delay + Queuing\ Delay$$

Edge computing aims to reduce the total latency by lowering propagation delay and transmission delay components. It may significantly quicken processing data closer to the source, causing reduced latency and more efficient data processing. Implementing edge computing may involve several key considerations, such as adapting existing infrastructure or deploying new hardware, such as edge servers, to support edge computing capabilities; developing or modifying software to enable edge computing, including the implementation of specialized algorithms and techniques optimized for edge environments; implementing strategies for managing data across edge and cloud environments, including data synchronization, consistency, and storage. Moreover, ensuring the security and privacy of data processed at the edge, as well as addressing potential vulnerabilities introduced by edge devices and networks, is mandatory along with balancing the energy consumption of edge devices and systems with the performance benefits of edge computing. One example of edge computing in product development is the use of edge devices for real-time monitoring and control of manufacturing processes. By processing sensor data and making control decisions locally at the edge, these systems can respond more quickly to changes in process conditions, leading to improved product quality and reduced waste. Another example is the use of edge computing for collaborative robotics in product assembly processes. By processing data from sensors and cameras locally, collaborative robots can quickly adapt to changes in their environment, such as the movement of human workers, ensuring safe and efficient operation.

Conclusions

In conclusion, the application of artificial intelligence (AI) tools in product development has demonstrated great potential to enhance efficiency, stimulate innovation and boost competitiveness in a broad range of industries. Product development specialists can make educated decisions about how to integrate AI tools into their processes and workflows by understanding the different AI technologies, their applications, and challenges. As AI continues to develop and mature, the possibilities for its application will only continue to expand, offering exciting possibilities for innovation and transformation in the industry.

Throughout the investigation of the product development process, it's become evident that the different phases, from planning and concept development to system-level design, detailed design, testing and refinement, and finally, production ramp-up, play crucial roles in the overall success of a product. By thoroughly managing and coordinating these phases, product development teams can make sure the smooth progression of projects from the very beginning to market launch. Furthermore, the integration of operation research methods and the emphasis on cognitive activities involved in product development have highlighted the importance of leveraging analytical techniques and cognitive skills to improve decision - making, optimize processes, and eventually, generate much more innovative and valuable products.

The most recent AI technologies examined in this thesis - machine learning, neural networks, evolutionary computation, computer vision, robotics, expert systems, speech processing, natural language processing, planning, and large language models - have demonstrated great potential when paired with product development. Product development professionals can evaluate the potential of AI tools and techniques to guide their design and innovation processes, resulting in improved results and more competitive products and services by understanding the latest advancements in AI and the underlying principles of these solutions.

This thesis has analysed the link between AI and product development and highlighted the possibilities generated by AI technologies in supporting design and innovation activities: machine learning techniques, particularly supervised learning, unsupervised learning, and reinforcement learning, have emerged as pivotal when *identification* activities are required. Typically, that occurs during product planning since designers have to identify the feature of the new products and in the phase of concept development which involves identifying potential product features, designs, and solutions that meet the identified requirements and constraints, but also throughout the entire product development process, as new information and insights may emerge at any point, requiring adjustments and adaptations in response. Through these techniques, we can discern intricate patterns and insights from vast, complex datasets, thereby catalyzing the process of identifying key aspects in product development.

Moreover, within the realm of *control* activities and *structuring* activities, several AI methods have showcased their vast potential. Both types of activies play a significant role in each phase of the product development process, though the extent and focus may vary depending on the specific phase. For example, during the planning and concept development phases, control activities may focus on evaluating potential concepts against established criteria. In the detailed design and testing phases, control activities may involve monitoring progress to ensure that designs meet technical specifications and performance requirements. Finally, during the production ramp-up phase, control activities may center on managing resource allocation and production schedules to ensure a successful product launch. Same reasoning applies to structuring activities: in the planning phase, structuring helps in defining project objectives, scope, and resources, while in the concept development and system-level design phases, it supports the organization and synthesis of design alternatives and requirements. In this context, the inclusion of robotics, in tandem with intelligent control systems and autonomous exploration capabilities, can automate routine tasks and foster an environment of precision and efficiency. Simultaneously, expert systems, such as decision support systems and teaching systems, can establish a more structured and controlled setting, aiding in decision-making and enabling a continuous learning process.

The *development* activities, are most prominent in concept development phases when teams concentrate on producing a wide range of possible solutions to identify the most promising concepts to carry forward, and in the detailed design phase when the selected ideas are refined and elaborated upon, and teams work to finalize design

specifications, manufacturing processes, materials, and other crucial aspects of the product. These kinds of activities are characterized by creative problem-solving and ideation, and stand to benefit significantly from the computational power and adaptability of AI technologies. Deep learning and neural networks can process and interpret massive volumes of data, yielding novel solutions that may not be readily apparent. Additionally, evolutionary computation methods such as genetic algorithms and hyperparameter optimization techniques like grid search, random grid search, and Latin hypercube sampling can iteratively enhance designs to optimize performance and efficacy.

The *communication* phases, critical to any development process phase, can leverage the advancements in natural language processing, speech processing, and large language models. It is essential during planning to gather customer requirements, define project objectives and establish expectations among stakeholders. During the concept development stage, it encourages brainstorming, idea generation, and the evaluation of feasible solutions; throughout system - level design and detailed design phases it helps in coordinating cross functional teams, discussing design alternatives, and also dealing with technical problems or constraints. It is essential throughout testing and refinement as well to discuss test results, identify areas for improvement and implement changes based on feedback while during the production ramp - up stage it helps make sure that all parties involved are informed about production schedules, quality standards and progress updates, leading to a successful product launch. AI can aid in understanding, interpreting, and generating human language, thereby enhancing the clarity, precision, and efficiency of information exchange among stakeholders. Concurrently, computer vision technologies can further augment communication by providing visual data analysis and interpretation, opening up more avenues for insightful dialogue. By bridging AI and cognitive science, it's become evident that AI tools can boost cognitive activities in product development, leading to far more efficient and effective processes, and the presentation of commercial software examples demonstrates the practical application of AI in the industry, showcasing its tangible impact.

Nevertheless, it has become strikingly clear that the integration of artificial intelligence into product development, while rife with opportunities, presents a complex landscape of challenges. Unintentional bias, embedded in the data we use, could lead to prejudiced AI results, thereby underlining the critical necessity of thorough bias evaluation and mitigation. As AI technology rapidly advances, we must not overlook the potential implications on our workforce and labor market, underscoring the need for innovative solutions for societal adjustment and re-skilling. Data security and stringent adherence to legal mandates, especially in light of the sensitive nature of product development data, can't be overstated. Furthermore, technical obstacles around the retrofitting of existing systems, future-proofing through upgradability, seamless migration, and edge computing need thoughtful navigation. As we stand at the precipice of an AI-driven era in product development, it is incumbent upon us to adopt a mindful, ethical, and strategic approach, carefully weighing the benefits against the challenges, and forging ahead with caution and foresight.

Bibliography and Sitography

Abernathy, W. J., & Utterback, J. M. (1978). Patterns of industrial innovation. Technology Review, 80(7), 40–47.
Brown, T. (2009). Change by design: How design thinking transforms organizations and inspires innovation. Harper Business.
Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.
Koza, J. R. (1992). Genetic programming: on the programming of computers by means of natural selection. MIT Press.
Lake, B. M., Ullman, T. D., Tenenbaum, J. B., & Gershman, S. J. (2017). Building machines that learn and think like people. Behavioral and Brain Sciences, 40.
McKinsey & Company (2017). Artificial intelligence: The next digital frontier? McKinsey Global Institute.
Sutton, R. S., & Barto, A. G. (2018). Reinforcement Learning: An Introduction. MIT press.
Tegmark, M. (2017). Life 3.0: Being Human in the Age of Artificial Intelligence. Vintage.
Ulrich, K. T., & Eppinger, S. D. (2015). Product Design and Development. McGraw-Hill Education.
Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer.
Kusiak, A. (2018). Innovation: The Interplay of Machine Learning and Human Insight. AI Magazine, 39(2), 15-24.
LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436-444.
Russell, S. J., & Norvig, P. (2016). Artificial intelligence: a modern approach. Malaysia; Pearson Education Limited.
Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.
Al Harbi, S. H., Tidjon, L. N., & Khomh, F. (2023). Responsible Design Patterns for Machine Learning Pipelines. arXiv preprint arXiv:2306.01788.
De Silva, D., & Alahakoon, D. (2021). An Artificial Intelligence Life Cycle: From Conception to Production. arXiv preprint arXiv:2108.13861.
Lu, Q., Zhu, L., Xu, X., Whittle, J., Zowghi, D., & Jacquet, A. (2022). Responsible AI Pattern Catalogue: A Multivocal Literature Review. arXiv preprint arXiv:2209.04963.
Higgins, D. (2020). OnRAMP for Regulating AI in Medical Products. Advanced Intelligent Systems, 2021, 42. doi:10.1002/aisy.202100042.
Higgins, D., & Madai, V. I. (2020). From Bit To Bedside: A Practical Framework For Artificial Intelligence Product Development In Healthcare. Advanced Intelligent Systems, 2020, 52. doi:10.1002/aisy.202000052.
Slominski, A., Muthusamy, V., & Ishakian, V. (2019). Towards Enterprise-Ready AI Deployments Minimizing the Risk of Consuming AI Models in Business Applications. arXiv preprint arXiv:1906.10418.
Peng, H. (2021). A Brief Survey of Associations Between Meta-Learning and General AI. arXiv preprint arXiv:2101.04283.
Schöning, J., & Westerkamp, C. (2023). AI-in-the-Loop -- The impact of HMI in AI-based Application. arXiv preprint arXiv:2303.11508.
Butler, T., Espinoza-Limón, A., & Seppälä, S. (2023). Towards a Capability Assessment Model for the Comprehension and Adoption of AI in Organisations. Journal of AI, Robotics & Workplace Automation, 1 (1), 18-33.
Isaac, E. R. H. P., & Reno, J. (2023). AI Product Security: A Primer for Developers. arXiv preprint arXiv:2304.11087.
Zhu, J., Villareale, J., Javvaji, N., Risi, S., Löwe, M., Weigelt, R., & Harteveld, C. (2021). Player-AI Interaction: What Neural Network Games Reveal About AI as Play. arXiv preprint arXiv:2101.06220.
Komendantskaya, E., Stewart, R., Duncan, K., Kienitz, D., Le Hen, P., & Bacchus, P. (2019). Neural Network Verification for the Masses (of AI graduates). arXiv preprint arXiv:1907.01297.
Rajagopal. A., & Nirmala. V. (2019). Strategies to architect AI Safety: Defense to guard AI from Adversaries. arXiv preprint arXiv:1906.03466.
Xue, Z., Li, R., & Li, M. (2022). Recent Progress in Conversational AI. arXiv preprint arXiv:2204.09719.
d'Avila Garcez, A., & Lamb, L. C. (2020). Neurosymbolic AI: The 3rd Wave. arXiv preprint arXiv:2012.05876.
Butler, T., Espinoza-Limón, A., & Seppälä, S. (2023). Towards a Capability Assessment Model for the Comprehension and Adoption of AI in Organisations. Journal of AI, Robotics & Workplace Automation, 1 (1), 18-33.
Prorok, A. (2018). Graph Neural Networks for Learning Robot Team Coordination. Federated AI for Robotics Workshop, IJCAI-ECAI/ICML/AAMAS 2018.
Vanderelst, D., & Winfield, A. (2016). The Dark Side of Ethical Robots. arXiv preprint arXiv:1606.02583.
Sanneman, L., & Shah, J. A. (2020). Trust Considerations for Explainable Robots: A Human Factors Perspective. arXiv preprint arXiv:2005.05940.
Freedman, R. G. (2022). AI-HRI Brings New Dimensions to Human-Aware Design for Human-Aware AI. AAAI 2022 Fall Symposium Series, Artificial Intelligence for Human-Robot Interaction.
Aman Jantan, Abiodun Esther Omolara, Kemi Victoria Dada, Nachaat AbdElatif Mohamed, Humaira Arshad. (2018). State-of-the-art in artificial neural network applications: A survey. Heliyon, 4(11), e00938.
Yanqing Duan, John E. Edwards, Yogesh K. Dwivedi. (2019). Artificial intelligence for decision making in the era of Big Data – evolution, challenges and research agenda. International Journal of Information Management, 48, 63-71.
Terrence Fong, Illah Nourbakhsh, Kerstin Dautenhahn. (2003). A survey of socially interactive robots. Robotics and Autonomous Systems, 42(3-4), 143-166.
Gupta, S., & George, J. F. (2016). Artificial intelligence for decision making in the era of Big Data – evolution, challenges and research agenda. International Journal of Information Management, 48, 63-71.
Dabbous, A., & Boustani, N. M. (2023). Digital explosion and entrepreneurship education: Impact on promoting entrepreneurial intention for business students. Journal of Risk and Financial Management, 16(1).
https://en.wikipedia.org/wiki/Markov_decision_process
An Example Illustrating the Density-Based DBSCAN Clustering Method... | Download Scientific Diagram : https://www.researchgate.net/figure/An-Example-Illustrating-the-Density-Based-DBSCAN-Clustering-Method-Applied-to-SMLM-Data_fig4_342141592
https://www.statisticshowto.com/hierarchical-clustering/
https://blog.mindmanager.com/decision-tree-diagrams/
https://www.tibco.com/it/reference-center/what-is-a-random-forest
https://www.collimator.ai/reference-guides/what-is-reinforcement-learning
https://www.linkedin.com/pulse/computer-vision-cloud-computing-yash-tavse/
https://www.computershub.org/wp-content/uploads/2015/08/Flow-Chart-Of-New-Product-Development-Process.png

Appendix

Fasi lvl 1	Fasi lvl 2	Fasi lvl 3	Esempio documento formale	Classe	Tecniche risolutive	Formato dati	Fonti dati	Finalità strumento A.I.			Branches	Modelli	Algoritmi	Software Commerciali
Planning	Opportunity identification	Definire variabili tecnologiche	Mappa delle opportunità nello spazio delle diverse variabili	Action structuring	Ricerca tramite google (adesso anche Chat GPT), previsioni mercato, previsione evoluzione tecnologica, PEST analysis...		Database brevetti, riviste specializzate, report/articoli scientifici, indagini statistiche già presenti in letteratura, traiettorie tecnologiche, …	Raccolta dati	Elaborazione dati		Natural Language Processing, Speech processing, Computer vision	Sentiment analysis, Text classification, Competitor analysis, Object tracking, Voice recognition, Speaker identification	Valence Aware Dictionary and sEntiment Reasoner, Support Vector Machines, Named Entity Recognition, Particle Filter algorithm, Google's Speech-To-Text API, Stanford Sentiment Analysis tool, Gaussian Mixture Model-Universal Background Model Algorithm	Microsoft Azure Cognitive Services, Amazon Comprehend, Brandwatch, OpenCV, Dragon NaturallySpeaking, IBM Watson Tone Analyzer, Speaker Recognition System
		Definire variabili di mercato			Template, 5 forze di Porter, SWOT, roadmap ...					Generazione di direzioni				
		Creare charter innovativo	Product innovation charter		Modelli funneling/tournament, Patent scanning, VRIO, focus group, survey	Formati testuali (docx, pdf, txt...), formati dati (xlsx...),formati immagine (jpeg, png, xcf...)	Focus group, brainsotrming, questionari/sondaggi, analisi dei trend dei segmenti, forum appasionati...		Elaborazione dati	Generazione di direzioni				
		Generare e sondare opportunità	Lista grezza opportunità		Real win worth it, Star matrix, BCG matrix, Mc-Kinsey matrix, Ad little mstrix					Generazione di direzioni	Expert systems, Evolutionary computation	Genetic algorithms, Delft Design Method	Binary Genetic Algorithm, Case-based Reasoning, Fuzzy Logic, Multi-Objective Genetic Algorithm	GenOpt, GE Digital's Predix, Delft Toolkit, ANSYS
		Selezionare opportunità	Lista cernita		AHP, Electre, Olistiche, tutti quei metodi che vanno a fare analisi multicriteri		Interviste più approfondite, focus group, brainsotrming, questionari, osservazioni dirette del cliente, resoconto concept prototype (virtuale o fisico)			Generazione di direzioni				
		Selezionare opportunità promettenti	Lista valutata superficialmente							Generazione di direzioni				
		Selezionare opportunità eccezionali	Lista "finale"											
	Product planning	Valutare e priorizzare le diverse opportunità di progetto	Portfoli di progetti		NPV calc,Payback period, ECV (expected commercial value), AHP (analisi multicriteri)	Formati gestione dati (xlsx...), Formati legati a software project mangement (mpp...)	Traiettorie tecnologiche, Product process change matrix, product technology roadmap, segment map, screening matrix,			Generazione di direzioni	Evolutionary computation, Planning	Genetic algorithms, Swarm intelligence optimization, Scheduling	Tabu Search Algorithm, Bynary Genetic Algorithm, Real-Coded Genetic Algorithm, Particle Swarm Optimization, Multi-Objective Genetic Algorithm	OptaPlanner, GenOpt, MATLAB, modeFRONTIER, DesignXplorer, ANSYS
		Definire una schedule di progetto, allocare risorse e definire rischi	Product plan		CPM (critical path method), PERT (project evaluation and review technique), PDM (precedence diagram method), CCM (metodo catena libeera) CRT, GERT		Aggregate resource planning, WBS/PBS (work/project breakdown structure), Gaant, AOA/AON , ResourceBS, RiskBS	Raccolta dati	Elaborazione dati	Generazione di direzione				
		Definizione della direzione complessiva di progetto	Mission statement/Visione di progetto				Resoconti/verbali delle attività precedenti							
	Identifying customer needs	Raccogliere raw data	Lista customer statements		Focus group, direct observation, scenarios of ouse, user trail, CRM feedback...	Formati testuali (docx, pdf, txt...), formati dati (xlsx...)	Questionari, resoconti interviste/focus groups, pareri associazioni di settore	Raccolta dati			Computer vision, Natural Language Processing, Speech processing	Requirements Gathering, Sentiment analysis, Text classification, Chatbots, Object tracking, Voice Recognition, Speaker identification	Latent Dirichlet Allocation, Valence Aware Dictionary and sEntiment Reasoner, Support Vector Machines, Natural Language Understanding, Particle Filter algorithm, Google's Speech-To-Text API, Stanford Sentiment Analysis tool, Gaussian Mixture Model-Universal Background Model Algorithm	IBM Rational DOORS, Microsoft Azure Cognitive Services, Amazon Comprehend, Diagflow, OpenCV, Dragon NaturallySpeaking, IBM Watson Tone Analyzer, Speaker Recognition System
		Interpretare e tradurre in customer needs	Lista customer needs		Linee guida sull'interpretazione degli statements		Lista customer statements		Elaborazione dati					
		Gerarchizzare i needs	Lista primary, secondary e tertiary needs		Cluster analysis, Factorial analysis				Elaborazione dati		Evolutionary computation	Genetic algorithms	Binary Genetic Algorithm, Real-Coded Genetic Algorithm, Multi-Objective Genetic Algorithm	GenOpt, MATLAB, ANSYS
		Definire il peso/ l'importanza relativa dei diversi needs	Lista finale dei needs		Interviste/questionari molto specifici					Generazione di direzione	Evolutionary computation	Swarm intelligence optimization	Particle Swarm Optimization	modeFRONTIER, DesignXplorer
Concept development	Product specifications	Creare una prima lista di specifications su designata scala	Lista grezza specifications	Action development	QFD, Needs-metric matrix, Parametric analysis, feedback analysis, Electre II		Liste needs e customer statement, lista prodotti concorrenza	Raccolta dati	Elaborazione dati		Computer vision, Speech processing, Natural Language Processing, Neural Networks	Principal Component Analysis, Q-Learning, Sentiment analysis, Text classification, Recurrent Neural Networks, Object tracking, Voice recognition, Speaker identification, Convolutional Neural Networks	Singular Value Decomposition, State-Action-Reward-State-Action, Valence Aware Dictionary and sEntiment Reasoner, Support Vector Machines, Backpropagation Through Time, Particle Filter algorithm, Google's Speech-To-Text API, Stanford Sentiment Analysis tool, Gaussian Mixture Model-Universal Background Model Algorithm	Scikit-learn, PyTorch, Microsoft Azure Cognitive Services, Amazon Comprehend, TensorFlow, OpenCV, Dragon NaturallySpeaking, IBM Watson Tone Analyzer, Speaker Recognition System, Keras
		Legarle opportunamente ai needs (cercando correlazioni)	Lista correlata di specifications							Generazione di direzione				
		Fissare estremi dell'intervallo di accettabilità per le diverse specifiche	Lista target specifications		Benchmark analysis	Formati testuali (docx, pdf, txt...), formati dati (xlsx...)	Report dettagliato sulla concorrenza rispetto ad una serie di specifications		Elaborazione dati	Generazione di direzione	Neural Networks, Evolutionary computation	Feedforward Neural Networks, Genetic Algorithms	Binary Genetic Algorithm, Real-Coded Genetic Algorithm, Multi-Objective Genetic Algorithm	GenOpt, MATLAB, ANSYS
		Sviluppare un modello tecnico del prodotto	Product champion, External decision, Intuition, Hierarchy decomposition,		Simulazioni virtuali o al banco prova		Storico dati prove prototipi precedenti, prodotti simili	Raccolta dati	Elaborazione dati	Generazione di direzione	Computer vision, Speech processing, Natural Language Processing, Neural Networks, Evolutionary computation, Expert systems	Deep Q-Network, Q-Learning, Competitor analysis, Recurrent Neural Networks, Feedforward Neural Networks, Genetic Algorithms, GE Design-Advisor	Experience Replay, State-Action-Reward-State-Action, Named Entity Recognition, Backpropagation Through Time, Binary Genetic Algorithm, Real-Coded Genetic Algorithm, Case-based Reasoning, Multi-Objective Genetic Algorithm	TensorFlow, PyTorch, Brandwatch, IBM Watson Studio, GenOpt, MATLAB, GE Digital's Predix, ANSYS, Keras
		Sviluppare un modello di coSto del prodotto	Modello di costo		Quantitative (Target costing, parametric costing, activity costing, life cycle costing), Qualitative (tecniche per analogia), analisi marginalità/redditività		BoM, Cicli di produzione, Cicli di vita di prodotti simili	Raccolta dati	Elaborazione dati	Generazione di direzione				

		Rifinire e eseguire ulteriori trade-offs tra target specifications	Modelli di confronto	Analisi multicriteri, benchmark analysis		Competitive map, analisi di redditività attesa nei diversi segmenti di mercato		Elaborazione dati	Generazione di direzione	Natural Language Processing, Neural Networks	Principal Component Analysis, Decision Tree, Q-Learning, Text classification, Recurrent Neural Networks, Feedforward Neural Networks	Singular Value Decomposition, Classification and Regression Trees, State-Action-Reward-State-Action, Support Vector Machines, Backpropagation Through Time	Scikit-learn, Weka, PyTorch, Amazon Comprehend, TensorFlow, Keras, IBM Watson Studio
		Far ricadere le specifiche di sistema nei sottosistemi (solo prodotti complessi)	Final list of specifications					Elaborazione dati	Generazione di direzione	Neural Networks, Evolutionary computation	Principal Component Analysis, Decision Tree, Recurrent Neural Networks, Feedforward Neural Networks, Genetic Algorithms	Singular Value Decomposition, Classification and Regression Trees, Backpropagation Through Time, Binary Genetic Algorithm, Real-Coded Genetic Algorithm, Multi-Objective Genetic Algorithm	Scikit-learn, Weka, TensorFlow, Keras, IBM Watson Studio, ANSYS, MATLAB, GenOpt
		Chiarire e comprendere il problema che si vuole risolvere	Diagrammi funzionali	Functional analysis, value analysis		Rodenacker diagrams, FAST diagrams, Functional trees,		Elaborazione dati	Generazione di direzione	Neural Networks, Evolutionary computation	Principal Component Analysis, Decision Tree, Recurrent Neural Networks, Feedforward Neural Networks, Genetic Algorithms	Singular Value Decomposition, Classification and Regression Trees, Backpropagation Through Time, Binary Genetic Algorithm, Real-Coded Genetic Algorithm, Multi-Objective Genetic Algorithm	Scikit-learn, Weka, TensorFlow, Keras, IBM Watson Studio, ANSYS, MATLAB, GenOpt
	Concept generation	Ricerca esterna	Report potenziali soluzioni ext da implementare	Patent scanning/screening, expert interview, literature research, survey	Formati testuali (docx, pdf, txt...), formati dati (xlsx...), formati immagine (jpeg, png, xlm...), formato CAD (step, dxf, ipt, dwg, stp...)	Database brevetti, conferenze, report di mercato, riviste/associazione consumatori, raccolte di progetti, handbooks (=manuali tecnicissimi), forum amatori e professionisti, prodotti concorrenza e non (reverse engineering).	Raccolta dati	Elaborazione dati		Computer vision, Natural Language Processing, Speech processing, Neural Networks	Sentiment analysis, Text classification, Competitor analysis, Object tracking, Voice recognition, Speaker identification, Random Forest, Principal Component Analysis, Decision Tree, K-Means Clustering, Recurrent Neural Networks, Feedforward Neural Networks, Convolutional Neural Networks	Valence Aware Dictionary and sEntiment Reasoner, Support Vector Machines, Named Entity Recognition, Particle Filter algorithm, Google's Speech-To-Text API, Stanford Sentiment Analysis tool, Gaussian Mixture Model-Universal Background Model Algorithm, Breiman's Algorithm, Singular Value Decomposition, Classification and Regression Trees, Lloyd's Algorithm, Backpropagation Through Time	Microsoft Azure Cognitive Services, Amazon Comprehend, Brandwatch, OpenCV, Dragon NaturallySpeaking, IBM Watson Tone Analyzer, Speaker Recognition System, R, Scikit-learn, Weka, TensorFlow, IBM Watson Studio, Keras
		Ricerca interna	Lista concept di soluzioni interne	Brainstorming (e sue varianti), biomimetics, gamification, ricerca database interni ...		database interni azienda (esperienza pregressa, registrata)	Raccolta dati	Elaborazione dati					
		Esplorazione sistematica delle soluzioni (e effetto di una loro combinazione)	**Lista di soluzioni integrate**	Concept combination tables, Concept classification tree, analisi multicriterio		digrammi funzionali, delle componenti, bozzetti di soluzioni specifiche, risultati di test condotti..			Generazione di direzione	Evolutionary computation, Planning	GamePlaying, Genetic Algorithms, Swarm intelligence optimization	Rapid Action Value Estimation Algorithm, Binary Genetic Algorithm, Real-Coded Genetic Algorithm, Particle Swarm Optimization, Multi-Objective Genetic Algorithm	Siemens' Simcenter Amesim, GenOpt, MATLAB, DesignXplorer, ANSYS
	Concept selection	Concept screening	Lista preliminare concept definitivi	Multivoting (AHP), Multiobject optimization, Collaborative optimization methods, Structured Group Management techniques, Web method survey, decision matrix, **selection matrix**	Formati testuali (docx, pdf, txt...), formati dati (xlsx...), formati immagine (jpeg, png, xlm...), formato CAD (step, dxf, ipt, dwg, stp...)	Possono essere usati metodi benchmark tra competitor (confronti HoQ (QFD)), come anche i concepts trees/tables			Generazione di direzione	Neural Networks, Evolutionary computation	Principal Component Analysis, Decision Tree, Recurrent Neural Networks, Feedforward Neural Networks, Genetic Algorithms	Singular Value Decomposition, Classification and Regression Trees, Backpropagation Through Time, Binary Genetic Algorithm, Real-Coded Genetic Algorithm, Multi-Objective Genetic Algorithm	Scikit-learn, Weka, TensorFlow, Keras, IBM Watson Studio, ANSYS, MATLAB, GenOpt
		Valutare concept	Lista finale concepts definitivi	Product champion, External decision, intuition (boh), Decomposion							Principal Component Analysis, Decision Tree, Recurrent Neural Networks, Feedforward Neural Networks, Genetic Algorithms	Singular Value Decomposition, Classification and Regression Trees, Backpropagation Through Time, Binary Genetic Algorithm, Real-Coded Genetic Algorithm, Multi-Objective Genetic Algorithm	Scikit-learn, Weka, TensorFlow, Keras, IBM Watson Studio, ANSYS, MATLAB, GenOpt
		Definire lo scopo del/dei test	Panoramica del survey			Risultati test analoghi	Raccolta dati	Elaborazione dati	Generazione di direzione	Natural Language Processing, Speech processing, Computer vision, Neural Networks, Evolutionary computation	Sentiment analysis, Text classification, Competitor analysis, Object tracking, Voice recognition, Speaker identification, Random Forest, Principal Component Analysis, Decision Tree, K-Means Clustering, Recurrent Neural Networks, Feedforward Neural Networks, Convolutional Neural Networks, , Genetic Algorithms, Swarm intelligence optimization	Valence Aware Dictionary and sEntiment Reasoner, Support Vector Machines, Named Entity Recognition, Particle Filter algorithm, Google's Speech-To-Text API, Stanford Sentiment Analysis tool, Gaussian Mixture Model-Universal Background Model Algorithm, Breiman's Algorithm, Singular Value Decomposition, Classification and Regression Trees, Lloyd's Algorithm, Backpropagation Through Time, Binary Genetic Algorithm, Real-Coded Genetic Algorithm, Particle Swarm Optimization, Multi-Objective Genetic Algorithm	Microsoft Azure Cognitive Services, Amazon Comprehend, Brandwatch, OpenCV, Dragon NaturallySpeaking, IBM Watson Tone Analyzer, Speaker Recognition System, R, Scikit-learn, Weka, TensorFlow, IBM Watson Studio, Keras, GenOpt, MATLAB, DesignXplorer, ANSYS
	Concept testing	Definire il campione di riferimento del target market	Schema del survey	Test statistici su popolazione/campione. Tecniche di clustering/valutazione delle correlazioni tra campioni di sottoinsiemi/segmenti	Formati testuali (docx, pdf, txt...), formati dati (xlsx...), formati immagine (jpeg, png, xlm...), formato CAD	Buyer persona, customer journey, analisi dei potenziali segmenti di mercato eseguite nelle fasi di planning		Elaborazione dati		Neural Networks			
		Definire la tipologia/formato di survey	Modello survey					Elaborazione dati		Expert systems	Delft Design Method	Fuzzy Logic	Delft Toolkit
		Raccogliere risposte ed interpretare i risultati generali	Resoconto indagine				Raccolta dati	Elaborazione dati	Generazione di direzione	Evolutionary computation, Planning	GamePlaying, Genetic Algorithms, Swarm intelligence optimization	Rapid Action Value Estimation Algorithm, Binary Genetic Algorithm, Real-Coded Genetic Algorithm, Particle Swarm Optimization, Multi-Objective Genetic Algorithm	Siemens' Simcenter Amesim, GenOpt, MATLAB, DesignXplorer, ANSYS
		Definire la WTP	Pricing model	Van Westendorp, pricing methods					Generazione di direzione	Evolutionary computation, Planning	GamePlaying, Genetic Algorithms, Swarm intelligence optimization	Rapid Action Value Estimation Algorithm, Binary Genetic Algorithm, Real-Coded Genetic Algorithm, Particle Swarm Optimization, Multi-Objective Genetic Algorithm	Siemens' Simcenter Amesim, GenOpt, MATLAB, DesignXplorer, ANSYS
System level design	Scelta architettura prodotto	Creazione schema globale di prodotto	Schema di prodotto	Interaction matrix, FAST/Rodenacker diagram,	Formati testuali (docx, pdf, txt...), formati dati (xlsx...), formati immagine (jpeg, png, xlm...), formato CAD (step, dxf, ipt, dwg, stp...)	Diagrammi funzionali di diversp tipo, verbali/resoconti dei diversi team di lavoro, concepts trees/tables	Raccolta dati	Elaborazione dati		Natural Language Processing, Speech processing, Computer vision, Neural Networks	Sentiment analysis, Text classification, Competitor analysis, Object tracking, Voice recognition, Speaker identification, Random Forest, Principal Component Analysis, Decision Tree, K-Means Clustering, Recurrent Neural Networks, Feedforward Neural Networks, Convolutional Neural Networks	Valence Aware Dictionary and sEntiment Reasoner, Support Vector Machines, Named Entity Recognition, Particle Filter algorithm, Google's Speech-To-Text API, Stanford Sentiment Analysis tool, Gaussian Mixture Model-Universal Background Model Algorithm, Breiman's Algorithm, Singular Value Decomposition, Classification and Regression Trees, Lloyd's Algorithm, Backpropagation Through Time	Microsoft Azure Cognitive Services, Amazon Comprehend, Brandwatch, OpenCV, Dragon NaturallySpeaking, IBM Watson Tone Analyzer, Speaker Recognition System, R, Scikit-learn, Weka, TensorFlow, IBM Watson Studio, Keras
		Raggruppamento elementi di schema	Schema di prodotto raggruppato					Elaborazione dati		Neural Networks			
		Creazione layout geometrico grezzo di prodotto	Layout/modello geometrico prodotto					Elaborazione dati	Generazione di direzione	Neural Networks, Evolutionary computation, Planning	Principal Component Analysis, Decision Tree, Recurrent Neural Networks, Feedforward Neural Networks, Genetic Algorithms, GamePlaying, Swarm intelligence optimization	Singular Value Decomposition, Classification and Regression Trees, Backpropagation Through Time, Binary Genetic Algorithm, Real-Coded Genetic Algorithm, Multi-Objective Genetic Algorithm, Rapid Action Value Estimation, Particle Swarm Optimization	Scikit-learn, Weka, TensorFlow, Keras, IBM Watson Studio, ANSYS, MATLAB, GenOpt, Siemens' Simcenter Amesim, DesignXplorer, modeFRONTIER
		Identificazione interazione fondamentali/critiche e di quelle problematiche/indesiderate	Lista delle interazioni tra i raggruppamenti	Interaction matrix, interaction graph					Generazione di direzione	Evolutionary computation			

	Risoluzione trade-off differenziazione vs accomunamento	Valutazione delle influenze delle scelte dell'architettura su supply chain e manufacturing process			Commonality/differentiation plan		Flow chart prodotto nelle diverse fasi di produzione	Raccolta dati	Elaborazione dati		Computer vision, Natural Language Processing, Speech processing, Neural Networks	Sentiment analysis, Text classification, Competitor analysis, Object tracking, Voice recognition, Speaker identification, Random Forest, Principal Component Analysis, Decision Tree, K-Means Clustering, Recurrent Neural Networks, Feedforward Neural Networks, Convolutional Neural Networks	Valence Aware Dictionary and sEntiment Reasoner, Support Vector Machines, Named Entity Recognition, Particle Filter algorithm, Google's Speech-To-Text API, Stanford Sentiment Analysis tool, Gaussian Mixture Model-Universal Background Model Algorithm, Breiman's Algorithm, Singular Value Decomposition, Classification and Regression Trees, Lloyd's Algorithm, Backpropagation Through Time	Microsoft Azure Cognitive Services, Amazon Comprehend, Brandwatch, OpenCV, Dragon NaturallySpeaking, IBM Watson Tone Analyzer, Speaker Recognition System, R, Scikit-learn, Weka, TensorFlow, IBM Watson Studio, Keras
		Valutare impatto dell'aggiunta di una variante di prodotto sui profitti attesi						Raccolta dati	Elaborazione dati		Computer vision, Natural Language Processing, Speech processing, Neural Networks			
		Risoluzione dei trade-off												
Detailed design	Rifinitura singole componenti/subsystem	Definire ogni caratterstica dei sottosistemi secondari e sue componenti Definire l'architettura dei sottosistemi e definire ogni caratteristica delle componenti del sottoinsieme. Creare specifiche di interfaccia dei sottosistemi	Modello completo dei sottosistemi e delle componenti				Diagrammi funzionali di diverso tipo, verbali/resoconti dei diversi team di lavoro, concepts trees/tables		Elaborazione dati	Generazione di direzione	Neural Networks, Evolutionary computation	Principal Component Analysis, Decision Tree, Recurrent Neural Networks, Feedforward Neural Networks, Genetic Algorithms	Singular Value Decomposition, Classification and Regression Trees, Backpropagation Through Time, Binary Genetic Algorithm, Real-Coded Genetic Algorithm, Multi-Objective Genetic Algorithm	Scikit-learn, Weka, TensorFlow, Keras, IBM Watson Studio, ANSYS, MATLAB, GenOpt
		Eliminazione/risoluzione tarde-off specifici			TRIZ									
	Integrazione componenti/subsystem	Revisione dell'integrazione del componente/subsystem nel sistema complessivo						Raccolta dati	Elaborazione dati		Computer vision, Natural Language Processing, Speech processing, Neural Networks	Sentiment analysis, Text classification, Competitor analysis, Object tracking, Voice recognition, Speaker identification, Random Forest, Principal Component Analysis, Decision Tree, K-Means Clustering, Recurrent Neural Networks, Feedforward Neural Networks, Convolutional Neural Networks	Valence Aware Dictionary and sEntiment Reasoner, Support Vector Machines, Named Entity Recognition, Particle Filter algorithm, Google's Speech-To-Text API, Stanford Sentiment Analysis tool, Gaussian Mixture Model-Universal Background Model Algorithm, Breiman's Algorithm, Singular Value Decomposition, Classification and Regression Trees, Lloyd's Algorithm, Backpropagation Through Time	Microsoft Azure Cognitive Services, Amazon Comprehend, Brandwatch, OpenCV, Dragon NaturallySpeaking, IBM Watson Tone Analyzer, Speaker Recognition System, R, Scikit-learn, Weka, TensorFlow, IBM Watson Studio, Keras
		Risoluzione eventuali elementi di conflitto	Modello finale completo dei sottosistemi e delle componenti		TRIZ					Generazione di direzione	Evolutionary computation, Planning, Expert Systems	GamePlaying, Genetic Algorithms, Swarm intelligence optimization, GE Design Advisor, Delft Design Method	Rapid Action Value Estimation Algorithm, Binary Genetic Algorithm, Real-Coded Genetic Algorithm, Particle Swarm Optimization, Multi-Objective Genetic Algorithm, Rule-based Systems, Case-based Systems, Fuzzy Logic	Siemens' Simcenter Amesim, GenOpt, MATLAB, DesignXplorer, ANSYS, Ge Digital's Predix, Delft Toolkit
Prototyping		Definizione degli obiettivi del prototipo	Statement test/prototipo	Choice	DOE (design of experiment) method	Formati testuali (docx, pdf, txt...), formati dati (xlsx...), formati immagine (jpeg, png, xlm...), formato CAD (step, dxf, ipt, dwg, stp...)	Report virtual/ analytical testing software (CAD), resoconti test reali con protipo fisico		Elaborazione dati	Generazione di direzione	Neural Networks, Evolutionary computation, Planning	Principal Component Analysis, Decision Tree, Recurrent Neural Networks, Feedforward Neural Networks, Genetic Algorithms, GamePlaying, Swarm intelligence optimization	Singular Value Decomposition, Classification and Regression Trees, Backpropagation Through Time, Binary Genetic Algorithm, Real-Coded Genetic Algorithm, Multi-Objective Genetic Algorithm, Rapid Action Value Estimation, Particle Swarm Optimization	Scikit-learn, Weka, TensorFlow, Keras, IBM Watson Studio, ANSYS, MATLAB, GenOpt, Siemens' Simcenter Amesim, DesignXplorer, modeFRONTIER
		Stabilire similarità prototipo/prodotto finale atteso												
		Definire un piano d'esecuzione dei test su prototipo	Protocollo test							Generazione di direzione	Neural Networks, Planning, Evolutionary computation, Expert Systems	Principal Component Analysis, Decision Tree, Recurrent Neural Networks, Feedforward Neural Networks, Genetic Algorithms, GamePlaying, Swarm intelligence optimization, GE Design Advisor, Scheduling	Singular Value Decomposition, Classification and Regression Trees, Backpropagation Through Time, Binary Genetic Algorithm, Real-Coded Genetic Algorithm, Multi-Objective Genetic Algorithm, Rapid Action Value Estimation, Particle Swarm Optimization, Rule-based Systems, Case-based Reasoning, Tabu Search Algorithm	Scikit-learn, Weka, TensorFlow, Keras, IBM Watson Studio, ANSYS, MATLAB, GenOpt, Siemens' Simcenter Amesim, DesignXplorer, modeFRONTIER, GE Digital's Predix, OptaPlanner
		Definire un plan/schedule per la realizzazione dei test e del prototipo	Schedule test/prototipo		CPM , PERT , PDM , CCM , CRT, GERT					Generazione di direzione				
		Definire milestone del plan	Roadmap test/prototipo							Generazione di direzione				
Design for manufacturability	Modellizzazione del processo produttivo	Creazione modello di produzione del prodotto	Modello produttivo dettagliato	Action structuring	Modelizzazione virtuale (uso software di tipo CAM), Prove dirette di esecuzione dei cicli produttivi	formati dati (xlsx...), formati immagine (jpeg, png, xlm...), formato CAD (step, dxf, ipt, dwg, stp...), formato CAM (dxf...)	Cicli di lavoro, BOM, Cicli di allocazione		Elaborazione dati	Generazione di direzione	Computer vision, Neural Networks, Evolutionary computation, Planning, Expert systems, Robotics	Principal Component Analysis, Decision Tree, Recurrent Neural Networks, Feedforward Neural Networks, Genetic Algorithms, GamePlaying, Swarm intelligence optimization, GE Design Advisor, Scheduling, Object Recognition, Object Tracking, Path Planning, Simultaneous Localization and Mapping, FLEXpert, Material Selection, Actor-Critic	Singular Value Decomposition, Classification and Regression Trees, Backpropagation Through Time, Binary Genetic Algorithm, Real-Coded Genetic Algorithm, Multi-Objective Genetic Algorithm, Rapid Action Value Estimation, Particle Swarm Optimization, Rule-based Systems, Case-based Reasoning, Tabu Search Algorithm, Advantage Actor-Critic, Viola-Jones algorithm, Particle Filter algorithm, A-star, GraphSLAM	Scikit-learn, Weka, TensorFlow, Keras, IBM Watson Studio, ANSYS, MATLAB, GenOpt, Siemens' Simcenter Amesim, DesignXplorer, modeFRONTIER, GE Digital's Predix, OptaPlanner, Cognex Corporation's VisionPro, OpenCV, Google Cartographer, MATLAB Robotics System Toolbox, FLEXpert, CES Selector
		Stima del costo manifatturiero/di produzione	Modello di costo											
	Ottimizzazione del prodotto in ottica produzione	Riduzione del costo delle componenti			Target costing, parametric costing, activity based costing, distribuzione dei costi overhead, comparazione con prodotti simili				Elaborazione dati					
		Riduzione del costo di assemblaggio	Modelli di costo e produttivi raffinati											
		Riduzione delle produzione a supporto												
	Integrazione con il processo di design del prodotto in senso stretto	Valutazione delle conseguenze sul prodotto delle scelte fatte a livello di processo produttivo.	Richieste modifiche/ report nuove integrazioni necessarie					Raccolta dati	Elaborazione dati					
Product review	Analisi risposta del target market	Raccolta feedback clienti/utilizzatori	Lista feedback	Action development	Focus group, direct observation, scenarios of ouse, user trail, CRM feedback...	formati dati (xlsx...), formati immagine (jpeg, png, xlm...), formato CAD (step, dxf, ipt, dwg, stp...), formato CAM (dxf...)	Pacchetti survey, raccolte casi d'uso prodotto	Raccolta dati			Computer vision, Natural Language Processing, Speech processing, Neural Networks	Sentiment analysis, Text classification, Competitor analysis, Object tracking, Voice recognition, Speaker identification, Random Forest, Principal Component Analysis, Decision Tree, K-Means Clustering, Recurrent Neural	Valence Aware Dictionary and sEntiment Reasoner, Support Vector Machines, Named Entity Recognition, Particle Filter algorithm, Google's Speech-To-Text API, Stanford Sentiment Analysis tool, Gaussian Mixture Model-Universal Background Model Algorithm, Breiman's Algorithm, Singular Value Decomposition, Classification and Regression Trees, Lloyd's Algorithm, Backpropagation Through Time	Microsoft Azure Cognitive Services, Amazon Comprehend, Brandwatch, OpenCV, Dragon NaturallySpeaking, IBM Watson Tone Analyzer, Speaker Recognition System, R, Scikit-learn, Weka, TensorFlow, IBM Watson Studio, Keras

	Analisi dei processi per la messa sul mercato del prodotto	Identificazione needs non completamente soddisfatti e/o nuovi needs non considerati durante lo sviluppo del prodotto	Report risposta clienti/utilizzatori	Analisi statistiche sulle risposte dei clienti/utilizzatori			Elaborazione dati		Computer vision, Natural Language Processing, Speech processing, Neural Networks	Networks, Feedforward Neural Networks, Convolutional Neural Networks	
		Raccolta dati produzione (logistici, qualità, produzione stretta /manufacturing)	LIsta dati produttivi	Raccolta automatica/manuale dati dei diversi processi	Database dati effettivi dei processi	Raccolta dati				Sentiment analysis, Text classification, Competitor analysis, Object tracking, Voice recognition, Speaker identification, Random Forest, Principal Component Analysis, Decision Tree, K-Means Clustering, Recurrent Neural Networks, Feedforward Neural Networks, Convolutional Neural Networks	Valence Aware Dictionary and sEntiment Reasoner, Support Vector Machines, Named Entity Recognition, Particle Filter algorithm, Google's Speech-To-Text API, Stanford Sentiment Analysis tool, Gaussian Mixture Model-Universal Background Model Algorithm, Breiman's Algorithm, Singular Value Decomposition, Classification and Regression Trees, Lloyd's Algorithm, Backpropagation Through Time	Microsoft Azure Cognitive Services, Amazon Comprehend, Brandwatch, OpenCV, Dragon NaturallySpeaking, IBM Watson Tone Analyzer, Speaker Recognition System, R, Scikit-learn, Weka, TensorFlow, IBM Watson Studio, Keras
		Identificazione elementi critici delle diversi processi	Report criticità produzione	Analisi scostamenti dei dati effettivi			Elaborazione dati				
	Analisi risultati economici post lancio	Raccolta dati economici post lancio	Lista dati economici		Database dati effettivi economici	Raccolta dati			Computer vision, Natural Language Processing, Speech processing, Neural Networks	Sentiment analysis, Text classification, Competitor analysis, Object tracking, Voice recognition, Speaker identification, Random Forest, Principal Component Analysis, Decision Tree, K-Means Clustering, Recurrent Neural Networks, Feedforward Neural Networks, Convolutional Neural Networks	Valence Aware Dictionary and sEntiment Reasoner, Support Vector Machines, Named Entity Recognition, Particle Filter algorithm, Google's Speech-To-Text API, Stanford Sentiment Analysis tool, Gaussian Mixture Model-Universal Background Model Algorithm, Breiman's Algorithm, Singular Value Decomposition, Classification and Regression Trees, Lloyd's Algorithm, Backpropagation Through Time	Microsoft Azure Cognitive Services, Amazon Comprehend, Brandwatch, OpenCV, Dragon NaturallySpeaking, IBM Watson Tone Analyzer, Speaker Recognition System, R, Scikit-learn, Weka, TensorFlow, IBM Watson Studio, Keras
		Identificazione elementi critici di natura economica	Report performance economiche	Analisi marginalità			Elaborazione dati				
		Ricerca (interna, esterna) delle possibili soluzioni alle criticità individuate	Lista potenziali modifiche next release prodotto	Patent scanning/screening, expert interview, literature research, survey, brainstorming, TRIZ...	Database brevetti, handbooks, database soluzioni già sviluppate...						
	Elaborazione di una nuova versione di prodotto	Studio piano esecuzione modifiche/soluzioni individuate	Piano esecuzione modifiche	CPM , PERT , PDM , CCM , CRT, GERT	Report effetti modifiche sul prodotto, Report effetti modifiche sui processi produzione (logisitici, qualitò produzione stretta/manufacturing), database processi implementazione modifiche passati e/o simili...			Generazione di direzione	Expert systems, Evolutionary computation	Genetic algorithms, Delft Design Method	Binary Genetic Algorithm, Case-based Reasoning, Fuzzy Logic, Multi-Objective Genetic Algorithm	GenOpt, GE Digital's Predix, Delft Toolkit, ANSYS