

1. For the text below, summarise the problem and suggest a software system architecture solution. The main stakeholders in this case are: - Software system architects Problem, context, and related work Throughout the last couple of years, the research in the field of artificial intelligence (AI) evolved rapidly. That happens thanks to machine learning (ML) and deep learning (DL), which are sub-categories of AI, and concrete applications of AI. In ML, the target is, instead of programming with if-conditions, loops, and instructions as it is done in traditional programming approaches, to let the system learn how to behave. For this purpose, a tremendous amount of data is necessary for training similar to how we, as humans, are learning. Before a newborn is able to recognize a car and distinguish all different kinds of brands and forms, it needs to see hundreds of vehicles repetitively, getting explained by its parents that this is a car. That is why engineers try to apply the way the human brain is working to the software. This approach is referred to as deep learning, where the main idea is to create neural networks with the target to identify specific criteria of a given input in order to recognize it accordingly. The considerable benefit of this approach is its flexibility. Let us consider an example of distinguishing cars according to their brand. The creation of a system with a traditional software engineering approach that is able to identify different brands with its various models, shapes, and colors in any weather and lighting condition is almost impossible. But with AI, this is a solvable task. By feeding the neural network with immense data of all kinds of vehicles in any situation, it is possible to create a system that able to recognize these. Hence, tasks that could not be solved in the past might be solvable with the help of AI. And this is applicable for every primary industry – the consumer market, healthcare, automotive, transportation, retail, logistics, education, agriculture, and more. Some companies are already applying AI to their products and business philosophy to create added values that are enabling them to come up with new business models and new customer target groups. Although there is a significant level of mistrust in AI, particularly from the perspective of losing jobs, existing experiences prove the development of technology does not necessarily lead to higher unemployment. “The countries with the highest robot density – which is Germany, Japan, South Korea, and Singapore - all of them have more than 300 robots per 10,000 workers and have among the lowest unemployment rates. So what we need to learn is, technology and humans combined in the right way will drive prosperity and growth, will drive employment, will drive affordability of products, and will drive demand. And that means, altogether, that if we embrace technology in a responsible way, we can really get there.” - Ulrich Spiesshofer, President, and CEO, ABB Ltd. Nowadays, the majority of companies still do not use AI for their products and services. Although these companies consider AI and are excited about its possibilities (both realistic and unrealistic), they are still failing to adopt AI because of a lack of understanding about the technology itself or due to missing experience with its implementation. Furthermore, the existing architecting approaches so far have ignored or not appropriately dealt with the important question of the adequacy of AI for the existing software systems and for those systems that are to be developed. The entry point of any technology for companies is a system architecture of their software systems. The purpose of the thesis is to compensate for the shortcomings of the state of the art. We aim to address companies and decision-makers with less experience in the field of AI and present an approach that supports them, on an architectural level, in designing an AI powered system while taking the most relevant AI-specific properties under consideration. AI is a disrupting technology with the potential to solve problems that traditional software engineering cannot manage. It is not anymore a simple framework or a plugin. It requires a complete reorganization of the system and introduces too many changes (development, process, system, and software and hardware components). Therefore, it cannot only be adopted by development teams. Still, its effects (benefits and drawbacks, perspectives, and shortcomings) must be discussed on a higher level – hence system and software architecture. Actual integration and deployment of AI for small- and medium-sized

embedded systems companies is still a widely unestablished process. The main reasons for this are: 1. The lack of the understanding of the AI technology itself, which leads to the lack of understanding about 2. potential application areas, and 3. the uncertainty if a migration towards an AI-based solution will be of benefit. The main problem in applying AI in the industry is a lack of a systematic approach for mapping the concerns of stakeholders to potential AI-enabled solutions, which, from the architectural point of view, also discusses consequences in terms of benefits and drawbacks. The domain of AI is very complex and exists in several variations and implementations. It addresses various vertical markets such as medical, consumer, automotive, and the industrial market. Also, its realization can be cloud, fog, or edge-based. Because of the lag of experience of industrial companies that mainly develop embedded hard- and software and due to my personal, professional background, the scope of the thesis will mainly cover AI on the edge and its hardware/software architecture. The problem for companies that are mainly operating in the industrial and embedded domain is a missing basis for decision-making when it comes to the implementation of an AI-approach to their products. For them, the question "Is AI suitable for the existing or new system?" is just barely, if not impossible, to answer confidently. That is where the thesis will provide its contribution. The objective is to support the decision-making process for embedded system companies when adopting AI. This challenge includes getting a profound impression and an increased awareness about technical and business-relevant aspects that need to be under consideration during an AI-implementation. Therefore, engineering techniques are required to get an understanding of the possible impact of implementing AI. Although the SWOT analysis seems to be one of the most common and widely used methods for strategic planning, it has its disadvantages when it comes to addressing particular domain related questions. Its advantage lies in its flexible applicability in different domains and its straightforward usability. But in our case, where we want to evaluate a specific technology, it does not provide any kind of support structure that allows us to incorporate an extension to overcome domain-specific deficiencies. This disadvantage is reflected in the fact that SWOT only describes a vague process of procuring information that is, most of the time, based on subjective impressions (Ashish B. Sasankar and Dr Vinay Chavan. Swot analysis of software development process models.) (Hanieh Kashfi. Adopting cloud computing for enterprise architecture: Swot analysis. International Journal of Advanced Research in Computer Science and Electronics Engineering (IJARCSEE), 6(5), 2017.) In contrast, the Solution Adequacy Check is a research area that can be applied to many different use cases. Fraunhofer RATE, and especially the adequacy check, already provides a solid base when it comes to architecture evaluation. It describes a detailed process of how to acquire data necessary to get a rating for the underlying architecture and thereby enhance the awareness of possible pros, cons, and shortcomings of the architecture. The downside here is that it is not designed for specific domains and technologies that require a more detailed or domain related check as the implantation of AI would do. Therefore it is the aim of the thesis to enhance the adequacy check in such a way that it addresses AI-domain-related circumstances, and that allows evaluating AI-based architectures.

Requirements for the solution Requirement Artifacts • A01: Support Decision Makers who want to adopt AI • A02: Getting an overview of the state of the art ML implementation in edge computing • A03: Identify AI-specific architectural properties • A04: Identify engineering techniques to evaluate the existing system concerning the applicability of AI • A05: Provide a design decision process based on adequacy check or SWOT analysis that maps requirements to potential solutions • A06: Enable a discussion about benefits, constraints, and risks of adopting AI towards technical and business-relevant concerns Requirements • R01: Support companies with existing traditional embedded systems to evaluate the adoption of AI into their current portfolio • R02: Getting an overview of the state of the art ML implementation in edge computing • R03: Identify AI-specific architectural properties that influence business, system, and hardware architectures. •

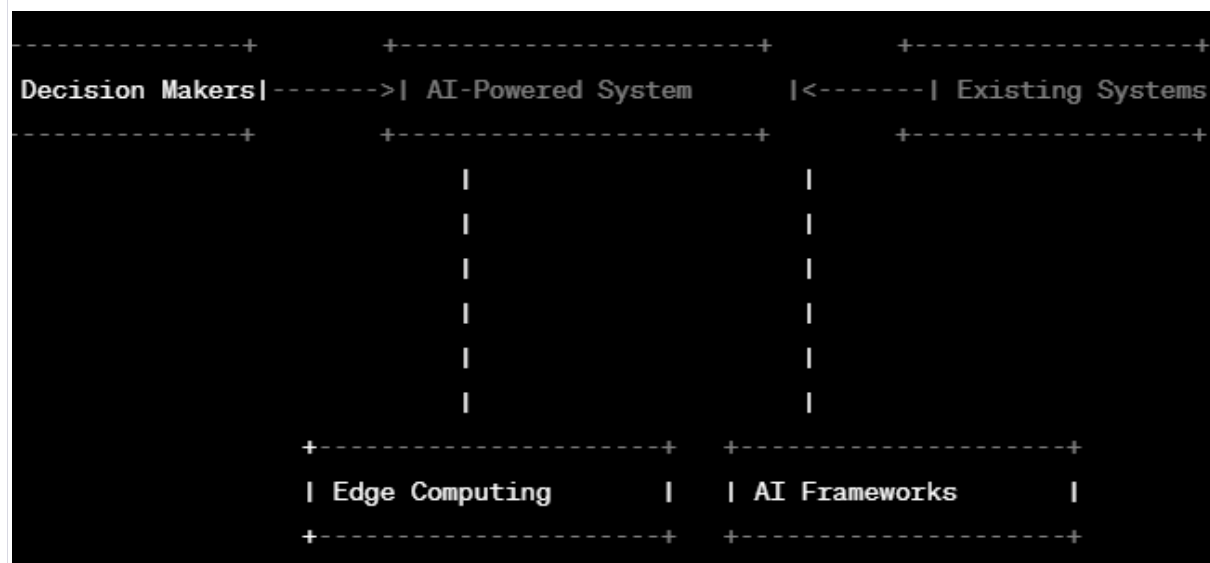
R04: Identify engineering techniques to evaluate the existing system concerning the applicability of AI • R05: Provide a design decision process based on the Solution Adequacy Check or SWOT analysis that maps requirements to potential solutions • R06: Enable a discussion (analysis and evaluation) about technical and business relevant concerns, such as added-values, benefits, constraints, and risks of adopting AI.

**Problem:** Many companies fail to adopt AI due to a lack of understanding about the technology itself and its implementation. There is a need for a systematic approach for mapping stakeholders' concerns to potential AI-enabled solutions, which also discusses consequences in terms of benefits and drawbacks.

**Solution:** The thesis proposes an approach that supports decision-makers in designing an AI-powered system while taking the most relevant AI-specific properties into consideration. The approach includes identifying AI-specific architectural properties, engineering techniques to evaluate the existing system concerning the applicability of AI, and providing a design decision process based on the Solution Adequacy Check or SWOT analysis that maps requirements to potential solutions. The objective is to support the decision-making process for embedded system companies when adopting AI and enable a discussion about technical and business-relevant concerns, such as added-values, benefits, constraints, and risks of adopting AI. The proposed approach aims to enhance the adequacy check in such a way that it addresses AI-domain-related circumstances and allows evaluating AI-based architectures.

2. Suggest architecture of a conceptual solution for this problem. We need a C4 system context diagram (<https://c4model.com/#SystemContextDiagram>). Please use text (arrows, dots, different shapes, and different colours) to visualise the solution. Consider the following: - Identify relevant external systems (if any) and add them to the diagram. - Identify relevant external stakeholders (if any) and add them to the diagram. - Identify high level abstract data types transferred between components and stakeholders. - Summarise data types transmitted between components. Add this information to the connections and interfaces between components.

The C4 system context diagram for the proposed solution is as follows:



The diagram includes three main components: Decision Makers, AI-Powered System, and Existing Systems. Decision Makers are external stakeholders who want to adopt AI and make decisions about the implementation of AI in their existing systems. The AI-Powered System is the solution proposed in the thesis that supports decision-makers in designing an AI-powered system.

Existing Systems are traditional embedded systems that the companies want to evaluate for the adoption of AI.

The AI-Powered System component consists of two sub-components: Edge Computing and AI Frameworks. Edge Computing represents the hardware and software architecture that supports the implementation of AI on the edge, which is essential for companies that mainly operate in the industrial and embedded domain. AI Frameworks represent the different frameworks that provide AI capabilities, such as machine learning and deep learning, and support the creation of AI-based systems.

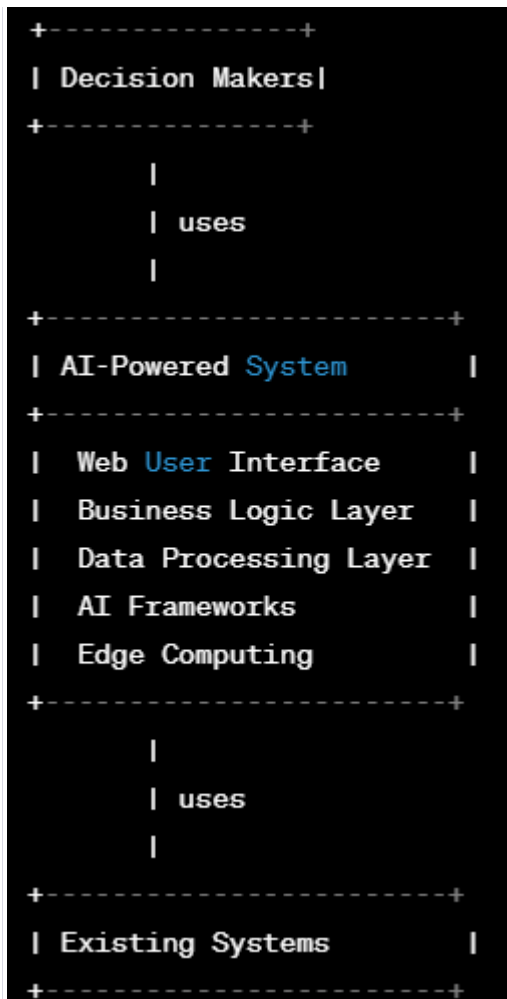
The high-level abstract data types transmitted between the components and stakeholders are requirements, architecture evaluation data, and design decision process. The Decision Makers transmit their requirements to the AI-Powered System component. The AI-Powered System component provides architecture evaluation data to Decision Makers, which include information on the applicability of AI to their existing systems and the benefits, constraints, and risks of adopting AI. The AI-Powered System component also provides a design decision process that maps requirements to potential solutions.

The connections and interfaces between the components include the transmission of requirements from Decision Makers to the AI-Powered System component, the transmission of architecture evaluation data from the AI-Powered System component to Decision Makers, and the transmission of the design decision process from the AI-Powered System component to Decision Makers. These connections and interfaces are depicted as arrows between the components in the diagram.

The AI-Powered System component receives data from two external systems: Decision Makers and Existing Systems. The connections between the AI-Powered System component and the external systems are depicted as arrows in different colors to indicate the different data types transmitted between them. Overall, the proposed solution architecture provides a systematic approach for decision-makers to evaluate the adoption of AI into their current systems and make informed decisions about the benefits and drawbacks of adopting AI.

3. Suggest architectural components that we need to solve this problem. We need a C4 container solution diagram (<https://c4model.com/#ContainerDiagram>). Please use text (arrows, dots, different shapes, and different colours) to visualise the solution. Consider the following: - Define how the main stakeholder will use the solution. Add it to the diagram.. - Which quality properties are important? Suggest adequate architectural solutions to address them (architectural style, design patterns, design tactics, components and their relationships). Structure the components according to it. - Define data types transferred between components. - Summarise data types transmitted between components. Add this information to the connections and interfaces between components. - Define interfaces between components. Add information about required and provided interfaces to the components, along with the data types. - Document which anti-patterns we need to avoid when implementing this solution.

The C4 container diagram for the proposed solution is as follows:



The main stakeholder, Decision Makers, will use the AI-Powered System to evaluate the adoption of AI into their existing systems. They will interact with the system through a Web User Interface, which will provide them with access to the AI Frameworks, Edge Computing, and Data Processing Layer. The Business Logic Layer will provide the core functionality of the system.

The important quality properties for the solution are scalability, performance, and reliability. To address these properties, the following architectural solutions will be implemented:

- Architectural style: Microservices architecture to enable scalability and modularity.
- Design patterns: Caching to improve performance and Circuit Breaker to improve reliability.
- Design tactics: Load balancing to improve scalability and monitoring to improve reliability.
- Components and their relationships: The Web User Interface will interact with the Business Logic Layer, which will interact with the Data Processing Layer and AI Frameworks. The Edge Computing component will provide the hardware and software architecture to support the implementation of AI on the edge.

The data types transferred between components include requirements, architecture evaluation data, and design decision process. These data types will be transmitted between the Web User

Interface and the Business Logic Layer, the Business Logic Layer and the Data Processing Layer, and the AI Frameworks and the Data Processing Layer.

The connections and interfaces between the components include the transmission of requirements from the Web User Interface to the Business Logic Layer, the transmission of architecture evaluation data from the Data Processing Layer to the Business Logic Layer, and the transmission of the design decision process from the Business Logic Layer to the Web User Interface. These connections and interfaces are depicted as arrows between the components in the diagram, along with the required and provided interfaces and the data types transmitted.

To avoid anti-patterns, the following practices will be implemented:

- Avoid tight coupling between components by using interfaces to decouple components.
- Use the principles of the SOLID design to ensure modularity and maintainability.
- Implement a robust error handling mechanism to prevent cascading failures.

4. Suggest architectural components that we need to solve this problem. We need a C4 component solution diagram (<https://c4model.com/#ComponentDiagram>). Please use text (arrows, dots, different shapes, and different colours) to visualise the solution. Consider following:

- Define how the main stakeholder will use the solution. Add it to the diagram.
- Which quality properties are important? Suggest adequate architectural solutions to address them (architectural style, design patterns, design tactics, components and their relationships). Structure the components according to it.
- Define data types transferred between components.
- Summarise data types transmitted between components. Add this information to the connections and interfaces between components.
- Define interfaces between components. Add information about required and provided interfaces to the components, along with the data types.
- Document which anti-patterns we need to avoid when implementing this solution.

The C4 component diagram for the proposed solution is as follows:



The main stakeholder, Decision Makers, will use the Web User Interface to interact with the AI-Powered System. The Web User Interface will consist of a Web App, Client-side components, and Server-side components. The Business Logic Layer will provide the core functionality of the

system and will include an API Gateway, Authentication, Authorization, Caching, Circuit Breaker, and Load Balancer components. The Data Processing Layer will include Data Storage, AI Frameworks, and Edge Computing components.

The important quality properties for the solution are scalability, performance, and reliability. To address these properties, the following architectural solutions will be implemented:

- Architectural style: Microservices architecture to enable scalability and modularity.
- Design patterns: Caching to improve performance and Circuit Breaker to improve reliability.
- Design tactics: Load balancing to improve scalability and monitoring to improve reliability.
- Components and their relationships: The Web App and Client-side components will interact with the Server-side components, which will interact with the Business Logic Layer. The Business Logic Layer will interact with the Data Processing Layer components.

The data types transferred between components include requirements, architecture evaluation data, and design decision process. These data types will be transmitted between the Web User Interface and the Business Logic Layer, the Business Logic Layer and the Data Processing Layer, and the AI Frameworks and the Data Processing Layer.

The connections and interfaces between the components include the transmission of requirements from the Web User Interface to the Business Logic Layer, the transmission of architecture evaluation data from the Data Processing Layer to the Business Logic Layer, and the transmission of the design decision process from the Business Logic Layer to the Web User Interface. These connections and interfaces are depicted as arrows between the components in the diagram, along with the required and provided interfaces and the data types transmitted.

To avoid anti-patterns, the following practices will be implemented:

- Avoid tight coupling between components by using interfaces to decouple components.
- Use the principles of the SOLID design to ensure modularity and maintainability.
- Implement a robust error handling mechanism to prevent cascading failures.
- Avoid code duplication by implementing reusable components.

5. Please suggest a toolchain to implement the solution. Consider which tools we can reuse. For the components we need to develop, please suggest a programming language. Please use text (arrows and dots) to visualise the solution. Define data types transferred between components. Summarise data types transmitted between components. Add this information to the connections and interfaces between components.

To implement the proposed solution, the following toolchain can be used:

- Programming Language: Python for the AI Frameworks and Data Processing Layer components, and JavaScript for the Web App and Client-side components.
- Frameworks and Libraries: Flask for the API Gateway, TensorFlow for the AI Frameworks, React for the Web App, and Redux for state management.
- Containerization: Docker to containerize the different components and enable portability.
- Orchestration: Kubernetes to manage the containers and enable scaling and deployment.



The data types transferred between components include requirements, architecture evaluation data, and design decision process. These data types will be transmitted between the Web User Interface and the Business Logic Layer, the Business Logic Layer and the Data Processing Layer, and the AI Frameworks and the Data Processing Layer.

The connections and interfaces between the components include the transmission of requirements from the Web User Interface to the Business Logic Layer via REST API, the transmission of architecture evaluation data from the Data Processing Layer to the Business Logic Layer via a message queue, and the transmission of the design decision process from the Business Logic Layer to the Web User Interface via WebSocket. These connections and interfaces are depicted as arrows between the components in the diagram, along with the required and provided interfaces and the data types transmitted.

The toolchain can be visualized as follows:

Web App + Client-side components <-REST API-> API Gateway + Authentication + Authorization + Caching + Circuit Breaker + Load Balancer <-Message Queue-> Data Storage + AI Frameworks + Edge Computing

The Python programming language will be used to develop the AI Frameworks and Data Processing Layer components, while the JavaScript programming language will be used to develop the Web App and Client-side components. The Flask framework will be used to develop the API Gateway component, while the TensorFlow framework will be used to develop the AI Frameworks component. The React library will be used to develop the Web App component, while the Redux library will be used for state management.

Overall, this toolchain will enable the implementation of the proposed solution and provide the necessary components and frameworks to ensure scalability, performance, and reliability.