

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 - DevOps engineers - Developers

Problem, context, and related work Software engineering today is a challenging task. As the complexity of software keeps increasing, software engineers find it ever more challenging to cope with this complexity and produce high-quality source code. Furthermore, developers today spend a significant amount of their time solving similar problems as they struggle with code reuse (e.g. they are aware that the solution exists but it requires tedious changes so re-writing a new code seems easier, and they can misinterpret the requirements, or they are not aware if a solution is available and being used in a different context). Ideally, developers could profit from systems that assist them during the development process by suggesting already existing good (and tested) solutions. This could reduce development efforts. The developers could further benefit from systems that have the capabilities to understand the software requirements. This would enable the creation of assistance mechanisms that can index problem and solution domains and be aware if for an existing or similar problem there exist solutions. We are focusing on supporting developers when dealing with these challenges:

1. Avoidance of repetitive tasks (realizing code that already exists increases time efficiency during development and keeps developers motivated)
2. Avoiding bugs in new code (tested code is often far more valuable because potentially many bugs have been discovered and removed)

These challenges are both a drain on motivation that developers have (e.g., influence their productivity and motivation to keep the existing work), and introduce a potential overhead due to all the activities that are related to testing newly implemented solutions. Even if solutions already exist, developers still would need to invest time in understanding them and reusing them. In particular, how well these solutions fit certain requirements. Because understanding source code is a time-consuming task, developers often find it quicker to write the new code. Ideally, there would exist solutions that can understand existing source code (and logic behind it) and relate it to the problems they are solving (requirements). AI is a technology that simulates human intelligence processes through computing systems. In an ideal case, AI is capable to rationalize and take actions towards a specific goal. AI, in this context, is interesting because it can recognize patterns in textual data. In software engineering, both requirements and source code are represented mostly in textual form. Therefore, if we can train AI models to understand requirements and to produce results in a restricted format that matches the semantics of programming languages, this could enable the creation of solutions that assist developers. Ideally, this could lead to the use of AI-enabled technologies as assistance tools. However, this is not an easy task. Gathering requirements for human is challenging as the requirement elicitation process need experience and constant communication between the stakeholders for understanding requirements. Expressing requirements is even hard for software engineers, as they imply many things (e.g. different use cases). They acquire customers' requirements, refine them into tasks and then use AI to assist them with the solutions. Another challenging task is to prioritize goals for development which are often not represented in the requirements and need an expert opinion. When developing software solutions, developers often can choose between various ways of implementation, different coding styles, various algorithms, and different programming languages. For example, if developers want to optimize for a reduced memory footprint, they might use processing algorithms that need more processing power but keep fewer states alive in memory. Finally, it is necessary to test the implementation. This is a challenge, as developers need to be skilled in the implemented technology stack. Gathering requirements can benefit from AI techniques, especially from NLP that can be used to automate this process by extracting valuable information from user requirements by semantic analysis. In an ideal scenario, AI can understand requirements and generate portions of the code to assist

developers. These portions can be code snippets or function templates. The focus of this work is on generating full functions to assist developers. While research in AI and software engineering is potentially providing a variety of solutions in this domain, there still do not exist solutions that can outmatch humans in terms of building program logic and fully understanding the requirements. Even more, although some very impressive solutions can generate source code (e.g., GPT-3), there are no practical solutions that can integrate with development practices and assist developers with writing particularly interesting code and choosing optimal solutions based on the code context and requirements. Therefore, we define our problem statement as follows: Are AI-enabled solutions mature enough to integrate with development processes, index problem and solutions domains, and assist developers by suggesting solutions to common problems? If this is not the case, what is preventing the adoption of AI-based solutions in this way, in software engineering? Therefore, in this thesis, we aim to identify challenges that prevent the adoption of AI from being used as an assistance tool in software development by providing an elaborate list of identified issues and an AI-based approach that can aid software development by generating solutions for already solved problems using user requirements as input. This considers the understanding of how the machine learning algorithms associate source code with natural language and how it understands it. Although there have been several studies focused on source code generation tools that use AI ("Imam, Ayad Tareq and Rousan, Thamer and Aljawarneh, Shadi; An expert code generator using rule-based and frames knowledge representation techniques; 2014", "Soni, Naitik Chetan and Pawar, Dhruv Ashok and Tambe, Namita Sandeep and Deolekar, Rugved Vivek; Automated system for code generation from unstructured algorithm; 2016", "Shah, Meet and Shenoy, Rajat and Shankarmani, Radha; Natural Language to Python Source Code using Transformers; 2021", "Lu, Chao and Cao, Xin and Zhu, Yi and Huang, Tao and Pan, Zhaoming and Li, Xiu; Procedure2Command: an AI-based Nuclear Power Plant Control Command Code Generation Prototype System; 2020", "Liliane do Nascimento Vale and Marcelo de Almeida Maia; Towards a question answering assistant for software development using a transformer-based language model; 2021", "Zhang, Chen and Niu, Xinyi and Yu, Bin; A Method of Automatic Code Generation Based on AADL Model; 2018"), there haven't been significant contributions toward their classification or identification of problems that prevent them to be usable in practice.

Requirements for the solution

- Functional requirements •
- RQ.1: An approach is able to use dataset for supervised learning and testing & Input •
- RQ.2: An approach is able to generate a training/testing dataset (For training an algorithm, we would need a specific formatted dataset or a structured dataset. The generated dataset will be the consolidated data collected by a design template (including fields like inputs, function requirements, source code, and output) for users to input, hence making our training data. The same logic can also be applied to gathering the testing data. However, there can be other means to gather datasets, for instance, crawling/scrapping/REST-API calls from open source projects or public/private repositories.) from taking user input via a design template. •
- The design template will include fields to structure the natural language input provided, such as function input (e.g. parameter integer A, parameter string B, etc.), function requirements (which can include function comments, description, use cases, etc.), function source code and function output (e.g. return type, etc.) •
- RQ.3: An approach can extract and generate a training/testing dataset from open source projects and public or private repositories (e.g. GitHub, BitBucket, etc.) •
- RQ.4: An approach is able to persist the consolidated training/testing dataset in a storage system (e.g. database, file storage, etc.) &
- Pre-Processing •
- RQ.5: An approach is able to pre-process the stored dataset for training and testing for the AI models (e.g. tokenization, word2vec, stop word removal, etc.) •
- RQ.6: An approach is able to transform the extracted repository data to a structured format, such that it aligns with the design template dataset •
- RQ.7: An approach is able to train on Python programming language source code &
- Training •
- RQ.8: An approach is

able to train on the structured dataset that is prepared using the design template. • RQ.9: An approach is able to train on deep neural networks by using existing models (e.g. GPT2, GPT3, etc.) • RQ.10: An approach is able to understand the requirements in the structured dataset to generate the source code • RQ.11: An approach is able to generate source code for Python programming language based on a set of input requirements (testing design template) & Output • RQ.12: An approach is able to generate multiple source code functions or snippets as a suggestion for the user to select • RQ.13: An approach is able to evaluate the generated source code accuracy based on syntax and logical correctness (by using the following metrics e.g. BLEU score, F-Score, WER, etc.) & Testing and Evaluation • RQ.14: The approach is able to integrate in an IDE (e.g. Eclipse, NetBeans, VS Code, etc.) by a plugin. This will allow the end user to take benefit of the deployed approach in a real-world use case when they are programming in on-going projects. & Deployment • RQ.15: The approach is able to save the trained model in a compressed format (e.g. ZIP, PKL, etc.) in a storage system for testing, evaluating and deploying

Quality requirements • Q.1: - Environment: ♣ An approach generating a function source code based on a structured input template ♣ Number of syntax faults = E; Parameters = p - Stimulus: ♣ Re-training the model with hyperparameter tuning ♣ Parameters = q - Response: ♣ Better syntax accuracy ♣ Number of syntax faults = $E/2$ • Q.2: - Environment: ♣ An approach generating a function source code based on a structured input template ♣ Cyclomatic complexity = CE ♣ Parameters = p - Stimulus: ♣ Re-training the model with hyper-parameter tuning ♣ Parameters = q - Response: ♣ Better syntax accuracy ♣ Number of syntax faults = $E/2$ • Q.3: - Environment: ♣ An approach generating a function source code in an on-going project ♣ Lack of cohesion in methods (LCOM) = C - Stimulus: ♣ Existing source code in an on-going project which are logically equivalent to the generated source code ♣ Parameters = q ♣ Response Low Cohesion [6] LCOM = 0 • Q.4: - Environment: ♣ An approach generating a function source code in an on-going project ♣ Size of the training dataset and its trained model accuracy; Size = S; Accuracy = A% - Stimulus: ♣ Re-training the model with the increased data size ♣ Size = 2S - Response: ♣ Improved training model accuracy ♣ Accuracy = 2A%

The problem addressed is the increasing complexity of software engineering, resulting in a drain on developers' productivity and motivation due to repetitive tasks and potential errors. The proposed solution involves an AI-based approach to assist developers by generating solutions to common problems and automating the requirement elicitation process. The functional requirements include the ability to use and generate datasets for supervised learning, extract data from open source projects and repositories, preprocess and transform data, train on deep neural networks, understand structured input templates to generate source code, evaluate generated code accuracy, and integrate into an IDE. Quality requirements include better syntax accuracy, reduced cyclomatic complexity, increased cohesion in methods, and improved model accuracy with increased data size. The proposed solution involves the use of AI models such as GPT-2 and GPT-3 to generate code snippets or function templates to assist developers.

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.

Here is a C4 system context diagram for the proposed solution:



The AI System component is the main component responsible for generating code snippets and function templates based on input requirements. It receives input through a front-end interface component, which is responsible for providing a structured input template for the user to fill in, including function input, requirements, source code, and output. The AI System can also gather training data from open source projects and repositories through an API Service component, and store the training data and generated code snippets in a data store component. The AI System can be integrated into an IDE through a plugin, which allows the user to access the generated code snippets while programming.

The external stakeholders are the users who input the requirements and receive the generated code snippets. The external systems are the IDE and any external systems from which the AI System gathers training data.

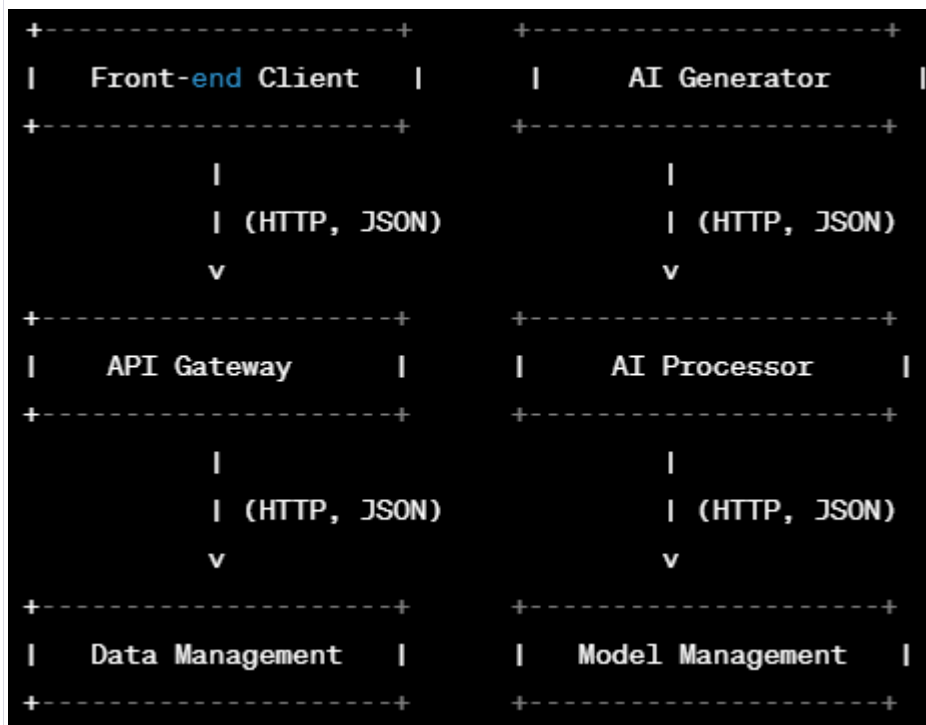
The data types transmitted between components are in JSON format, including the structured input template, training data, and generated code snippets. The connections and interfaces between components specify the data types transmitted and received, as well as any necessary processing or transformation of the data.

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.

Here is a C4 container diagram for the proposed solution:



The Front-end Client component is used by the main stakeholder, the developer, to input requirements and receive generated code snippets. The API Gateway component receives requests from the Front-end Client and directs them to the appropriate components. The Data Management component is responsible for storing and managing training data and generated code snippets, while the Model Management component is responsible for training and managing the AI model.

The important quality properties for this solution are performance, reliability, and maintainability. To address these properties, we can use a microservice architecture style to allow for scalability and fault tolerance. The components can be designed using design patterns such as the Facade pattern to simplify their interfaces and improve maintainability. The AI model can be trained using deep learning techniques, such as using GPT-3, to improve accuracy.

The data types transmitted between components are in JSON format, including the input requirements, generated code snippets, and training data.

The connections and interfaces between components specify the data types transmitted and received, as well as any necessary processing or transformation of the data. The Front-end Client

provides the API Gateway with input requirements in JSON format using HTTP requests. The API Gateway then directs the requests to the appropriate component based on the endpoint. The Data Management and Model Management components both provide and receive data in JSON format using HTTP requests.

The required interfaces for the components are as follows:

- The Front-end Client provides a RESTful API with endpoints for input requirements and receiving generated code snippets.
- The API Gateway provides a RESTful API with endpoints for directing requests to the appropriate components.
- The Data Management component provides a RESTful API with endpoints for managing training data and generated code snippets.
- The Model Management component provides a RESTful API with endpoints for managing the AI model.

The anti-patterns that we need to avoid when implementing this solution include the Monolithic architecture anti-pattern, as this can lead to scalability and maintainability issues. We also need to avoid the God Object anti-pattern, by separating responsibilities and concerns between components.

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.

As this question requires a more detailed diagram, I will provide a textual description of the components needed to solve the problem.

Components:

1. User Interface Component: This component will be responsible for providing a user-friendly interface to the user, allowing them to input requirements, generate source code and view the generated code. It will also handle error messages and provide feedback to the user.
2. Natural Language Processing (NLP) Component: This component will be responsible for processing natural language input from the user and extracting relevant information such as the function name, parameters, and requirements. It will also be responsible for converting natural language requirements into a structured format that can be used by the Machine Learning (ML) component.
3. Machine Learning (ML) Component: This component will be responsible for training and testing the ML models that will generate the source code. It will use the data provided by

the NLP component and the dataset from open source projects and public or private repositories to train and test the ML models.

4. Code Generation Component: This component will generate source code based on the ML model trained by the ML component. It will use the structured input from the NLP component and generate code snippets or full functions.
5. Code Evaluation Component: This component will be responsible for evaluating the generated code for syntax and logical correctness. It will use metrics such as BLEU score, F-Score, WER, etc. to evaluate the code.
6. Dataset Component: This component will be responsible for storing the consolidated training and testing datasets in a storage system.
7. Model Storage Component: This component will be responsible for storing the trained ML models in a compressed format (e.g., ZIP, PKL) in a storage system.
8. Integration Component: This component will be responsible for integrating the solution with different Integrated Development Environments (IDEs) such as Eclipse, NetBeans, VS Code, etc. It will provide a plugin that will allow the end-users to take advantage of the solution in a real-world use case when they are programming in ongoing projects.

Stakeholders:

- Software System Architects
- DevOps Engineers
- Developers

Quality properties:

- Accuracy
- Reliability
- Maintainability
- Scalability

Interfaces:

- User Interface component: Receives structured input from the user and displays generated code to the user.
- NLP component: Receives natural language input from the user and provides structured input to the ML component.
- ML component: Receives structured input from the NLP component, trains the ML models and provides generated source code to the Code Generation component.
- Code Generation component: Receives structured input from the ML component and generates source code.
- Code Evaluation component: Receives generated code from the Code Generation component and evaluates it for syntax and logical correctness.
- Dataset component: Stores the consolidated training and testing datasets.
- Model Storage component: Stores the trained ML models in a compressed format.
- Integration component: Integrates the solution with different IDEs.

Data types:

- Natural language input

- Structured input
- Generated source code
- Training and testing datasets
- Trained ML models

Anti-patterns:

- Tight Coupling: Avoid tightly coupling the components to enable independent development, testing, and deployment.
- Monolithic Architecture: Avoid developing a monolithic architecture to enable scalability and maintainability.

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, we can use a variety of existing tools and technologies:

1. Data collection and pre-processing: For data collection, we can use web scraping tools like BeautifulSoup, Scrapy, or Selenium to collect data from online sources. For pre-processing, we can use Python libraries like NLTK and spaCy for natural language processing.
2. Model training and testing: We can use deep learning frameworks like TensorFlow, PyTorch, or Keras for training and testing our AI models. We can use these frameworks to build and train deep neural networks, which can be used to generate source code based on natural language input.
3. Deployment: We can deploy our AI models using cloud platforms like AWS or Google Cloud Platform, or we can use containerization tools like Docker and Kubernetes to deploy our models on-premise.
4. IDE Integration: To integrate our solution with an IDE, we can use plugins or extensions. For example, we can use the Language Server Protocol (LSP) to provide language features to IDEs like VSCode or Eclipse.

For the components we need to develop, we can use Python as the programming language. We can use Flask, a Python web framework, to create RESTful APIs that will be used to communicate between the components.

The data types transferred between components are structured datasets that include natural language input, function input, function requirements, function source code, and function output. These datasets will be pre-processed using techniques like tokenization, stop word removal, and word embedding.

The connections and interfaces between components will use RESTful APIs to communicate between the different components. The APIs will use JSON as the data format for transmitting the structured datasets.

Overall, our toolchain will consist of a variety of existing tools and technologies, and we will develop new components using Python and Flask to build RESTful APIs for communication between components.