Gradus Vol 10, No 2 (2023)
ISSN 2064-8014

DEVELOPING OF NLP MODELS BY MODEL BASED SOFTWARE DEVELOPMENT

Zsolt Krutilla [0009-0000-3607-4309] [1,2]

[1] Department of Software Development and Application, Institute of Information Technology, University of Dunaújváros, Hungary
[2] Doctoral School of Applied Informatics and Mathematics, Óbuda University, Hungary
https://doi.org/10.47833/2023.2.CSC.024

Keywords:
natural language processing,
model based development,
applied Informatics,
artificial Intelligence

Article history:
Received 14 July 2023
Revised 20 November 2023
Accepted 2 December 2023

Abstract
In the case of software development, we can talk about several methodologies, such as the waterfall model, test-driven software development, agile software development, but the question rightly arises whether software development methods in the traditional sense are applicable to the development of natural language processing language models, especially from the aspect of AI and dictionary-based NLP models. Learning AI models is substantially different from the classical software development process. Whereas in classical software development, developers explicitly describe the operation and behavior to the computer, in AI model learning, the models themselves learn from the input data. This paper presents a possible solution that applies the agile Scrum methodology used in classical software development to the development of dictionary-based NLP models, and identify the agile development opportunities in case of the machine learning-based NLP models.

1 Introduction

Since software development is a social process [1], understanding how people work together to create software is critical. The importance of software in our society is coupled with the difficulties that arise during its development in terms of understanding the business need and the application that is produced. [2] Generally, software development teams are brought together to develop new products - or to improve existing ones. The team itself usually consists of two or more software developers involved in the creation of a specific piece of software, each to be delivered within a certain timeframe, who rely on their collective skills, given that the scope, complexity and number of tasks required to develop modern software usually exceeds the capabilities of individual developers. [3] In today's modern, fast-paced world, we are increasingly demanding criteria from a development team that we previously thought unthinkable. This is nowadays also true for the development of language models, so that a shorter development time interval will be expected in the future for the learning and development of NLP (Natural Language Processing) models in the classical sense. In this context, we need to explore the possibility of using, for example, the agile software development methodology in the development of language models. In this article I will present the classic software development methodologies and their general advantages and disadvantages that I have experienced in my more than 14 years of development career. In addition, a dictionary-based NLP model based on the Agile Scrum methodology is presented too. In order to gain an insight into how the agile development of NLP models can be achieved, we need to summarize in broad terms the most popular classical software development methods.

2 Software as a product

We often must treat software as products that we produce, and like any production technology [4], software "production" has its own technologies and tools. Software is treated as a product in its own right [5], with market value and customer requirements. As a result, development teams focus on the features, functionality and value of the product to create a product that meets customer and market needs. In the product-oriented approach [6], customer requirements, functional and non-functional requirements, and development tasks and priorities are captured in the product backlog. This allows the team to select from the backlog the tasks and priorities needed for sprints. The software-as-a-product approach provides flexibility to manage changing customer requirements. The product backlog allows for flexible prioritization and consideration of customer feedback. This allows the team to adapt to new demands and the changing market environment. [7] Iterative development cycles are often used in software development, where the product is constantly evolving, and new versions are released. This allows for early market introduction and incorporation of customer feedback into subsequent versions. [8]

3 Software development from a process perspective

The software development process usually consists of several steps, which we can monitor continuously to ensure efficient and structured project implementation. [1] The general process of software development can be broken down into the following steps [9-11]:

- Defining needs and requirements: in this initial phase, developers, customers and other stakeholders jointly define the software's goals, functionality and requirements. This is usually done through interviews, working groups and documentation.
- Design: in this phase, developers create detailed designs of the software architecture, the relationships between components, the user interface and other important elements. Diagrams, such as UML diagrams and data flow diagrams, are often used in the design process.
- Development: this phase is about the actual coding of the software. Based on the specifications developed in the design phase, developers use programming languages, frameworks and tools to implement the software. This phase usually includes unit testing, which is aimed at testing the various components (functions, classes, etc.) and identifying errors.
- Testing: software testing is a key part of the development process. Testing is a key part of the development process. Various testing methods can be used, such as unit testing, integration testing, system testing and acceptance testing.
- Release: after testing and bug fixes, the software is ready for release. This phase involves installing and configuring the software in a real or test environment, as well as engaging users and monitoring their use of the software.
- Maintenance phase: once the software is released, further improvements are called software evolution or maintenance. This involves considering user feedback, bug fixes, updates and introducing new features to the software.

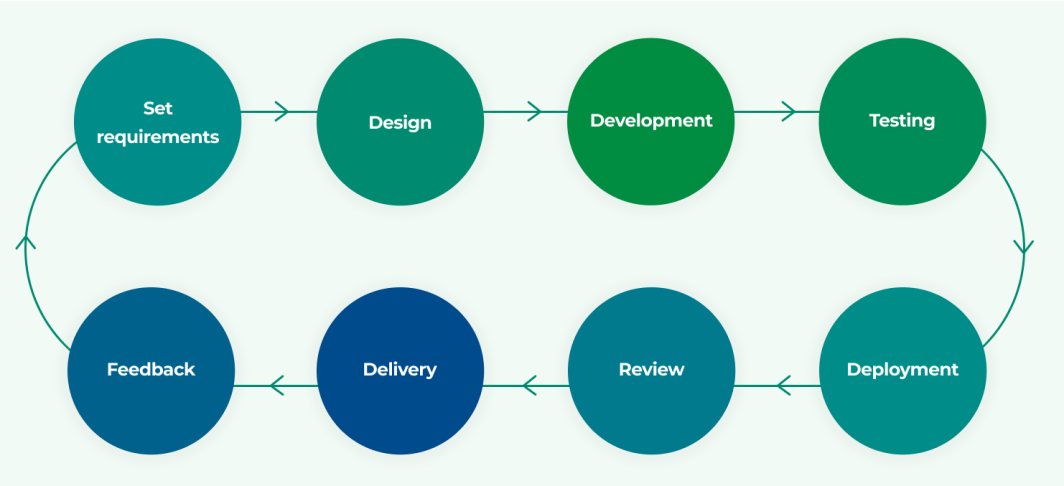

Figure 1: General software development process steps [54]

It is important to note, however, that the software development process is not a rigid linear process but is often iterative and agile. [12] During iterations, developers continuously receive feedback, improve and enhance the software based on user needs. Agile development methods such as Scrum or Kanban are extremely popular because of their flexibility and customer-centric approach. [13-14]

These needs are usually inspired by human needs or business needs, the use/application of which supports a company's operations or gives it an advantage over its competitors. These needs can also be called the drivers of software development. Inducers of software development are factors that act as drivers to initiate software development. These drivers can be business needs, problems or opportunities that the organization identifies and wants to respond to by developing software.

Business needs are one of the most common drivers for software development. [15] This can be the introduction of a new product or service that requires software. For example, a company wants to create a new e-commerce website for online sales. Increasing the efficiency of a company's processes or improving existing systems may also be a reason for software development. [16] For example, a transport company may develop new logistics software that automates data capture and tracking, reducing manual labour and increasing efficiency. Increasing competitiveness can also be a trigger for software development. A company can try to stand out from the competition by developing new, innovative software solutions. [17-18] For example, developing a smart home system that allows remote control and automation of home appliances. In addition to increasing competitiveness, changes in the business environment can also motivate software development, such as changes in regulations or legal requirements, a frequent change demand in the banking sector.

These needs are usually part of a company's lifecycle, based on business strategy and market opportunities, and require software development to achieve objectives or solve problems. They are identified by company management, experts, market researchers or customers. The business needs are then usually documented in more detail and passed on to software developers, who then design and implement the software to meet the objectives.

4 Software development methodologies

Software development companies use a variety of methods and methodologies to manage their projects and make the software development process more efficient.

4.1 Waterfall model

This is the traditional linear development model, which involves the following steps: requirements definition, design, implementation, testing and maintenance. Each step follows a linear sequence, and it is usually only after the completion of each phase that it is possible to move on to the next phase. [19-20]

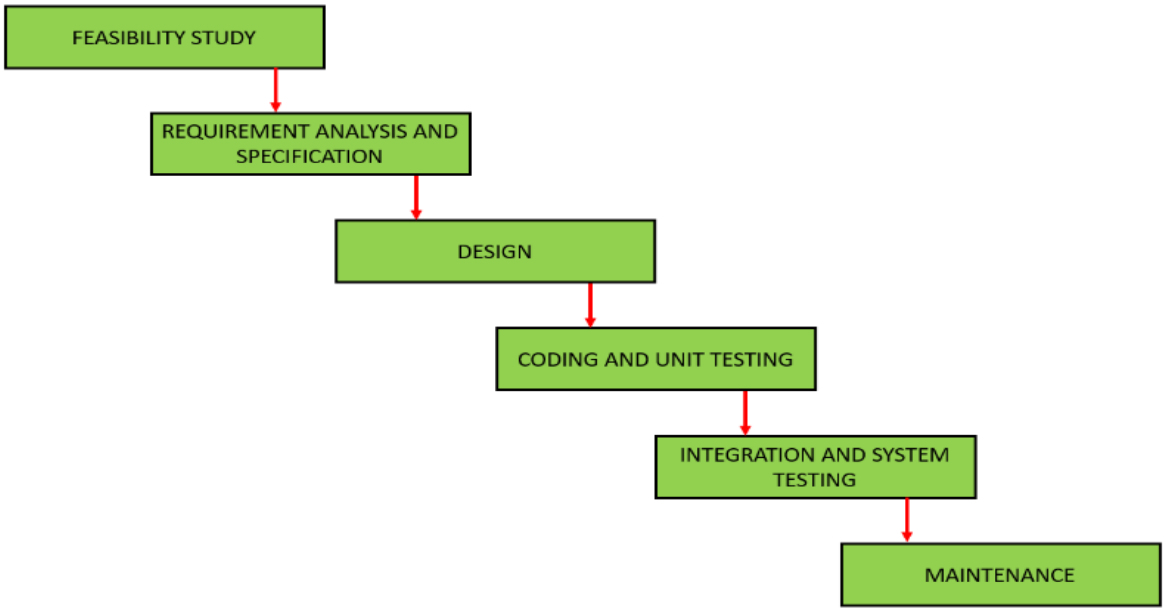

Figure 2: Waterfall model [55]

Advantages:

- Structured and well-defined process: the steps of the development are predefined.
- The development process is well defined and well-defined.
- Good choice for projects with stable and well-defined requirements.

Disadvantages:

- Inflexibility: Difficult to adapt to changing requirements.
- Late feedback: users only see the finished product at the end.
- Long development cycles.

4.2 Agile methodology

Agile software development involves methodologies that take a more flexible and iterative approach to development. Some common agile methodologies include Scrum, Kanban, and Extreme Programming (XP). Agile methodologies divide development into several short iterations (sprints) in which developers receive regular feedback and continuously apply changes and improvements to the software. [3, 21]

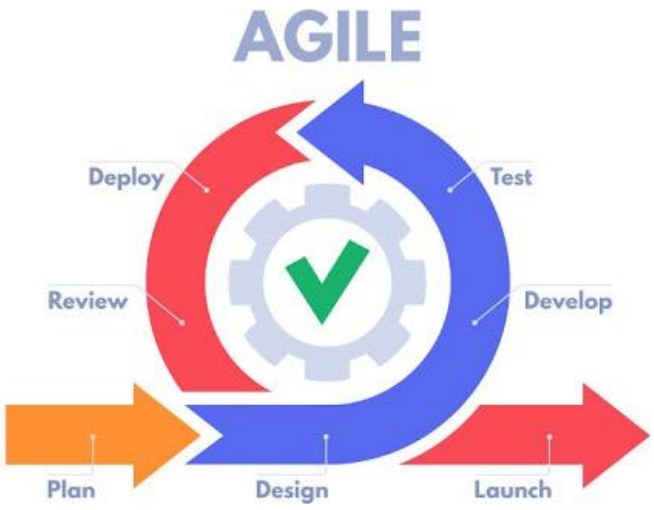

Figure 3: Agile software development process [56]

Advantages:

- Flexibility: easily adapts to changing requirements.
- Early feedback: the product is available and repairable sooner.
- Promote teamwork: teams can work together more effectively.

Disadvantages:

- Lack of structure: sometimes too much flexibility can make it difficult to manage the project.
- Not ideal for all types of projects.

4.3 Lean development

Lean development aims to eliminate unnecessary work and resources from the process. This methodology is based on the premise that software development should be continuously improved and non-value-adding activities minimized. Lean development is characterized by continuous feedback, process optimization and shortening development time. [22]

Figure 4: The process of Lean Software Deployment [57]

Advantages:

- Unnecessary or non-value-adding activities during development are minimized.
- Helps to ensure that requirements remain stable and do not change abruptly.
- Supports continuous improvement, allowing rapid iterations and continuous improvement.
- Focus on customer needs and the frequent use of customer feedback in development.

Disadvantages:

- Changes and culture shifts can be time consuming.
- Not suitable for all projects.
- Need for continuous monitoring.
- Lack of an appropriate culture.

4.4 DevOps

DevOps is a culture and a practical approach that combines application development and operations. DevOps aims to increase efficiency and collaboration between software development and software operations. Developers and operators work together to ensure faster software releases, high availability and reliable operations. [23-24]

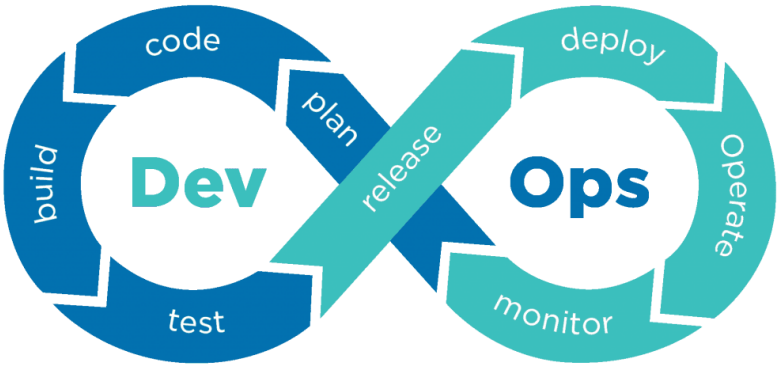

Figure 5: The DevOps software release process [58]

Business needs are usually defined through communication between the company and the customer. Business analysts, project managers and customer service representatives help to assess, document and understand needs and requirements. Software developers then design and develop the software based on the identified needs. The project is usually managed by a project manager, who is responsible for planning the project, supervising the work and coordinating resources. Project management methods include using Gantt charts, scheduling tasks, tracking project status and organizing team communication. [25]

Advantages:

- Continuous delivery: allows frequent, automated delivery and testing.
- Improved collaboration between developers and operators.
- Improves product quality and stability.

Disadvantages:

- Requires major changes from traditional operation.
- Requires complex infrastructure and tools.

The choice and application of a specific method or methodology depends on the requirements and circumstances of the project and company. Often, companies combine or tailor methodologies

to best fit their own development processes and needs, including plan-driven and agile methodologies.

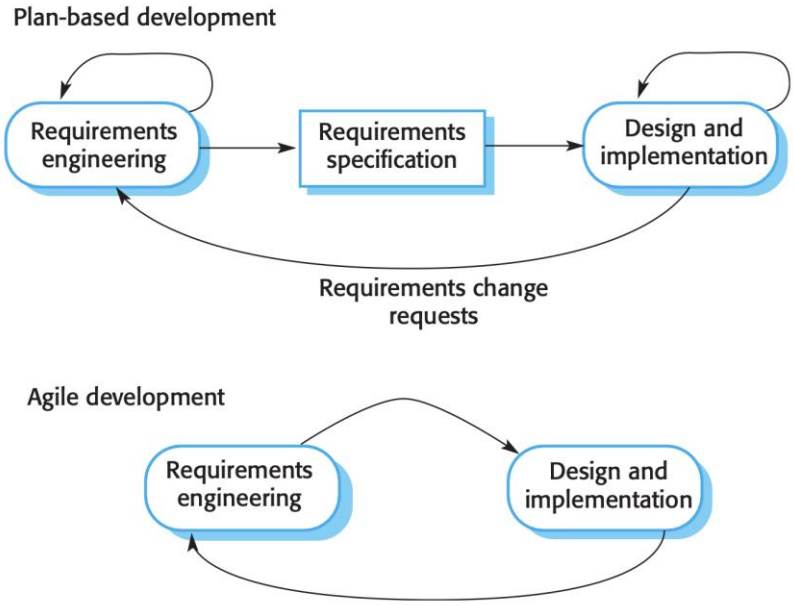

Figure 6: Plan-driven and agile development [3]

5 Agile software development

Agile methodologies offer a flexible and iterative approach to software development, allowing for rapid response to change, continuous improvement and meeting end-user needs. Agile methodologies are diverse and can be tailored to project-specific needs and team preferences. In some cases, hybrid solutions tailored to projects are also used, combining agile methodologies with other development frameworks or methodologies to better fit the specific characteristics of the projects. Agile software development methodologies, especially Scrum, are popular and widespread among companies for several reasons. Overall, agile methodologies allow projects to be implemented in a more flexible and adaptable way. As mentioned above, development teams work in shorter iterations (sprints) where they receive regular feedback from customers or users. This allows for rapid changes and revisions to the development direction as business needs change. Agile methodologies would focus on customers, who work closely with the development team, taking an active part in planning and prioritizing sprints. This allows developers to get real-time feedback from customers and better understand real needs. They greatly enhance transparency and communication between the development team and customers. Regular scrum meetings (e.g. daily standing reception) and end of sprint presentations allow for continuous communication and feedback between developers and customers. This helps developers to quickly resolve issues and understand the project process.

They emphasize iterative development, where software is developed incrementally and incrementally. This allows early results and prototypes for customers to build on for further development and improvements. Teamwork and ownership among developers are strengthened. Teams plan and organize their work independently within sprints and take joint responsibility for results. This can increase team motivation and efficiency.

5.1 Scrum

Scrum is one of the most widely used and best-known agile methodologies. [26] In the Scrum framework, the project is divided into short iterations, called sprints, usually 1-4 weeks. In each sprint, the development team reviews the backlog (list of tasks to be completed), selects the tasks planned for the sprint, and commits to completing them. A daily stand-up meeting is held during the sprint

where team members share progress and problems. At the end of the sprint, they present the work done and review the results achieved.

5.2 Kanban

Kanban is an agile methodology based on visualization. [27] On a Kanban board or virtual tool, cards represent development tasks. The columns of the board represent the states of the tasks, such as "Waiting", "In Progress" and "Done". The team reviews the board, moves the cards between the columns according to the progress of the tasks, and limits the number of cards on the board to optimize workload and minimize congestion.

5.3 Extreme Programming (XP)

Extreme Programming is an agile methodology that focuses on the development process and software quality. [28] XP emphasizes continuous feedback, small incremental development and test-driven development. In XP, developers work in pairs, often swapping partners. In addition, XP has a strong emphasis on code quality, refactoring and automated testing.

5.4 Lean Software Development

Lean Software Development combines agility with Lean manufacturing methodology. [22] The principle is to maximize the creation of value for the end user while minimizing waste and redundancy. Lean Software Development emphasizes continuously improving processes, transparency, a steady workload, and small batches that are often released based on end-user feedback.

The reasons described above contribute to making agile methodologies, especially Scrum, the preferred choice of companies in software development projects [3,8,13,21,26-28], and thus Scrum is also a preferred methodology in my research. There are other preferences and variants of agile methodologies, such as Kanban and Extreme Programming (XP) mentioned earlier, which have a different focus and approach, but are equally based on agile principles.

5.5 Test Driven Development and the Scrum methodology

The agile Scrum methodology is often combined with TDD (Test-Driven Development) [29], as both focus on the agility and quality of the development process. In the Scrum framework, TDD is usually integrated into sprints, where writing tests, implementing code and running tests are part of the iteration. Tests can help to monitor the progress of the development work and the client requirements and improve the efficiency of flexible and iterative development cycles.

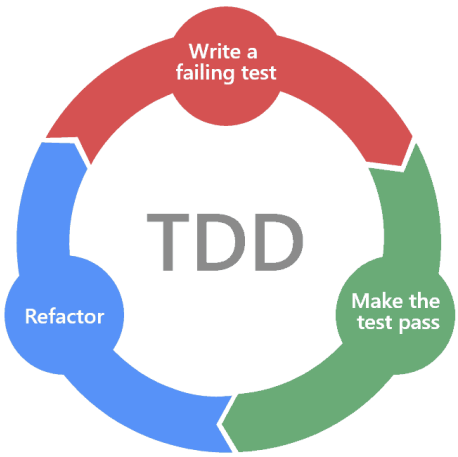

Figure 7: Test-Driven Development process [59]

Test-Driven Development (TDD) is a development method that focuses on testing. The basic principle is that developers first write tests for the desired functionality, and only then implement the code to ensure that the tests run successfully. [30] Developers first write tests for the desired functionality, which capture the expected inputs, outputs, and expected behavior of the software.

Initially, these defined tests will not run because the code for them has not yet been implemented, so as a first step, developers implement the code necessary to successfully run the tests. The code will be extended only as much as the tests require, thus minimizing redundant code. The tests are re-run with the freshly implemented code and if the tests run successfully, it means that the new functionality is working properly, or if the tests fail, the developers continue to work on improving the code and re-running the tests.

6 "Development" of artificial intelligence (AI) based models

Learning AI models is very different from the classical software development process. Whereas in classical software development, developers explicitly describe the operation and behavior of the computer, in AI model learning, the models learn from the input data themselves. [31] First, it is necessary to collect data of sufficient quantity and quality. This data can be labeled data, where the data is assigned an expected output or label, or unlabeled data, where there is no predefined output. The collected data must be prepared to train the models. This may include normalizing the data, ensuring scale, removing noise or filling in missing data. The appropriate AI model must then be chosen to solve the problem. The structure and parameters of the model must also be set. [32] The model is trained with the available data. This means that the data is applied to the model and the model iteratively fine-tunes its own weights or parameters to make the outputs as accurate as possible to the inputs. After training, we evaluate and validate the model. This can be done using separate test data or cross validation. We examine the performance of the model, such as accuracy, efficiency or other relevant metrics. If necessary, we can fine-tune the model in further iterations to achieve even better results. This may involve adjusting hyperparameters, data or model structure. Learning AI models is therefore based on automatic learning from training data, where models produce outputs based on inputs. [33-34] This is an iterative and experimental process that improves the performance and accuracy of the models by fine-tuning the data and the model.

6.1 Applying Agile methodology to AI models

So, the question rightly arises: can agile methodology (e.g. scrum) be used to teach AI models? In short, the answer is that yes, Scrum methodology and teaching AI models can be combined to effectively manage the AI project and development process, but let's look at how and at what level this is possible. Although Scrum was originally designed for software development projects, the Agile principles and the Scrum framework can be applied to teaching AI models. [35] Agile methodologies, such as Scrum or Kanban, are based on the principles of flexibility, rapid iterations and customer focus.

Before the start of an AI project, we can create a product backlog that contains a prioritized list of AI models and features to be implemented in the project. This will help clarify business needs and project goals. In the Scrum framework, development work is divided into sprints. You can also plan sprints for teaching AI models, where you define the learning objectives and the work to be done. This includes data collection, data processing, model building and teaching. The ceremonies defined in the Scrum framework, such as sprint planning, daily stand-up reception and end-of-sprint presentation, can also be useful during the teaching process. AI developers and the team can communicate regularly, share development status and progress, and customers or users can provide feedback on the performance of the models. At the end of the sprint, an updated model or "product increment" is delivered as a result of the AI model learning. This allows the models to be tested, evaluated and further fine-tuned. Further modifications can be made based on feedback from customers or users. When teaching an AI model, priorities and tasks can be managed in an agile way. Backlog and sprint planning can be modified based on business needs and user feedback. Flexibility allows you to react quickly to changing circumstances and new requirements. A Jira Scrum board can be used to keep a record of the "project", which specifically supports an agile organization.

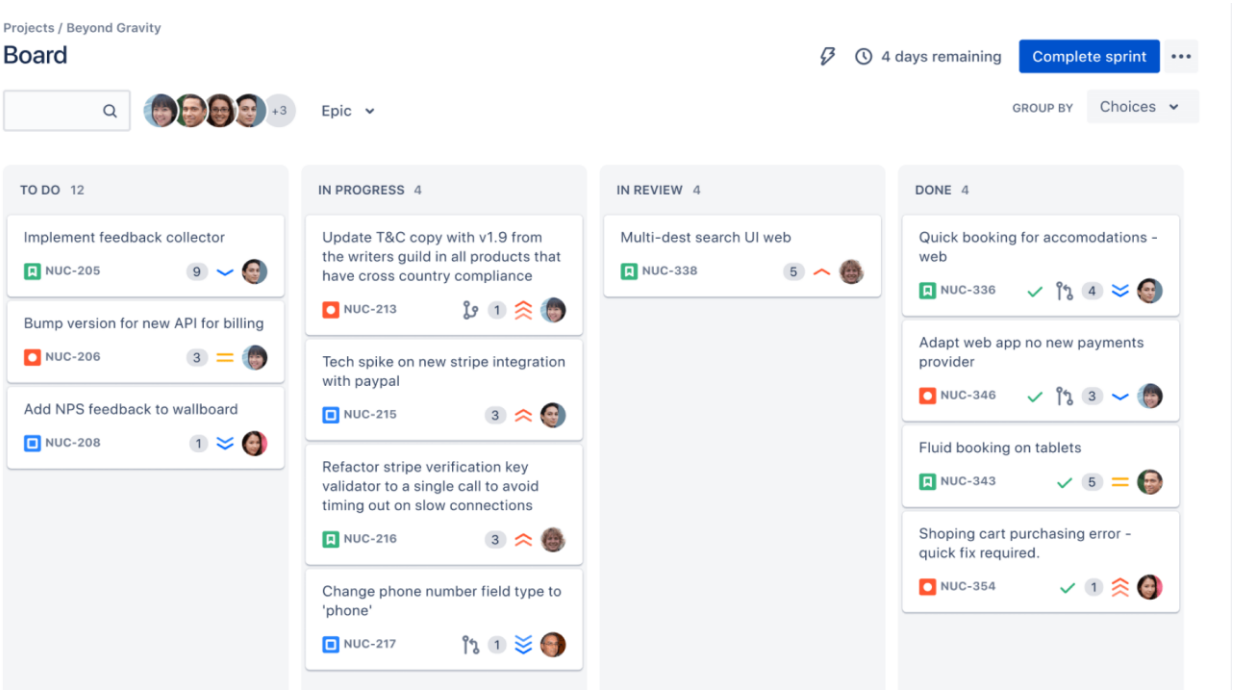

Figure 8: Jira software – Scrum board [60]

Learning the AI model is a complex process and using the Scrum methodology is only one possible approach. Depending on the nature of the AI project, the size of the team and the project context, other agile methodologies or hybrid solutions may be useful. The key is to incorporate agility and iterative development into AI project management to respond quickly to changes and achieve the desired results.

There are several methodologies and approaches to agile software development that can be applied to AI projects, depending on the nature of the project, the size of the team and the project context. Kanban, for example, is an agile methodology that focuses on workflow visualization and process continuity and can be used in AI projects. In this methodology, development tasks are represented as cards on visual boards or virtual devices. The cards represent each task and indicate its status (e.g., "waiting", "in progress", "done"). The team moves the tasks continuously around the board, allowing for process transparency and a steady workload. In addition to the Kanban methodology, even the Lean methodology is an excellent application for an AI project, as Lean development focuses on minimizing waste and creating the best possible value for the end user. This methodology strongly emphasizes continuous improvement, customer centricity, and small batches released on a regular basis.

Due to the specificities of AI projects, it is therefore often the case that Agile methodologies are combined with other development frameworks and methodologies, as it allows flexibility and the ability to meet project-specific needs. The choice of the right combination between Agile methodologies and AI project management is project and team driven. Project specific requirements, team size and skills, and client and user needs should be considered to select the most appropriate methodology.

6.2 "Developing" NLP models with agile tools

The NLP is a branch of artificial intelligence and computational linguistics that aims to enable machines to interpret and generate natural language. [36] This includes the recognition of entities (people, places, things, etc.), syntactic and semantic analysis, text categorization and the recognition of emotional content. It enables machines to generate texts, for example to provide answers or to communicate with chatbots. This process can include syntactic and semantic analysis, the use of language models and machine learning algorithms. In addition, NLP can be applied to machine translation and sentiment analysis, which allows automatic translation of texts from one language to

another and identification of the emotional sentiment of texts. Machine translation involves text segmentation, the use of language models, translation rules and data analysis.

Developers and researchers are constantly working to improve and extend NLP technologies to make natural language processing even more efficient and accurate. In my research position, I am mainly involved in research on text analytics, including classification models to categories incoming customer enquiries (e.g. categorizing complaint letters by MNB (Magyar Nemzeti Bank, in English Hungarian National Bank) subject or processing group.

6.3 AI-based NLP models

Natural Language Processing is closely related to AI technology, as NLP applies AI methods and algorithms to process and interpret natural language data. [41] NLP can therefore be seen as a subset of AI, which enables machines to understand and generate human language. NLP uses machine learning techniques applied in AI to analyse and model language data [42] Machine learning algorithms are used to identify language patterns, rules and features that enable machines to learn about language structures and the meaning of content. For example, recurrent neural networks (RNNs) or the generalized sequence-by-sequence model (Seq2Seq) used in deep learning can be effective in text translation or text generation. Natural language models are complex mathematical representations that can encode the meaning and structure of words, sentences and texts. Models, such as Word2Vec, GloVe or Transformer [44-46], can learn language vector representations and determine similarities or meaning relationships between words.

6.4 Dictionary-based language models

In the development of dictionary-based NLP models, a predefined dictionary or collection of terms is used to analyze texts and identify their meaning. [47] One of such models is the SPSS Modeler as a development environment, which is a widely used data mining and predictive analytics tool [48-49] and offers several features and techniques in the field of text analytics. [50]

Text analytics allows you to analyze and interpret unstructured text data to extract valuable information. At my research site, the various tools of SPSS Modeler, on which my first classification model was built, are used extensively, so it is worthwhile to cover these features from the point of view of the level and spectrum of use of classical software development methodologies in the development of such a language model. They allow the scanning and preparation of text data which includes the import of text sources, the pre-processing of text (e.g. text cleaning, normalization, tokenization, etc.) and the structuring of data for analysis.

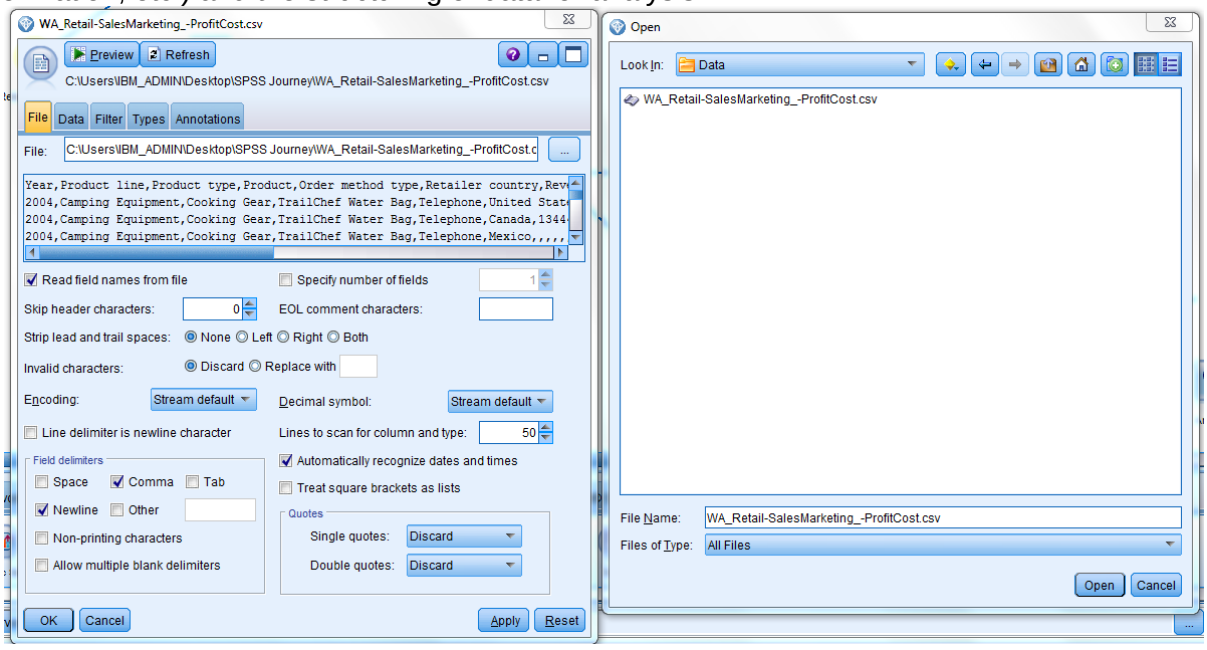

Figure 9: SPSS modeler data reading [61]

SPSS Modeler offers a range of text analysis techniques and algorithms to help you process and interpret your data. These include text tagging, term frequency calculation, synonym identification, categorization, sentiment analysis, entity recognition and others. This means that predictions and models can be created based on the analysis of text data. For example, we can model user preferences, customer behavior or other predictions based on textual data. The use of text analytics features allows users to discover and make sense of textual data. With its systematic analysis workflows, built-in algorithms and graphical interface, SPSS Modeler effectively supports text analytics projects and facilitates the extraction and exploitation of information from text data, just as you would in developing software in the traditional sense.

The development of dictionary-based NLP models can consist of several steps [51-52], which first require data collection. This can be textual data sources, documents, websites or other sources that contain the terms or words on which the dictionary is built. The data may be corpora or manually compiled databases. After data collection, the next step is to create the dictionary. This involves selecting relevant terms or words and assigning synonyms, word types or other features. The dictionary can be manually compiled or created using automated methods based on the data collected. Once the dictionary is complete, the models must learn the meaning or semantics of each word. Dictionary-based models often also identify relationships between terms, for example, they may define categories or topics associated with words in the dictionary, so that the models can categorize texts or identify topics.

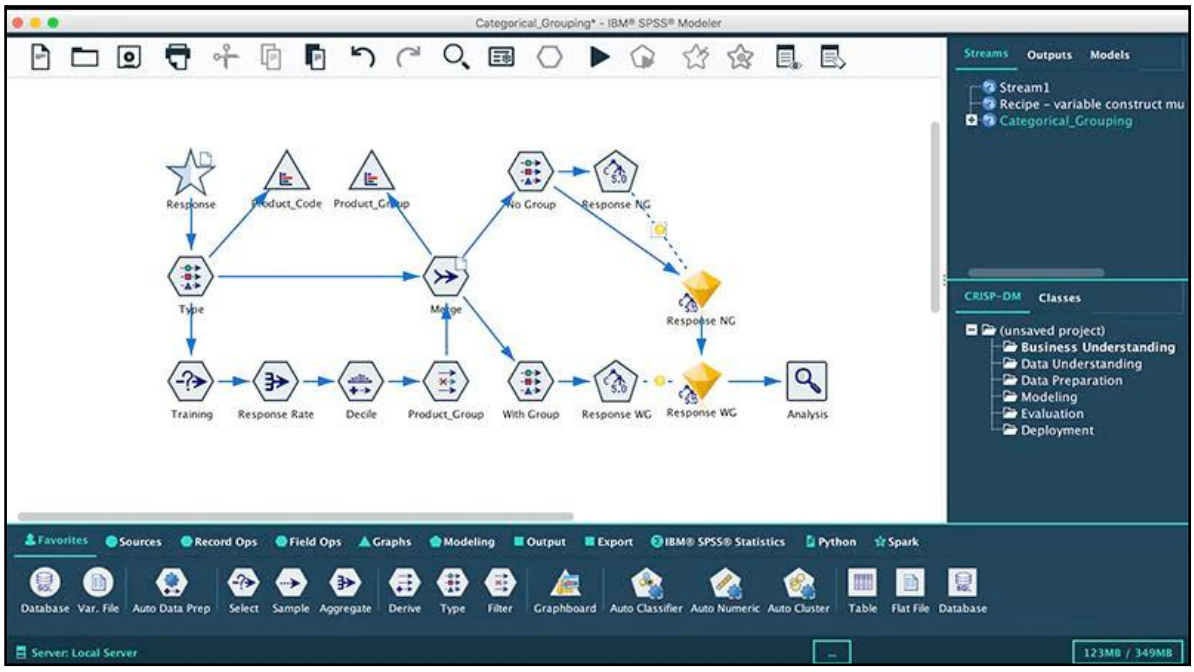

Figure 10: SPSS Modeler IDE [53]

The use of SPSS Modeler as a development tool greatly supports and ensures the development of dictionary-based NLP models, and provides a graphical interface for data preparation, model design and predictive analysis that is relatively simple and straightforward to use.

7 Developing NLP models using agile methodology

I started my engineering career as a DevOps developer, during which we built an agile team and introduced the agile methodology, so when I started developing dictionary-based NLP models, the idea to develop these models using the tools used in the agile methodology came almost immediately. When dictionary-based NLP models are developed using agile methodologies, they usually allow for flexible design where the primary goal is continuous improvement and an iterative approach. When developing dictionary-based NLP models, it may also be important to use iterative

processes, where models are progressively refined and improved in response to feedback and customer needs. Since agile methodologies typically divide the project into sprints, specific goals and tasks defined in sprints can also help to ensure efficient progress and regular feedback when developing dictionary-based NLP models. These principles can also be applied to the development of artificial intelligence (AI) models. The flowchart below shows how the agile methodology can be applied in practice to the development of the artificial intelligence models used in our research and development:

Table 1. Agile methodology can be applied in practice to the development of the artificial intelligence models

1. Project initiation and planning	*2. Identify customer needs*	*3. First iteration (Sprint)*	*4. Feedback and iteration*	*5. Further iterations*
The first step is to define the MI project and define its objectives.	Work closely with customers or stakeholders to capture customer needs and requirements.	During the first iteration, build an MI model that meets the initial requirements.	Continuously communicate with customers or users and collect their feedback.	Repeat the previous steps in several iterations to further improve the AI model.
The first step is to start with the initiation of the MI project.	Define goals and priorities for the first iteration of the project.	The first step is to establish the first iteration.	Modify and refine the models and th e system based on the first iteration.	Continue to refine the models throughout the iterations based on customer needs and feedback.
		Develop and test the model to produce the first functional prototype.	Regularly monitor and evaluate the performance of models during iteration cycles.	
6. Testing and validation	*7. Display and finalization*	*8. Maintenance and further development*	*9. Closure Evaluation*	
The MI model is tested and validated on different data sets and scenarios.	Make the completed AI models and system available to customers or users.	Ongoing maintenance and updating of the MI models and system according to changing requirements and customer needs.	Evaluate the results and lessons learned at the end of the project.	
Test the performance and reliability of the models.	Prepare documentation and instructions for use.	Add new ideas and features during further iterations.	Document achievements and areas that need further improvement.	

Figure 11: Process steps and description of the agile AI model development

Before using this agile methodology, the language models were developed as a traditional project task, which means that only the major milestones and their commitment dates were defined and recorded. In order to monitor the project, regular project meetings and status meetings were scheduled, where tasks were measured back with the development teams using an Excel-based action tracker. The composition of the teams (experts) was defined according to the expertise and, as in the waterfall model, the individual process steps were agreed within the teams concerned. We had used the agile methodology and matrix-type team building in classic software development, so after the agile implementation it was trivial for me to develop the language models based on a new methodology that better fit with the rest of the team, and also to rethink the team composition. This meant that the team composition was not only composed of model developers, but also mixed, with

a regrouping of experts from the business domain, thus building a matrix-like agile squad that included agile ceremonies (e.g. daily stand up, retrospective and 2-week scrum meetings). My experience that an "agile matrix" organization is an organizational structure that applies agile methodologies and adopts a different approach to work than classical hierarchical organizations. An agile matrix organization can have several advantages over a classical hierarchical organization, as it allows for faster response to changing market conditions and better collaboration between teams. The biggest advantage of an agile organization is that they facilitate faster decision making, as decisions are made by teams on the ground, without long hierarchical approval processes. There is a strong emphasis on teamwork and effective communication. Experts work directly together, which increases problem-solving ability and reduces communication barriers.

When developing dictionary-based NLP models, regular feedback, tests and user feedback can be used to refine and improve the models. As in traditional software development, the use of agile methodologies in NLP model development allows developers to be flexible to customer needs, to continuously develop and refine dictionary-based NLP models, and to receive direct feedback from customers during the development process. This methodology can facilitate the development of more efficient and effective dictionary-based NLP models. In the field of language models, however, we can talk not only about dictionary-based language models, but also about the highly successful AI-based models, where the question of whether classical software development methodologies (e.g. scrum) can be used to teach AI-based models is also an interesting one. However, it is important to first clarify the difference between AI and dictionary-based NLP models, as their "development" is different.

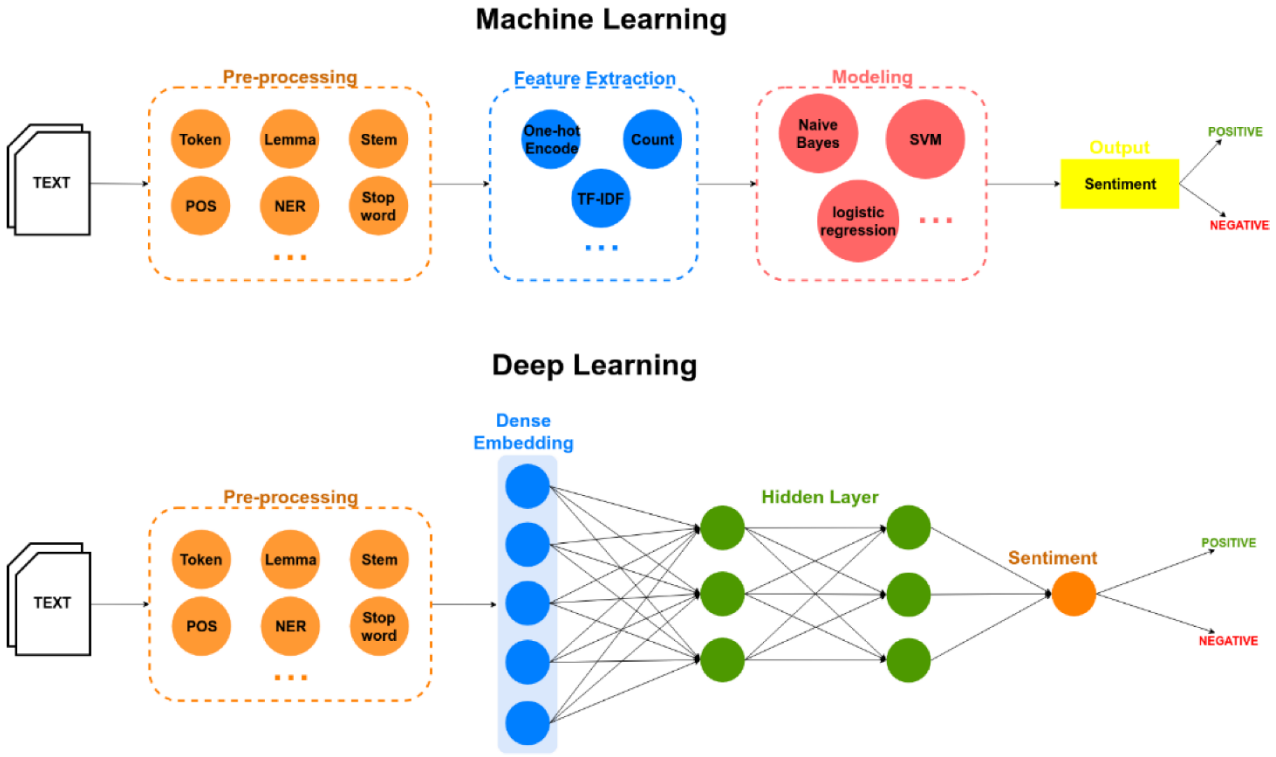

Figure 12: Difference between the AI-based and Dictionary-based NLP model architecture [62]

AI-based models are based on prior learning, where they learn language structures and features from large amounts of raw text and are thus able to generate and make predictions about textual data. Dictionary-based NLP models, on the other hand, rely on predefined dictionaries, in which the meanings, properties and other information of words are predefined. AI-based models are highly flexible and capable of discovering linguistic patterns and meanings without relying on pre-defined rules or dictionaries. This allows them to handle language tasks for which there is no prior dictionary or definition. Dictionary-based NLP models, on the other hand, rely on specific dictionaries and can only operate based on predefined words and meanings. The AI bases models requires a

large amount of pre-learned data and computational resources to operate efficiently, and their generalization ability is stronger because they learn a wide range of language structures and features. Dictionary-based NLP models, on the other hand, rely on less data, as dictionaries are generally smaller and require fewer resources to operate. Overall, AI-based models have greater flexibility and generalization capabilities, while dictionary-based NLP models are more specific and rely on predefined dictionaries. The choice depends on the specific task, the amount of data available and the objectives, and the application of a method tailored to the needs of a project or application may be optimal.

8 Conclusion

The development of the NLP model using agile methodologies has brought many benefits. NLP model development is often a project with complex and changing requirements, and the agile approach allows flexibility to respond to changing requirements. The results of the applied methodology are summarized in the table below:

Table 2. Result of the applied agile NLP model development

Advantages	*Disadvantages*
Agile methodologies allow rapid prototyping to validate initial concepts of the NLP model.	Agile development processes can sometimes seem chaotic and lead to a loss of control. Managing agile projects requires good leadership and effective coordination of teams.
Customer needs and data often change during NLP projects. Agile methodologies allow for flexibility in responding to these changes, enabling project management based on the most up-to-date requirements.	They may tend to develop rapid prototypes and short-term solutions, which can lead to long-term technical debt.
Agile methodologies continuously incorporate customer feedback and fresh requirements into the development process.	Changing requirements and iterations can increase the complexity and cost of projects, especially if they are not properly planned and controlled.
Agile methodologies promote collaboration and effective communication between different professionals such as developers, designers, data analysts and domain experts.	The high pace and density of iterations in agile projects can increase workload and lead to team burnout.
Enable early feedback from users or customers, allowing the model to be refined and improved to meet current needs and expectations.	Agile methodologies are not ideal for all projects. For complex, well-defined and long-term projects, traditional project management methods may be more effective.
Allow continuous testing and validation to ensure the quality and performance of the model. Errors are detected early and immediate corrections are possible.	Maintaining quality standards can be challenging as priorities are constantly changing and testing timescales can be tight.
An agile organizational structure creates an environment where teams have more autonomy and are accountable for their tasks. This increases staff motivation and contributes to better results.	
Allow the model to be continuously improved and maintained after the project is completed. This keeps the model up-to-date and efficient.	

Based on the information and methodologies presented in this article, we can say that the principles of scrum, such as sprints, product and sprint backlog, can be effective in teaching GPT models. The agile scrum methodological aspects can be applied to the teaching of GPT models as follows, since we can apply the concepts in an equivalent way to GPT models. When teaching GPT models, the desired features, suggestions for improving the performance of the model and

development objectives can be recorded in the product backlog. The backlog can be prioritized, and the scrum team can select tasks from the backlog during sprints. During the sprints, the scrum team works on specific tasks to progress the learning of the GPT model. At the end of each sprint, progress and lessons learned are evaluated. At the beginning of each sprint, the scrum team selects the tasks from the product backlog that will be executed in the sprint. These tasks will be included in the sprint backlog, in which the tasks, expected results and required resources are described in detail. During the training of the GPT models, the scrum team holds a Daily Scrum meeting where the results of the previous day, current tasks and possible obstacles are discussed. This gives the team the opportunity to coordinate and keep up to date on the development process.

Applying agile methodologies to NLP model development can help increase project efficiency, achieve customer satisfaction and improve model quality in the face of changing requirements. It is important to understand and clarify, however, that agile methodologies are not applicable to all projects, and it may be important to combine agile and traditional methodologies in the development of NLP models or to select the best approach based on the project specifics. Good planning, well-functioning teamwork and good project management can be essential for success.

References

[1] Sawyer, S., & Guinan, P. J. (1998). Software development: Processes and performance. IBM systems journal, 37(4), 552-569.

[2] Goble, C. (2014). Better software, better research. IEEE Internet Computing, 18(5), 4-8.

[3] Sommerville, I. (2007). Sommerville: Software Engineering.

[4] Bosch, J. (2001, May). Software product lines: organizational alternatives. In Proceedings of the 23rd International Conference on Software Engineering. ICSE 2001 (pp. 91-100). IEEE.

[5] Weiss, D. M., & Lai, C. T. R. (1999). Software product-line engineering: a family-based software development process. Addison-Wesley Longman Publishing Co., Inc.

[6] Kadmiry, M. (2021). The comparison between the process-oriented approach and the product-oriented approach in teaching writing the case of Moroccan EFL students in preparatory classes for the grandes ecoles. Arab World English Journal (AWEJ) Volume, 12.

[7] Sedano, T., Ralph, P., & Péraire, C. (2019, May). The product backlog. In 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE) (pp. 200-211). IEEE.

[8] Larman, C. (2004). Agile and iterative development: a manager's guide. Addison-Wesley Professional.

[9] Zahran, S. (1998). Software process improvement: practical guidelines for business susccess. Addison-Wesley Longman Ltd..

[10] Balsamo, S., Di Marco, A., Inverardi, P., & Simeoni, M. (2004). Model-based performance prediction in software development: A survey. IEEE Transactions on Software Engineering, 30(5), 295-310.

[11] Budgen, D. (2003). Software design. Pearson Education.

[12] Szalvay, V. (2004). An introduction to agile software development. Danube technologies, 3.

[13] Stoica, M., Mircea, M., & Ghilic-Micu, B. (2013). Software development: agile vs. traditional. Informatica Economica, 17(4).

[14] Kumar, G., & Bhatia, P. K. (2012). Impact of agile methodology on software development process. International Journal of Computer Technology and Electronics Engineering (IJCTEE), 2(4), 46-50.

[15] Rodríguez, P., Markkula, J., Oivo, M., & Garbajosa, J. (2012). Analyzing the drivers of the combination of lean and agile in software development companies. In Product-Focused Software Process Improvement: 13th International Conference, PROFES 2012, Madrid, Spain, June 13-15, 2012 Proceedings 13 (pp. 145-159). Springer Berlin Heidelberg.

[16] Kasim, T., Haracic, M., & Haracic, M. (2018). The improvement of business efficiency through business process management. Economic Review: Journal of Economics and Business, 16(1), 31-43.

[17] Vanhaverbeke, W., & Chesbrough, H. (2014). A classification of open innovation and open business models. New frontiers in open innovation, 6, 50-68.

[18] Sakas, D., Vlachos, D., & Nasiopoulos, D. (2014). Modelling strategic management for the development of competitive advantage, based on technology. Journal of Systems and Information Technology, 16(3), 187-209.

[19] Petersen, K., Wohlin, C., & Baca, D. (2009). The waterfall model in large-scale development. In Product-Focused Software Process Improvement: 10th International Conference, PROFES 2009, Oulu, Finland, June 15-17, 2009. Proceedings 10 (pp. 386-400). Springer Berlin Heidelberg.

[20] Adenowo, A. A., & Adenowo, B. A. (2013). Software engineering methodologies: a review of the waterfall model and object-oriented approach. International Journal of Scientific & Engineering Research, 4(7), 427-434.

[21] Reifer, D. J. (2002). How good are agile methods? IEEE software, 19(4), 16-18.

[22] Poppendieck, M., & Cusumano, M. A. (2012). Lean software development: A tutorial. IEEE software, 29(5), 26-32. DOI: 10.1109/MS.2012.107

[23] Jabbari, R., bin Ali, N., Petersen, K., & Tanveer, B. (2016, May). What is DevOps? A systematic mapping study on definitions and practices. In Proceedings of the Scientific Workshop Proceedings of XP2016 (pp. 1-11).

[24] Banica, L., Radulescu, M., Rosca, D., & Hagiu, A. (2017). Is DevOps another project management methodology? Informatica Economica, 21(3).

[25] White, D., & Fortune, J. (2002). Current practice in project management—An empirical study. International journal of project management, 20(1), 1-11. DOI.: 10.1016/S0263-7863(00)00029-6

[26] Schwaber, K. (1997). Scrum development process. In Business Object Design and Implementation: OOPSLA'95 Workshop Proceedings 16 October 1995, Austin, Texas (pp. 117-134). Springer London.

[27] Kirovska, N., & Koceski, S. (2015). Usage of Kanban methodology at software development teams. Journal of applied economics and business, 3(3), 25-34.

[28] Erickson, J., Lyytinen, K., & Siau, K. (2005). Agile modeling, agile software development, and extreme programming: the state of research. Journal of Database Management (JDM), 16(4), 88-100.

[29] Aggarwal, V., & Singhal, A. (2019). Empirical study of test-driven development with scrum. In Advances in Computing and Data Sciences: Third International Conference, ICACDS 2019, Ghaziabad, India, April 12–13, 2019, Revised Selected Papers, Part II 3 (pp. 13-21). Springer Singapore. DOI: 10.1007/978-981-13-9942-8_2

[30] Hammond, S., & Umphress, D. (2012, March). Test driven development: the state of the practice. In Proceedings of the 50th Annual Southeast Regional Conference (pp. 158-163).

[31] Chen, B., Jiang, J., Wang, X., Wan, P., Wang, J., & Long, M. (2022, February). Debiased Self-Training for Semi-Supervised Learning. In Advances in Neural Information Processing Systems.

[32] Wiyono, S., & Abidin, T. (2019). Comparative study of machine learning KNN, SVM, and decision tree algorithm to predict student's performance. International Journal of Research-Granthaalayah, 7(1), 190-196.

[33] Learning, D. (2020). Deep learning. High-dimensional fuzzy clustering.

[34] Kelleher, J. D. (2019). Deep learning. MIT press.

[35] Kristiadi, D. P., Sudarto, F., Sugiarto, D., Sambera, R., Warnars, H. L. H. S., & Hashimoto, K. (2019, November). Game Development with Scrum methodology. In 2019 International Congress on Applied Information Technology (AIT) (pp. 1-6). IEEE.

[36] Liddy, E. D. (2001). Natural language processing.

[37] Chopra, A., Prashar, A., & Sain, C. (2013). Natural language processing. International journal of technology enhancements and emerging engineering research, 1(4), 131-134.

[38] Hirschberg, J., & Manning, C. D. (2015). Advances in natural language processing. Science, 349(6245), 261-266.

[39] Jiang, K., & Lu, X. (2020, November). Natural language processing and its applications in machine translation: A diachronic review. In 2020 IEEE 3rd International Conference of Safe Production and Informatization (IICSPI) (pp. 210-214). IEEE. DOI: 10.1109/IICSPI51290.2020.9332458

[40] Abbaszade, M., Salari, V., Mousavi, S. S., Zomorodi, M., & Zhou, X. (2021). Application of quantum natural language processing for language translation. IEEE Access, 9, 130434-130448. DOI: 10.1109/ACCESS.2021.3108768

[41] Qiu, X., Sun, T., Xu, Y., Shao, Y., Dai, N., & Huang, X. (2020). Pre-trained models for natural language processing: A survey. Science China Technological Sciences, 63(10), 1872-1897.

[42] Khan, W., Daud, A., Nasir, J. A., & Amjad, T. (2016). A survey on the state-of-the-art machine learning models in the context of NLP. Kuwait journal of Science, 43(4).

[43] Lopez, M. M., & Kalita, J. (2017). Deep Learning applied to NLP. arXiv preprint arXiv:1703.03091.

[44] Afshinpour, B., Groz, R., Amini, M. R., Ledru, Y., & Oriat, C. (2020, December). Reducing Regression Test Suites using the Word2Vec Natural Language Processing Tool. In SEED/NLPaSE@ APSEC (pp. 43-53).

[45] Pennington, J., Socher, R., & Manning, C. D. (2014, October). Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).

[46] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30.

[47] Das Dawn, D., Khan, A., Shaikh, S. H., & Pal, R. K. (2022). A dictionary-based model for bengali document classification. Applied Intelligence, 1-20.

[48] McCormick, K., & Salcedo, J. (2017). IBM SPSS Modeler essentials: Effective techniques for building powerful data mining and predictive analytics solutions. Packt Publishing Ltd.

[49] Wendler, T., & Gröttrup, S. (2016). Data mining with SPSS modeler: theory, exercises and solutions. Springer.

[50] Achananuparp, P. SOCIAL MEDIA & PUBLIC OPINION: EFFECTIVENESS OF SENTIMENT TEXT ANALYTICS ON SOCIAL MEDIA DATA.

[51] Reveilhac, M., & Morselli, D. (2022). Dictionary-based and machine learning classification approaches: a comparison for tonality and frame detection on Twitter data. Political Research Exchange, 4(1), 2029217.

[52] Song, M., Yu, H., & Han, W. S. (2015). Developing a hybrid dictionary-based bio-entity recognition technique. BMC medical informatics and decision making, 15(1), 1-8.

[53] Figure reference – IBM SPSS Modeler: https://www.ibm.com/products/spss-modeler (downloaded: 24/10/2023)

[54] Figure reference – Cleveroad: https://www.cleveroad.com (downloaded: 24/10/2023)

[55] Figure reference – GeeksforGeeks: https://www.geeksforgeeks.org/software-engineering-classical-waterfall-model/ (downloaded: 24/10/2023)

[56] Figure reference – ProMan Consulting Kft.: https://promanconsulting.hu/mi-az-agilis-modszertan-legelterjedtebb-agilis-modszertanok/ (downloaded: 24/10/2023)

[57] Figure reference – TMS: https://tms-outsource.com/blog/posts/lean-software-development/ (downloaded: 24/10/2023)

[58] Figure reference – BairesDev: https://www.bairesdev.com/devops/ (downloaded: 24/10/2023)

[59] Figure reference – Marsner: https://marsner.com/blog/why-test-driven-development-tdd/ (downloaded: 24/10/2023)

[60] Figure reference – Atlassian: https://www.atlassian.com/software/jira/features (downloaded: 24/10/2023)

[61] Figure reference – IBM Developer: https://developer.ibm.com (downloaded: 24/10/2023)

[62] Figure reference – MDPI: https://www.mdpi.com/2079-9292/9/3/483 (downloaded: 24/10/2023)