AUTOMATIC TRANSFORMATION OF USER STORIES IN TO UML USE CASE DIAGRAM

PROJECT REPORT

Submitted By

ASIFA SIRAJ (Reg. No: MCT21MCA-2017)

to

APJ Abdul Kalam Technological University

In partial fulfillment of the requirements for the award of Degree in

MASTER OF COMPUTER APPLICATIONS

DEPARTMENT OF COMPUTER APPLICATIONS

MOHANDAS COLLEGE OF ENGINEERING & TECHNOLOGY

Anad, Nedumangad,

Thiruvananthapuram 695541

2023

DEPARTMENT OF COMPUTER APPLICATIONS
MOHANDAS COLLEGE OF ENGINEERING AND TECHNOLOGY
Anad, Nedumangad, Trivandrum-695541

CERTIFICATE

This is to certify that the main project report entitled **"AUTOMATIC TRANSFORMATION OF USER STORIES IN TO UML USE CASE DIAGRAM "** submitted by **MS. ASIFA SIRAJ (Register No: MCT21MCA-2017)** to the **APJ Abdul Kalam Technological University**, in partial fulfillment of the requirements for the award of the Degree of **Master of Computer Applications,** is a bonafide record of the project work carried out by her under my/our guidance and supervision. This report, in any form, has not been submitted to any other University or Institute for any purpose.

Internal Supervisor(s) **Project Coordinator**

Head of the Department **External Examiner**

DECLARATION

I undersigned hereby declare that the main project report for the "**AUTOMATIC TRANSFORMATION OF USER STORIES IN TO UML USE CASE DIAGRAM** " submitted for partial fulfillment of the requirements for the award of the degree of Master of Computer Applications from APJ Abdul Kalam Technological University, Kerala, is a bonafide work done by me under the supervision of **Prof. Dr. JEEJA G.S**. This submission represents my ideas in my own words and where ideas or words of others have been included, I have adequately and accurately cited and referenced the original sources.

I also declare that I have adhered to academic honesty and integrity ethics and have not misrepresented or fabricated any data, idea, fact, or source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the Institute and or the University and can also evoke penal action from the sources which have thus not been properly cited, or from whom proper permission has not been obtained. This report has not been previously formed as the basis for the award of any degree, diploma, or similar title of any other university.

Place: Trivandrum **Submitted by**

Date: **ASIFA SIRAJ**

ACKNOWLEDGEMENT

I am overwhelmed in all humbleness and gratefulness to acknowledge in depth all those who have helped me to put these ideas, well above the level of simplicity into something concrete. I would like to express my special thanks and gratitude to our principal **Dr. S. SHEELA** and our Director **Dr. ASHALATHA THAMPURAN** for providing all the necessary facilities.

I am greatly thankful to **Prof. SREEJA K**, (HOD, Department of Computer Applications) for her kind cooperation and guidance throughout the course of my project.

I am also thankful to my Guide **Prof. JAYANTHI.T,** (Associate Professor, Department of Computer Applications) for her kind cooperation and guidance throughout the course of my project.

I am also thankful to my Guide **Prof. JEEJA.G.S,** (Associate Professor, Department of Computer Applications) for her kind cooperation and guidance throughout the course of my project.

I would also like to acknowledge the contributions of my people from discord and other communities who have been a source of motivation and encouragement throughout this project.

Last but certainly not least, I would also like to thank all the staff of the Department of Computer Applications for their help and cooperation.

With gratitude

ASIFA SIRAJ

TABLE OF CONTENTS

ABSTRACT

The User Story format has become the most popular way of expressing requirements in Agile methods.UML is a way of visualising a software program using a collection of diagrams.The purpose of this paper is to present a new approach that automatically transforms user stories into UML diagrams. This approach aims to automatically generate UML diagrams, mainly usecase diagrams. User stories are written in natural language (English), so the use of a natural language processing tool was necessary for their processing.Hence, we are using NLP techniques.Our proposed system utilizes a deep learning model trained on a large dataset of user stories and corresponding UML use case diagrams. The model is designed to learn the underlying patterns and relationships between user story descriptions and their corresponding use case representations. By utilizing a deep neural network architecture, the model can effectively capture the semantic meaning of the user stories and extract the relevant information for constructing use case diagrams.

1. INTRODUCTION

1.1 ABOUT THE PROJECT

Automatic transformation of user stories into UML use case diagrams refers to the process of generating UML use case diagrams from textual descriptions of user requirements or use cases. User stories typically describe the needs, requirements, and actions of users in a software system in a natural language format. UML (Unified Modeling Language) use case diagrams are graphical representations of the interactions between users and a system, depicting the various use cases and actors involved in the system. The automatic transformation process involves analysing the textual descriptions of user stories and extracting the relevant information to create a UML use case diagram. This may involve identifying the actors, use cases, and relationships between them, and mapping them to UML use case diagram elements such as actors, use cases, and associations. Automating this process can save time and effort compared to manually creating use case diagrams, especially in large-scale software projects with many use cases. However, it is important to note that the quality of the resulting UML use case diagrams heavily depends on the accuracy and completeness of the input user stories

1.2 OBJECTIVE

Efficiency: Automating the process of transforming user stories into UML use case diagrams can save time and effort compared to manually creating them, especially in large-scale software projects with many use cases.

Consistency: Automated transformation can help ensure consistency in the representation of user requirements and use cases across different teams and stakeholders.

Accuracy: Automated transformation can reduce errors and inconsistencies that may arise from manual interpretation and representation of user stories.

Clarity: UML use case diagrams provide a clear and concise representation of the interactions between users and the system, which can help improve communication and understanding among project stakeholders.

Traceability: UML use case diagrams can help track the progress of user requirements and use cases throughout the software development lifecycle, from requirements gathering to testing and deployment.

Overall, the automatic transformation of user stories into UML use case diagrams can help streamline the software development process and improve the quality of the resulting software product.

1.3 SCOPE

The scope of automatic transformation of user stories into UML use case diagrams can vary depending on the specific needs and requirements of a software development project. However, some common scopes of this process include:

Requirement Analysis: Automatic transformation can help in analyzing the requirements gathered from users and transforming them into UML use case diagrams, which can serve as a basis for designing the system's functionality.

Design: UML use case diagrams are commonly used to depict the functional requirements of the system and its interactions with users. Automatic transformation can help in designing these use case diagrams efficiently and accurately.

Testing: UML use case diagrams can also be used to generate test cases that can be used to verify the functionality of the system. Automatic transformation can help in generating these test cases by transforming the use case diagrams into executable test scripts.

Maintenance: UML use case diagrams can help in the maintenance of the system by providing a clear and concise representation of its functionality. Automatic transformation can assist in updating these diagrams as the system evolves and new requirements arise.

Documentation: UML use case diagrams can serve as a useful documentation tool for the system's functionality. Automatic transformation can help in generating these diagrams and keeping them up-to-date as changes are made to the system

1.4 EXISTING SYSTEM

Several software tools and systems are available that offer automatic transformation of user stories into UML use case diagrams. These systems provide features for importing user stories from various formats, such as Excel or Word documents, and automatically generating UML use case diagrams from them.By using these existing systems, software development teams can streamline the process of requirements analysis, design, testing, maintenance, and documentation, thereby improving the quality of the final software product.

Existing systems for automatic transformation of user stories into UML use case diagrams offer several benefits, but they also have some limitations that should be considered. One of the main limitations is the accuracy and completeness of the transformation. While automatic transformation can reduce errors and inconsistencies, it may not capture the full scope of user requirements, leading to incomplete or inaccurate use case diagrams.

1.5 PROPOSED SYSTEM

The proposed system includes, Firstly, a user story parsing module is developed to extract relevant information from the user stories. This module identifies actors, actions, and system requirements described in the user stories.

Next, natural language processing (NLP) techniques are employed to analyze and understand the parsed information. Part-of-speech tagging, named entity recognition, and sentiment analysis are used to further refine the understanding of the user stories.

Based on the parsed information, the system identifies primary actors and their corresponding actions. Similar actions are grouped together to form individual use cases. Relationships between actors and use cases, such as associations and generalizations, are defined.

Once the use cases and their relationships are identified, the system generates the UML use case diagram. This can be achieved by using a diagramming library or tool that supports UML.

Advantages of the proposed system are, it saves time and effort by automating the process of creating use case diagrams. Manually translating user stories into UML use case diagrams can be a time-consuming task, especially for large and complex systems. The automated system reduces the burden on analysts and developers, allowing them to focus on other important aspects of the software development process.

Secondly, it enhances accuracy and consistency. Manual transformations of user stories to diagrams are prone to errors, inconsistencies, and misinterpretations. By leveraging natural language processing and automated parsing techniques, the proposed system minimizes the risk of human error and ensures a more accurate and consistent representation of user requirements

2. METHODOLOGY

2.1 AGILE METHODOLOGY

For my final year academic project, I implemented Agile methodology to manage the development of automatic transformation of user stories into uml use case diagram Agile methodology is a project management framework that emphasizes flexibility, collaboration, and continuous improvement. This report describes the process of implementing Agile in my project and discuss the benefits, challenges, and lessons learned. Agile methodology is a project management approach that focuses on delivering value to the customer through flexible, iterative development.Agile teams work in short cycles called sprints, with each sprint delivering a working increment of the project.

2.2 ROLES

For my final year academic project on automatic transformation of user stories into uml use case diagram, I decided to implement Agile methodology to ensure that I could work efficiently and effectively. Since I was working alone, I adapted the roles of the team members to fit my needs.

Head of Department, Prof Sreeja K, acted as my designated Scrum master, helped to facilitate meetings and ensure that I followed the Scrum framework.

Prof. Jeeja, served as my product owner, providing guidance on the project goals and priorities.

2.3 IMPLEMENTATION

To begin, I defined the product vision for my automatic transformation of user stories into uml use case diagram project, which will reduce the overall time and effort required to create use case diagrams. Then, I created a product backlog by identifying the features and functionalities required for the platform.

Next, I planned the sprints based on the time required for each task in the product backlog. For example, the first sprint focused on dataset collection and prepration, while the second sprint focused on Deep Learning Model Selection and execution. During each sprint, I worked on the tasks identified in the product backlog and held daily stand-up meetings to track progress and identify any obstacles that needed to be addressed.

At the end of each sprint, I conducted a sprint review to evaluate the progress made and identify any areas for improvement. For example,it is important to ensure that the dataset covers the diverse range of requirements and functionalities and it is important to clean the dataset for the transformation process, I identified the root cause of the issue and made necessary adjustments. I also held a sprint retrospective to reflect on the development process and identify ways to improve for the next sprint. For example, if I realized that I spent too much time on a particular task, I adjusted the sprint planning accordingly to better allocate time for the remaining tasks.

One of the key benefits of implementing Agile methodology in a solo project is that it can help you stay organized and on track. By breaking your project down into smaller, manageable sprints, you can focus on one task at a time and avoid feeling overwhelmed. Additionally, holding daily stand-up meetings can help you stay accountable and motivated, as you'll have to report on your progress and identify any obstacles that need to be addressed.

Another major benefit of implementing Agile methodology in a automatic transformation of user stories into uml use case diagram project is that it allows you to be flexible and adapt to changes in your project requirements. For example, if I realized that a particular feature was taking longer than expected to implement, I could adjust my sprint planning to better allocate time for that feature. This allowed me to remain agile and ensure that I was delivering the most value to my users.

Another important aspect of implementing Agile methodology is adapting it to fit your specific needs and constraints. For example, as a solo developer, you may need to adjust the length of your sprints or the number of tasks you tackle in each sprint. You may also need to adjust your approach to accommodate any unexpected challenges or changes in your project requirements.

Overall, implementing Agile methodology in my automatic transformation of user stories into uml use case diagram project was a valuable experience that allowed me to work efficiently and effectively. By following the Agile framework, I was able to continuously improve my development process and achieve my project goals.

2.4 PRODUCT BACKLOG

SL NO	USER STORIES	PRIORITIES	COMMENT FROM SCRUM MASTER	COMMENT FROM PRODUCT OWNER
1	Developer creates an user interface	Very High		
2	Develop a module to parse user stories and extract relevant information	Very High		
3	Implement NLP techniques to analyze and understand the parsed user story information.	Very High		
4	Apply part-of-speech tagging, named entity recognition, and sentiment analysis.	High		
5	Gather a diverse dataset of user stories and their UML use case diagrams.	Very High		
6	Develop algorithms to identify primary actors and actions	High		
7	To identify the relationships between actors and actions	High		
8	Create a use case diagram based on the relationship	Very High		
9	Develop a feature for user to save the use case diagram	High		
10	Can detect the errors in the generated use case diagram	Low		

Sl No	USER STORIES	PRIORITIES	COMMENT FROM SCRUM MASTER	COMMENT FROM PRODUCT OWNER
11	Can easily verify the accuracy of generated use case diagram	High		
12	Provide options for customizing	Low		

2.5 SPRINT BACKLOG

Sl.No.	USER STORY	NOT STARTED	IN PROGRESS	COMPLETED
1	Creating development environment			Completed
2	Downloading the required assets			Completed
3	Dataset collection			Completed
4	Learn about NLP technique			Completed
5	Data analyzing and cleansing using NLP technique			Completed
6	Data preprocessing			Completed
7	Search for algorithem			Completed
8	Design and develop the deep learning model architecture			Completed
9	Applying transformation model			Completed
10	Training and Testing the data			Completed
11	Performance Optimization			Completed
12	Creating interface			Completed
13	Connecting interface with model			Completed

2.6 FEASIBILITY STUDY

A feasibility study is a preliminary investigation or analysis conducted to evaluate the practicality, viability, and potential success of a proposed project or idea. It is typically conducted at the beginning of a project to determine if it is worthwhile, given the resources, time, and effort required to complete it. The main purpose of a feasibility study is to assess the technical, economic, social, and environmental aspects of a proposed project, and to identify any potential issues, risks, or challenges that may affect its success. This analysis includes gathering and evaluating data, determining project requirements, defining objectives, assessing potential benefits and drawbacks, analyzing the market, and estimating costs and financial returns. The results of a feasibility study help project managers and stakeholders make informed decisions about whether to proceed with a project or not, and to develop a plan for how to move forward if the project is deemed feasible.

Considerations involved in the feasibility analysis:

- Economic feasibility
- Operational feasibility
- Technical feasibility
- Schedule feasibility

2.6.1 ECONOMIC FEASIBILITY

The economic feasibility of automatic transformation of user stories into UML use case diagrams involves assessing the costs and benefits associated with developing and implementing the system. Here are some factors to consider:

1. Development Costs: Evaluate the costs associated with developing the automatic transformation system. This includes the salaries of data scientists, machine learning experts, and software developers involved in building the deep learning model and the necessary infrastructure. Consider the expenses related to data collection, labeling, and any required third-party tools or libraries.

2. Maintenance Costs: Determine the ongoing costs of maintaining the system. This includes regular updates to the deep learning model to improve its accuracy and

efficiency, infrastructure maintenance, and addressing any bug fixes or issues that may arise. Consider the cost of retaining skilled personnel to ensure the system's smooth operation over time.

3. Infrastructure Costs: Assess the infrastructure requirements for running the automatic transformation system. This may involve high-performance computing resources, storage, and networking infrastructure. Consider whether existing infrastructure can be leveraged or if additional investments are needed.

4. Training and Support Costs: Determine the cost of training and supporting users of the system. This includes providing training materials, documentation, and user support resources. Consider the resources required to assist users in effectively utilizing the transformed UML use case diagrams and addressing any questions or issues they may have.

2.6.2 OPERATIONAL FEASIBILITY

The operational feasibility of automatic transformation of user stories into UML use case diagrams using is crucial to ensure the practicality and effectiveness of the system within the operational context. It involves assessing various factors to determine if the system can be successfully implemented and integrated into existing processes.To begin, a stakeholder analysis should be conducted to understand the needs, expectations, and challenges of key stakeholders such as business analysts, developers, and clients.User acceptance is a critical aspect of operational feasibility. It is essential to evaluate whether users find the transformed diagrams accurate, clear, and valuable for their work. Training and familiarization of users play a vital role in operational feasibility. Providing comprehensive training materials, documentation, and user support resources enables users to effectively understand and utilize the system.By thoroughly assessing the operational feasibility, organizations can determine if the automatic transformation system aligns with their operational context, meets user expectations, integrates smoothly with existing workflows, and ultimately enhances the requirements analysis and software development processes.

2.6.3 TECHNICAL FEASIBILITY

The technical feasibility of automatic transformation of user stories into UML use case diagrams using deep learning techniques involves evaluating the availability of relevant data, the technical

expertise required to develop and train the deep learning model, and the computational resources required for training and deployment.

The availability of relevant data, such as labeled datasets containing user stories and corresponding UML use case diagrams, is crucial for training the deep learning model. If such data is not readily available, data collection and labeling efforts may be required, which can be time-consuming and resource-intensive. However, if the dataset is available, it can be used to train the model and improve its accuracy.

The technical expertise required to develop and train the deep learning model includes knowledge of machine learning algorithms, natural language processing techniques, and software development. Deep learning experts and data scientists with expertise in these areas are needed to design and develop an effective model.

The computational resources required for training and deployment of the model depend on the size and complexity of the dataset and the model architecture. A powerful GPU or distributed computing infrastructure may be needed to train the model efficiently. Additionally, the system should be optimized for performance and scalability, allowing it to handle large datasets and perform real-time transformations.

2.6.4 SCHEDULE FEASIBILITY

The schedule feasibility of implementing the automatic transformation of user stories into UML use case diagrams is a critical aspect to ensure the project's timely completion. It involves careful planning and assessment of various factors to determine if the project can be realistically executed within the desired timeframe. The first step is to define a clear and realistic project timeline. Assessing resource availability is crucial for schedule feasibility. This includes evaluating the availability of skilled personnel, data, and infrastructure required for the project. Risk management is essential for schedule feasibility. Identifying potential risks, such as data availability issues, technical challenges, or unforeseen obstacles, allows for proactive mitigation strategies.Sufficient time should be allocated for testing and validation of the automatic transformation system. Thoroughly evaluating the system's performance, accuracy, and reliability is essential to ensure it meets the desired quality standards. By considering these factors and incorporating them into the project plan, project managers can assess the schedule feasibility of implementing the automatic transformation of user stories into UML use case diagrams using deep learning. This ensures that the project progresses smoothly, tasks are completed within the desired timeframe, and the final deliverables are delivered on schedule.

3.REQUIREMENT SPECIFICATION

3.1 HARDWARE REQUIREMENTS

RAM	:	8 GB
Processor	:	12th Gen Intel(R) Core(TM) i7-12650H 2.30 GHz
Storage	:	250 GB and above

3.2 SOFTWARE REQUIREMENTS

Frontend	:	Streamlit
Backend	:	Python 3.9
Environment	:	Anaconda
Operating System	:	Windows 11
Browser	:	Mozilla Firefox

STREAMLIT

Streamlit is a popular Python library used for building interactive and customizable web applications for data science and machine learning projects. It provides a simple and intuitive way to create user interfaces that allow users to interact with machine learning models and visualizations. Streamlit can be used in the frontend of the automatic transformation system to create a user-friendly interface for users to input user stories and visualize the generated UML use case diagrams.

The main advantage of using Streamlit is its simplicity. It provides an easy-to-use API that allows developers to quickly create web applications with minimal code. The declarative syntax of Streamlit makes it straightforward to define the layout and components of the user interface. Developers can easily add input fields or file upload functionality to gather user stories, and display the generated UML use case diagrams using Streamlit's interactive visualization capabilities.

Streamlit's real-time updates enhance the user experience by enabling users to see the immediate impact of their input. As users enter or modify user stories, the UML use case diagrams can

dynamically update, providing instant feedback. This interactive feature helps users understand the transformation process and make necessary adjustments to achieve the desired results.

Another advantage of Streamlit is its versatility in data visualization. It offers a range of visualization options, such as plots, tables, and interactive components, that can be used to present the transformed user stories and UML use case diagrams. Developers can leverage these capabilities to create visually appealing and informative representations of the transformed data.

Furthermore, Streamlit simplifies the deployment process. It provides a command-line interface to start the application server, allowing for easy deployment on local machines or cloud platforms. Streamlit also supports seamless integration with popular cloud services, enabling efficient hosting and sharing of the automatic transformation system.

Overall, Streamlit is an excellent choice for developing the frontend of an automatic transformation system for user stories into UML use case diagrams using deep learning. Its simplicity, real-time updates, interactive visualization capabilities, and easy deployment make it an efficient and user-friendly tool for creating engaging web applications.

PYTHON

Python is a suitable choice for the backend implementation of the automatic transformation of user stories into UML use case diagrams using deep learning. Python is widely used in the field of data science and machine learning due to its extensive libraries, ease of use, and strong community support. Here's why Python is a good fit for the backend:

Deep Learning Frameworks: Python has several powerful deep learning frameworks such as TensorFlow, PyTorch, and Keras. These frameworks provide a wide range of tools and APIs for developing and training deep learning models. Python offers compatibility with these frameworks, allowing developers to leverage their capabilities for implementing the transformation process.

Language Libraries: Python boasts a rich ecosystem of libraries and modules specifically designed for data processing, natural language processing, and machine learning tasks. Libraries like NLTK (Natural Language Toolkit) and spaCy can be utilized for preprocessing and analyzing user stories. Additionally, libraries like scikit-learn provide useful tools for feature extraction and modeling.

Scalability and Performance: Python's performance has significantly improved over the years, and Python introduces optimizations and enhancements. While Python may not match the raw speed of lower-level languages, it offers excellent scalability and performance for most deep learning tasks. By leveraging parallel computing techniques and optimizing code, Python can efficiently handle the transformation process.

Integration with Web Frameworks: Python web frameworks like Flask and Django offer robust solutions for developing backend services and APIs. These frameworks provide the necessary infrastructure for receiving user story inputs, invoking the deep learning model, and returning the generated UML use case diagrams as responses. Python compatibility with these web frameworks facilitates seamless integration and efficient backend development.

ANACONDA

Anaconda is a popular environment for data science and machine learning projects, making it a suitable choice for developing the automatic transformation system for user stories into UML use case diagrams using deep learning. Here's why Anaconda is beneficial for this project:

Virtual Environments: Anaconda allows the creation of isolated virtual environments. Virtual environments provide a clean and separate environment for the project, enabling better package management and avoiding conflicts between different projects. Developers can create a dedicated virtual environment for the automatic transformation system, ensuring that the required packages and versions are properly isolated and maintained.

Cross-Platform Compatibility: Anaconda is available for Windows, macOS, and Linux, making it a versatile choice for developers working on different operating systems. It ensures cross-platform compatibility and facilitates collaboration among team members using different platforms.

Jupyter Notebooks: Anaconda includes Jupyter Notebook, an interactive web-based environment for developing and sharing code, visualizations, and documentation. Jupyter Notebook is particularly useful for data exploration, experimentation, and sharing project findings. It enables developers to integrate code, visualizations, and explanations, making it easier to demonstrate the automatic transformation process and showcase results.

Integrated Development Environment (IDE) Support: Anaconda seamlessly integrates with popular Python IDEs such as PyCharm and Spyder. These IDEs provide advanced features like

code autocompletion, debugging, and project management, enhancing the development experience and productivity.

WINDOWS 11

Windows 11 is the latest version of the Windows operating system, released by Microsoft in 2021. It is designed to provide users with a modern and intuitive interface, advanced productivity features, and enhanced security and privacy features. One of the key features of Windows 11 is its redesigned user interface, which includes a new Start menu and Taskbar, improved window management, and new snap layouts that make it easier to organize and work with multiple windows simultaneously. The interface has also been updated with new icons, animations, and themes, providing a fresh and modern look and feel. Windows 11 also includes a range of productivity features designed to help users work more efficiently. These include virtual desktops, improved touch controls, and enhanced support for voice commands and dictation. The operating system also includes new features for gaming, such as Auto HDR, DirectStorage, and Xbox integration, making it an ideal choice for gamers. Another significant improvement in Windows 11 is its enhanced security and privacy features. The operating system includes built-in security features such as Windows Hello, which enables secure biometric authentication, and Microsoft Defender, a comprehensive security solution that protects against viruses, malware, and other threats. Windows 11 also includes new privacy settings, giving users more control over how their data is collected and used.

3.3 FUNCTIONAL REQUIREMENTS

User Story Parsing: The system should be able to parse and extract relevant information from user stories written in natural language. It should identify key actors, actions, and system functionalities described in the user story.

Natural Language Understanding: The system should employ natural language processing techniques to understand the context and semantics of user stories. It should accurately interpret the meaning and intent behind the user story, identifying the underlying requirements.

Entity Extraction: The system should be able to extract entities, objects, or system components mentioned in the user stories. It should identify nouns or noun phrases that represent actors, system functionalities, and other key elements.

Relationship Extraction: The system should identify relationships between the extracted entities. It should recognize associations, dependencies, and interactions described in the user stories, which can be represented as relationships in the UML use case diagram.

Deep Learning Model: The system should employ a deep learning model specifically trained for transforming user stories into UML use case diagrams. The model should be capable of learning patterns and mappings between user story features and corresponding use case diagram elements.

Use Case Generation: The system should generate use cases based on the extracted information from the user stories. It should create use case templates and populate them with relevant actors, actions, and relationships identified from the user stories.

Accuracy and Reliability: The system should strive to achieve a high level of accuracy and reliability in the transformation process. It should consistently generate correct and meaningful use case diagrams that accurately reflect the requirements described in the user stories

User-Friendly Interface: The system should provide a user-friendly interface for users to input user stories and view the generated use case diagrams. It should be intuitive and easy to navigate, even for non-technical users, facilitating the adoption and usage of the system.

3.4 NON-FUNCTIONAL REQUIREMENTS

- Accuracy and Precision
- Robustness and Stability
- Scalability
- Performance

4.TECHNOLOGY

4.1 OVERVIEW

The technology used in Automatic Transformation of User Stories into UML Use Case Diagram typically involves a combination of several technologies and techniques. Here are some of the key technologies commonly used in this context:

4.2 NATURAL LANGUAGE PROCESSING

Natural Language Processing (NLP) techniques play a crucial role in the Automatic Transformation of User Stories into UML Use Case Diagram using Deep Learning. Here are some key NLP techniques that are commonly utilized in this process:

Text Preprocessing: Before analyzing user stories, text preprocessing techniques are applied to clean and normalize the text. This may involve tasks such as removing punctuation, converting text to lowercase, handling abbreviations, and eliminating irrelevant information or noise.

Tokenization: Tokenization involves breaking down the text into individual tokens or words. It helps in segmenting the user stories into meaningful units, which can be further processed and analyzed.

Part-of-Speech Tagging: Part-of-speech (POS) tagging assigns grammatical tags to each word in the user stories, indicating their syntactic role. This information is useful in understanding the structure of the sentences and identifying the different parts of speech, such as nouns, verbs, adjectives, etc.

Lemmatization: is a useful NLP technique that can be employed in the Automatic Transformation of User Stories into UML Use Case Diagram using Deep Learning. Lemmatization involves reducing words to their base or canonical form, known as the lemma. By reducing words to their base form, lemmatization helps in standardizing and normalizing the text, improving the accuracy of analysis and understanding of the user stories.

spaCy: spaCy is a popular open-source NLP library that provides efficient and accurate lemmatization capabilities. It offers pre-trained models for various languages and allows easy integration into NLP pipelines.

4.3 VECTORIZATION

Vectorization is a fundamental step in the Automatic Transformation of User Stories into UML Use Case Diagram. It involves converting textual data, such as user stories, into numerical representations that can be processed and analyzed by deep learning models. Vectorization enables the use of machine learning algorithms that operate on numerical inputs. Once the user stories are vectorized, the resulting numerical representations can be fed into deep learning models as input. These models can learn the relationships between the vectorized user stories and the corresponding UML use case diagram elements, allowing for the generation of accurate and meaningful diagrams.

4.4 ALGORITHM

TRANSFORMER MODEL

The transformer model is a powerful deep learning architecture that can be employed in the Automatic Transformation of User Stories into UML Use Case Diagram. Transformers are specifically designed to capture long-range dependencies and contextual relationships in sequential data. This makes them well-suited for understanding the complex structure and semantics of user stories and generating the corresponding UML use case diagram elements.

In the context of this task, the transformer model can be utilized to learn the mapping between the input user stories and the output UML use case diagram elements. The user stories are tokenized and fed into the transformer model, which consists of an encoder-decoder architecture. The encoder part processes the input tokens and captures their contextual representations, while the decoder part generates the UML use case diagram elements based on the encoded representations.

During training, pairs of user stories and their corresponding UML use case diagram elements are used to optimize the model's parameters. This process involves minimizing a suitable loss function that measures the discrepancy between the predicted diagram elements and the ground truth. By iteratively adjusting the model's parameters, the transformer learns to accurately generate the UML use case diagram elements that align with the input user stories.

The transformer's attention mechanism is a key component that enables it to capture the dependencies and relationships between different parts of the user stories and generate coherent diagram elements. The attention mechanism allows the model to assign varying levels of importance to different tokens based on their relevance to the generation of each diagram element. This helps the transformer model effectively capture the nuances and intricate details in the user stories, resulting in more accurate and meaningful UML use case diagram generation.

By leveraging the power of the transformer model, the Automatic Transformation system can automate the process of generating UML use case diagrams from user stories. The transformer's ability to capture long-range dependencies, handle variable-length input sequences, and generate output sequences makes it a suitable choice for this task. It empowers the system to transform user stories into UML use case diagram representations, facilitating better communication and understanding of system requirements.

4.5 BERT MODEL

The BERT (Bidirectional Encoder Representations from Transformers) model is a highly effective deep learning architecture that can be applied in the Automatic Transformation of User Stories into UML Use Case Diagram using Deep Learning. BERT has the ability to capture contextualized word representations by leveraging the power of transformer-based architectures.

In this task, the BERT model can be fine-tuned to learn the mapping between user stories and UML use case diagram elements. The pre-training of BERT involves training on vast amounts of text data to learn general language representations. However, in the transformation task, BERT is fine-tuned using labeled data specific to user stories and UML use case diagram elements. The labeled data consists of pairs of user stories and their corresponding UML use case diagram elements, such as use cases, actors, and relationships.

The input to the BERT model consists of tokenized user stories, where each token represents a word or subword unit. These tokenized user stories are then formatted into BERT's input structure, which includes special tokens like [CLS] for classification and [SEP] for separation between user stories and UML use case diagram elements.

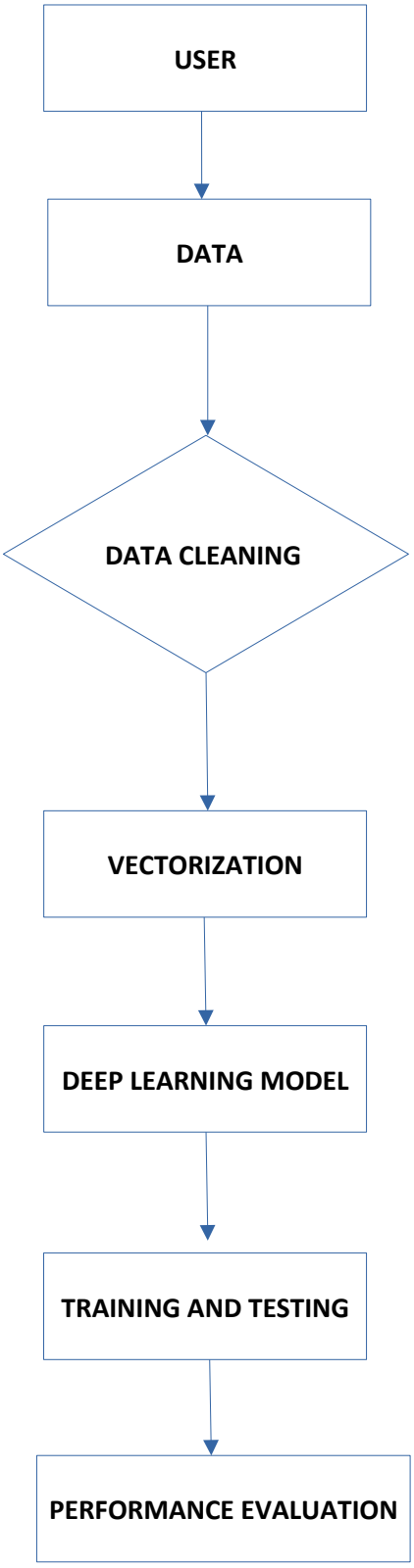
During training, the fine-tuned BERT model learns to capture the contextual information from the user stories and generate the corresponding UML use case diagram elements based on the

learned representations. BERT's contextualized word embeddings enable it to capture complex relationships and dependencies between words, resulting in more accurate and meaningful diagram generation.

The fine-tuned BERT model can be used as part of an end-to-end pipeline for automatic transformation. User stories are fed as input to the BERT model, which generates the UML use case diagram elements based on the learned representations. This approach leverages BERT's powerful language understanding capabilities to automate the process of transforming user stories into UML use case diagrams, enhancing the efficiency and accuracy of requirements analysis and system design.

3. SYSTEM DESIGN

5.1 WORK FLOW ARCHITECTURE



5.2 MAJOR FUNCTIONALITY

The Automatic Transformation of User Stories into UML Use Case Diagram encompasses several major functionalities that contribute to the efficient and accurate conversion process. This system leverages the power of deep learning techniques to automate the transformation of user stories into UML use case diagrams, providing numerous benefits to the requirements analysis and system design process..

Firstly, the system incorporates natural language processing (NLP) techniques to effectively understand and process the user stories. It utilizes advanced NLP models, such as BERT or other transformer-based architectures, to capture the semantics and contextual information embedded within the user stories. This allows for a more accurate interpretation of the user requirements and their relationships with the system functionalitie

Secondly, the system employs deep learning models specifically designed for sequence generation or sequence labeling tasks. These models are trained on labeled data, where user stories are paired with their corresponding UML use case diagram elements. Through extensive training, the models learn to generate the appropriate use cases, actors, and relationships based on the input user stories. This functionality automates the tedious and time-consuming task of manually translating user stories into UML use case diagrams.

Furthermore, the system incorporates vectorization techniques to represent the user stories in a format that is compatible with deep learning models. This involves transforming the textual data into numerical representations that can be effectively processed by the deep learning algorithms. Vectorization techniques, such as word embeddings or subword embeddings, enable the system to capture the semantic meaning and context of the user stories, enhancing the accuracy of the transformation process

Overall, the major functionality of the Automatic Transformation of User Stories into UML Use Case Diagram in its ability to automate and streamline the process of converting user stories into UML use case diagrams. By leveraging deep learning techniques, the system enhances the efficiency, accuracy, and consistency of requirements analysis and system design, ultimately contributing to the successful development and implementation of software systems.

6. SYSTEM TESTING

6.1 TESTING STRATEGIES

Input validity testing in the Automatic Transformation of User Stories into UML Use Case Diagram is crucial to ensure that the system can effectively handle various types of valid and well-formed user stories. This testing aims to verify the system's ability to process and interpret user stories accurately, capturing the essential information required for generating meaningful UML use case diagrams.

Transformation accuracy testing in the Automatic Transformation of User Stories into UML Use Case Diagram is a critical aspect of the evaluation process. It aims to assess the system's ability to generate precise and meaningful UML use case diagrams from the input user stories. The accuracy of the transformation process directly impacts the quality and reliability of the generated diagrams and their alignment with the user requirements.

Performance evaluation testing in the Automatic Transformation of User Stories into UML Use Case Diagram using Deep Learning is essential to assess the system's efficiency, speed, and resource consumption during the transformation process. It aims to ensure that the system can handle a significant volume of user stories while maintaining acceptable performance levels.

Scalability testing in the Automatic Transformation of User Stories into UML Use Case Diagram using Deep Learning involves assessing the system's ability to handle increasing workloads and larger datasets without compromising performance or functionality. This type of testing is essential to ensure that the system can scale effectively to meet the demands of real-world usage

6.2 SYSTEM MAINTENANCE

System maintenance for the Automatic Transformation of User Stories into UML Use Case Diagram using Deep Learning is essential to ensure the continued reliability, performance, and effectiveness of the system.

Model Updates and Retraining: Keep the deep learning models up to date by periodically assessing their performance and considering retraining or fine-tuning them. As new labeled data becomes available or when the system encounters new patterns or domain-specific vocabulary, updating and retraining the models can enhance the accuracy and relevance of the transformation process.

Data Quality Assurance: Regularly review and maintain the quality of the labeled training data. Ensure that the data remains relevant and representative of the user stories and UML use case diagrams encountered in the system. Perform data validation checks and handle any

inconsistencies or anomalies in the data to maintain the integrity of the training process.

Security and Privacy: Ensure the system adheres to appropriate security and privacy measures to protect sensitive user information and maintain compliance with relevant regulations. Implement data encryption, access controls, and other security measures to safeguard the user stories and UML use case diagrams processed by the system.

Error Analysis and Improvement: Analyze the errors or discrepancies encountered during the transformation process. Identify the root causes and patterns of errors and make necessary adjustments to the system, such as refining the preprocessing steps, modifying the training data, or updating the deep learning models.

6.1.1 MAINTENANCE SCHEDULE

- Regular maintenance activities will be scheduled and communicated to the team, with clear timelines and responsibilities assigned.
- Stay updated with the latest security practices and ensure that the system adheres to relevant privacy regulations.

6.1.2 IMPLEMENTATION PLAN

MAINTENANCE TASK	FREQUENCY	RESPONSIBLE PARTY
Project Scope and Requirements	Monthly	Developer
Ongoing Maintenance and Support	Daily	System Admin
Data Collection and Preparation	Quarterly	Developer
System Development and Integration	Weekly	System Admin
Testing and Quality Assurance	Daily	System Admin

Note: For this project, I was solely responsible for both development and system administration tasks.

7. CONCLUSION

In conclusion, the project of Automatic Transformation of User Stories into UML Use Case Diagram presents a valuable solution for streamlining the process of translating user stories into UML use case diagrams. By harnessing the power of deep learning models, such as BERT or transformer-based architectures, the system can automatically extract and analyze the information from user stories, accurately map them to corresponding use cases, and generate UML diagrams that capture the essential functionalities and interactions.Through the implementation of this project, organizations can benefit from increased efficiency, reduced manual effort, and improved accuracy in transforming user stories into UML diagrams. The system's ability to handle a variety of user stories, including complex sentence structures and special characters, ensures its versatility and applicability across different domains and industries.Overall, the Automatic Transformation of User Stories into UML Use Case Diagram project showcases the potential of deep learning techniques in automating and enhancing the requirements engineering process. By leveraging the power of natural language processing and machine learning, this project opens doors for improved communication, productivity, and accuracy in software development, ultimately leading to the delivery of high-quality software products that meet users' needs effectively.

8. FUTURE ENHANCEMENT

The project of Automatic Transformation of User Stories into UML Use Case Diagram has significant potential for future enhancements and advancements. Here are some potential areas for improvement:

Handling Ambiguity: Enhancing the system's ability to handle ambiguous or incomplete user stories can be a valuable improvement. This can involve incorporating techniques such as context-aware processing, dialogue modeling, or user interaction to seek clarification or additional information when faced with ambiguous user stories.

Multi-Lingual Support: Extending the system's capabilities to support multiple languages would enable its application in a more diverse range of projects and global contexts. Adapting the deep learning models and training data to handle languages other than English would be a valuable enhancement.

Support for Other Diagram Types: Expanding the system's capabilities to generate not only UML use case diagrams but also other types of diagrams, such as activity diagrams, sequence diagrams, or class diagrams, would provide a more comprehensive solution for requirements engineering.

Domain-Specific Adaptation: Customizing the deep learning models and training data to specific domains or industries can enhance the system's accuracy and relevance for particular use cases. Incorporating domain-specific knowledge and terminology can improve the quality of the generated UML use case diagrams.

Continuous Learning and Feedback Integration: Implementing mechanisms for continuous learning and feedback integration would enable the system to adapt and improve over time. This can involve leveraging user feedback, expert annotations, or active learning techniques to refine the models and enhance their performance.

Visualization and Interaction: Enhancing the visualization capabilities of the generated UML use case diagrams can provide a more interactive and user-friendly experience. This can involve

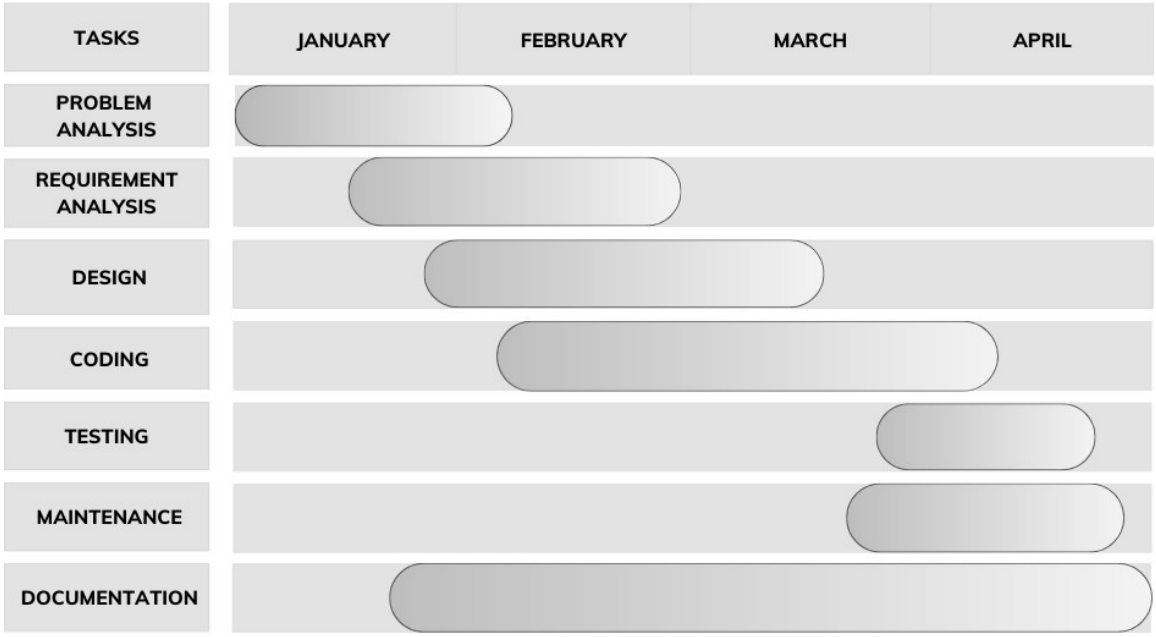
incorporating features such as zooming, panning, highlighting, or linking with other related artifacts.

Integration with Requirements Management Tools: Integrating the system with existing requirements management tools or project management systems can streamline the overall requirements engineering process. This would allow for seamless transfer of transformed user stories and UML diagrams, ensuring consistency and traceability.

Explainability and Interpretability: Providing explanations or justifications for the generated UML diagrams can enhance transparency and trust in the system. Techniques such as attention mechanisms, rule-based generation, or natural language explanations can be explored to improve the system's interpretability.

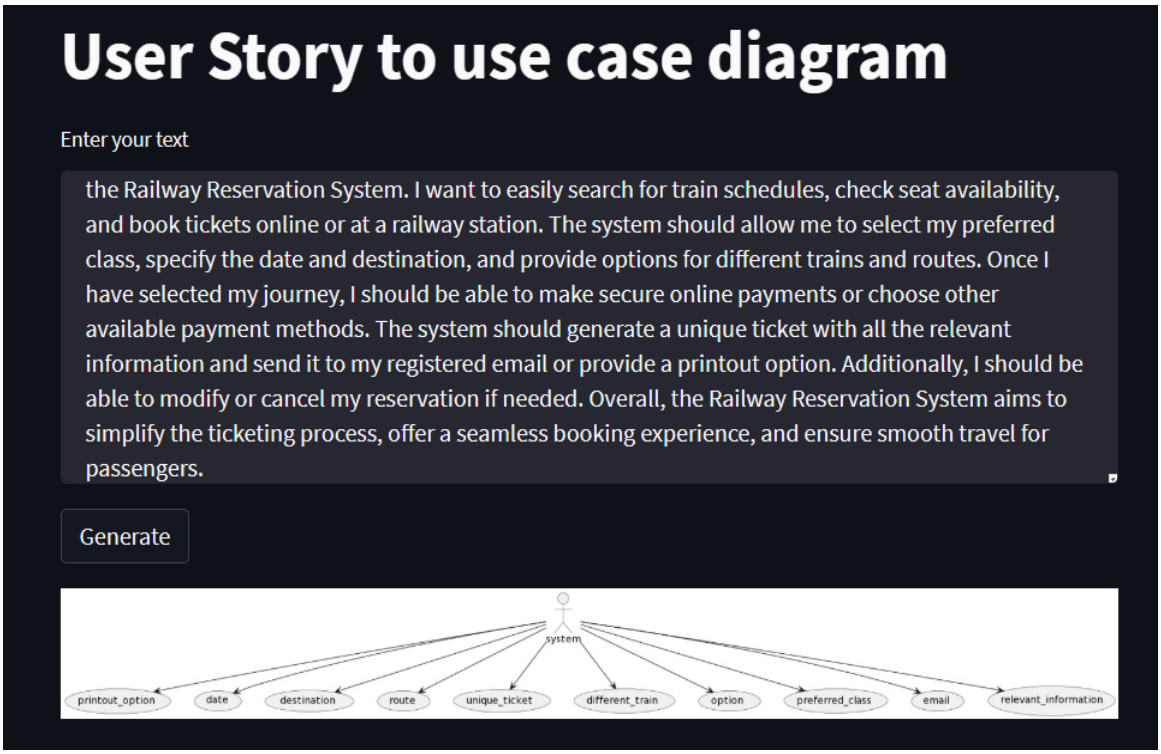
9. RESULTS

9.1 GANTT CHART



9.2 SCREENSHOTS

User Interface



Use case output

9.3 PROJECT LOG

Date : 12 / 01 / 2023

Place : Mohandas College of Engineering & Technology

Duty Executed : Search for topic

Date : 17 / 01 / 2023

Place : Mohandas College of Engineering & Technology

Duty Executed : Research about uml usecase diagram

Date : 23 / 01 / 2023

Place : Mohandas College of Engineering & Technology

Duty Executed : Prof. Sreeja K (HOD) Approved Project Topic

Date : 23 / 01 / 2023

Place : Mohandas College of Engineering & Technology

Duty Executed : Abstract Submitted

Date : 24 / 01 / 2023

Place : Mohandas College of Engineering & Technology

Duty Executed : Started learning about user stories and
uml use case diagram.

Date : 02 / 02 / 2023

Place : Mohandas College of Engineering & Technology

Duty Executed : started learning about nlp technique

Date : 09 / 02 / 2023

Place : Mohandas College of Engineering & Technology

Duty Executed : search for dataset

Date : 01 / 03 / 2023

Place : Mohandas College of Engineering & Technology

Duty Executed : Started creating documentation

Date : 02 / 03 / 2023

Place : Mohandas College of Engineering & Technology

Duty Executed : Preprocessed the dataset

Date : 03 / 03 / 2023

Place : Mohandas College of Engineering & Technology

Duty Executed : Explore and select the BERT model

Date : 08 / 03 / 2023

Place : Mohandas College of Engineering & Technology

Duty Executed : Trained the BERT model using the labeled dataset

Date : 16 / 03 / 2023

Place : Mohandas College of Engineering & Technology

Duty Executed : Error occurred in training

Date : 17 / 03 / 2023

Place : Mohandas College of Engineering & Technology

Duty Executed : Correction in dataset

Date : 21 / 03 / 2023

Place : Mohandas College of Engineering & Technology

Duty Executed : Training completed

Date : 22 / 03 / 2023

Place : Mohandas College of Engineering & Technology

Duty Executed :Started designing the system architecture

Date : 25 / 03 / 2023

Place : Mohandas College of Engineering & Technology

Duty Executed :Designed the system architecture

Date : 26 / 03 / 2023

Place : Mohandas College of Engineering & Technology

Duty Executed : Conducted comprehensive testing

Date : 01 / 04 / 2023

Place : Mohandas College of Engineering & Technology

Duty Executed : Completed testing

Date : 03 / 04 / 2023

Place : Mohandas College of Engineering & Technology

Duty Executed : Validated the accuracy of the transformed UML use case diagrams

Date : 20 / 04 / 2023

Place : Mohandas College of Engineering & Technology

Duty Executed : Monitored system performance

Date : 24 / 04 / 2023

Place : Mohandas College of Engineering & Technology

Duty Executed : Completed Documentation

10. BIBILIOGRAPHY

1. Ghasemi, A., & Pahl, C. (2020). Automatic generation of UML use case diagrams from user stories. In Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MODELS) (pp. 251-261).

2. Sahin, C., & Acar, A. E. (2020). Use case diagram generation from user stories using deep learning. In 2020 11th International Conference on Information Technology in Medicine and Education (ITME) (pp. 321-326).

3. Sinha, A., & Roy, A. (2020). Automated generation of UML use case diagrams from user stories using BERT. In 2020 2nd International Conference on Data, Engineering and Applications (IDEA) (pp. 1-6).

4. Verma, S., Bhattacharya, S., & Choudhury, T. (2019). Use case diagram generation from textual requirements using deep learning. In Proceedings of the 6th International Conference on Internet of Things: Systems, Management and Security (IOTSMS) (pp. 213-219).

11. REFERENCES

1. K. Schwaber, Scrum Development Process. In OOPSLA Business Object Design and Implementation Workshop, Eds. London: Springer, pp.117-134. (1997) . DOI: 10.1007/978-1-4471-0947-1 11.

2. L. Cao, B. Ramesh, Agile requirements engineering practices: an empirical study, IEEE Software, vol: 25, Issue: 1. (2008) . DOI:10.1109/MS.2008.

3. A. Rodrguez, I. G.-R.de Guzmn, E. Fernndez-Medina and M. Piattini, ”Semi-formal transformation of secure business processes intoanalysis class and use case models: An mda approach,” in Information and Software Technology, 52(9):945 971, (2010). DOI:10.1016/j.infsof.2010.03.015

4. D. Bell, UML basics: An introduction to the unified modeling language, IBM Developer Works. (2003).http://www.ibm.com/developerworks/rational/library/769

5. D.D. Kumar, D.D., M.A. Babar, An Automated Tool for Generating UML Models from Natural Language Requirements. Automated SoftwareEngineering, pp. 680-682. (2009). DOI: 10.1109/ASE.2009.48