Proceedings of the ASME 2024
International Design Engineering Technical Conferences and
Computers and Information in Engineering Conference
IDETC-CIE2024
August 25–28, 2024, Washington, DC

IMECE2024-142594

GST: A FRAMEWORK TO AUTOMATICALLY GENERATE SYSML DIAGRAMS FROM TEXT BASED ON DEEP LEARNING

Haomin Guo[1], Qibo Peng[2], Yusheng Liu[1,*]

[1]Zhejiang University Hangzhou, China
[2]China Astronaut Research and Training Center Beijing, China

ABSTRACT

One of the primary objectives in MBSE is to transform ambiguous and inconsistent natural language into coherent and concise SysML models. By automating this process, individuals can significantly benefit from reduced manual effort. However, recent research has encountered various challenges, including limited input, reliance on user involvement, impractical application of results, and poor generalization of methods. In this study, we introduce a novel application of deep learning to generate SysML diagrams and present a framework GST. GST uses deep learning techniques to extract SysML diagram fragments from the text that has been split into sentences, and integrates them to form a SysML model. We have constructed a labeled dataset and implemented an instance of GST to confirm its feasibility. The experimental results demonstrate that GST is capable of automatically converting unstructured natural language inputs into SysML models. The entire generation process does not necessitate user involvement, and the resulting outputs in XMI format can be directly imported into modeling software, allowing for subsequent modifications and additional modeling tasks. The implementation code for GST instance and the labeled dataset is available at GST-example-code.

Keywords: SysML diagram, Deep learning, Text-to-diagram, Automated diagrams from text, XMI

1. INTRODUCTION

The inherent fuzziness, incompleteness, and complexity of natural language pose significant challenges in the domains of knowledge retrieval, reuse, analysis of design solutions, and management of design changes[1]. Model-Based Systems Engineering (MBSE) strives to generate coherent and precise models as a replacement for traditional documentation by using Systems Modeling Language (SysML). Manual construction of SysML models is a time-intensive endeavor. Therefore, the development of a fully automated approach to transform natural language into SysML models becomes imperative. Current studies either utilize inputs that are not in natural language [1, 2], which imposes limitations, or present results using separate diagramming software [1, 3], without actual integration into modeling software. Alternatively, they employ rule-based approaches [1, 3], resulting in poor generalization, or rely on varying degrees of user involvement [3], thus lacking sufficient automation. Deep Learning (DL) can eliminates the need for manual rule formulation, minimizes human intervention, and achieves superior performance and generalization. However, exploring the application of deep learning in automating the generation of SysML diagrams remains an underexplored research topic.

Our research has designed a framework named GST (Generate SysML diagrams from Text), which facilitates the seamless transformation of unstructured natural language into SysML diagrams autonomously. GST adheres to the "split-extract-integrate" principle. It initially splits text into semantically complete sentences. An entity-relation extraction model is then employed to extract implied triples from each sentence, with each triple representing a fragment of the SysML model. Subsequently, rule-based algorithms are utilized to integrate the disparate triples into a cohesive SysML model. Finally, graphical information is incorporated into the model, resulting in the output of an XMI file containing both the model information and graphical representation. The results indicate that GST can generate initial BDD and IBD automatically, requiring no human intervention, and the generated XMI file can be directly imported into modeling software, enabling users to refine and conduct subsequent work based on the preliminary results.

2. RELATED WORKS

2.1 Automatic Generation of SysML Diagrams and UML Diagrams

Currently, there is a scarcity of research on the automatic generation of SysML diagrams. [1] proposed a method for gener-

*Corresponding author: ysliu@cad.zju.edu.cn

ating SysML diagrams, from a recursive object model, by defining transformation rules. [2] proposed a specification guideline that commences with a partial SysML model and a set of requirements to automatically generates SysML diagram. [3] utilizes traditional NLP tools and heuristic rules to directly generate SysML diagrams from unstructured natural language text. This research introduces a dedicated software to display the SysML diagrams, requiring users to adjust threshold parameters. The structural diagrams in SysML are highly similar to UML diagrams, including class diagrams and composite structure diagrams [4]. Therefore, we conducted a thorough review of previous research on generating UML diagrams. Past research on UML diagram generation primarily relied on traditional NLP techniques, augmented with heuristic rules or rule algorithms [5–13]. Moreover, research such as [6, 9, 10, 12, 13] prescribed stringent requirements for the input text, specifying adherence to particular specifications, formats, templates, or constraint conditions. [14] summarized the existing work, finding that much of this research necessitates substantial user intervention and interaction, with only a few methods achieving full automation. [15] merged natural language patterns with machine learning to achieve complete automation. [16] introduced deep learning models to discover diverse rules for transforming natural language text into models. Similarly, [17] showed that deep learning eliminates the need for manually defining an extensive set of rules, exhibits superior generalization, fully automates the entire process, and requires no additional effort from users.

The findings from the aforementioned research indicate that traditional approaches entail various limitations, including imposing high input requirements that do not involve natural language text, utilizing semi-automated methods, depending on customized rules, requiring user involvement. And most of researches tend to generate results on specific drawing software which impedes its application in actual modeling software, hindering subsequent user modifications and adjustments. In contrast, deep learning-based methods no longer depend on rules, eliminating the manual definition and maintenance of rules, thereby expediting the automation process and enhancing the method's applicability.

2.2 Joint Entity and Relation Extraction

It is evident from previous research that the generation of structural diagrams primarily involves extracting named entity nouns and their relationships. It was called Named Entity Recognition (NER), and Relation Extraction (RE) in deep learning. Recent studies have demonstrated that simultaneously addressing NER and RE can lead to improved outcomes. The traditional sequential approach of independently handling NER and RE may entail challenges such as error propagation and information redundancy [18–20]. [18] centers around deep learning and presents a systematic exploration and summary of the most recent advancements in NER and RE. The model introduced in [21] presents a pioneering labeling scheme that harmonizes entity and relationship tagging, thereby transforming the joint entity-relation extraction task into a sequence labeling approach. Various studies, including[19, 20, 22–24], have proposed distinct models for jointly extracting entity relations. These models aim to tackle the diverse challenges in this field and achieve exceptional performance.

These studies employ entity-relation extraction models to retrieve triples that encompass entities and their relationships. Applying entity-relation extraction models to SysML diagram generation associates the subject and object entities in the triples with the blocks, while the predicate relationships in the triples correspond to the relationships between these blocks. With the advancement of this field, a plethora of models are available for reference.

2.3 XMI

Current methods pose limitations in various aspects. Some methods lack an interactive interface, preventing user from updating the extracted models [25]. Others rely on dedicated software or specialized drawing software to display the models. Consequently, the generated results are confined to specific software, resulting in a lack of independence from the modeling software and limited model portability.

XMI (XML Metadata Interchange) effectively resolves the mentioned issue. It is utilized to exchange metadata information of models through XML (eXtensible Markup Language) representation. And it offers a standardized representation of metadata information in models[26]. Major CASE tools support XMI export and import [27, 28]. Storing models as XMI format files allows for their transfer between various software applications that support XMI. These studies [5–7, 29, 30] employ XMI to store and represent the models, which are subsequently imported into modeling tools for visualization. Moreover, These studies[31, 32] utilize XMI to represent models, which facilitates their subsequent work.

XMI is employed to store the resulting models, enabling their importation into any XMI-compatible software for presentation and convenient modification by subsequent users. Given its status as an industry standard, nearly all modeling tools provide support for XMI.

3. GST

GST framework divides the generation process of SysML diagrams into three key stages, including five steps in total. The initial stage named pre-processing, which comprises the processing and segmentation of input documents into sentences to fulfill the input requirements of the subsequent stage's entity relation extraction model. The second stage is dedicated to entity relation extraction, utilizing mainstream joint extraction models for entities and relations, extracting Subject-Predicate-object(SPO) triples from the sentences. Input for this stage consists of a group of sentences composed in natural language, while the output comprises a group of triples consisting of entities and their relationships. The third and final stage is post-processing. Initially, all the triples are consolidated into a hierarchical structure, which concisely and comprehensively describes the model structure information. Subsequently, the content of this hierarchical structure, along with added graphical information, is used to generate XMI. Consequently, XMI files encompass the model information and view information, serving as the ultimate output of the GST framework. Users can import the XMI files into familiar

modeling software to view the constructed model and facilitate further work. Each step's output becomes the subsequent step's input, providing a fully automated process. Users need only submit documents containing natural language texts to receive XMI files containing model information. GST framework is visually presented in Figure. 1.

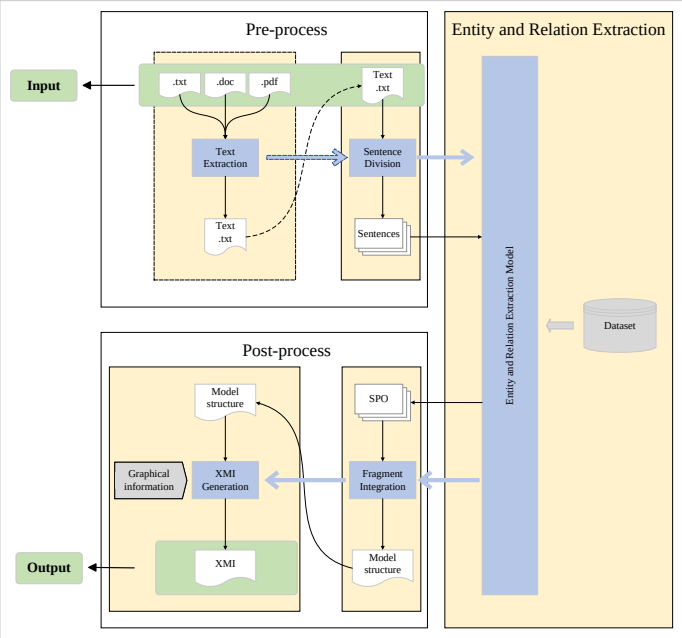


FIGURE 1: The GST framework

3.1 Text Extraction

This step aims to manage input from various document formats. If the input consists of plain text documents, this step can be omitted. Otherwise, different formats of text content should be initially extracted into a .txt document during this stage. The purpose is to convert natural language into plain text format, facilitating subsequent processing steps. However, as the focus of this study does not lie on these techniques, no specific implementation methods are provided. Users are granted the autonomy to choose and integrate their preferred methods for extracting text content from other formats within the framework. This step yields a .txt document that encompasses the extracted plain text content.

3.2 Sentence Division

The objective of this step is to partition the entire text into a set of semantically coherent sentences. The input for this stage is a .txt document generated in the preceding step. The quality of the resulting sentence segmentation directly impacts the effectiveness of the entity relation extraction model. Because the model operates at the sentence level to extract SPO. Only when the segmented sentences possess a relatively comprehensive semantic coherence, can the model genuinely achieve its intended efficacy. Sentence segmentation commonly employs rule-based methods, such as utilizing punctuation marks. Deep learning techniques can also be employed for sentence segmentation, enabling to autonomously discern patterns and leading to improved generalization and performance, which needs manual segmentation to establish the training dataset. The output of this step comprises a collection of individual sentences.

3.3 Entity and Relation Extraction

The entity relation extraction stage represents the fundamental step of this framework. This stage operates on a set of sentences generated in the preceding step. Our study employs the entity relation joint extraction model to extract SPO triples from each sentence. The subject and object components within each SPO align with blocks or part properties in BDD and IBD, while the predicate verb corresponds to relationships in BDD and IBD. Notably, these relationships are predefined within BDD and IBD and may not manifest explicitly within the original sentence. An example of extracting SPO from text is illustrated in Figure. 2. The resulting output of this step is a collection of SPO triples, encapsulating SysML fragments.

Text	Extracted SPO	Corresponding SysML elements
A computer consists of a mainframe and a monitor, with the mainframe connected to the monitor.	computer-composite association-mainframe	block- composite association-block
	computer-composite association-moniter	block- composite association-block
	mainframe-connector-moniter	part property-connector-part property

FIGURE 2: Example of SPO Extraction

3.4 Fragment Integration

In this step, the group of SPO triples containing SysML fragments, acquired in the previous step, undergoes an initial integration process. Disorganized textual descriptions of SPO impede efficient data processing within the algorithmic program of next step. Therefore, through the application of specific rule-based algorithms, these scattered SPO triples are consolidated and organized into a tree-like structure, expressed in XML format. Next step can subsequently create corresponding XMI elements systematically, following the established hierarchy within the tree structure. During the integration process, tailored rules can be implemented to handle exceptional scenarios, such as the identification and removal of duplicate elements. Hence, the output of this step entails an initial depiction of the SysML model structure, presented in the preliminary form of BDD.xml.

3.5 XMI Generation

In this step, the previously obtained BDD.xml file undergoes further refinement according to the XMI specifications of SysML. This process involves accurately mapping the information derived from BDD and IBD diagrams to their corresponding labels and attributes within the XMI representation. However, the crucial graphical visualization information required for displaying the BDD and IBD diagrams is absent. It fails to incorporate specific instructions on how these elements should be visually depicted and arranged on the diagrams[28]. Within the GST step, visualization information is manually specified. For our specific case, we have assigned a default value to these visual attributes.

Upon completion of these steps, the final XMI is generated, encompassing the extracted model information from natural language texts as well as the incorporated graphical visualization

details. By effortlessly importing this XMI file into their preferred modeling software, users can effectively present the automatically generated model and corresponding views. The entire workflow is fully automated, eliminating the necessity for any manual intervention.

4. EXAMPLE

In this section, we present specific methods or models for each step in the framework, implement a code-based instance of GST, validate its feasibility by inputting text content, and demonstrate the results using modeling software. Our example is based on Chinese text. The text content is provided in a .txt file as our input in Figure. 3. It is crucial to highlight that the main focus of our research is not on the formatting of input text. Consequently, we chose not to utilize alternative file formats like PDF, but rather opted for the most basic format, namely text files (txt). Consequently, the example we provide does not involve the initial step of text extraction.

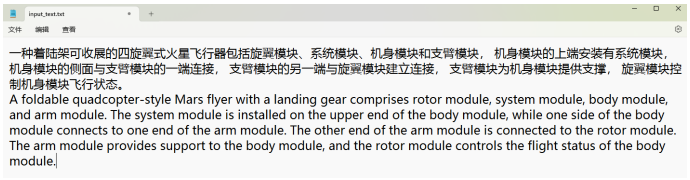


FIGURE 3: Input text

4.1 Sentence Division

We employ a straightforward rule-based method, utilizing standard Chinese punctuation marks "。！？；：，" in conjunction with line breaks to demarcate sentence boundaries. As the next step necessitates JSON format input, we incorporate a "text" field to denote the original text and store the segmented sentences in a JSON file. The specific segmentation outcomes are visually depicted in Figure. 4, serving as the input for the subsequent stage of the entity relation joint extraction model.

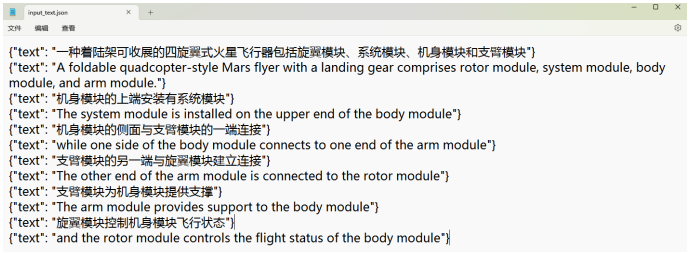


FIGURE 4: Result of pre-process

4.2 Entity and Relation Extraction

To train the entity relation joint extraction model used in this study, we manually constructed a dataset. The construction of the dataset drew inspiration from the DuIE dataset[33], while the source of the original texts comprised 34 invention patents from diverse domains obtained from the Chinese Patent website. The dataset is meticulously annotated with blocks and their corresponding composite association in BDD, as well as part properties and connector relations in IBD. For example, the raw text is:

"四旋翼式火星飞行器包括系统模块、机身模块，机身模块的上端安装有系统模块"

"A quadcopter-style Mars flyer includes a system module and a body module, with the system module installed on the upper end of the body module"

The obtained data after annotation is illustrated in Figure. 5.

FIGURE 5: The obtained data after annotation

In the example, we constructed a total of 1655 annotated data points, which were subsequently divided into training and validation sets at a ratio of 9:1. A dedicated test set was not created. Instead, a larger portion of the data was allocated for training the model.

Based on the baseline model provided in [21] and the LIC2021 Relation Extraction Competition organized by Baidu, we convert the joint entity relation task into a sequence labeling task. We assign three labels to each token based on its position within the entity span (Begin, Inside, Other)(B, I, O)[21]. Furthermore, we distinguish the B label by associating it with a specific predicate. Our focus lies on two types of predicates: composite associations in BDD and connector in IBD, as specified in a defined schema collection. Taking into account the predicates, along with the cases of head entity and tail entity, we assign a total of four B labels. We subsequently merge the I and O labels, resulting in a total of six labels for each token, as depicted in Figure. 6. For sequence labeling and extraction of the SPO triples, we employ a three-layer roBERTa pre-trained model, which provides enhanced support for Chinese compared to models like BERT [34]. As the final step, we feed the set of sentences acquired from the pre-processing procedures into the trained model, ultimately yielding a collection of SPO triplets encompassing SysML fragments.

4.3 Fragment Integration

In order to construct a comprehensive tree structure that delineates the SysML model framework (in XML format), we adhere to explicit guidelines to systematically organize the disparate SPO elements procured from the preceding step.

We have formulated the following rules to generate a comprehensive tree structure that encompasses the BDD and IBD information of the model:

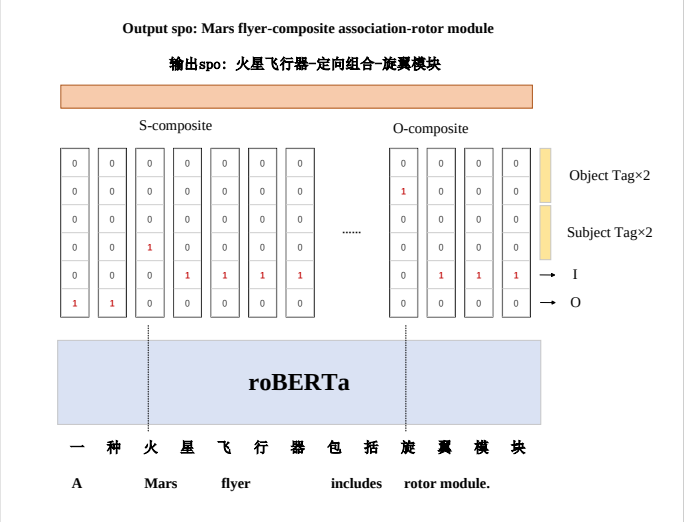


FIGURE 6: The model and sequence labeling example

a) The first entity in the text is created as the root node.

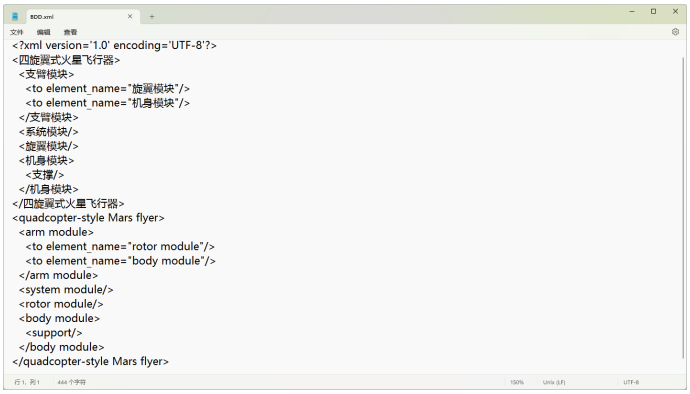
b) Before creating, check if the node is already created, and ignore the duplicated node.

c) In addition to being created as part properties, the head entity and tail entity of the connector relationship are also created as blocks and added to the root node.

d) The connector relationship is created as the "to" node under the node corresponding to the subject of that relationship, with the "element_name" being the object.

e) The connector relationships between part properties that do not belong to the same block are ignored.

Upon applying the rule-based algorithm to the series of SPO generated in the preceding step, we have obtained a comprehensive XML file that elucidates the model structure, as demonstrated in Figure. 7. This XML file has been designated as "BDD.xml" and will serve as input for the subsequent phase.



```
<?xml version='1.0' encoding='UTF-8'?>
<四旋翼式火星飞行器>
  <支臂模块>
    <to element_name="旋翼模块"/>
    <to element_name="机身模块"/>
  </支臂模块>
  <系统模块/>
  <旋翼模块/>
  <机身模块>
    <支撑/>
  </机身模块>
</四旋翼式火星飞行器>
<quadcopter-style Mars flyer>
  <arm module>
    <to element_name="rotor module"/>
    <to element_name="body module"/>
  </arm module>
  <system module/>
  <rotor module/>
  <body module>
    <support/>
  </body module>
</quadcopter-style Mars flyer>
```

FIGURE 7: Result of fragment integration

4.4 XMI Generation

The input for this step is the BDD.xml generated in the previous step. The XMI to be generated consists of two main parts: the model part and the graphical part. The model part contains the information describing the SysML model structure from the BDD.xml generated in the previous step. The graphical part is used to visualize the SysML model and present the BDD and IBD diagrams within the model. By combining the model with the graphical data, it can be imported into modeling software to generate the SysML model, BDD, and IBD, and visualize the pre-drawn BDD and IBD diagrams for the user.

Firstly we introduce the overall structure of the model part illustrated in Figure. 8. The root node of the whole model part is the node labeled as "uml:Model" located under the XMI root node. All the information of the model is encompassed in the "PackagedElement" named "System Architecture" under this root node. The system architecture package comprises three main components: BDD that describes the entire model architecture, the blocks contained in the BDD, and the composite associations among these blocks. For the entirety of the model, we have only designated a single BDD as we believe it sufficiently captures the entire model architecture. This BDD is named "Overall Architecture". Within each block, based on the composite association information of the block, the part properties owned by the block will be created. If a block possesses part property, we generate an IBD specific to this block. Subsequently, based on the connector relationships among the part properties, we establish connectors. These three elements, part properties, IBD, and connectors, exist at the same level and are associated with the same owner block.

FIGURE 8: The SysML model in XMI

The XMI components corresponding to blocks, part properties, composite associations, and connector relationships are specifically depicted in Figure. 8. Our primary concern centers around establishing the correspondence between SysML model elements and their respective XMI labels. Detail map between SysML elements and XMI tags is illustrated in Table. 1. Furthermore, the "Extension" under the "system architecture" "packagedElement" corresponds to the BDD, while the IBD corresponds to the "Extension" present within the block. The process of mapping SysML model elements to XMI labels and attributes is conducted either recursively or sequentially, according to the BDD.xml tree generated in the preceding step.

For every SysML diagram generated within the model part, an Extension node is appended beneath the XMI root node to

TABLE 1: Map between SysML elements and XMI tags

SysML element	XMI tag	tag attribute
block	packagedElement	name=block
composite association	packagedElement	name=""
part property	ownedAttribute	aggregation="composite"
connector	ownedConnector	type="Connector"

hold the graphical data necessary for rendering the diagram. This encompassing information encompasses the coordinates and dimensions of each block, as well as the coordinates, types of the connections and so on. The Extension node maintains close association with the diagram nodes in the previous model part, enabling seamless navigability.

The Extension, exhibited in Figure. 9, encapsulates the graphical data of the BDD. The first "mdElement" node assumes the role of delineating the diagram frame characteristics, thereby defining the area for diagrammatic representation. Sequentially, the "mdElement" nodes portray the graphical attributes of the blocks. Each block within the model is associated with a singular "mdElement" of this type, illustrating the specific block coordinates, dimensions, colors, and other pertinent details. Finally, the "mdElement" nodes correspond to the graphical information of composite associations, portraying them as distinct arrow symbols within the software application. We specified that all connection lines are in straight form. In the Extension that represents the graphical information of the IBD, as depicted in Figure. 10, the "mdElement" nodes portray the graphical information of the DiagramFrame, part property, and connectors, the same as the BDD.

FIGURE 9: The graphical part of BDD in XMI

Default values have been set for these graphical representations. In BDD, block coordinates are generated based on a predetermined interval, following a top-to-bottom and left-to-right hierarchy of these blocks. Blocks at the same level are aligned in rows. Conversely, the IBD employs a random function to generate block coordinates within the drawing area. This randomness stems from the fact that IBD relationships are structured as graphs rather than hierarchies, making it challenging to algorithmically arrange coordinates systematically.

In conclusion, the model part encompasses information pertaining to the composition structure and internal relationships of the model, whereas the graphical part addresses SysML diagram

FIGURE 10: The graphical part of IBD in XMI

presentation. It is worth noting that real-world scenarios may necessitate additional information, customized extensions, and related elements within the XMI file. However, these are beyond the scope of this study. The robust extension capabilities of the XMI standard, along with the flexibility of XML, allow us to avoid the inclusion of excessive and unnecessary information.

Finally, the generated XMI file will be named "import_MD.xml" and imported into the modeling software M-Design for visualization. M-Design is a robust commercial modeling software known for its strong support for both Chinese and English. It has extensive applications in practical engineering modeling and offers import and export functionality for XMI files. The resulting model, along with BDD and IBD, is displayed in M-Design, which can be found in Appendix. A.

5. DISCUSSION

During the pre-processing stage, the consideration of different text input formats was neglected. Thus, the content extraction step was not implemented in the provided examples. However, it is important to note that there is a scarcity of pure text documents that solely consist of natural language, while document formats like PDF are more prevalent. Therefore, within the GST framework, we propose the integration of a content extraction pre-processing step, envisioning the future capability of GST to process inputs across all common formats. Regarding sentence segmentation, we employed a simplistic rule that adequately handles typical semantically complete sentences. Nevertheless, this rule is not without its limitations. For instance, a sentence such as *"A car, contains four wheels"* would be segmented into two sentences based on our rule, despite its semantic coherence as a single sentence. Consequently, the implementation of sentence segmentation that considers semantics is crucial. It may be advantageous to explore deep learning methods to discover semantic relationships between sentences, offering a promising avenue for sentence segmentation.

```
"spo_list" :
[
   {
      "predicate" : "配音",
      "subject" : "王雪纯",
      "subject_type" : "娱乐人物",
      "object" : {"@value" : "晴雯", "inWork" : "红楼梦"},
      "object_type" : {"@value" : "人物", "inWork" : "影视作品"}
   }
]
```


FIGURE 11: Example of DuIE dataset

In the entity relation extraction stage, we have trained a model using an annotated dataset and applied it to the example of GST framework. However, our annotated dataset is limited in scope, focusing solely on the block and composite associations of BDD, as well as the part property and connector relationships of IBD. Users have the flexibility to augment the annotated dataset based on their specific requirements. Users can expand the dataset by incorporating additional natural language raw texts or enhance the annotated dataset by annotating attributes of BDD blocks, additional relationships between blocks, port information in IBD, and so on. By following the data annotation patterns of DuIE dataset described in [33], as illustrated in Figure. 11, we can successfully expand our dataset and generate more comprehensive and refined BDD and IBD results through deep learning techniques.

The key distinction between the example in Figure. 11 and our annotated dataset is the incorporation of additional fields within the "object" and "object_type" category to provide more comprehensive object-related information. Furthermore, our dataset allows for the inclusion of supplementary sub-fields under the "subject," "predicate," and "object" fields, enabling the description of additional details pertaining to the entity and relationship, such as inherent properties of block and connection multiplicities. As for the model aspect, since our research primarily focuses on other aspects rather than optimizing extraction performance, we employ a basic pre-trained model for sequence labeling, aligning with our annotation scheme. Regrettably, this model only achieves an F1-score of 0.6799 on our constructed training set. However, due to the inherent flexibility of the GST approach, various components within the intermediate steps, including the model used in this particular phase, can be interchanged. Recent studies have produced numerous highly effective models for jointly extracting entity relations. Moreover, users have the option to construct models more tailored to the extraction of SysML model elements based on the characteristics of the annotated dataset.

In the post-processing stage, specifically in the fragment integration part, we have defined distinct rules for generating comprehensive BDD and IBD structures. Rule c results in a significant proliferation of first-level nodes beneath the root node. Because our aim is to generate a SysML model with a single root block, hence the adoption of this rule. However, it can inadvertently result in the creation of incorrect composite associations. According to Rule e, our research refrains from considering inter-block connections for part properties and does not generate IBD to describe such across-levels relationships. These connections pose significant challenges for the graphical representation part of the XMI, as it requires displaying the owner block of part property and even port information on block boundaries. For generating the XMI, we construct a single BDD for the entire SysML model. However, if there is a large number of blocks, this approach might lead to an excessively complex BDD. It may be more appropriate to divide it into multiple BDD. Additionally, an IBD is created for each block featuring part property. In cases where there are no connector relationships between part properties, the IBD solely consists of scattered part properties without any connecting lines. In this case, IBD are unnecessary for such block, but our rules inadvertently generate redundant IBD. Random coordinates are assigned to the blocks during the generation of graphical information for IBD. This can result in disorderly distribution and partial overlapping of blocks within the generated IBD. Straight lines are used to connect the blocks throughout the entire graphical representation. However, it is difficult to avoid line crossings in the IBD, leading to an aesthetically less pleasing layout. Readability may be compromised when there are numerous blocks and lines.

From the results displayed in the modeling software, it is visually apparent that a redundant block named "Support" was extracted, while the connector relationship between the body module and the system module was not captured. These issues can be attributed to poor performance in extracting entity relationships model. Further examination of the input and output in the entity relation extraction stage revealed the extraction of redundant and incorrect SPO triples, with some SPO triples not being extracted. Subsequent processing addressed overlapping and logically erroneous SPO triples, although there were still logically correct SPO triples included in the generation of BDD and IBD, despite the absence of such semantics in the original text. These issues can be attributed to the inadequate extraction capability and domain suitability of the model, as well as the limited annotated dataset that failed to cover a wide range of natural language sentence patterns.

In conclusion, the GST framework demonstrates a preliminary capability to generate BDD and IBD from natural language text. However, there remain several areas that necessitate further improvement.

6. CONCLUSION AND FUTURE WORK

Our research proposes a deep learning based framework GST that automatically generates BDD and IBD in the SysML model from natural language text. We implemented a instance of instance, validated its feasibility, and imported the results into modeling software to demonstrate the visualized outcomes. The results showed that GST automatically generated the majority of the expected content in BDD and IBD. GST offers high flexibility, allowing users to freely replace components. The entire GST process is fully automated and does not require manual intervention. Moreover, GST imposes no specific requirements on the input text and is independent of modeling tools, enabling direct import into modeling software for subsequent tasks. Future research can explore to incorporate external knowledge bases into the input phase of GST to enhance the input text and generate perfect BDD, IBD or other SysML diagrams.

REFERENCES

[1] Wan, Wei, Cheong, Hyunmin, Li, Wei, Zeng, Yong and Iorio, Francesco. "Automated transformation of design text ROM diagram into SysML models." *Advanced Engineering Informatics* Vol. 30 No. 3 (2016): pp. 585–603. DOI https://doi.org/10.1016/j.aei.2016.07.003. URL https://www.sciencedirect.com/science/article/pii/S1474034616302105.

[2] de Biase, Maria Stella, Marrone, Stefano and Palladino, Angelo. "Towards Automatic Model Completion: from Requirements to SysML State Machines." (2022). URL 2210.03388.

[3] Zhong, Shaohong, Scarinci, Andrea and Cicirello, Alice. "Natural Language Processing for systems engineering: Automatic generation of Systems Modelling Language diagrams." *Knowledge-Based Systems* Vol. 259 (2023): p. 110071. DOI https://doi.org/10.1016/j.knosys.2022.110071. URL https://www.sciencedirect.com/science/article/pii/S0950705122011649.

[4] Friedenthal, Sanford, Moore, Alan and Steiner, Rick. *A Practical Guide to SysML: The Systems Modeling Language*, 2nd ed. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2011).

[5] Deeptimahanti, Deva Kumar and Babar, Muhammad Ali. "An Automated Tool for Generating UML Models from Natural Language Requirements." *2009 IEEE/ACM International Conference on Automated Software Engineering*: pp. 680–682. 2009. DOI 10.1109/ASE.2009.48.

[6] Amdouni, Soumaya, Karaa, Wahiba Ben Abdessalem and Bouabid, Sondes. "Semantic annotation of requirements for automatic UML class diagram generation." (2011). URL 1107.3297.

[7] Herchi, Hatem and Abdessalem, Wahiba Ben. "From user requirements to UML class diagram." (2012). URL 1211.0713.

[8] Abdelnabi, Esra A., Maatuk, Abdelsalam M., Abdelaziz, Tawfig M. and Elakeili, Salwa M. "Generating UML Class Diagram using NLP Techniques and Heuristic Rules." *2020 20th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*: pp. 277–282. 2020. DOI 10.1109/STA50679.2020.9329301.

[9] Shweta and Sanyal, Ratna. "Impact of passive and negative sentences in automatic generation of static UML diagram using NLP." *Journal of Intelligent Fuzzy Systems* Vol. 39 No. 2 (2020): pp. 2047–2059. DOI 10.3233/JIFS-179871.

[10] Alashqar, Abdelkareem. "AUTOMATIC GENERATION OF UML DIAGRAMS FROM SCENARIO-BASED USER REQUIREMENTS." *Jordanian Journal of Computers and Information Technology* Vol. 7 (2021): p. 1. DOI 10.5455/jjcit.71-1616087318.

[11] Bashir, Nauman, Bilal, Muhammad, Liaqat, Misbah, Marjani, Mohsen, Malik, Nadia and Ali, Mohsin. "Modeling Class Diagram using NLP in Object-Oriented Designing." *2021 National Computing Colleges Conference (NCCC)*: pp. 1–6. 2021. DOI 10.1109/NCCC49330.2021.9428817.

[12] Jahan, Munima, Abad, Zahra Shakeri Hossein and Far, Behrouz. "Generating Sequence Diagram from Natural Language Requirements." *2021 IEEE 29th International Requirements Engineering Conference Workshops (REW)*: pp. 39–48. 2021. DOI 10.1109/REW53955.2021.00012.

[13] Hnatkowska, Bogumiła and Cebinka, Mateusz. "Activity Diagram Generation Based on Use-Case Textual Specification." *Computing and Informatics* Vol. 40 No. 4 (2021): pp. 772–795.

[14] Abdelnabi, Esra A., Maatuk, Abdelsalam M. and Hagal, Mohammed. "Generating UML Class Diagram from Natural Language Requirements: A Survey of Approaches and Techniques." *2021 IEEE 1st International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering MI-STA*: pp. 288–293. 2021. DOI 10.1109/MI-STA52233.2021.9464433.

[15] Yang, Song and Sahraoui, Houari. "Towards automatically extracting UML class diagrams from natural language specifications." *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*: p. 396–403. 2022. Association for Computing Machinery, New York, NY, USA. DOI 10.1145/3550356.3561592. URL https://doi.org/10.1145/3550356.3561592.

[16] Rigou, Yves and Khriss, Ismaïl. "A Deep Learning Approach to UML Class Diagrams Discovery from Textual Specifications of Software Systems." Arai, Kohei (ed.). *Intelligent Systems and Applications*: pp. 706–725. 2023. Springer International Publishing, Cham.

[17] Zhu, Rui, Li, Wenxin and Jin, Canchang. "TAG: UML Activity Diagram Deeply Supervised Generation from Business Textural Specification." *2023 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*: pp. 956–961. 2023. DOI 10.1109/SANER56733.2023.00116.

[18] Nasar, Zara, Jaffry, Syed Waqar and Malik, Muhammad Kamran. "Named Entity Recognition and Relation Extraction: State-of-the-Art." *ACM Comput. Surv.* Vol. 54 No. 1 (2021). DOI 10.1145/3445965. URL https://doi.org/10.1145/3445965.

[19] Shang, Yu-Ming, Huang, Heyan and Mao, Xianling. "OneRel: Joint Entity and Relation Extraction with One Module in One Step." *Proceedings of the AAAI Conference on Artificial Intelligence* Vol. 36 No. 10 (2022): pp. 11285–11293. DOI 10.1609/aaai.v36i10.21379. URL https://ojs.aaai.org/index.php/AAAI/article/view/21379.

[20] Wang, Jiaxin, Zhang, Lingling, Liu, Jun, Ma, Kunming, Wu, Wenjun, Zhao, Xiang, Wu, Yaqiang and Huang, Yi. "TGIN: Translation-Based Graph Inference Network for Few-Shot Relational Triplet Extraction." *IEEE Transactions on Neural Networks and Learning Systems* (2022): pp. 1–15DOI 10.1109/TNNLS.2022.3218981.

[21] Zheng, Suncong, Wang, Feng, Bao, Hongyun, Hao, Yuexing, Zhou, Peng and Xu, Bo. "Joint Extraction of Entities and Relations Based on a Novel Tagging Scheme." Barzilay, Regina and Kan, Min-Yen (eds.). *Proceedings of*

the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers): pp. 1227–1236. 2017. Association for Computational Linguistics, Vancouver, Canada. DOI 10.18653/v1/P17-1113. URL https://aclanthology.org/P17-1113.

[22] Li, Zhe, Fu, Luoyi, Wang, Xinbing, Zhang, Haisong and Zhou, Chenghu. "RFBFN: A Relation-First Blank Filling Network for Joint Relational Triple Extraction." Louvan, Samuel, Madotto, Andrea and Madureira, Brielen (eds.). *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*: pp. 10–20. 2022. Association for Computational Linguistics, Dublin, Ireland. DOI 10.18653/v1/2022.acl-srw.2. URL https://aclanthology.org/2022.acl-srw.2.

[23] Tang, Wei, Xu, Benfeng, Zhao, Yuyue, Mao, Zhendong, Liu, Yifeng, Liao, Yong and Xie, Haiyong. "UniRel: Unified Representation and Interaction for Joint Relational Triple Extraction." Goldberg, Yoav, Kozareva, Zornitsa and Zhang, Yue (eds.). *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*: pp. 7087–7099. 2022. Association for Computational Linguistics, Abu Dhabi, United Arab Emirates. DOI 10.18653/v1/2022.emnlp-main.477. URL https://aclanthology.org/2022.emnlp-main.477.

[24] Wang, Zhen, Nie, Hongyi, Zheng, Wei, Wang, Yaqing and Li, Xuelong. "A Novel Tensor Learning Model for Joint Relational Triplet Extraction." *IEEE Transactions on Cybernetics* (2023): pp. 1–12DOI 10.1109/TCYB.2023.3265851.

[25] Saini, Rijul, Mussbacher, Gunter, Guo, Jin LC and Kienzle, Jörg. "Automated, interactive, and traceable domain modelling empowered by artificial intelligence." *Software and Systems Modeling* Vol. 21 No. 3 (2022): pp. 1015–1045. DOI 10.1007/s10270-021-00942-6.

[26] Alanen, Marcus, Lundkvist, Torbjörn and Porres, Ivan. *A Mapping Language from Models to XMI [DI] Diagrams*. Citeseer.

[27] Wei, Ran, Kolovos, Dimitrios S., Garcia-Dominguez, Antonio, Barmpis, Konstantinos and Paige, Richard F. "Partial loading of XMI models." *Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems*: p. 329–339. 2016. Association for Computing Machinery, New York, NY, USA. DOI 10.1145/2976767.2976787. URL https://doi.org/10.1145/2976767.2976787.

[28] Hameed, Kashif, Bajwa, Imran Sarwar and Naeem, Muhammad Asif. "A Novel Approach for Automatic Generation of UML Class Diagrams from XMI." Chowdhry, Bhawani Shankar, Shaikh, Faisal Karim, Hussain, Dil Muhammad Akbar and Uqaili, Muhammad Aslam (eds.). *Emerging Trends and Applications in Information Communication Technologies*: pp. 164–175. 2012. Springer Berlin Heidelberg, Berlin, Heidelberg.

[29] Alami, Nermeen, Arman, Nabil and Khamayseh, Faisal. "Generating Sequence Diagrams from Arabic User Requirements using MADA+TOKAN Tool." *The International Arab Journal of Information Technology* (2019): pp. 65–72DOI 10.34028/iajit/17/1/8.

[30] Elallaoui, Meryem, Nafil, Khalid and Touahni, Raja. "Automatic generation of UML sequence diagrams from user stories in Scrum process." *2015 10th International Conference on Intelligent Systems: Theories and Applications (SITA)*: pp. 1–6. 2015. DOI 10.1109/SITA.2015.7358415.

[31] *Functional Reasoning of System Architecture in the System Modeling Language (SysML) With XML Representation*, Vol. Volume 2: 43rd Computers and Information in Engineering Conference (CIE) of *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference* (2023). DOI 10.1115/DETC2023-117193. URL https://asmedigitalcollection.asme.org/IDETC-CIE/proceedings-pdf/IDETC-CIE2023/87295/V002T02A044/7061019/v002t02a044-detc2023-117193.pdf, URL https://doi.org/10.1115/DETC2023-117193.

[32] Durai, Anand Deva, Ganesh, Mythily, Mathew, Rincy Merlin and Anguraj, Dinesh Kumar. "A novel approach with an extensive case study and experiment for automatic code generation from the XMI schema Of UML models." *The Journal of Supercomputing* Vol. 78 No. 6 (2022): pp. 7677–7699. DOI 10.1007/s11227-021-04164-x.

[33] Li, Shuangjie, He, Wei, Shi, Yabing, Jiang, Wenbin, Liang, Haijin, Jiang, Ye, Zhang, Yang, Lyu, Yajuan and Zhu, Yong. "DuIE: A Large-Scale Chinese Dataset for Information Extraction." Tang, Jie, Kan, Min-Yen, Zhao, Dongyan, Li, Sujian and Zan, Hongying (eds.). *Natural Language Processing and Chinese Computing*: pp. 791–800. 2019. Springer International Publishing, Cham.

[34] Cui, Yiming, Che, Wanxiang, Liu, Ting, Qin, Bing and Yang, Ziqing. "Pre-Training With Whole Word Masking for Chinese BERT." *IEEE/ACM Transactions on Audio, Speech, and Language Processing* Vol. 29 (2021): pp. 3504–3514. DOI 10.1109/TASLP.2021.3124365.

APPENDIX A. RESULT DISPLAYED IN M-DESIGN

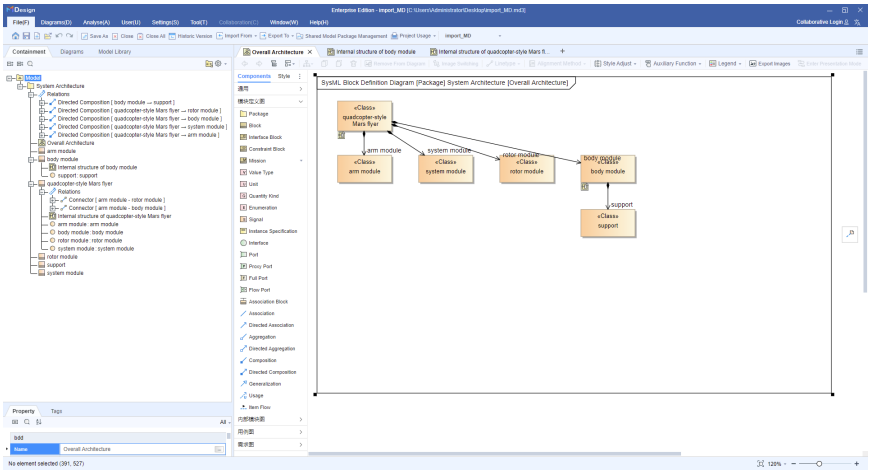


FIGURE 12: The generated BDD displayed in M-Design

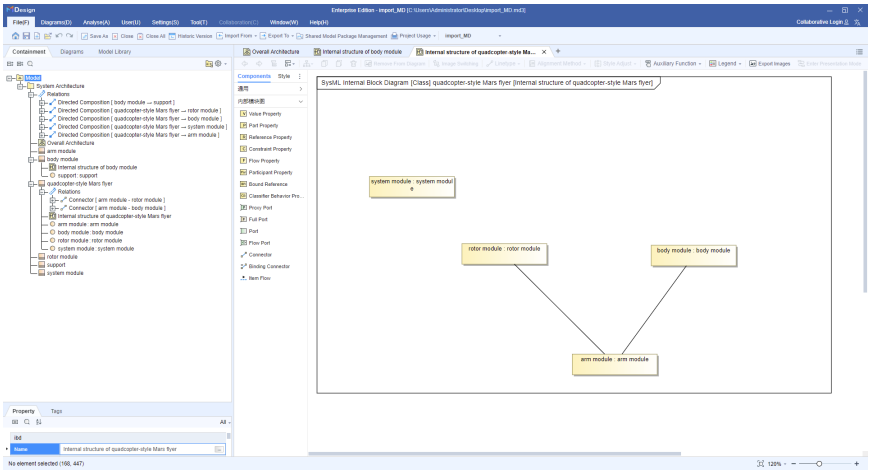


FIGURE 13: The generated IBD-1 displayed in M-Design

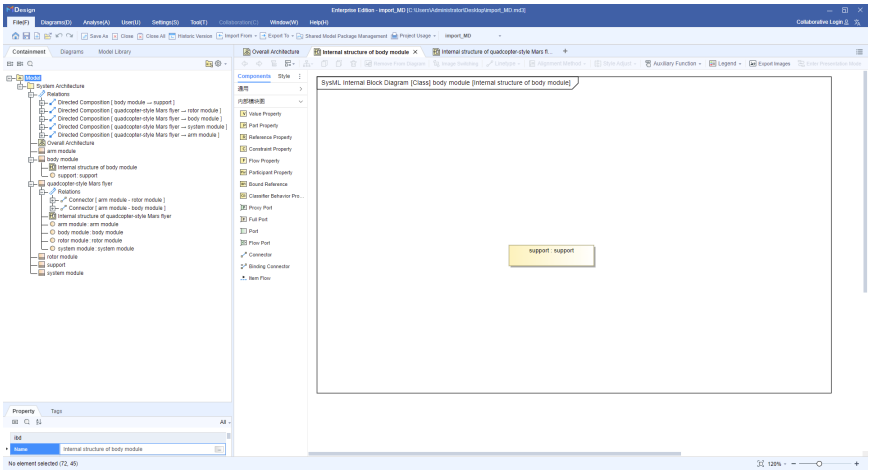


FIGURE 14: The generated IBD-2 displayed in M-Design