Original scientific paper

Event-based Approach for Analyzing and Designing System: A Case Study of Designing Curriculum System

Yanti Andriyani, Al Aminuddin, Evfi Mahdiyah, and Riki Ario Nugroho

Abstract — **Designing a system is an important step in the software development process. A use case diagram (UCD) and a class diagram (CD) are the most used diagrams in designing a system. This case study aims to explore the implementation of an event-based approach using the event table (ET) to design the Outcome-based Education Curriculum System (OBECS). In generating a UCD of OBECS, the event-based approach involves three processes: (1) identifying actors and the relationship between actors; (2) identifying use cases and the relationships of the use cases; and (3) generating UCD. Meanwhile, there are four processes in the event-based approach which can be used to generate a CD of OBECS namely: (1) identifying the classes for each event or action using the ET;(2) identifying the relationships between sources and objects; (3) identifying the class' attributes and methods; and (4) integrating all classes. Our study proposes a clear and simple concept to generate a UCD and CD in designing a system. It is expected that the result of the current study could help a software designer in modelling the system from the system requirement.**

Keywords — **Class diagram, Design System, Event, Event Table, System Requirements, Use Case Diagram.**

I. Introduction

Designing a system is an important step in software development as designing a system requires a visualization in the form of a diagram or modeling to present the requirement specifications. A use case diagram (UCD) and a class diagram (CD) are the most used diagrams in designing a system. However, developing a system diagram is a challenging process and requires deep understanding of the system requirements. A growing body of research has shown different kinds of approach which can be used to identify UCD and CDs.

Paper received July 07, 2021; revised June 09, 2022; accepted July 15, 2022. Date of publication August 05, 2022. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Miroslav Lutovac.

This work was supported by the Research and Community Service Institutions The University of Riau under the research grant number 703/UN.19.5.1.3/PT.01.03/2021.

Yanti Andriyani is a lecturer of Computer Science Major, Faculty of Mathematics and Natural Science, The University of Riau, Indonesia. (email: yanti.andriyani@lecturer.unri.ac.id).

Al Aminuddin is a lecturer of Computer Science Major, Faculty of Mathematics and Natural Science, The University of Riau, Indonesia. (email: al.aminuddin@lecturer.unri.ac.id).

Evfi Mahdiyah is a lecturer of Computer Science Major, Faculty of Mathematics and Natural Science, The University of Riau, Indonesia. (email: evfi.mahdiyah@lecturer.unri.ac.id).

Riki Ario Nugroho is a lecturer of Computer Science Major, Faculty of Mathematics and Natural Science, The University of Riau, Indonesia. (email: riki.ario@lecturer.unri.ac.id).

More recently, attention has been paid to the Natural Language Processing technique to generate a UCD [1], [2]. In her case study, Elallaoui et al [2] generated a UCD by extracting user stories using Natural Language Processing (NLP). As investigated by Osman [1], the written text of the user requirements was analyzed using the natural language processing. Some studies have attempted to derive and visualize a UCD by using a business value [3] and a business process modeling notation (BPMN) [4]. In a similar vein, some recent studies have used NLP to generate CD [5]–[8]. Samnyal and Nasiri conducted a study by collecting a corpus of user expectations [5] and user stories [7] which were then analyzed and processed using NLP to draw a CD.

An event is something that occurs [9] which is important to be captured and recorded in the process of designing a system. Its importance lies on its ability to capture specific data that are used as inputs, the logic flow and the outputs. The event specification provides insights for software development teams to identify and analyze the system requirements from external and internal systems [9].

Software practitioners implement event analysis in a system design by documenting the event analysis in a table called an Event Table (ET) [10]. There are three types of events involved in the system namely external events, temporary events and state events. An external event is an event that happens and is initiated by external users. A temporary event is an event that happens when the system reaches the limited time or state. A state event is triggered by internal system and continued to the next process as a further response. In general, an ET has five fields consisting of event, source, action, object, and destination. A source field is the source of an event involved internal or external users. An object field is the object involved in the system and affected by an action in a process. A destination field is an object goal that receives the result of the event execution [10]. While ET basically consists of five columns, it can be modified and extended to be more specific. Such an ET is called an extended ET [10] – the one that is used in the current study. The extended table specifies actions into includes 'action', extends 'action', and specializes 'action'. Our study implements the extended ET since this study specifically aims to identify system requirements.

Nowadays, Indonesian universities are required to implement the Outcome-based Education Curriculum System (OBECS). OBECS is a platform designed to integrate curriculum elements.

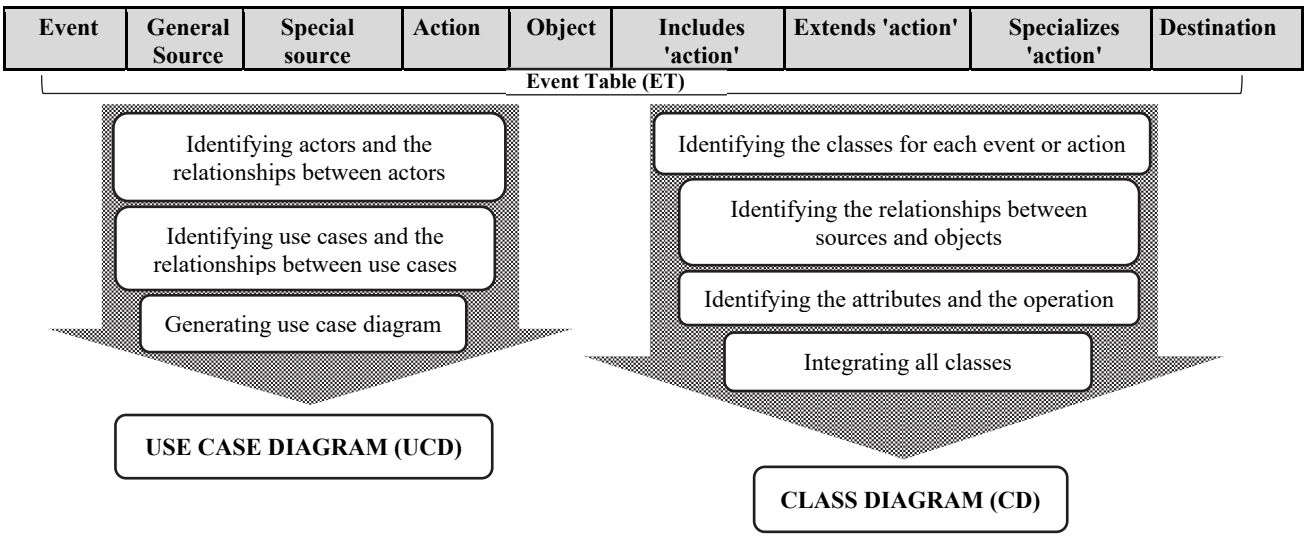

Fig. 1. Research Method.

The curriculum feature enables curriculum teams to manage fundamental curriculum data such as foundations, Program Educational Objectives (PEO), Program Learning Outcomes (PLO), Body of Knowledges (BoK), Courses, and Lecturers. It also manages matrix data such as matrix PEO-PLO, matrix PLO-Courses, matrix PLO-BoK and matrix BoK-Courses. Feature Course Plan facilitates lectures to organize course data including Course Learning Outcomes (CLO), Sub-CLO, teaching methods, activities and resources, assignment plans, and assessment plans. Designing OBECS platform is an issue under the circumstances of curriculum elements correlation. Thus, a system developer needs to understand the matrix of curriculum elements which represents the correlation between the curriculum elements and the OBECS design. The correlation between curriculum elements in the matrices form needs to be included in the platform as the matrices are the main data that are involved in the OBECS process. The matrices in the OBECS are used to generate lesson plans.

Most studies in the field of OBE curriculum have only focused on learning management systems [11], online courses [12] and the structure of the curriculum [12], [13]. In addition, the most implemented approach in generating use case and CD has only focused on NLP approach. Up to now, far too little attention has been paid to designing OBECS using an event-based approach. This study therefore aims to explore the implementation of an ET to design OBECS.

II. Data and Method Transformation

A. Data Collection

This case study was conducted to design the OBECS platform by implementing an ET approach. We collected data (i.e., system requirement) through document analysis, observations, and interviews. Document analysis was conducted to identify the structure and the curriculum element relationships. Observations were conducted to observe how lecturers and curriculum team members (CTM) correlate the curriculum elements and generate the curriculum documentation. The observations are aimed to explore the problems existed in curriculum management. A main problem in curriculum management is defined by lecturers and curriculum team solely. Interviews were conducted to capture the details of system requirements. There were two curriculum teams consisting of 10 team members from different study programs and 15 lecturers who were not included as curriculum team members.

The research method in this study was divided into two sections: Use Case Diagram and Class Diagram generations. In each section we elaborated the process of deriving each diagram.

B. Use Case Diagram (UCD)

1) Deriving UCD

Deriving a UCD in our study involved three processes:

a. Identifying actors. It is a process of identifying and defining the user's system as a source in an ET. The source in an ET has two types: a general source and a special source. A general source in an ET is related to the main actor as the user of a system in general. A special source defines a specific actor that refers to a general source field (i.e., a main actor).
b. Identifying relationships between actors. This process aims to identify the association between the identified actors from the previous process. The relationships between actors are in the form of generalization or specialization. Generalization and specialization between actors occur when an actor exists in the special source.
c. Generating UCD. This process is used to emphasize deriving the use case from the domain process which provides general scenarios of a system that captures the system or subsystem behavior. In this process, the captured behavior is defined as an action. Identifying relationships between use cases aims to identify the association between the identified use cases from the previous process. The relationships between use cases have three forms, such as includes 'action', extends 'action' and specializes 'action' in the ET. Includes 'action' is the action that is involved in the base use case. Fig. 2 depicts an example of the deriving process from an ET to use case symbols. The Extends 'action' is the action that is included in the base action, and it is not mandatory to operate the base action. Specialization 'action' is the sub action of the base action.

Event	General Source	Special source	Action	Object	Includes 'action'	Extends 'action'	Specializes 'action'	Desti-nation
Manage data course	CTM	-	Manage data course	Course	Allocate Lecturer to course	-	-	CTM

*)CTM= Curriculum Team Member

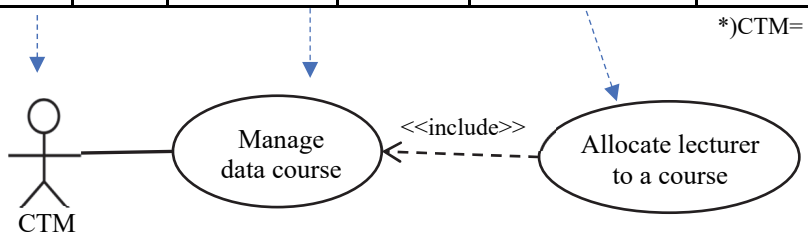

Fig. 2. An example of the deriving process from an ET to use case symbols.

Event	General Source	Special source	Action	Object	Includes 'action'	Extends 'action'	Specializes 'action'	Destination
Manage data course	CTM	-	Manage data course	Course	Allocate Lecturer to course	-	-	CTM

*)CTM= Curriculum Team Member

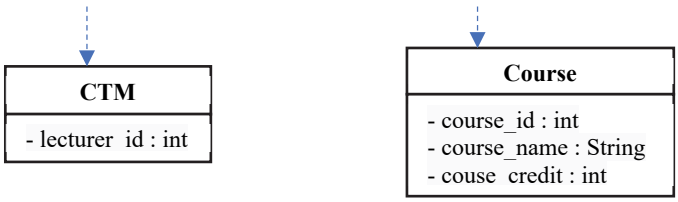

Fig. 3**.** An example of the deriving process from an ET to CD symbols.

C. Deriving Class Diagram

Deriving a CD in our study consisted of four processes. Fig. 3 depicts an example of the deriving process from an ET to use case symbols. The four processes are explained below:

a. Identifying classes for each event or action. It is a process to determine classes. Event is used to understand the context of the domain process. General source and special source fields are mapped as classes. Object field in the ET is also mapped as classes in a CD. For example, CTM in the general source is derived as CTM class and Course in the object is derived as course class (see Fig. 3).
b. Identifying the relationships between sources and objects if it exists. The relationships between classes that are derived from sources and objects can be determined by understanding the context and the existing item in the general source and special source. Understanding the context helps to identify the multiplicity between classes. General and special sources field can be used to identify generalization relationships.
c. Identifying the attributes and the operation. Identifying classes' attributes can be elaborated by adding two columns 'input message' and 'output message' on the ET. The additional fields provide a new element to specify the classes' attributes. In our study, we focus on a standard ET (i.e., without additional fields) and do not add new columns to identify the attributes. Next, action field in the ET is derived as an operation/a method in a CD. For example, manage data course can be detailed into add, update, and delete operation.
d. Integrating all classes to generate CD. This process focuses on integrating all elements into one CD. Integrating a CD involves understanding the whole context of the system that needs to be combined with the deriving process of the ET.

III. Result and Discussion

The ET approach was used to derive a UCD in our case study of Outcome-based Curriculum Platform for Indonesian Universities. Fig. 4 depicts the transformation process from the ET to UCD and CD in this paper.

A. Transformation from ET to UCD

The UCD integration process in our case study referred to Table 1 and Table 2. The identified events then mapped into the ET's fields. The first process was started by identifying system requirement and identifying events involved in the requirement. The actors involved in the events were mapped into a general source field in the ET. Based on the identification, there were two actors involved: lecturer and curriculum team members (CTM). A special source in the ET was identified by analysing the role on the actors. If the actors had a similarity in performing a particular action (i.e., activity), it can be mapped to a special source. For example, in our case study CTM had similar actions with lecturer. CTM was defined as a general actor because CTM had more actions that cannot be performed by lecturer, and the lecturer's actions can be performed by CTM.

Sequentially, the activities and data/objects involved in the events were identified and mapped into action and object fields in the ET.

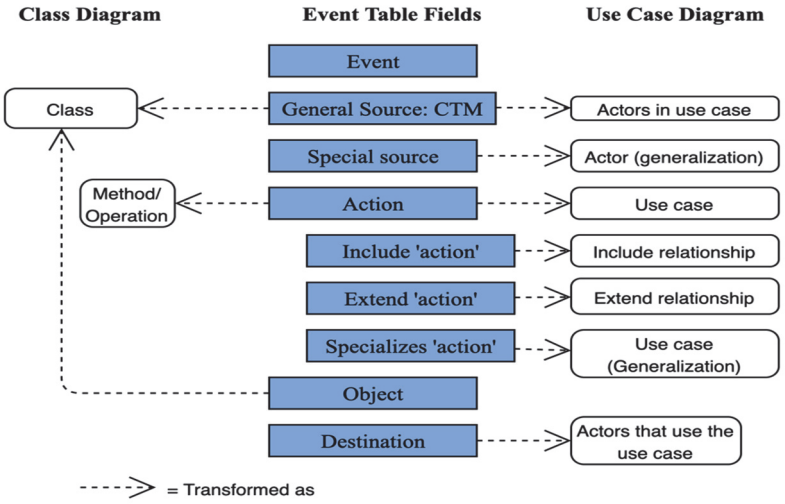

Fig. 4. The transformation processes.

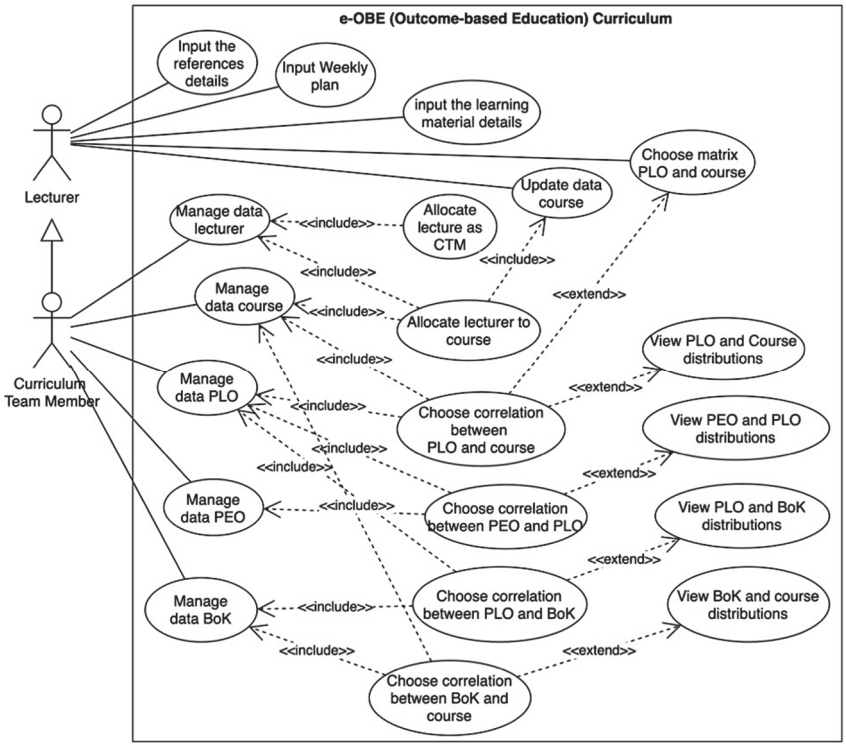

Fig. 5. UCD of OBECS derived from ET.

In addition, the subset and the extension of the action can be mapped into includes 'action' or extends 'action'. Based on the identification, there were 20 use cases, five of which were use cases related to lecturer, five use cases directly related to CTM, six were the included use cases, and the other four were extended use cases. Based on the relationship identification, the lecturer had one inclusion, and one extends relationships. CTM had 11 inclusions and 5 extends relationships. In our case study, the specializes 'action' was not available. The final process of UCD generation was realized by analyzing the relationships all the use case elements from the ET (Table 1 and Table 2). The UCD generation of OBECS was presented in Fig. 5.

B. Transformation from ET to CD

The detail CD integration process in our case study referred to Table 1 and Table 2. The transformation from ET to CD was started by focusing on general source and object fields in the ET. A general source field was transformed as a class in CD. Action field in the ET can be used as a method in class diagram. Based on the identification, there were 12 classes identified from deriving general, special source and object. The 12 classes were Lecturer, CTM, Course, PLO, BoK, PL, References, Weekly_Plan, matrix, PLO, learning_materials and course.

Sequentially, the relationships between the classes occurred between lecturer and CTM in the form of generalization. The relationships between classes were identified by understanding the system needs and the context. Integrating all classes with all relationship types in one CD was a process that focuses on integrating all elements into one CD. Integrating a CD involved understanding the whole context of the system that needs to be combined with the deriving process of the ET. The details of the CD relationships can be seen in Fig. 6.

TABLE 1 ET CURRICULUM TEAM MEMBERS (CTM: CURRICULUM TEAM MEMBERS; BOK: BODY OF KNOWLEDGE; PEO: PROGRAM EDUCATIONAL OBJECTIVES; PLO: PROGRAM LEARNING OUTCOMES; DISTRIB: DISTRIBUTION).

Event	General Source	Special source	Action	Object	Includes 'action'	Extends 'action'	Speci-alizes 'action'	Desti-nation
CTM wants to manage data course for allocating lecturer	CTM	-	Manage data course	Course	Allocate Lecturer to course	-	-	CTM
	CTM	-		BoK, Course	Choose correlation between BoK and Course	-	-	CTM
CTM wants to manage data lecturer and CTM	CTM	-	Manage data lecturer	Lecturer	Allocate Lecturer to course	-	-	CTM
	CTM	-		Lecturer, CTM	Allocate Lecturer as CTM	-	-	CTM
CTM wants to manage data PEO, PLO, BoK and Course for choosing the relationships the curriculum components	CTM	-	Manage data PEO	PEO, PLO	Choose correlation between PLO and PEO	View PLO and PEO distrib.	-	CTM
	CTM	-	Manage data PLO	PLO, PEO	Choose correlation between PLO and PEO	View PLO and PEO distrib.	-	CTM
	CTM	-		PLO, BoK	Choose correlation between PLO and BoK	-	-	CTM
	CTM	-		PLO, Course	Choose correlation between PLO and Course	-	-	CTM
	CTM	-	Manage data BoK	BoK, PLO	Choose correlation between PLO and BoK	View PLO and BoK distrib.	-	CTM
	CTM	-		BoK, Course	Choose correlation between BoK and Course	View BoK and Course distrib.	-	CTM

TABLE 2 ET LECTURER.

Event	General Source	Special source	Action	Object	Includes 'action'	Extends 'action'	Specializes 'action'	Desti-nation
Lecturer wants to update data course	Lecturer	CTM and lecturer	Update data course	Teaching Plan	Allocate lecturer to course	-	-	CTM, Lecturer
Lecturer wants to choose PLO components in a course	Lecturer	CTM and lecturer	Choose matrix PLO and Course	Teaching Plan	-	Choose PLO and Course	-	CTM, Lecturer
Lecturer wants to add learning materials	Lecturer	CTM and lecturer	Input learning materials	Materials, weekly Plan	-	-	-	CTM, Lecturer
Lecturer wants to add lesson plan	Lecturer	CTM and lecturer	Input weekly plan	Weekly plan	-	-	-	CTM, Lecturer
Lecturer wants to add references details	Lecturer	CTM and lecturer	Input the references	References, weekly plan	-	-	-	CTM, Lecturer

C. Discussion

The result of this study indicated that the event table (ET) was able to generate UCD and CD for a system design. Although we did not perform a validation or verification in terms of how accurate UCD and CD generated using ET, this study has presented a simple and clear concept to generate UCD and CD by identifying events as system requirements. Previous studies in generating UML diagrams explored various techniques, such as event-driven process chains or using Natural Language Processing Technique. Many of studies focused on technical and required specific skills (i.e., coding skill), and programming tools which are not fully understood. For example, Amjad et al [14] explored how to generate UML diagram using event-driven approach. The event-driven approach focused on elaborating each element into atomic event and transform it into the XML format. Another example was the research in generating UML diagrams which required the code generator and java code [6], [15]. Such a study was similar to the studies using NLP to generate UCD and CD which focused on transforming the semantic meaning to the symbols (i.e., UCD and CD) [16].

IV. CONCLUSION

Requirement analysis is a crucial process in designing a system. It is a challenge for software teams or system analysts to comprehend system specifications. At the same time, most software teams face difficulties and need more time in understanding and summarizing the system requirements. This paper has presented a simple approach in generating UCD and CD for designing a system using the ET. In contrast to previous approaches which generate visual models (i.e., UML diagrams) using a technical approach, we generate UCD and CD by analyzing the events involved in the system design and transforming the fields from the ET to UCD and CD symbols into one model.

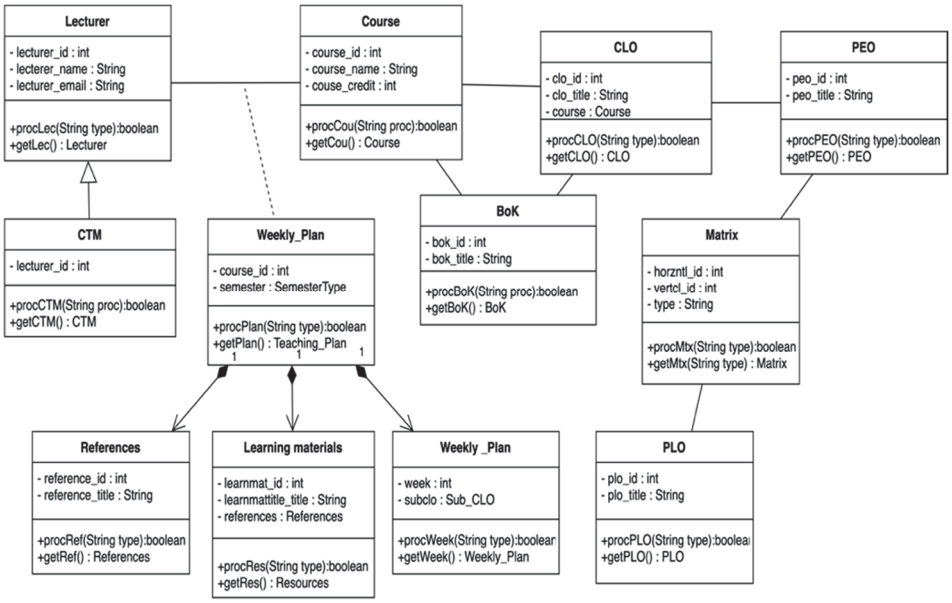

Fig. 6. CD of OBECS derived from ET.

Three processes are involved in generating a UCD using the ET, such as identifying actors and the relationship between actors, identifying use cases and the relationships between use cases, and generating a UCD. There are three processes included in generating a CD using the ET: identifying the classes for each event or action, identifying the relationships between sources and objects, identifying the attributes and the operation, and integrating all classes. Based on our study, an ET can be used to generate a structured analysis model. Furthermore, we allow mapping the event to the event table and transform the event table to use case diagrams to (a) show the bigger picture of events involved in the system, (b) highlight behavioral and structural elements of the events from a specific element of the event table.

In future work, the exploration can be expanded to other UML diagrams, such as sequence diagrams or state machine diagrams. Further research can be conducted by developing an automatic tool that can generate ETs to use case and CDs automatically. The tools can apply to all processes in the system that supports the UML diagram.

REFERENCES

[1] M. S. Osman, N. Z. Alabwaini, T. B. Jaber, and T. Alrawashdeh, "Generate use case from the requirements written in a natural language using machine learning," in *2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT)*, 2019, pp. 748–751.

[2] E. A. Abdelnabi, A. M. Maatuk, T. M. Abdelaziz, and S. M. Elakeili, "Generating UML Class Diagram using NLP Techniques and Heuristic Rules," in *International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*, Dec. 2020, vol. 130, pp. 277–282. doi: 10.1109/STA50679.2020.9329301.

[3] N. Kharmoum, S. Retal, S. Ziti, and F. Omary, "A novel automatic transformation method from the business value model to the UML use case diagram," in *International Conference on Advanced Intelligent Systems for Sustainable Development*, 2019, pp. 38–50.

[4] N. E. Ben Ayed and H. Ben Abdallah, "From an Annotated BPMN Model to a Use Case Diagram: DESTINY Methodology," in *Conference on Complex, Intelligent, and Software Intensive Systems*, 2019, pp. 379–390.

[5] R. Sanyal and B. Ghoshal, "Automated class diagram elicitation using intermediate use case template," *IET Softw.*, vol. 15, no. 1, pp. 25–42, 2021.

[6] C. R. Narawita and K. Vidanage, "UML generator – use case and class diagram generation from text requirements," *Int. J. Adv. ICT Emerg. Reg.*, vol. 10, no. 1, p. 1, Jan. 2018, doi: 10.4038/icter.v10i1.7182.

[7] S. Nasiri, Y. Rhazali, and M. Lahmer, "Towards a generation of class diagram from user stories in agile methods," in *Advancements in Model-Driven Architecture in Software Engineering*, IGI Global, 2021, pp. 135–159.

[8] E. A. Abdelnabi, A. M. Maatuk, T. M. Abdelaziz, and S. M. Elakeili, "Generating UML Class Diagram using NLP Techniques and Heuristic Rules," in *2020 20th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*, 2020, pp. 277–282.

[9] Z. Zhaoyin, L. Yanfang, and C. Chao, "Software Requirement Analysis Research Based on Event-Driven," in *2009 International Forum on Computer Science-Technology and Applications*, 2009, vol. 1, pp. 247–250.

[10] M. I. Muhairat and R. E. Al-Qutaish, "An approach to derive the use case diagrams from an event table," 2009.

[11] S. Midraj, "Outcome-Based Education (OBE)," *TESOL Encycl. English Lang. Teach.*, pp. 1–7, 2018.

[12] N. A. Bakar and S. Rosbi, "Framework of Outcome-Based-Education (OBE) for Massive Open Online Courses (MOOCs) in Islamic Finance Education," *Int. J. Adv. Eng. Res. Sci.*, vol. 6, no. 10, 2019.

[13] T. Mubeen, S. K. Hussain, and F. Aqeel, "TALEM(The Advanced Learning and Education Management) System With OBE(Outcome-based Education)," in *2019 International Conference on Information Science and Communication Technology (ICISCT)*, Mar. 2019, pp. 1–5. doi: 10.1109/CISCT.2019.8777424.

[14] A. Amjad, F. Azam, M. W. Anwar, W. H. Butt, M. Rashid, and A. Naeem, "UMLPACE for modeling and verification of complex business requirements in event-driven process chain (EPC)," *IEEE Access*, vol. 6, pp. 76198–76216, 2018.

[15] M. K. Shiferaw and A. K. Jena, "Code Generator for Model- Driven Software Development Using UML Models," in *Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, Mar. 2018, pp. 1671–1678. doi: 10.1109/ICECA.2018.8474690.

[16] E. V Sunitha and P. Samuel, "Automatic code generation from UML state chart diagrams," *IEEE Access*, vol. 7, pp. 8591–8608, 2019.