RESEARCH ARTICLE | JANUARY 25 2023

An investigation of class diagram builder system from the translation of BPMN and database

Farida Utami; Syafira Nurilhaq Maulidya; Bilal Hidayaturrohman; Nadlir Mubarak; Muhammad Zege Zain; Indra Kharisma Raharjana; Badrus Zaman

AIP Conf. Proc. 2554, 040006 (2023)
https://doi.org/10.1063/5.0103679

An Investigation of Class Diagram Builder System from The Translation of BPMN and Database

Farida Utami[1], Syafira Nurilhaq Maulidya[1], Bilal Hidayaturrohman[1], Nadlir Mubarak[1], Muhammad Zege Zain[1], Indra Kharisma Raharjana[1], and Badrus Zaman[1, a)]

[1]Information Systems Study Program, Faculty of Science and Technology, Universitas Airlangga, Surabaya, Indonesia

[a)] Corresponding author: badruszaman@fst.unair.ac.id

Abstract. BPMN (Business Process Model and Notation) is a standard business process model that is accepted both nationally and internationally. The class diagram is a UML diagram that describes the classes in a system and their relationship between one class and another. As part of the process innovation, this research proposes a generator system that can be time-efficient in creating complex Class Diagrams with BPMN and Database. This research identifies and analyzes every element of BPMN (XPDFL file) and Database (XML file) relevant to making a Class Diagram. While the output of the system is a class diagram that displays classes, attributes contained, methods used, and relations between classes. Evaluating the performance of the Generator System was done by comparing and assessing the class diagrams generated by the system with class diagrams designed by experts. The confusion matrix method was used to measure the performance of the Generator System. The Generator System achieved an accuracy of 97.4%, which means that this proposed system's performance is good.

INTRODUCTION

Class Diagram is a UML diagram that describes classes and their relationships in the system equipped with attributes and methods [1]. The Class diagram can represent the display status of the application with the main constituents, namely classes and their relationships [2,3]. However, designing a Class Diagram is not something easy to do. Analyst cannot design a Class Diagram effectively without knowing what to domain problem is [4]. Therefore, designing a Class Diagram would cause a loss of time when not using suitable tools. For this reason, the requirements elicitation stage is important in the success of software development [4]. The problem and solution domains are usually described in the BPMN diagram, as one of result of elicitation output. The BPMN diagram can be used as a basis for compiling a class diagram.

Rhazali et al. proposed a methodology to control the transformation of the model from Computational Independent Model (CIM) to Platform Independent Model (PIM) into a model-driven architecture (MDA) [5]. The methodology stated is creating a transformable model at the CIM level to facilitate the task of transforming to the PIM level. This method ensures the recommendation of the MDA approach by presenting business processes at the CIM level through BPMN, an OMG standard for modeling business processes. The strength of this research is the emergence of the first idea that there is a possibility in translating Generator diagrams. This paper proposed a solution to the problem of transforming from a business model (CIM level) to a design model (PIM level) [6]. The paper about the transformation of BPMN to use cases and class diagrams defined in UML proposed a method for automatic transformation of the CIM level to the PIM level by taking notice of the MDA approach. Their proposal is based on creating a good level of CIM through well-defined rules that enable the achievement of a rich model containing relevant information to facilitate the task of transforming to the PIM level [6]. This research provides an approach that allows displacement from a model that describes business operations to a model that provides software analysis and design. The result of this research is that the approach provides an efficient solution to the problem of transforming a business model represented at the CIM level to an analysis and design model modeled at the PIM level and produces a set of practical classes in the software development process.

A tool that can create a relational database system based on Object-Oriented Analysis [7] designs class diagrams by creating XML documents extracted from class diagrams based on class names, attribute names, data types of relationships between classes, multiplicity, and inheritance [7]. This study provides guidelines for working on class diagram generators, but the drawback is that there is no input method for class diagrams.

In a study related to the translation of class diagrams into relational databases, the author explains in-depth algorithm concepts in translating Class Diagrams into Relational Databases. The advantage of this research is that this algorithm can reduce errors, costs, and translation time. The drawback is that this algorithm cannot directly translate many to many relationships [8]. Although it differs from the aim of our paper to translate into class diagrams, the research provides questions for the missing method part for us to find out how to get methods for class diagrams later.

In the analysis model paper from the BPMN model to UML, the author proposes a transformation-based approach to generate use cases, system sequences, and class diagrams from business process models. This approach has accounting benefits for both the semantic and structural aspects of the business process model. However, there are no attributes that are generated from the business process for the class diagram. It will be our task to figure out how to assign attributes to be included in the class diagram [9]. Just like the research on deriving class diagrams from BPMN, the author proposes guidelines to help software designers build software models—in this case, UML class diagrams, from business process models—BPMN diagrams, provided by Business Analysts. This guide borrows the idea of object-oriented domain analysis to identify UML classes from business process models and increase knowledge of application domains such as domain-specific patterns and other types of semantics. This research can be used as a general guideline for us in developing our software. Nevertheless, there is a lack of inclusion attributes in class diagrams. The inclusion of attributes in class diagrams is our task for further research [10].

A study of the Model-Driven approach for generating UML class diagrams describes the principles of MDA and model transformations applied to the creation of UML class diagrams from a two-part model, which is presented in the form of a business process model related to the conceptual model. Model two is developed for the problem domain related to its application to driving schools, and a UML class diagram is generated using the approach offered in this study. The strength defined in this study is the transformation model within the MDA framework, where the source model in the business process model is linked to the concept model. Also, the target model is defined in the framework of the class diagram. This research still needs to be investigated further what concept models can be displayed with class transformation diagrams [11].

This research will propose process innovation through a system that can save time making Class Diagrams by using two important artifacts. The proposed system will convert two artifacts, namely BPMN (Business Process Model and Notation) and database, into a complex Class Diagram. BPMN aims to provide a notation that is easily understood by business users, like business analysts that need to make initial drafts and technical developers who have a role in implementing technology on the system [12], While the database represents entities and their relationships. The database has been considered a powerful artifact because of its ability to apply data normalization [13]. With this system, it is expected to assist the task of software developers in designing Class Diagrams for the software that is being developed. Other than increasing time efficiency, this system can reduce costs and improve quality in terms of the suitability of a given artifact to another. No previous studies have attempted to transform BPMN artifacts and a database into a Class Diagram. Some of those studies only showed Class Diagrams only as input in transforming other artifacts [7,8]. A similar study also transforms BPMN to Class Diagram, but this study has not been able to explain the types of relationship between classes, attributes, and data types through BPMN. The transformation is done using the help of the Use Case artifact. Therefore the detailed definition of each class cannot be done [5]. This research can provide innovation by using a database as input. Databases can provide attributes, data types, and relationships between entities, so almost all components of a class diagram can be well defined. Our study cannot still show multiplicity between related classes. It is caused by the database format being unable to provide the information related to multiplicity, even though the type of relationship could be identified.

The inability of the Class Diagram to show multiplicity does not make this research fail because the generated Class Diagram is still able to show all of the components other than multiplicity. In addition, there are also limitations of BPMN and Database inputs that must follow a particular pattern. To improve the result of this research, a feature to add multiplicity manually by the user can be implemented to the system. Improving the system's algorithm can also be done to allow the system to accept more variations of input.

This paper aims to provide an understanding of how the Class Diagram Generator system works and then assess the performance of this system by comparing the results of the Class Diagram generated by the system and the Class Diagram designed manually by experts.

METHOD

The generator system that we propose is a web application—generator system development utilizing Django as a full-stack framework. This study was conducted by adopting design science research, which develops products as a proof-of-concept [14]. The software development methodology used for the development of this generator system is Scrum. We choose Scrum because it is one of the popular frameworks for agile development [15]. The first stage of system development using Scrum is to create a backlog list. The second stage is a sprint, which begins with a sprint meeting to determine the workload for the upcoming sprint. The last stage is daily meetings to discuss progress, obstacles and evaluate the system. The implementation of the method in this study can be seen at the URL https://github.com/AgileRE-2021/CDG.

Dataset

The compiled dataset consists of two types of artifacts, namely BPMN and Database. BPMN and database are input files. The dataset must have certain conditions as BPMN must contain data object and data object's state. Bizagi Modeller is used as a tool to create BPMN easily. BPMN datasets must have an XPDL extension. The Database dataset must contain tables and relations between tables using foreign keys. We can use Power Designer to create databases then saved them in SQL. Converting SQL to XML, we can use PhpMyAdmin to import SQL files and then export files as XML. The total datasets used in this study are 12 files with XPDL and XML extension. The datasets used are accounting, bank, cafe, driver's license making, online shop A, boarding school, car rental, hospital, SCM, supermarket, online shop B, and transport. These datasets were built independently by taking references from websites and papers. The 12 dataset has different components and processes. Overall, the BPMN dataset collects 44 data objects and 60 data object states. At the same time, the database dataset collects 205 attributes and 29 foreign keys. Details for each dataset can be seen in Table 1.

TABLE 1. Dataset Details

No.	Dataset's Name	BPMN		Database	
		Data Object	State	Attribute	Foreign Key
1	Accounting	3	3	14	1
2	Bank	3	7	15	3
3	Café	3	5	15	2
4	Driver's License Making	3	3	18	2
5	Online Shop A	4	8	18	3
6	Boarding School	4	4	16	3
7	Car Rental	4	5	16	3
8	Hospital	4	4	27	2
9	Supply Chain Management	5	6	12	4
10	Supermarket	4	5	23	3
11	Online Shop B	4	4	19	1
12	Transport	3	6	12	2
	Total	**44**	**60**	**205**	**29**

BPMN and Database Translation Process to Class Diagram Architecture

The procedure for developing a generator system used to generate BPMN and Database artifacts into Class Diagrams can be seen in Fig.1. The procedure involves entering files as system inputs, reading files, taking file information, and translating to Class Diagrams. An explanation for each stage can be seen below.

Read Files

The first stage to use this generator system is entering a file BPMN with XPDL and Database with XML extensions. Files in PC local storage are entered by uploading them to the system. The generator system uses Django's file upload feature. Files are identified and analyzed to ensure that information required for the translation process conforms to system criteria. If errors do not match the system criteria, we will handle the errors by displaying error notifications.

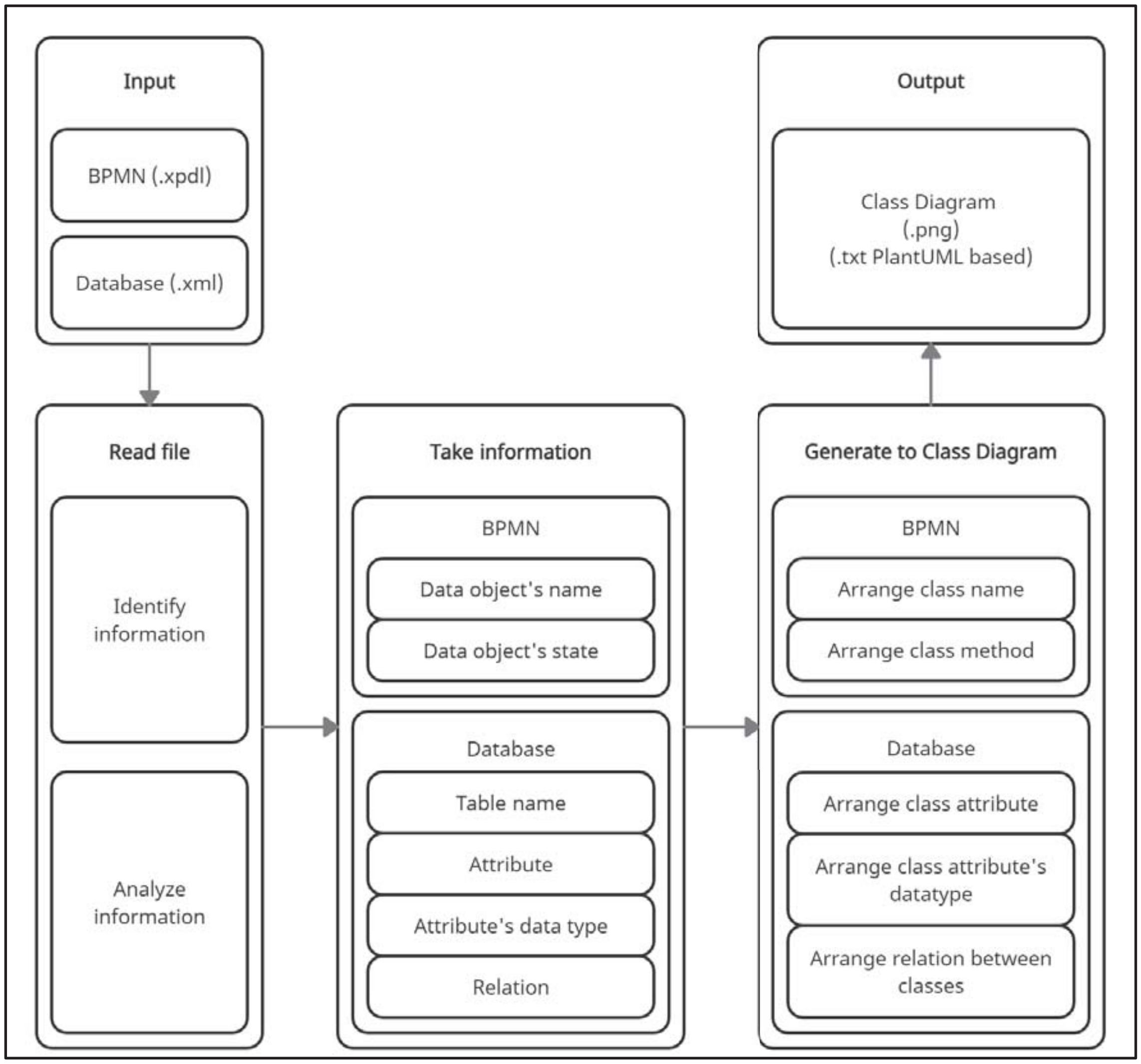

FIGURE 1. Method of BPMN and Database translation process to Class Diagram

Take File Information

After the information in the input file is safe, the next stage is taking the file information needed for the translation process to the Class Diagram. The information taken from the BPMN file is the name of the data object and the data object's state. The two types of information are used to build class names and class methods. The information taken from the Dataset file is the table name, attribute, the attribute's data type, and relation. This database will help build other Class Diagram components such as class attributes, attribute data type, and relationships between classes.

Generate to Class Diagram

The information that was successfully retrieved, then it will be processed for the preparation of Class Diagrams. Class Diagram preparation employs the PlantUML library in Python. PlantUML was chosen because it can define a diagram using a simple and intuitive language. Using the PlantUML library also aims to download the translation results in PNG and TXT.

Generator Systems Evaluation

The Class Diagram that the generator system has generated will go through an evaluation process with manual calculations. We evaluate by comparing and assessing the class diagrams generated by the system with class diagrams designed by experts. The expert who collaborated with our research has professional software development skills as a system analyst. So, it will be easy to create Class Diagram artifacts precisely according to system needs. The translation from BPMN and Database experts need to analyze each component of BPMN and Database to match the Class Diagram. Analysis of BPMN by identifying the name of the data object and the data object's state. While in the database, experts must identify attributes and their types, table names, and relationships between tables.

We calculated precision, recall, and F-Measure values for each BPMN dataset and database dataset. The calculation is based on the value of True Positive (TP), False Positive (FP), False Negative (FN) [16]. True Positive (TP) results from translation from BPMN and Database to Class Diagram identified by the generator system and manual conversion by experts. False Positive (FP) is the result of translation from BPMN and Database to Class Diagram that has been identified by the generator system but not identified by experts. False Negative (FN) is the result of translation from BPMN and Database to Class Diagram that not identified by the generator system but identified by experts. The equation for calculating these three values can be seen in Eq. (1), (2), and (3).

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

$$F - Measure = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{3}$$

RESULTS

Translation Process Implementation

Input Files

The generator system proposed in this research utilizes a particular file format for the data input. The file extension must be XPDL for the BPMN input process, while the extension for the Database input process must be XML. Figure 2 displays the BPMN sample of the Supermarket dataset. The dataset has two main processes named Product Order and Product Payment. Overall, this BPMN has 4 data objects and five object states. The Database sample of the Supermarket dataset can be seen in Fig.3.

Output Files

The translations resulted from the class diagram can be seen in two types, namely UML and PNG, see Fig.4. UML type of output can be accessed through the plantUML platform to visualize or change it according to user wishes. The class diagram will display classes, attributes contained, methods used, and relations between classes.

Performance Evaluation

An evaluation must be done to rate the system's performance. The variable that will be analyzed is the accuracy of the translation. Twelve dataset projects consist of BPMN and Database that will be the object and the experts' class diagram. In creating class diagrams considered ground truth, the class diagrams made are only of the association type. The class diagram created by experts will be compared to the class diagram created by the system. The four experts who will create the class diagrams have a strong level of business analysis from upstream to downstream and logical thinking. They have had joined a system development project before.

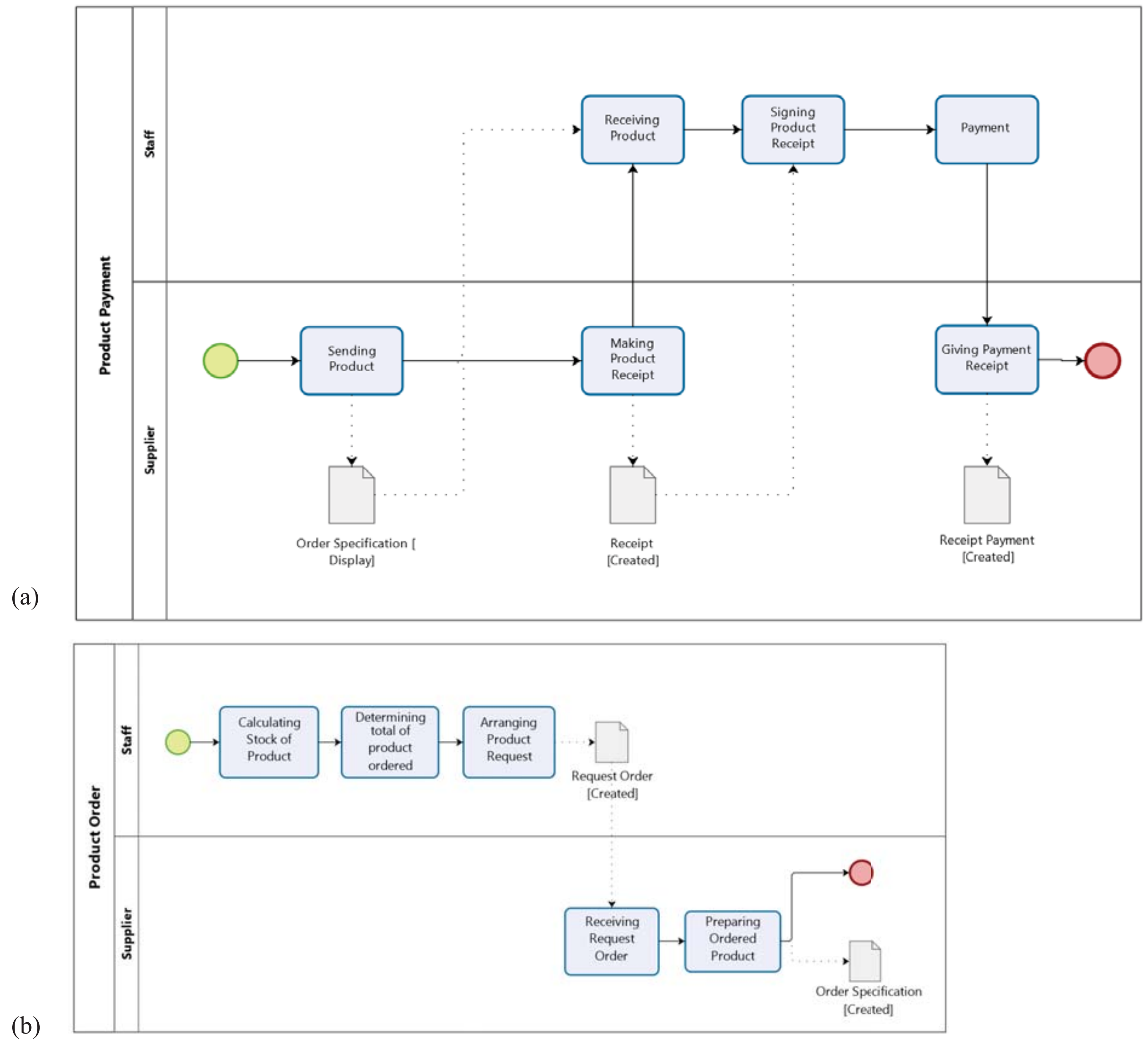

FIGURE 2. BPMN of Supermarket dataset. (a) process of product payment. (b) process of product order

For the system, datasets input has elements that are involved in constructing class diagrams. The elements are Data object and Object state, which is contained in BPMN, while attributes and relations are from a database. All of the elements in the class diagram created manually by experts were compared with each of their pairs in the class diagram resulted in the system. We used the confusion matrix method to evaluate the performance of the generator system. Table 2 compares the classification results performed by the system with the manual expert's classification results with the calculation of each Precision, Recall, and F-Measure.

True Positive value for each dataset project is determined by calculating the results of the same system elements with the elements of the results expert. The value of False Positive is calculated based on the elements contained in the system results but not on the expert results. For False Negative, we calculated the number of elements in the expert results not contained in the system results. From the comparison results in Table 2, perform calculations to measure the average accuracy of the system as a whole. The average accuracy system can be calculated using formula (3). As a result, the average F-measure from the generator system for the 12 project datasets is 97.74%.

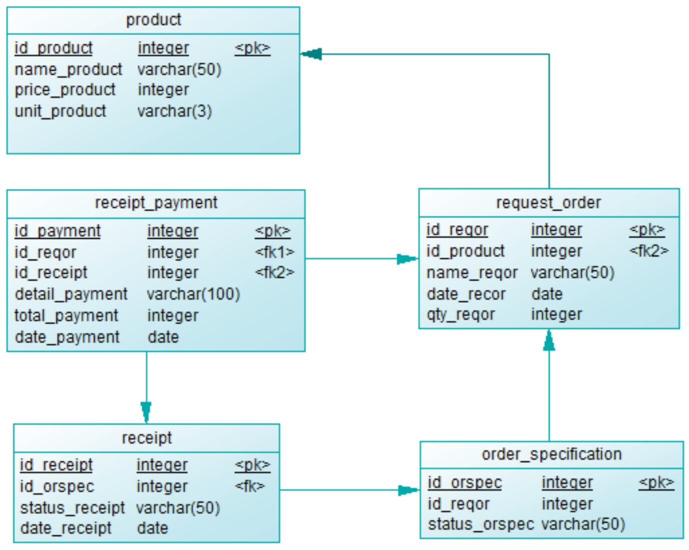

FIGURE 3. Supermarket Database Design

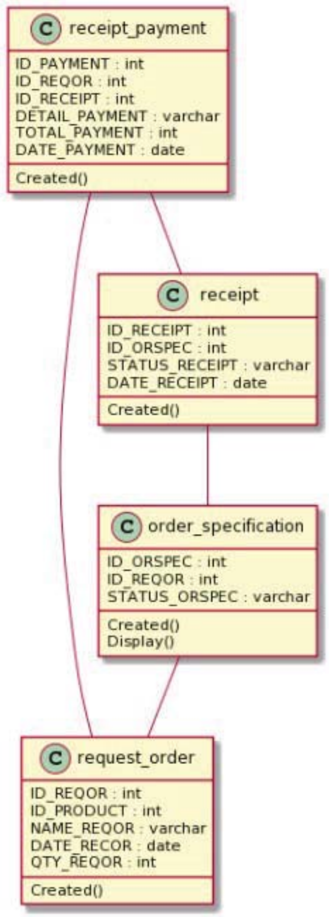

FIGURE 4. Output Class Diagram

TABLE 2. Performance assessment of BPMN to Class Diagram Generator system

Dataset Project's Name	True Positive	False Positive	False Negative	Precision	Recall	F-Measure
Accounting	19	2	2	0.9	0.9	0.9
Bank	31	0	0	1	1	1
Cafe	26	1	0	0.96	1	0.98
Online Shop A	34	2	0	0.94	1	0.96
Driver's License Making	24	4	0	0.86	1	0.92
Boarding School	30	0	0	1	1	1
Car Rental	30	1	0	0.97	1	0.98
Hospital	23	0	0	1	1	1
Supply Chain Management	31	0	0	1	1	1
Supermarket	35	0	0	1	1	1
Online Shop B	29	2	1	0.94	0.96	0.95
Transportation	25	0	0	1	1	1
Averages				**0.964**	**0.988**	**0.974**

DISCUSSION

Table 2 explained the system generator performance accuracy measured by the confusion matrix method. The average precision of datasets is 0.964, which means the accuracy between the information requested by the user and the answer given by the system is 96.4%. The average recall of datasets is 0.988 which means, the percentage of the success rate of the system in retrieving information is 98.8%. The system is almost perfect for detecting information from datasets. To evaluate the system's performance, it is better to use F-Measure because there is an uneven distribution in the cost of False Negative and False Positive [16]. The F-Measure average was calculated to find the harmonic mean of the Precision and Recall. The average of F-Measure is 0.974, which means the percentage of measure that displays the trade-off between Recall and Precision is 97.4%.

The use case diagram model interprets the previous paper's CIM to PIM transformation approach for the functional view. The state diagram model presents the dynamic view, and the class diagram model [5]. Our approach collects data objects in BPMN to make classes and transform the object states into class methods. We use the database to collect attributes and relations. The False Positive value refers to the system that displays elements that did not exist in the experts' Class Diagram. It means this system has a probability of displaying the wrong results. The False Negative value indicates that the system failed to detect one of the Online Shop B project methods and failed to detect a relation in the accounting project. Errors in data input are the main reason why the system cannot translate some elements.

In this research, we struggled to display the multiplicity in the class diagram because the database did not have information about multiplicity. We also have some restrictions requiring the data input rules to be followed, or the system cannot translate the BPMN to Class Diagram. The restrictions are about the format files of the input data, which has to be XPDL for BPMN data input, and XML for database data input because the system cannot read SQL format. The rules of naming the tables in the database must be the exact name of the data object contained in BPMN so that the system can read and combine both. It will be challenging for the Database Administrator to ensure the rules are applied well.

CONCLUSION

This study has conducted a generator that translates two artifacts, namely BPMN and Database, to the Class Diagram model. The generator systems inputs are XPDL files for BPMN and XML files for the database. While The system output is class diagrams that display classes, attributes contained, methods used, and relations between classes. The Generator System achieved an accuracy of 97.4%, which means that this proposed system's performance is good. A System Analyst expert did the performance evaluation of the system proposed. We defined the accuracy by the confusion matrix method and got 97.4% as the accuracy results.

ACKNOWLEDGEMENTS

This paper is part of the final project of the software development course (2020/2021) at the Information Systems Study Program, Faculty of Science and Technology, Universitas Airlangga.

REFERENCES

1. F. Shaari, A. B. Azuraliza, and R. H. Abdul, “A design of an assessment system for uml class diagram,” in *Proc. - 2007 Int. Conf. Comput. Sci. its Appl. ICCSA 2007,* (ICCSA, 2007), pp. 123–130.
2. G. Booch, J. Rumbaugh, and I. Jacobson, *The Unified Modelling Language Reference Manual* (Addison Wesley, 2004).
3. V. Y. Sien, Comput. Sci. Educ. **21**, 317–342 (2011).
4. I. K. Raharjana, V. Aprillya, B. Zaman, A. Justitia, and S. S. M. Fauzi, Computers **10**, 36 (2021).
5. Y. Rhazali, Y. Hadi, and A. Mouloudi, “A new methodology cim to pim transformation resulting from an analytical survey,” in *MODELSWARD 2016 - Proceedings of the 4th International Conference on Model-Driven Engineering and Software Development,* (MODELSWARD, 2016), pp. 266–273.
6. Y. Rhazali, Y. Hadi, and A. Mouloudi, Int. J. Comput. **8**, 1467–1471 (2014).
7. A. A. Fadhil, Int. J. Comput. Appl. **181**, 14–17 (2018).

8. W. Kuskorn and S. Lekcharoen, “An adaptive translation of class diagram to relational database,” in *2009 Int. Conf. Inf. Multimed. Technol. ICIMT 2009,* (ICIMT, 2009), pp. 144–148.
9. W. Khlif, N. E. Ben Ayed, and H. Ben-Abdallah, “From a BPMN model to an aligned UML analysis model,” in *ICSOFT 2018 - Proc. 13th Int. Conf. Softw. Technol,* (ICSOFT, 2019), pp. 623–631.
10. W. Rungworawut and T. Senivongse, Research Gate, 1526–1533 (2005).
11. O. Nikiforova, e-Informatica **3**, 59–72 (2009).
12. S. A. White, BPTrends, (2004).
13. C. Coronel, S. Morris, and P. Rob, *Database Systems (9th Edition)* (Cengage Learning, 2010).
14. D. A. P. Sari, A. Y. Putri, M. Hanggareni, A. Anjani, M. L. O. Siswondo, and I. K. Raharjana, “Crowdsourcing as a tool to elicit software requirements,” in *AIP Conf. Proc.* 2329, (IEEE, 2021).
15. A. Hermawan and L. P. Manik, J. Inf. Syst. Eng. Bus. Intell. **7**, (2021).
16. B. Zaman, A. Justitia, K. N. Sani, and E. Purwanti, Cybern. Inf. Technol. **20**, 82–94 (2020).