International Journal of Engineering & Technology, 7 (2.29) (2018) 776-780

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Tracing the Linkage of Several Unified Modelling Language Diagrams in Software Modelling Based on Best Practices

Dian Sa'adillah Maylawati[1]*, Muhammad Ali Ramdhani[2], Abdusy Syakur Amin[3]

[1]Department Of Informatics, Sekolah Tinggi Teknologi Garut, Indonesia
[2]Department Of Informatics, UIN Sunan Gunung Djati Bandung, Indonesia
[3]Department Of Industrial Engineering, Universitas Pasundan, Indonesia
**Corresponding Author E-Mail: Dsaadillah@Sttgarut.Ac.Id*

Abstract

Designing software aims to ensure that the systems will be developed truly reflects the needs of users. To ensure the software and the needs of users requires designing software with a good methodology, which can capture a wide range of user needs properly and accurately. One method of designing software is applied through the implementation of Unified Modeling Language (UML) concept. The objective of this article is to reveal the general system and concept of object-oriented software design with UML. The methodology used in writing articles using literature reviews as the primary source of analysis object oriented software design with UML The method that used in this article is literature review as a major source of object-oriented software design analysis with UML. In the next phase, the main topic of the article is clarified, validated, and verified by the model of the Focus Group Discussion. The respondents of this study are informatics lecturers at UIN Sunan Gunung Djati Bandung. This article describes the approach of object oriented software designing with UML, especially how to use use-case, class, and sequence diagram well with the right linkage. And also, how to trace the linkage between those diagrams thus meet all software requirements. Through a good UML method, the produced software is expected to meet the standards of good software quality, which is able to solve problems quickly and accurately.

Keywords: *Object Oriented; Software Engineering; Software Traceability; Unified Modelling Language*

1. Introduction

Software analysis and design aim to ensure that the systems that will be developed truly reflects the requirements and achieve the desire of quality [1], [2]. To ensure the software reflects requirements, software analysis and design need a good methodology which is capable of capturing a wide range of user needs are properly and accurately. Software analysis and design have to be modelled, one method of software analysis and design modelling uses object oriented viewpoint. Object oriented analysis and design method is evolved as the development trend of object-oriented programming [3]–[7]. Different with procedural or functional software development, object-oriented software development use an object as a view point [3], [4] So, modeling an object for software requirement is the first step in object-oriented method.

Information systems (IS) is a combination of information technology utilizations and human activity upon a set of agreed procedure[8], generally is used to support management and operation[9]. IS is an organized data process[10], IS has a high level of flexibilities to develop and scalable[11]. Refers to several research, the information system has an accurate data accessibility and efficient run-time [12], high accuracy [13], and to support a proper decision [14], low cost [15], extended accessibility [16], intensify user knowledge [17], increase productivity [18], provide a better data and information [19], and used as data storage [20].

As many research and literature prove that Unified Modelling Language (UML) is capable to modelling object oriented analysis and design [21]–[27]. Moreover, UML is a popular modelling language for specifying, visualizsing, constructing, and documenting software system that has been proven has a good performance [23], [24], [26], [28]. In several reserach, UML model is genereted into programming language automatically [24], [29]. Based on observations and experiences, the theory of UML is not balanced with practice. Therefore, this paper discuss the systematic, general concept, and how to uses and trace the linkage of several diagrams in UML based on practice. UML diagrams that are used among others use case diagram, class diagram, and sequence diagram. Use case, class, and sequence diagram are choosen because the result of observation the diagrams must exist on software specification and design documents. And is also found errors in practice the UML modelling moreover pay attention to the linkages between those diagrams.

In the upcoming section, we describe requirement elicitation Section 2, tracing requirements in use case diagram in Section 3. Tracing requirements in class diagram in Section 4, tracing use case and class in sequence diagram in Section 5. Next, we describe about traceability of several UML diagram in Section 6. And finally, conclusion and discussion of this research stated in Section 7.

2. Requirement Elicitations

In software development process, the first step is elicitation all of requirements from stakeholders who concerned and interested in the software. Good software is when all of the needs of all stakeholders are met. However, the fact that not all stakeholders need to be met, when not in accordance with the treaty or agreement, the

Copyright © 2016 Authors. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

defiance of the laws or policies of an organization, and the request have a negative impact and harm other stakeholders, and depend to the priority needs. Elicitation is a process of search, browse, and understand with more detail, within, and that it had little likelihood there is information left behind [1], [2]. So, the requirements elicitation itself is a process in which the search for information more deeply and thoroughly of all stakeholders who has been associated with the software [1], [2], [22]. Results of requirements elicitation is in the form of exposure to any natural language related stakeholders with the wishes and interests of software.

Next, the natural language from requirement elicitation is mapped into software requirement that can be a statement with natural sentence until mathematic symbol which is mode detail [1], [2]. Software requirement can be categorized into functional and non-functional requirements. Functional requirement is a statement containing a services or functions of software, whereas non-functional requirement is a statement that give a constraint which can be measured for services or functions of software [1], [2]. For tracing the consistency of software modelling with UML easily, the requirements have been coded. Table 1 [30], is described the example of functional requirement about text mining application that will be used until traceability of several UML diagrams are reached.

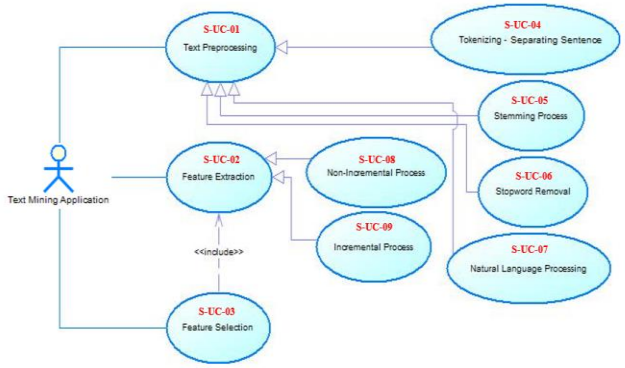


Fig 1 Example of Use-case Diagram for Functional Requirement in Table 1.

Table 1: Example of List of Functional Requirements

ID Func. Req.	Decription
S-FR-01	System is able to receive input collection of text document. Hereinafter is referred to input 1.
S-FR-02	System is abble to accept the result of pre-processing as input for extracting text representation. Next, referred to input 2.
S-FR-03	System can recieve minimum support value as a constraint for extracting text representation. Next, referred to input 3.
S-FR-04	System is able to read all the files on input 1.
S-FR-05	System is able to separate sentence in the document collection as a needs of text representation.
R-06	System is able to change the entire document to lowercase.
S-FR-07	System is able to remove stopwords, including punctuation, characters non-letters, and symbols that exist in input 1.
S-FR-08	System is able to do stemming process to text document from input 1 for creating the best text representation.
S-FR-09	System is able to store the result of pre-processing from input 1 to input 2. Hereinafter is referred to file 1.
S-FR-10	System is able to read the input 2 as input for creating text representation.
S-FR-11	System can create text representation using Frequent Pattern Growth algorithm. Hereinafter is referred to output 1.
S-FR-12	System capable of storing text representation, hereinafter is referred to file 2
S-FR-13	System is able to select redundant text representation in file 2.
S-FR-14	System capable of storing selected text representation into a file, hereinafter is referred to file 3.
S-FR-15	System is able to create the final text representation from file 3. Hereinafter is referred to output 2.
S-FR-16	System capable of storing the final text representation into a file, hereinafter is referred to file 4.
S-FR-17	System is able to correct abbreviation words into the real word accordance with natural language processing concept.

3. Tracing Requirements in Use Case Diagram

Use case diagram is used for modelling or describing interaction between actor and system with point of view by actor, what is actors do or give to system, and what is actors need or get from system [1], [2], [21], [27]. Use case diagram usually be used for describing bussiness process of system and modelling the analysis of system. Modelling the analysis of system automatically associated with functional requirement. Every use case in use case diagram must be represent at least one requirement. From the list of functional requirements in Table 1 is modelled with use case diagram in Figure 1.

Use case "Text Preprocessing" in Figure 1 represents several functional requirements in Table 1 among others S-FR-01, S-FR-02, S-FR-04 until S-FR-09, and S-FR-17, use case "Feature Extraction" represents several functional among others S-FR-03, S-FR-04, S-FR-10 until S-FR-12. And the last use case "Feature Selection" represents several functional requirements, among others S-FR-13 until F-FR-16. Another use case is specific use case from use case "Text Preprocessing", "Feature Extraction", and "Feature Selection". Every use case is better if given special code for tracing process easliy as shown in the Figure 1 with red colour of text.

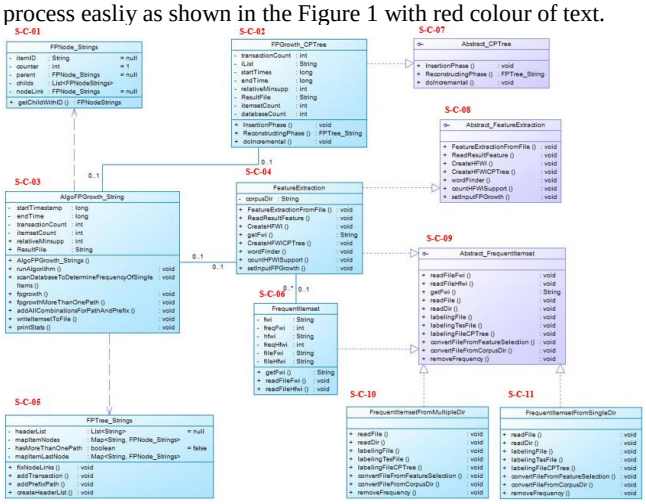


Fig 2: Example of pieces of Class Diagram for Functional Requirement in Table 1 and Use Case "Feature Extraction"

4. Tracing Requirements in Class Diagram

Class diagram is main diagram which characterizes object oriented software analysis and design, because class diagram represents an objects and describes the relation between objects of system [1]–[3], [7]. In object-oriented software analysis and design, indentifying an objects is the important steps, several object-oriented modelling do this step first before another step [3], [4], [7], [21]. Identifying an objects or classes of system easly from the result of requirement elicitation process. Every requirement of course has an object that related to the fuctionalities of system. Other than that, from use case can be identified an objects of system. From the list of functional requirements in Table 1 and use case "Feature Extraction" in Figure 1, the example of pieces of the class diagram model of system is describes in Figure 2 [30]. Same with use case diagram, for tracing the linkage of diagrams easily, it is better to give unique code for each class in class diagram.

Table 2: Example of List of Functional Requirements

Use Case ID	S-UC-02 : Feature Extraction
Brief Description	System run feature extraction function if actor call it.
Preconditions	System is running.
Basic Flow (S-UC-02: Feature Extraction)	
Assumptions	-
System Actor Action	**System Response**
Actor input document collection directori, name of output file, minimum support value, and call feature extraction function.	System read the content of file of document collection.
	System do text representation forming process with Frequent Pattern Growth algorithm, then save the result into output file.
	System produce set of text representation and re-check the appearance of set of text representation in document collection, then save the result into output file.
Post Condition	System success to save set of text representation into output file.

5. Tracing Use Case and Class in Sequence Diagram

Sequence diagram is one of interaction diagram that describes interaction between objects in a time sequence with series of messages flowing which requested or granted from one object to another object [4], [6], [7], [24], [27]. Every use case in use case diagram has scenario use case that describes detail process inside the use case. The flow of message in sequence diagram in accordance with the flow process from scenario use case. Ideally, sequence diagram modeling each of normal and alternative flow process from scenario use case.

Different from the flow of message in sequence diagram that related with scenario use case, an object in sequence diagram is related with an object or class in class diagram. Then, the message at the flow of message is an operation from object or class which is used. Relation between objects in sequence diagram must be consistent with relation between objects in class diagram. Figure 3 and Figure 4 [30], show the example of sequence diagram for normal flow of use case "Feature Extraction" that explained in Table 2 and use class from piece of class diagram in Figure 2. Actor in Fig. 3 and Fig. 4 is equal with actor "Pengguna" for this case study. Every sequence diagram is better has an unique code for tracing process of diagrams easily.

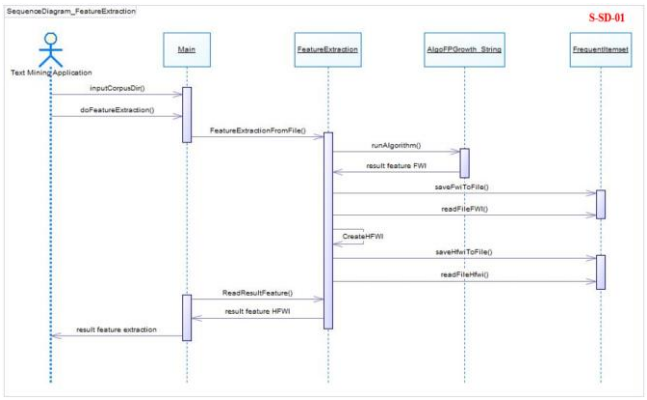


Fig 3: Example of Sequence diagram for use case "Feature Extraction – Non-Incremental Process"

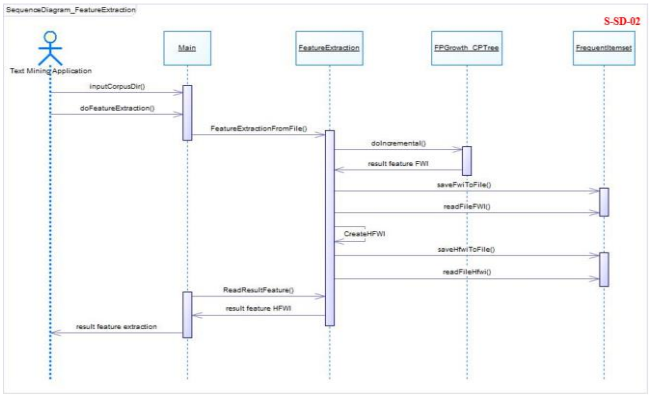


Fig 4: Example of Sequence diagram for use case "Feature Extraction – Incremental Process"

6. Traceability of Several Unified Modelling Language Diagrams

Tracing the relation or linkage between several diagrams in UML is an important process. Because, traceability of several diagrams can be measured how well modelling software has been created and whether all of user and software requirements are met. Figure 6 described the linkage between use case, class, and sequence diagram. Then, Table 3, Table 4, and Table 5 explain about traceability between use case, class, and sequence diagram as a result of the linkage of those three diagrams. From the example between use case, class, and sequence diagram have been had a right relation or linkage and have been meet all or functional requirements. Figure 5 is described about the flow process of modelling software analysis and design with UML, especially using use case, class, and sequence diagram, until resulting traceability of those diagrams.

Table 3: Traceability between Functional Requirement, Use Case, and Class

Traceability of Func. Req., Use case, and Class	**ID Func. Req.**	**ID Use case**	**ID Class**
	S-FR-01	S-UC-01	In the other class
	S-FR-02	S-UC-04, S-UC-05, S-UC-06, S-UC-07	S-C-04, S-C-08, S-C-09
	S-FR-03	S-UC-02, S-UC-08, S-UC-09	S-C-02, S-C-03, S-C-04
	S-FR-04	S-UC-02, S-UC-08, S-UC-09	S-C-02, S-C-03, S-C-04
	S-FR-05	S-UC-01, S-UC-04	In the other class
	S-FR-06	S-UC-01	In the other class
	S-FR-07	S-UC-01, S-UC-06	In the other class
	S-FR-08	S-UC-01, S-UC-05	In the other class
	S-FR-09	S-UC-01, S-UC-04, S-UC-05, S-UC-06, S-UC-07	In the other class
	S-FR-10	S-UC-02, S-UC-08, S-UC-09	S-C-02, S-C-03, S-C-04, S-C-07
	S-FR-11	S-UC-02, S-UC-08	S-C-01, S-C-03, S-C-04, S-C-05, S-C-06
	S-FR-12	S-UC-02, S-UC-08, S-UC-09	S-C-06, S-C-10, S-C-11
	S-FR-13	S-UC-03	In the other class
	S-FR-14	S-UC-03	In the other class
	S-FR-15	S-UC-02, S-UC-03	In the other class and S-C-06, S-C-10, S-C-11
	S-FR-16	S-UC-01, S-UC-02, S-UC-	In the other class and S-C-06, S-C-10, S-C-11

Trac	ID Func. Req.	ID Use case	ID Class
		03	
	S-FR-17	S-UC-01, S-UC-07	In the other class

Table 4: Traceability between Use Case and Class

Traceability between Use case and Class	ID Use case	ID Class
	S-UC-01	In the other class
	S-UC-02	S-C-01 until S-C-11
	S-UC-03	In the other class
	S-UC-04	In the other class
	S-UC-05	In the other class
	S-UC-06	In the other class
	S-UC-07	In the other class
	S-UC-08	S-C-01, S-C-03, S-C-04, S-C-05, S-C-06, S-C-08, S-C-09, S-C-10, S-C-11
	S-UC-09	S-C-02, S-C-04, S-C-06, S-C-07, S-C-08, S-C-09, S-C-10, S-C-11

Table 5: Traceability between Use Case, Sequence, and Class

Traceability between Use case, Sequence, and Class	ID Seq. Diagram	ID Use case	ID Class
	S-SD-01	S-UC-02, S-UC-08	S-C-03, S-C-04, S-C-06
	S-SD-02	S-UC-02, S-UC-09	S-C-02, S-C-04, S-C-06

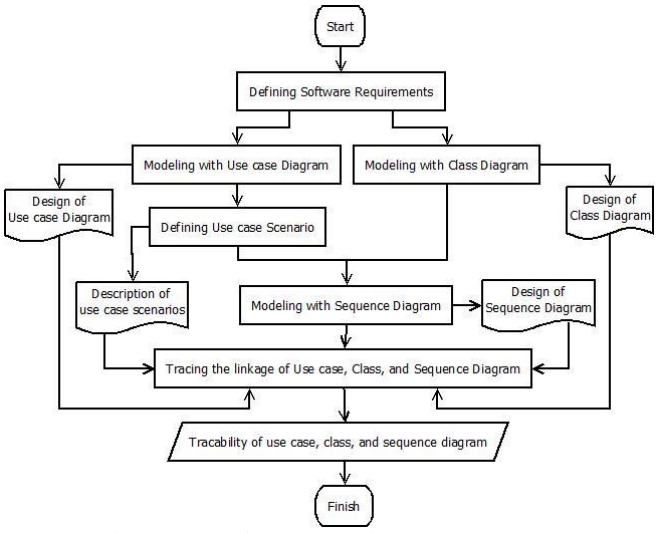


Fig 5: The flow process of modeling software analysis and design with UML

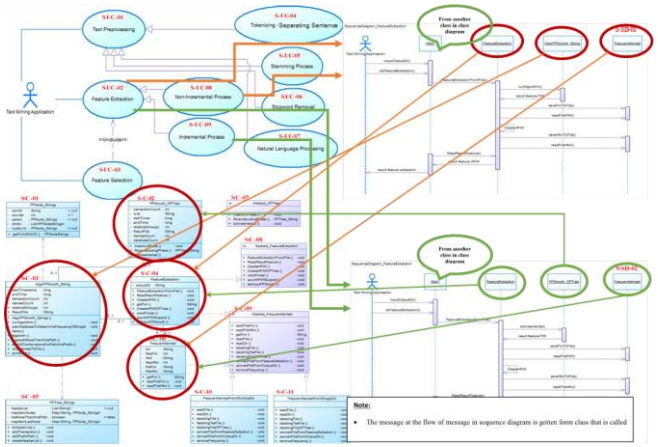


Fig 6: Traceability of several diagram in software analysis and design with UML

7. Conclusion

Object-oriented software modelling is one of point of view in software engineering method. In modelling software on analysis and design process, UML is still popular for specifying, visualising, constructing, and documenting software and has been proven has a good performance. Using modelling with UML diagrams are not arbitrary, one diagram to other diagrams have a rule and linkage. So, traceability of UML diagrams, especially use case, class, and sequence diagram, is an important steps to know whether the result of analysis and design software has been met the requirements or not yet. This paper is discuss about how to trace the linkage between use case, class, and sequence diagrams, and how to trace those diagrams until meet all of software and user requirements. For discussion, this paper explains only three of many UML diagrams. Every diagram in UML has a unique notations and rules, and of course have the linkage with another diagrams. Therefore, traceability for the other diagrams are also needed to achieve a good modelling software with good quality.

References

[1] R. S. Pressman, *Software Engineering: A Practitioner's Approach*, 7th ed. New York: McGraw-Hill, 2011.

[2] I. Sommerville, *Software Engineering*. 2010.

[3] P. Coad and E. Yourdon, "Object Oriented Analysis," *Comput. Aided Archit. Des.*, 1991.

[4] P. Coad and E. Yourdon, *Object-Oriented Design*, 1st ed. Yourdon Computing Press Series, 1990.

[5] G. Booch, "Object-Oriented Development," *IEEE Trans. Softw. Eng.*, vol. SE-12, no. 2, pp. 211–221, 1986.

[6] G. Booch, *Object Oriented Analysis and Design with Applications*, vol. 2. 1994.

[7] G. Booch, *Object-Oriented Analysis and Design*, 2nd ed. Santa Clara, California: Addison-Wesley, 1998.

[8] A. Pamoragung, K. Suryadi, and M. A. Ramdhani, "Enhancing the implementation of e-Government in indonesia through the high-quality of virtual community and knowledge portal," in *Proceedings of the European Conference on e-Government, ECEG*, 2006, pp. 341–348.

[9] M. A. Ramdhani, *Metodologi Penelitian untuk Riset Teknologi Informasi*. Bandung: UIN Sunan Gunung Djati Bandung, 2013.

[10] D. S. Maylawati, W. Darmalaksana, and M. A. Ramdhani, "Systematic Design of Expert System Using Unified Modelling Language," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 288, no. 1, p. 12047, 2018.

[11] H. Aulawi, M. A. Ramdhani, C. Slamet, H. Ainissyifa, and W. Darmalaksana, "Functional Need Analysis of Knowledge Portal Design in Higher Education Institution," *Int. Soft Comput.*, vol. 12, no. 2, pp. 132–141, 2017.

[12] C. Slamet, A. Rahman, A. Sutedi, W. Darmalaksana, M. A. Ramdhani, and D. S. Maylawati, "Social Media-Based Identifier for Natural Disaster," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 288, no. 1, p. 12039, 2018.

[13] C. Slamet, R. Andrian, D. S. Maylawati, W. Darmalaksana, and M. A. Ramdhani, "Web Scraping and Naïve Bayes Classification for Job Search Engine," vol. 288, no. 1, pp. 1–7, 2018.

[14] Y. A. Gerhana, W. B. Zulfikar, A. H. Ramdani, and M. A. Ramdhani, "Implementation of Nearest Neighbor using HSV to Identify Skin Disease," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 288, no. 1, p. 012153 1234567890 Implementation, 2018.

[15] A. Rahman, C. Slamet, W. Darmalaksana, Y. A. Gerhana, and M. A. Ramdhani, "Expert System for Deciding a Solution of Mechanical Failure in a Car using Case-based Reasoning," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 288, no. 1, p. 12011, 2018.

[16] C. Slamet, A. Rahman, M. A. Ramdhani, and W. Darmalaksana, "Clustering the Verses of the Holy Qur'an Using K-Means Algorithm," *Asian J. Inf. Technol.*, vol. 15, no. 24, pp. 5159–5162, 2016.

[17] D. S. Maylawati, M. A. Ramdhani, W. B. Zulfikar, I. Taufik, and W. Darmalaksana, "Expert system for predicting the early pregnancy with disorders using artificial neural network," in *2017 5th International Conference on Cyber and IT Service Management, CITSM 2017*, 2017.

[18] W. B. Zulfikar, Jumadi, P. K. Prasetyo, and M. A. Ramdhani, "Implementation of Mamdani Fuzzy Method in Employee

Promotion System," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 288, no. 1, p. 12147, 2018.
[19] D. S. A. Maylawati, M. A. Ramdhani, A. Rahman, and W. Darmalaksana, "Incremental technique with set of frequent word item sets for mining large Indonesian text data," in *2017 5th International Conference on Cyber and IT Service Management, CITSM 2017*, 2017.
[20] A. Taofik, N. Ismail, Y. A. Gerhana, K. Komarujaman, and M. A. Ramdhani, "Design of Smart System to Detect Ripeness of Tomato and Chili with New Approach in Data Acquisition," in *IOP Conference Series: Materials Science and Engineering*, 2018, vol. 288, no. 1, p. 12018.
[21] Booch, "The Unified Modeling Language User Guide," *Addison-Wesley Object Technol. Ser.*, p. 496, 2005.
[22] F. Schneider, B. Bruegge, and B. Berenbach, "The unified requirements modeling language: Shifting the focus to early requirements elicitation," in *2013 3rd International Workshop on Comparing Requirements Modeling Approaches, CMA@RE 2013 - Proceedings*, 2013, pp. 31–36.
[23] V. C. Gerogiannis, P. Fitsilis, and L. Anthopoulos, "Role of unified modelling language in software development in Greece – results from an exploratory study," *IET Softw.*, vol. 8, no. 4, pp. 143–153, 2014.
[24] D. Kundu, D. Samanta, and R. Mall, "Automatic code generation from unified modelling language sequence diagrams," *IET Softw.*, vol. 7, no. 1, pp. 12–28, 2013.
[25] V. S. W. Lam, "Theory for classifying equivalences of unified modelling language activity diagrams," *IET Softw.*, vol. 2, no. 5, p. 391, 2008.
[26] R. Pooley and P. King, "Unified modelling language and performance engineering," *IEE Proc. Softw.*, vol. 146, no. 1, pp. 1–10, 1999.
[27] K.-J. Yang and R. Pooley, "Process modelling to support the Unified Modelling Language," in *Computer Software and Applications Conference, 1997. COMPSAC '97. Proceedings., The Twenty-First Annual International*, 1997, pp. 467–472.
[28] C. Canevet, S. Gilmore, J. Hillston, M. Prowse, and P. Stevens, "Performance modelling with the Unified Modelling Language and stochastic process algebras," *Comput. Digit. Tech. IEE Proc.* -, vol. 150, no. 2, pp. 107–120, 2003.
[29] S. E. Viswanathan and P. Samuel, "Automatic code generation using unified modeling language activity and sequence models," *IET Softw.*, 2016.
[30] D. S. A. Maylawati, "PEMBANGUNAN LIBRARY PRE-PROCESSING UNTUK TEXT MINING DENGAN REPRESENTASI HIMPUNAN FREQUENT WORD ITEMSET (HFWI) Studi Kasus: Bahasa Gaul Indonesia," Bandung, 2015.