Journal of Korea Multimedia Society Vol. 22, No. 6, June 2019(pp. 706-715)
https://doi.org/10.9717/kmms.2019.22.6.706

Development of UML Tool using WPF Framework and Forced-Directionality Graph Algorithm

Ahmad Zulfiana Utama[†], Duk-Sung Jang[††]

ABSTRACT

This research implemented grammatical rules for relationship extraction from class diagram candidate. The problem statement is generated by our algorithm to yield class diagram and candidate relationship candidates. The relationships of class diagrams are extracted automatically from the problem statement by using Natural Language Processing (NLP). The extraction used the grammatical rules that obtained from various sources and translated into our algorithm. The performance evaluation of the extraction algorithm used ATM problem statements. The application captures the problem statement and draws automatically the relations of class diagrams using Forced-Directionality Graph algorithm. The performance evaluations show refining methods for class diagram and relationships extraction improve recall score.

Key words: Natural Language Processing, Problem Statement, Class Diagram, Relationships, Force-Directionality Graph Algorithm

1. INTRODUCTION

Software development is prone to failure. Bad planning is one of the factors that led to failure. Inappropriate plans lead to communication fraud among software developers, programmers, and stakeholder[1]. Software development is commonly used with object-oriented paradigm. The object-oriented paradigm is modeling the real world into the object. Developers need a standard notation to represent the object and their relations[2].

Applying Natural Language Processing (NLP) techniques in understanding of the problem statement is recommended. NLP analyzes the words from the problem statement. NLP has several tasks such as POS Tagger function, chunking process, stemming, and named entity recognition. These tasks have used to analyzed class diagram and relationships among classes.

This study has developed an application to extract class diagram notation consist of a class name, attributes, and operations from a problem statement. The class diagram notation extracted by using NLP approach[3]. If the class diagram has been analyzed, the relationships of class diagrams are identified. The relationships are represented by the main verb in a sentence[4]. The algorithm of applications determines relationships among classes based on a transitive verb. The grammatical rules divided the relationships according to the type associations, aggregation, composition, and generalization.

This paper describes how to extract relationships based on NLP approach and visualizes the

※ Corresponding Author: Duk-Sung Jang, Address: (42601) Sungseo Campus of Keimyung University, Dalgubul-daero 1095, Dalseo-gu, Daegu, Korea, TEL: +82-53-580-5267, FAX: +82-53-580-5165, E-mail: dsjang@kmu.ac.kr
Receipt date: Mar. 5, 2019, Revision date: May 23, 2019
Approval date: June 4, 2019

[†] (studied at) Dept. of Computer Eng., Keimyung University, (working at) Indonesian National Institute of Aeronautic and Space
(E-mail: ahmad.zulfiana@lapan.go.id)
[††] Dept. of Computer Eng., School of Engineering, Keimyung University

results by Forced-Directionality Graph(FDG) algorithm. The FDG algorithm will be implemented in this UML application to draw automatically class diagrams with relationships.

2. RELATED WORKS

Some research has been developed to find grammatical rules. The grammatical rules are mandatory tasks in NLP. A Researcher can define their grammatical rules related to domain research. Previous studies about grammatical rules in class diagram extraction domain have explained by Elbendak[5] and Sagar[6].

Elbendak[5] represented Class-Gen tool, which can identify object/classes from natural language specifications automatically. Class-gen generated requirements from uses case description that is written in natural language. They are using a memory-based shallow parser(MBSP) for the preprocessing stage, and also developing design rules based on grammatical language.

Sagar[6] has proposed design rules build upon grammatical constructs. This rule aims to extract class diagram notation from problem statements automatically. Design rules categorized as class rules, attribute rules, operation rules, relations rules, and subject forms rules. Each rule has different formulations. For instance, class rules have four formulations to distinguish words as a class name from the sentence. Also, attribute rules have five formulations to gather information from sentences. The main contributions of their works that were created the strict pattern (design rules) to analyze problem statement. The results of the works reach around 90 percent for recall and precision measurement on ATM problem statements made by Rumbaugh[7].

This study implemented the grammatical rules from Elbendak[5] and Sagar[6]. Furthermore, we translated and implemented into the UML tools. In addition, this study developed refinement methods, such as identification of relationship using dependency parser, and checking similarity of words with WordNet library. Checking similarity function in WordNet has been used by Lee and Hwang[8] to differ semantic similarity that was used to find a correlation between image tags. Also, this study proposed a new technique to draw automatically class diagrams using FDG algorithm.

3. RELATIONSHIPS

3.1 Class Diagram Extraction

To determine a relationship of class diagrams, first, the program must find the class diagram from the problem statement. The detailed process for determining class has been discussed in our previous research. The algorithm of class diagram extraction used the grammatical rules from previous studies and refinement methods that were developed during implementation phases. The evaluation results for the algorithm achieved a high score, which is close to 95 percent for recall or precision[3].

3.2 Relationship Extraction

Our research has succeeded in define the candidate class automatically so that we would use as the basic formula for define candidates relationship. In general, the connection identifies by a verb/verb phrase. If there are two classes separated by a verb, the verb is used as a relationship candidate. To find out the relationship, if one class *possess*, *controls*, *is connected to*, *is related to*, *is a part of*, *has parts*, *is a member of* or *has/have* as members some other class in our system.

There are several stages to extract relationship. The algorithm extracted relationship type sequentially from association, aggregation, composition, and generalization. Class relationship modeling starts with creating an initial set of aggregation, composition, and generalization. After that, assign the associations to the classes from unused verbs.

The order based on the verb forming a relationship. The type of relation aggregation, composition, and generality has special verbs while associations fit all types of verbs.

3.2.1 Association Extraction

The association relations are the most general relationships. It is a glue among classes; if the class has not an association, it is means the class is isolated. The association relationship defines the semantic connections between individual of class candidates. Classes can interact with each other and visualize by associations relationship in a diagram. Association carries information or known as messages relationship among the classes in the system. We set tuple of association as:

A:= (*Class1*, *Class2*, *Message*)

In natural language text, association is marked by verb phrases. We are collected the grammatical rules to denote an association relationship is shown in Table 1.

3.2.2 Aggregation Extraction

Aggregation is a special kind of association that represents “whole/part” hierarchy. The whole side of the relationships is often called the assembly or the aggregate. Aggregation can express class is a combination of the other classes. For example, we can define “car” as an aggregation of “engine”, “body”, and so forth[11]. In other hands, the engine is “part of” the car. This “part of” relationships is known as aggregation. This relationship is a weak form of containment in that the lifetimes of the whole and its parts are independent. For example, in car problems, we can replace the type of engine without destroying the car object.

3.2.3 Composition Extraction

The composition is a strong kind of aggregation type. If the aggregate is destroyed, then the parts would be destroyed as well. The object parts of a composition cannot stand alone; the objects exist only to serve the aggregate. This relationship exists in the lifetimes of the whole, and its parts are dependent.

The rules in composition extraction from requirement specification are limited. It is hard to distinguish between aggregation and composition relationships, because the formula is very similar, shown in Table 3.

Table 1. Grammatical Rules to define association type.

No	Grammatical Rules	References
1	If the sentence has a form: C1 - VB - C2 where C1 and C2 are class candidates, VB is an association relationship. The VB determine as message in this relation.	[5][9][10]
2	A transitive verb is a candidate for a relationship type. A transitive verb links a subject and an object. C1(subj) - transitive verb - C2(obj)	[6][9][4]
3	A verb with a preposition is a candidate for a relationship. A verb that contains a prepositional object linked to a transitive verb along with a preposition combines with the preposition to form a relationship. C1 - verb - preposition - C2	[6]
4	A verb showing possession (e.g., “to have”) may also imply an aggregation or association. In this type of “has/have” phrase, the noun that occurs after the phrase does not usually denote an attribute. The possession would show an association relationship between two entity types (classes). C1 - has/have - C2	[9]
5	If the verb phrase cannot be categorized into aggregation, composition or generalization, or processed to trivial relations, is an association relation. if V not in (aggregation, composition, generalization) then V is association.	[6]

Table 2. Grammatical Rules to define aggregation type

No	Grammatical Rules	References
1	Find the aggregation by finding verb phrase in the form “something contains something”, “something is part of something”, and “something is made up of something”. The verb phrase patterns such as: C1 – “is made up of” – C2 C1 – “is part of” – C2 C1 – “contains” – C2 C1 – “comprises” – C2	[6][9] [12][13]
2	If the sentence has formed: C1 – has/have – C2 – C3 – ...	[12]

Table 3. Grammatical Rules to define composition type

No	Grammatical Rules	References
1	The “of” prepositions may indicate the composition. C1 – “of” – C2	[9]
2	If a verb is in the following list {include, involve, consists of, contain, comprise, divided to, embrace}, this indicates a relationship of aggregation or composition.	[14][15]

3.2.4 Generalization Extraction

The generalization uses to arrange classes into inheritance hierarchies. A high-level abstraction is generalized, and a low-level abstraction is specialized. The specialized classes receive all of the attributes and operations, which are defined in the parent class. The classes in a generalization relationship must be the same type/abstraction.

One of the most significant rules to define generalization is the “is-a” rule. The “is-a” rule states that class A can only be a valid subclass of class B. The rule of the “is-a” verb typically indicates a hierarchical relationship between two classes.

Organizing classes into inheritance hierarchies are critical in object-oriented design and programming. In the practical object-oriented programming, the generalizations help to evade duplication and make the body of the class is reusable.

3.3 Refining Algorithm

3.3.1 Object of Preposition

We can find out the object of prepositions by used POS tagging and knows the relation in a phrase by using the dependency tree. POS tagging marked the object of prepositions with the pobj tag.

Fig. 1 is visualization of the dependency tree object of prepositions. The "cart" word is an object of prepositions by using a preposition "to." The ob-

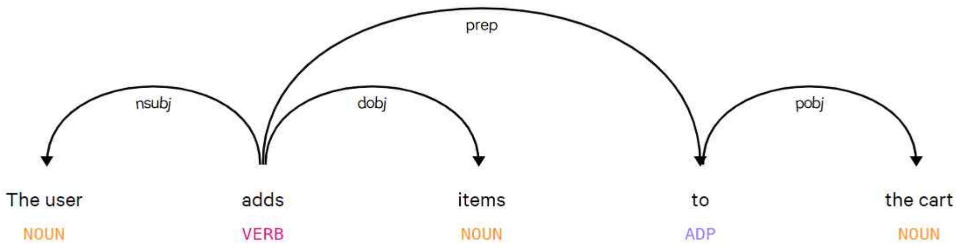


Fig. 1. Example of relationships for object of prepositions case.

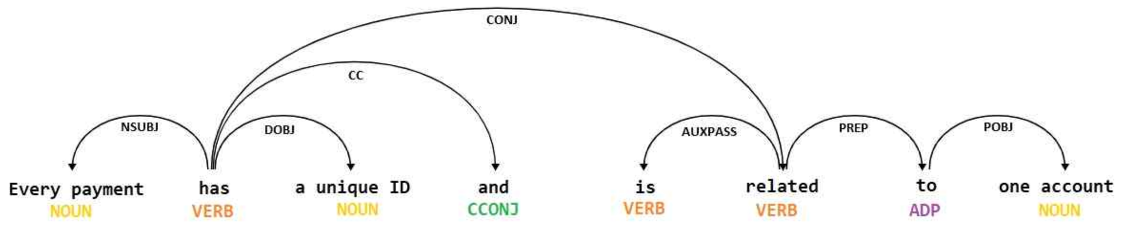


Fig. 2. Example of the finding relationships by using dependency parser.

ject of preposition will have no meaning if it does not refer to the verb in a sentence. To find out the verb of an object of prepositions can use the token.head function on spaCy. The spaCy uses the terms head and child to define the linking of words in a phrase. Each word can only have one head; this head is the parent of a word. The linking of words describes as "parent-child."

3.3.2 Dependency parser

Dependencies relation is a short description of grammatical relationships in a sentence to ease understanding and increase effectiveness. It represents all sentence relationships uniformly as typed dependency relations. The dependency analyzes triples of words to produce syntactic relations.

Dependency parsers can be used to detect compound verb for associate relations and detect relations based on the relationship between words. The sentence can consist of two verbs which refer to a subject known as a compound verb. Fig. 2. An example of a compound verb. For problems as illustrated in Fig. 2, the design rules can only identify one relationship, specifically subject in the first verb and objects.

Design rules will extract the relationship in Fig. 2. based on the first verb. Whereas the second verb "related to" is not determined. The advantages of this examination will produce more accurate results. Results obtained from Fig. 2. The dependency parser method will extract the association relationship that connects the payment class and account. The following is an example of a snippet list from the results of the preprocessing process that stores information on dependency parser:

[[. . . ('account', ['NP', 'NOUNPHRASE', 'pobj', 'account'], ['related to', 'IN'], ['one'])]],

Dependency parser stored in index 2 [‘related to ',‘ IN'] for head, and child in index 3 ['one']. The searching algorithm to get dependency information started from the last index, and each iteration traces the parent from the previous index. For example, the word account has a parent "related to," the word "related to" has a parent "has," and the "has" word has a child "payment." The searching algorithm would stop when the noun subject had found.

3.3.3 Word Similarity Methods

We have developed two methods to reduce the similarity of words contained in the problem statement; checking words using the similarity() function of the WordNet library and check each candidate class which consists of a noun phrase. this function detects words equality based on vector

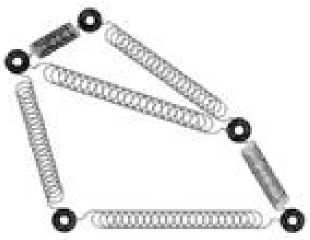


Fig. 3. Spring Model.

values, after that find and replace the information from synonym words. An Example of a redundancy noun phrase, the "item" word will be considered similar to the "loan item" noun phrase. While an examples of the similarity() function, if the system has a class candidate "order" and "purchase," the word checking algorithm will choose one candidate class.

4. VISUALIZATION OF RELATIONSHIP USING FDG

After the application has successfully extracted the class diagram and its relation, a new problem arises, namely how to draw a class diagram and its relation automatically to the canvas. Canvas uses position X, Y to place an object in it. The FDG algorithm will calculate the position of X, Y object automatically based on the number of objects and the number of relations that will be visualized.

The Force-Directed GraphFDG algorithm uses a spring model representation, where nodes are considered charged objects that are connected (attraction) with a line represented by a spring. The charged objects attract (repulse) each other.

There are four main steps in the FDG Algorithm[16]:

1. Calculate the repulsive force for each node.
2. Calculate the attractive force for all nodes connected by lines.
3. Sum the repulsive and attractive forces.
4. Replace the nodes with a new position.

The Fruchterman and Reingold method[16] uses the following equation to find the attractive and repulsive forces:

$$Fa(d) = \frac{d^2}{k} \tag{1}$$

$$Fr(d) = \frac{-k^2}{d} \tag{2}$$

In equation (1) d is the distance between 2 neighboring vertices. k is a constant for node placement that distributes nodes evenly. The constant k is stated by equation (3):

$$k = \sqrt{\frac{areaofframe}{|V|}} \tag{3}$$

Area of a frame is the multiplication of the length and width of the canvas area (Area), and | V | the number of nodes to be drawn. Fruchterman and Reingold add a variable t to decrease temperature/calculation formulated as

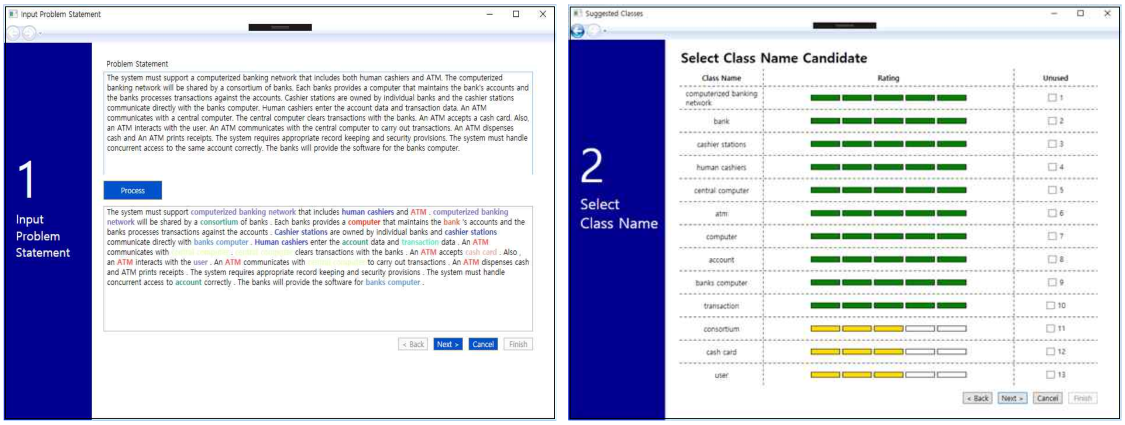
$$t = \frac{width}{10} \tag{4}$$

The t value is used to speed up the placement of the new node because the frame width is reduceds by one-tenth of the previous area.

5. IMPLEMENTATION

In the implementation section, we used the ATM problem statement made by Rumbaugh[7]. This problem statement has been used by previous studies[5,6]. Here is a way to extract the class diagram using the UML application that we have developed ourselves. First, users inputted the problem statement into the input column, shown in Fig. 4(left). The application shows the results of the problem statement analysis, which was marked by writing marked with color. The coloring makes it easier for users to see the results of the class, attributes, and operations that were successfully extracted by the application. Second, users will be given a choice of class names that have been successfully extracted and given a rating based on the grammatical rules that were successfully fulfilled by the classes. The user can delete the candidate class by clicking on the checklist box column, shown in Fig. 4(right).

Third, the user verifies every relation that belongs to a particular class, shown in Fig. 5(left). Finally, the application will visualize the class diagram along with the relation automatically, shown in Fig. 5(right). The advantage of using the FDG algorithm is that when the application places the



Problem Statement

The system must support a computerized banking network that includes both human cashiers and ATM. The computerized banking network will be shared by a consortium of banks. Each banks provides a computer that maintains the bank's accounts and the banks processes transactions against the accounts. Cashier stations are owned by individual banks and the cashier stations communicate directly with the banks computer. Human cashiers enter the account data and transaction data. An ATM communicates with a central computer. The central computer clears transactions with the banks. An ATM accepts a cash card. Also, an ATM interacts with the user. An ATM communicates with the central computer to carry out transactions. An ATM dispenses cash and An ATM prints receipts. The system requires appropriate record keeping and security provisions. The system must handle concurrent access to the same account correctly. The banks will provide the software for the banks computer.

Process

The system must support computerized banking network that includes human cashiers and ATM . computerized banking network will be shared by a consortium of banks . Each banks provides a computer that maintains the bank 's accounts and the banks processes transactions against the accounts . Cashier stations are owned by individual banks and cashier stations communicate directly with banks computer . Human cashiers enter the account data and transaction data . An ATM communicates with central computer . central computer clears transactions with the banks . An ATM accepts cash card . Also , an ATM interacts with the user . An ATM communicates with central computer to carry out transactions . An ATM dispenses cash and ATM prints receipts . The system requires appropriate record keeping and security provisions . The system must handle concurrent access to account correctly . The banks will provide the software for banks computer .

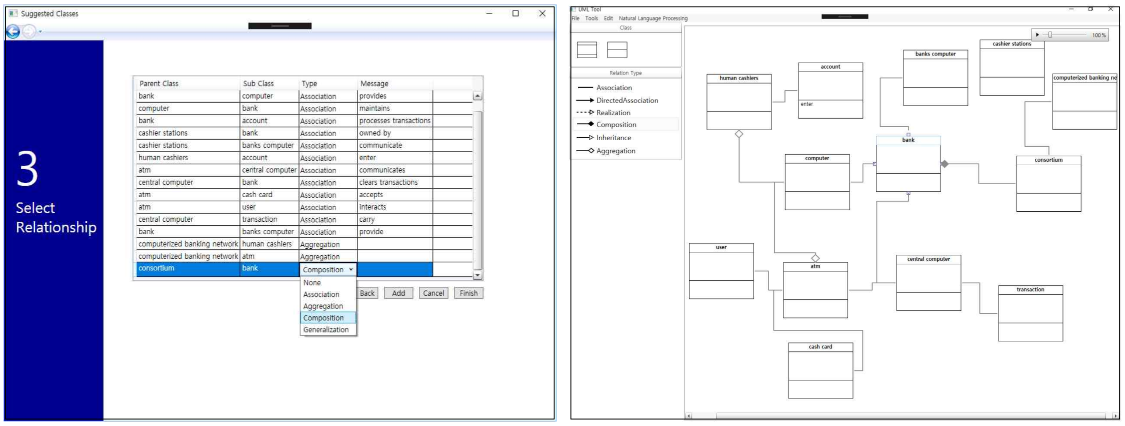
< Back　Next >　Cancel　Finish

Select Class Name Candidate

Class Name	Rating	Unused
computerized banking network	🟩🟩🟩🟩🟩	☐ 1
bank	🟩🟩🟩🟩🟩	☐ 2
cashier stations	🟩🟩🟩🟩🟩	☐ 3
human cashiers	🟩🟩🟩🟩🟩	☐ 4
central computer	🟩🟩🟩🟩🟩	☐ 5
atm	🟩🟩🟩🟩🟩	☐ 6
computer	🟩🟩🟩🟩🟩	☐ 7
account	🟩🟩🟩🟩🟩	☐ 8
banks computer	🟩🟩🟩🟩🟩	☐ 9
transaction	🟩🟩🟩🟩🟩	☐ 10
consortium	🟨🟨🟨⬜⬜	☐ 11
cash card	🟨🟨🟨⬜⬜	☐ 12
user	🟨🟨🟨⬜⬜	☐ 13

< Back　Next >　Cancel　Finish

Fig. 4. Wizard for input problem statement(left). Wizard for evaluation class candidate(right).



Parent Class	Sub Class	Type	Message
bank	computer	Association	provides
computer	bank	Association	maintains
bank	account	Association	processes transactions
cashier stations	bank	Association	owned by
cashier stations	banks computer	Association	communicate
human cashiers	account	Association	enter
atm	central computer	Association	communicates
central computer	bank	Association	clears transactions
atm	cash card	Association	accepts
atm	user	Association	interacts
central computer	transaction	Association	carry
bank	banks computer	Association	provide
computerized banking network	human cashiers	Aggregation	
computerized banking network	atm	Aggregation	
consortium	bank	Composition	

None
Association
Aggregation
Composition
Generalization

Back　Add　Cancel　Finish

Fig. 5. Wizard for evaluation relationship(left). Visualization for the final results(right).

class diagram on the canvas, the connected class will be close together, making it easier for users to change the position of the class diagram. Users can change the position of the class diagram, or additional information from the class diagram itself.

6. EVALUATION

The standard definition of correctness for extracting requirement by NLP approach does not exist. The standard guidelines to measure our work by using the problem statement from books/previous research. Books and previous research considered to have true validity. We used the ATM and library statement for performance evaluation. The class diagram and relationship had extracted by them used as the answer key. The answer key used to validate our results. The evaluation matrices used recall, precision, and over-specification. The recall attends to compare the results produced with an answer key table. Recall calculates the missed results. The following is the formula for recall[5]:

$$R = \frac{N_{correct}}{N_{correct} + N_{missing}} \tag{5}$$

N*correct* defines the value that matches to the answer key. The N*missing* counts the number of missing answers. Precision calculates how many values do not match to the answer key[5].

Table 4. Grammatical Rules to define composition type

No	Grammatical Rules	References
1	If two classes separate with “is-a” word.	[9]
2	If two classes separate by list of generalization words such as “type of”, “categories of”, “kinds of“.	[6]

$$P = \frac{N_{correct}}{N_{correct} + N_{incorrect}} \quad (6)$$

The incorrect value is a result that is not included in the answer key. In this study, the incorrect value is defined as a class candidate from the subject noun and not includes in the answer key. The values that are not included in the answer key but are included in requirements of grammar rules are stated as Nextra to report over-specification[5].

$$Os = \frac{N_{extra}}{N_{correct} + N_{missing}} \quad (7)$$

In this section, the results used design rules shown in table 1-4 are compared with our refining method, which described in section 3.3. Table 5 shows the results of system performance evaluations for three case studies. The answer key from ATM case is as follows {*consortium, bank, account, customer, central computer, bank computer, cashier, cashier station, atm, remote transaction, cash card*}. Our system produce class candidates are as follows {*bank, cashier station, human cashier, automatic teller machine, computerized banking network, atm, consortium, computer, account, transaction, central computer, cash card, user, customer*}. Our system produce class candidates are as follows {*bank, cashier station, human cashier, automatic teller machine, computerized banking network, atm, consortium, computer, account, transaction, central computer, cash card, user, customer*}.

The answer key of relationships for ATM case consists of 10 implicit relations and 6 explicit relations. When we are applying the design rules from previous studies that shown in table 1-4 were unable to detect the relationship of object prepositions and relationship based on the relation of words (dependency words). In ATM case consists of two relationships which were defined by the object of preposition. The recall and over-specification score before refining were 66% and 100%. After the application implemented the refining algorithm, the score increased to 100%, and 100% For redundancy words, the algorithm found one word; *[['cash', 'cash card', 13, 11]]*. The class candidate “cash” replaced by “cash card.”

The results for the library case[17] before refining show that recall, precision, and over-specification are 50%, 100% and 25% respectively. After the application used the refining method got 100%,

Table 5. The performance evaluation of relationships candidate

Problem Statements	Score					
	Before Refining			After Refining		
	Recall	Precision	Over Specification	Recall	Precision	Over Specification
	Explicit[%]	Explicit[%]	Explicit[%]	Explicit[%]	Explicit[%]	Explicit[%]
ATM[5]	66	100	100	100	100	100
Library Management System[17]	50	100	25	100	100	25
Online Shopping[18]	80	100	50	100	100	50

100%, and 25%. The generalization relations in library problem statements related by dependency words. If the application were only used design rules shown in table 4, the system failed to define relationships. After the application implemented dependency parser method to find a relationship in a sentence, it succeed to found generalization relationships. The performance evaluation after refining is 100%, 100%, and 25%. For redundancy words found three words included in [['member', 'customer', 9, 0], ['item', 'loan item', 6, 3], ['loan', 'loan item', 13, 3]]. The "member" class candidate replaced by "customer", and "item"/"loan" replaced by "loan item". The exciting findings from the library case, the redundancy words affected the results of extraction as explained by the Harmain [17]. His results showed incorrect values due to redundancies and synonyms. Our system can tackle these problems by using the refine redundancy function to reduce similarity words.

The results of the application from Online Shopping[18] show that recall, precision, and over-specifications for class candidates are 80% , 100%, and 50%. The performance results after refining methods are 100%, 100%, and 50%. We have found two relationships define by the object of prepositions. For redundancy words we have found two words [['order', 'customer order', 6, 5], ['item', 'line item', 10, 8]]. The "order" words replaced by "customer order" and "item" word replaced by "line item."

7. CONCLUSION

When the application implemented the design rules from previous studies, there were several problems. The design rules could not detect relationships on the object of prepositions and compound verb. Also, the design rules could not define the connectedness that was illustrated through word relations. After the application had used refining methods, which were consists of check word similarity, dependency parser, and object of prepositions, the results of performance evaluations were increased. In addition. If the high accuracy in determining the class candidate increased, it will affect the relationships extraction. For the visualization section, FDG algorithm is helpful to arrange class diagram position and draw the relationships in the canvas automatically.

REFERENCE

[1] R.S. Schach, *Object-oriented Software Engineering,* McGraw-Hill Publishers, Pennsylvania, 2008.

[2] J. Mylopoulos, L. Chung, and Y. Eric, "From Object-oriented to Goal-oriented Requirements Analysis," *Communications of the ACM*, Vol. 42, No. 1, pp. 31-37, 1999.

[3] A.Z. Utama and D.S. Jang, "An Automatic Construction for Class Diagram from Problem Statement Using Natural Language Processing," *Journal of Korea Multimedia Society,* Vol. 22, No. 3, pp. 386-394, 2019.

[4] S. Hartmann and L. Sebastian, "English Sentence Structures and EER Modeling," *Proceeding of ACM International Conference Proceeding Series*, Vol. 247, pp. 27-35, 2007.

[5] M. Elbendak, P. Vickers, and N. Rossiter, "Parsed Use Case Descriptions as a Basis for Object-oriented Class Model Generation," *Journal of Systems and Software,* Vol. 84, No. 7, pp. 1209-1223, 2011.

[6] V.B. Sagar, R. Vidya, and S. Abirami, "Conceptual Modeling of Natural Language Functional Requirements," *Journal of Systems and Software,* Vol. 88, pp. 25-41, 2014.

[7] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W.E. Lorensen, *Object Oriented Modeling and Design,* Prentice-Hall, New Jersey, 1991.

[8] S.H. Lee and D.H. Hwang, "Tag Ranking System Based on Semantic Similarity of Tag-pair," *Journal of Korea Multimedia Society,*

Vol. 16, No. 11, pp. 1305-1314, 2013.

[9] M. Ibrahim and R. Ahmad, "Class Diagram Extraction from Textual Requirements Using Natural Language Processing (NLP) Techniques," *Proceeding of International Conference on Computer Research and Development*, pp. 200-204, 2010.

[10] G. Lucassen, M. Robeer, F. Dalpiaz, J.M.E. Werf, and S. Brinkkemper, "Extracting Conceptual Models from User Stories with Visual Narrator," *Journal Requirements Engineering,* Vol. 22. No. 3, pp. 339-358, 2017.

[11] B. Meyer, *Object-oriented Software Construction,* Prentice Hall Publishers, New York, 1988.

[12] A. Dennis, B.H. Wixom, and D. Tegarden, *Systems Analysis and Design UML Version 2.0*, Wiley Publishers, New Jersey, 2009.

[13] T.C. Lethbridge and R. Laganière, *Object Oriented Software Engineering: Practical Software Development Using UML and Java*, McGraw-Hill Publishers, Pennsylvania, 2004.

[14] H. Herchi and W.B. Abdessalem, "From User Requirements to UML Class Diagram," *Proceeding of International Conference on Computer Related Knowledge*, pp. 68-71, 2012.

[15] C. Arora, M. Sabetzadeh, L. Briand, and F. Zimmer, "Extracting Domain Models from Natural-language Requirements: Approach and Industrial Evaluation," *Proceeding of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems*, pp. 250-260, 2016.

[16] T.M. Fruchterman and E.M. Reingold, "Graph Drawing by Force Placement," *Software: Practice and Experience*, Vol. 21, No. 11, pp. 1129-1164, 1991.

[17] H.M. Harmain and R. Gaizauskas, "CM-Builder: an Automated NL-based CASE Tool," *Proceedings ASE 2000, Proceeding of IEEE International Conference on Automated Software Engineering*, pp. 45-53, 2000.

[18] Online Shopping, https://www.uml-diagrams.org/examples/online-shopping-domain-uml-diagram-example.html (accessed Feb., 15, 2019).

Ahmad Zulfiana Utama

Ahmad Zulfiana is a junior researcher in space science center at Indonesian National Institute of Aeronautics and Space. He is candidate master degree in Keimyung University. His areas of interest include System Information Development, Database, Visualization and Natural Language Processing.

Duk-Sung Jang

He received the BS degree in computer engineering from Kyung pook National Univ., in 1979. He also obtained the MS and PhD degrees in computer engineering from Seoul National Univ., in 1981 and 1988 respectively. He is currently a professor with the Dept. of Computer Eng. at Keimyung Univ. His research interests include Compiler, Natural Language Processing, and Voice Recognition.