2019 1st International Conference on Cybernetics and Intelligent System (ICORIS)
22nd-23rd August 2019 at Institut Teknologi dan Bisnis (ITB) STIKOM Bali, Indonesia

Requirements Dependency Graph Modeling on Software Requirements Specification Using Text Analysis

Yudi Priyadi
Informatics Department
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
yudi.17051@mhs.its.ac.id

Arif Djunaidy
Information Sytems Department
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
adjunaidy@is.its.ac.id

Daniel Siahaan
Informatics Department
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
daniel@if.its.ac.id

***Abstract*—Understanding interdependency among requirements is one of the success factors in software development. Information on requirements interdependency explicitly and implicitly resided various design artifacts or diagrams. A software requirements specification document is the artifact delivered in the early phase of development. It drives its following development processes. It also contains information on interdependencies among the requirements, such as similar, part-of, and elaborate. This study proposes an approach to model the requirement dependency graph for a software requirements specification document. There is an extraction process for Text Preprocessing, which includes of Tokenization, Stopword Removal, and Stemming. Besides, there is a process of measuring semantic similarity through WS4J (WordNet Similarity for Java). The results of the extraction process, combined with Greedy Algorithms as the optimum value solution approach. Besides, a method for calculating similarity was used through the practices of Wu Palmer and Levenshtein. At the end of this process, Reliability is performed using the Gwet's AC1 formula.**

Keywords—software requirements specifications, requirements dependency graph, similarity, dependency type, text preprocessing, reliability

I. Introduction

Requirement engineering is a branch of software engineering, which focuses on several factors, namely: goals, functions, and limitations on software [1]. This branch includes discussing the relationship between these factors to determine the specifications of the software. The results of the requirements engineering process are documented as a software requirements specification document. It plays as an agreement between the developer and the customer on factors to be achieved at the stage of software design.

Dahlstedt and Persson [2] explain that most individual requirements developed during the requirement engineering process, cannot be repaired separately from the rest of the requirements during the software development. This behavior happens because all requirements are interrelated and affecting each other in a heuristic pattern, resulting in requirements interdependencies.

Understanding requirements interdependency is one of the success factors in the software development process. The information that explains the requirements interdependency may be distributed across the software development processes. The artifacts produced through the processes may be implicit and explicitly contains the interdependency information. For visualizing these requirements, there is a diagram notation that refers to the Interdependency Requirements Relationship Diagram [3]. A requirement change is an unavoidable event within the software development life cycle. Thus, a change in a requirement would not be solely independent to itself. Moreover, it may cause a chain effect on its associated requirements.

This study introduces a method to extract requirement dependencies from a software requirements specification document. It also graphically models the dependencies in a requirement dependency graph (RDG). The rest of the paper would provide the following information. First, it shows the theoretical background on RDG. Second, it describes the construction of the method. Third, it illustrates the implementation of the method. Fourth, it illustrates the RDG reliability. Fifth, it shows the result and its analysis. Finally, it concludes the work.

II. Theoretical Background

A. Requirements Dependency Graph

Based on Dahlstedt and Persson [2], in the interdependency process, there is an explanation of the relationship and the relationship between requirements. Even in previous studies, it was written that elements could not stand alone, but a condition is always related to and influence other needs in a complicated way [4].

To improve the understanding and the usage of natural language processing, the specification of software requirements decrypt the grammatical relationships of the software requirements in natural language sentences [5]. Therefore, the requirements interdependency information can be utilized by a software engineer without the need for linguistic expertise to extract textual relationships.

In carrying out the engineering activities on software, the collective vision between the client and the developer is a necessity, to achieve the development of a system that is following their goals and needs. A research activity conducted by [6], that delay in correcting needs, can result in costs as much as 200 times, compared to the corrective activity carried out during the phase of Requirement.

Given the size and complexity of the software system is expensive and time-consuming, the application of this dependency Requirement method can handle the dependency relationship in the software development cycle when changes occur. In addition to these benefits, it also has the potential to increase cost control. Through this method, then in carrying out engineering activities on software, the collective vision between the client and the developer can be achieved for the development of a system that is following their goals and requirements.

B. Research Framework

The rationale that became the idea of this research was software development activities that were not following the needs of the business processes of its users. The reason is that Software Requirement Specification has become an agreement between users and developers, changes occur and are not following system requirements. The effect of this event will result in changes to all conditions related to business processes that have been defined in the SRS so that it can be concluded that there is a concept of dependency between all the requirements involved.

In Fig. 1, an overall research idea is presented, regarding requirements graph interdependency that occurs in the Software Requirement Specification document. For example, there are three artifacts involved in Software Requirement Specification documents, namely:

- Software Requirement Specification. Dependencies can occur between Functional Requirements Statement or Non-Functional Requirement Statement.
- Use Case Diagram. Dependencies can occur through Use Case with the Requirement Statement.
- Class Diagram. Dependencies can occur through the Class with the Requirement Statement

Based on the results of the dependency on the three examples of UML artifacts, it will produce a requirement dependency mapping that has the nature of a relationship in the form of a series of graphs that can be integrated. (See Fig. 1).

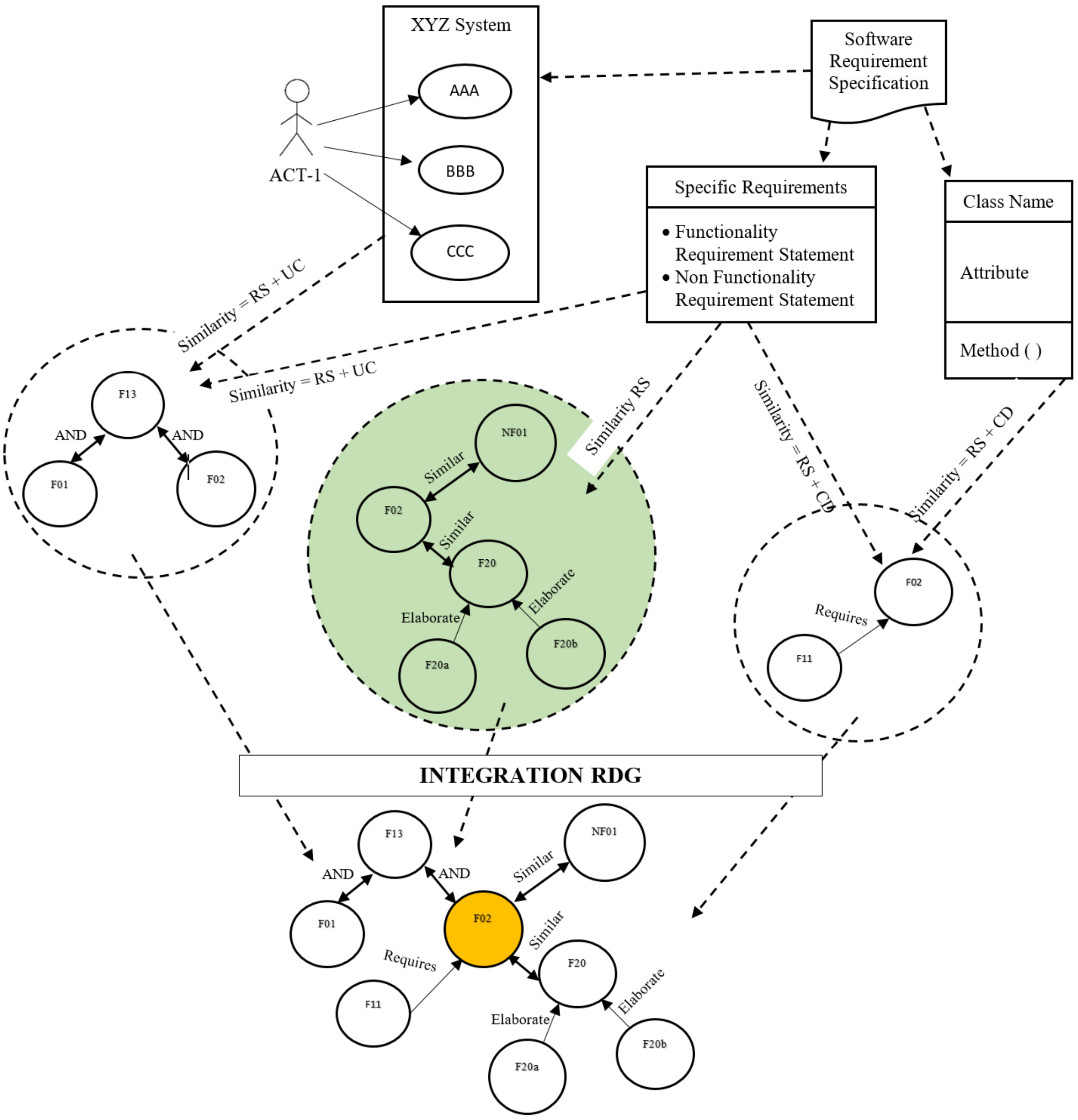


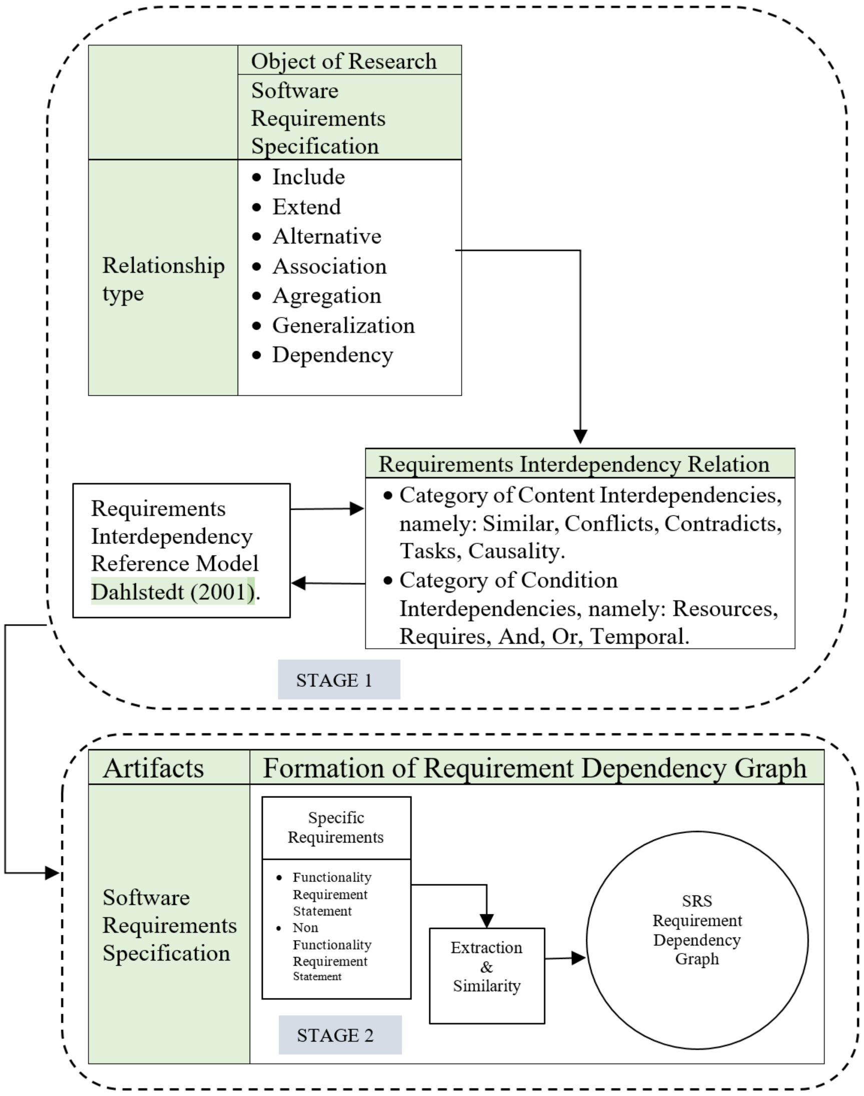
Fig. 1. Overall Research Ideas.

This research is a sub-part of a bigger project. The project focuses on extracting RDGs from various UML artifacts. The project also aims to develop a mechanism to integrate those RDGs and maintains the integrity of the unified RDG. The RDGs integrations may produce redundancy and discrepancy in model integrity. The focus of this research is to develop a method to produce the Requirements Dependency Graph, based on the requirements extraction results.

To describe the research process on SRS artifacts, in Fig. 2, a chart is presented, which represents the stages of the process as a framework for thinking, namely:

1) Conduct requirements traceability, which is the initial stage to identify the types of relations on artifacts that are the object of research. Furthermore, it is analyzed to determine the kind of Requirement Interdependency based on a model reference for the types of Interdependency. For reference, the type of mapping is based on the Dahlstedt reference model.

2) Extract requirement dependencies from the Software Requirements Specification. It is done to form the Requirement Dependency Graph based on the information resided in the artifact. As input material in this extraction process, it comes from a combination of Functional Requirements Statement and Non-Functional Requirement Statement.



Object of Research
Software Requirements Specification
Relationship type
Include
Extend
Alternative
Association
Agregation
Generalization
Dependency
Requirements Interdependency Relation
Category of Content Interdependencies, namely: Similar, Conflicts, Contradicts, Tasks, Causality.
Category of Condition Interdependencies, namely: Resources, Requires, And, Or, Temporal.
Requirements Interdependency Reference Model Dahlstedt (2001).
STAGE 1
Artifacts
Formation of Requirement Dependency Graph
Software Requirements Specification
Specific Requirements
Functionality Requirement Statement
Non Functionality Requirement Statement
Extraction & Similarity
SRS Requirement Dependency Graph
STAGE 2

Fig. 2. Research Framework Dependency Graph on SRS.

III. DEPENDENCY METHODS ON SRS

Referring to Figure 3 regarding the method of forming the SRS Requirement Dependency Graph, then in this section, an explanation of each stage is explained about the research process carried out on a Requirement Statement, consisting of a combination of Functional Requirement Statement and Non-Functional Requirement Statement.

In the extraction process, Text Preprocessing activities are carried out in each Requirement Statement, which consists of Tokenization, Stop Word Removal, and Stemming. For example, in the Statement = Requirements Engineer can create a new project, then the result becomes {Requirement; engineer}, {can; create}, {new; project}. This extraction is based on the formula for a triplet with the criteria {actor / system, action, object}.

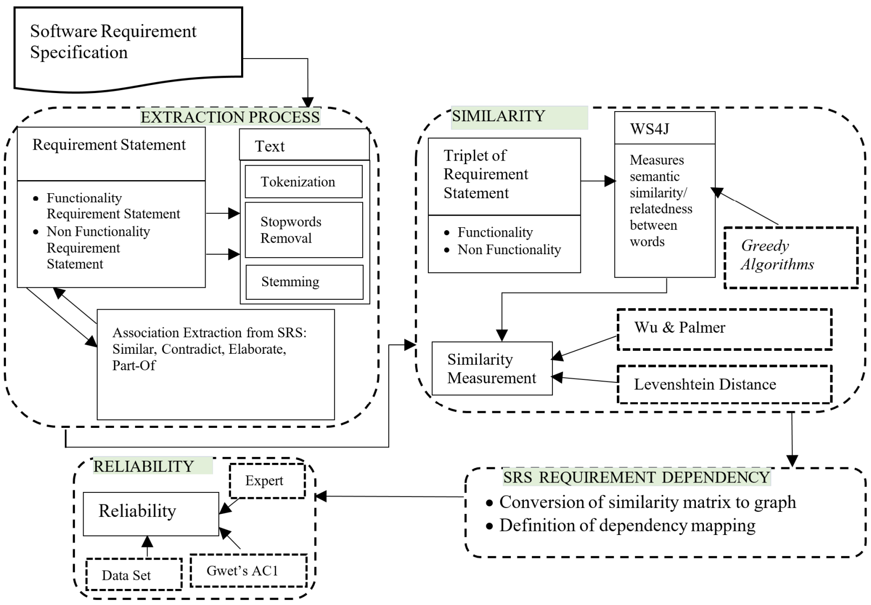

Software Requirement Specification
EXTRACTION PROCESS
Requirement Statement
• Functionality Requirement Statement
• Non Functionality Requirement Statement
Text
Tokenization
Stopwords Removal
Stemming
Association Extraction from SRS: Similar, Contradict, Elaborate, Part-Of
SIMILARITY
Triplet of Requirement Statement
• Functionality
• Non Functionality
WS4J
Measures semantic similarity/ relatedness between words
Greedy Algorithms
Similarity Measurement
Wu & Palmer
Levenshtein Distance
RELIABILITY
Reliability
Expert
Data Set
Gwet's AC1
SRS REQUIREMENT DEPENDENCY
• Conversion of similarity matrix to graph
• Definition of dependency mapping

Fig. 3. Dependency Method SRS.

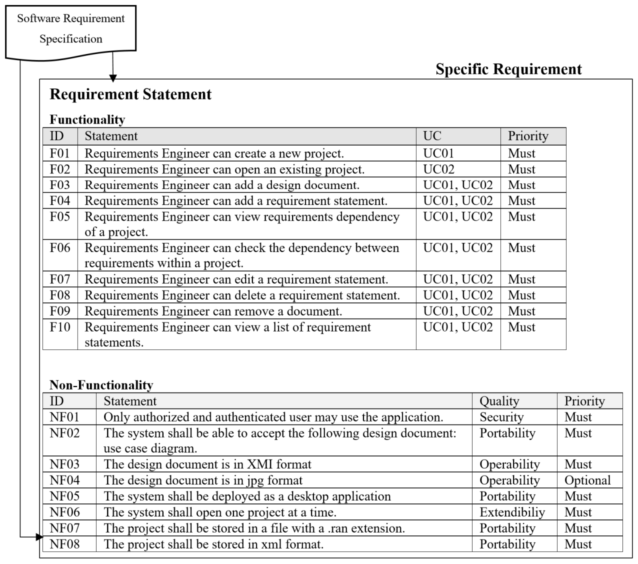
For the process of measuring semantic similarity, the method uses an open-source application called WS4J (WordNet Similarity for Java).

The results of this activity are combined with Greedy Algorithms as an approach to find the optimum value solution. Besides, a method for calculating similarity was used through the practices of Wu Palmer and Levenshtein. Finally, in the Reliability process, use the Gwet's AC1 formula that involves the Expert and Data Set as a reference. [7], [8]. The process illustration is shown in Fig. 3.

IV. ILLUSTRATION OF METHOD IMPLEMENTATION

A. *Mapping the Requirement Statement*

Based on an SRS document, it can be seen that there is a Specific Requirement tabulation which consists of Functional and Non-Functional. However, the next extraction and similarity process is applied equally, namely as a collection of Requirement Statement. Please see Fig. 4. as an illustration.


Software Requirement Specification

Specific Requirement

Requirement Statement

Functionality

ID	Statement	UC	Priority
F01	Requirements Engineer can create a new project.	UC01	Must
F02	Requirements Engineer can open an existing project.	UC02	Must
F03	Requirements Engineer can add a design document.	UC01, UC02	Must
F04	Requirements Engineer can add a requirement statement.	UC01, UC02	Must
F05	Requirements Engineer can view requirements dependency of a project.	UC01, UC02	Must
F06	Requirements Engineer can check the dependency between requirements within a project.	UC01, UC02	Must
F07	Requirements Engineer can edit a requirement statement.	UC01, UC02	Must
F08	Requirements Engineer can delete a requirement statement.	UC01, UC02	Must
F09	Requirements Engineer can remove a document.	UC01, UC02	Must
F10	Requirements Engineer can view a list of requirement statements.	UC01, UC02	Must

Non-Functionality

ID	Statement	Quality	Priority
NF01	Only authorized and authenticated user may use the application.	Security	Must
NF02	The system shall be able to accept the following design document: use case diagram.	Portability	Must
NF03	The design document is in XMI format	Operability	Must
NF04	The design document is in jpg format	Operability	Optional
NF05	The system shall be deployed as a desktop application	Portability	Must
NF06	The system shall open one project at a time.	Extendibiliy	Must
NF07	The project shall be stored in a file with a .ran extension.	Portability	Must
NF08	The project shall be stored in xml format.	Portability	Must

Fig. 4. Mapping Requirement Statement.

B. *Formation of Triplets*

The next step is to form triplet tabulations, whose criteria are divided into Actors, Actions, and Objects. For example, in this illustration, use some of the combined Requirements Statement from Functional and Non-Functional, namely for ID F01 to F05, and ID NF01 to NF04. The results of this grouping will be used in the process of Measurement Semantic Similarity. The formation of this triplet is the same as the sentence structure criteria consisting of Subjects, Verbs, and Objects. For the syntax of writing, it can be observed in Table I.

TABLE I. TRIPLET RS

ID	Triplet of Requirement Statement
F01	{Requirement;engineer},{can;create},{new;project}
F02	{Requirement;engineer},{can;open},{existing;project}
F03	{Requirement;engineer},{can;add},{design;document}
F04	{Requirement;engineer},{can;add},{requirement; statement}
F05	{Requirement;engineer},{can;view},{requirement; dependency;project}
NF01	{authorized; authenticated;user},{may;use},{application}
NF02	{system},{shall be able; accept},{following; design; document; use;case;diagram}
NF03	{design;document},{is},{xmi;format}
NF04	{design;document},{is},{jpg;format}

C. *Perform Semantic Similarity Measurement*

This process uses an open source-based online application called WS4J (WordNet Similarity for Java). It is a tool used for measuring the semantic similarities between two words. It also scales the connection of each word in a sentence as a result of triplet grouping. The results of this calculation tabulation. Then the method finds the optimum value through Greedy Algorithms. For example, given the statements from Table I, the set of requirements-pairs that produce the optimum value are presented in Table II. The selected pairs are marked by colored bars.

TABLE II. PROCESS RESULTS OF WS4J AND GREEDY ALGORITHM

Requirement Statement		NF01					
		authorized	authenticated	user	may	use	Application
F01	Requirement	0	0	0.5455	0	0	0.6667
	Engineer	0	0	0.8889	0	0	0.5333✓
	Can	0	0	0	0	0	0
	Create	0.6667	0	0	0	0.6667 ✓	0
	New	0	0	0	0	0	0
	Project	0	0	0.2667✓	0	0	0.8000

Based on the results of calculations in Table II, it can be observed that there are optimum values, through giving a checklist. This value then will be calculated using the similarity formula of Wu Palmer [9] for example, as follows:

$$\text{Sim } S_{mxn}, = \frac{2\ X\ (\sum_{i=1}^{\min[m][n]} maks\ tokenSim|m_i||n_i|)}{m+n} \qquad (1)$$

$$\text{Sim } S_{NF01xF01} = \frac{2\ X\ (0.2667+0.667+0.5333)}{6+6} = 0.24 \qquad (2)$$

Through the repetition process for the same stages, then, as an illustration, the results of the whole calculation can be examined in Table III. In this table, we use an interpretation of the threshold value from the agreement of experts using the Gwet's AC1 method; for example, the threshold result of this method is ≥ 0.4. For this threshold value, refer to the research of [10].

TABLE III. PROCESS RESULTS OF THE WU-PALMER

ID Code	F01	F02	F03	F04	F055
NF01	0.24	0.33	0.32	0.14	0.18
NF02	0.21	0.41	0.40	0.28	0.10
NF03	0.26	0.27	0.42	0.45	0.29
NF04	0.22	0.25	0.21	0.30	0.21

Threshold >= 0.4 Gwet's AC1

D. *Make Requirements Associations*

Refer to the tabulation results in Table III. Then the visual Dependency is made based on the threshold value, as in Table IV.

TABLE IV. THE WS4J PROCESS AND THE GREEDY ALGORITHM

ID Code	F01	F02	F03	F04	F055
NF01					
NF02		✓	✓		
NF03			✓	✓	
NF04					

In the next step is to design the Requirement Dependency Graph that refers to Table IV. In the table, it can observe that the Requirements Statement on Functionality that has interdependence is F02, F03, and F04. In addition, for the Requirements Statement on Non-Functionality that has mutual dependence is NF02 and NF03.

Based on Table IV, the next process is to design a Requirement Dependency Graph for SRS, by understanding the definitions of [4] to be applied to each Requirement Statement involved in the SRS (see the results of Fig. 5.), as follows:

- F02 = Requirements Engineer can open an existing project.
- F03 = Requirements Engineer can add a design document.
- F04 = Requirements Engineer can add a requirement statement.
- NF02 = The system shall be able to accept the following design document: use case diagram.
- NF03 = The design document is in XMI format

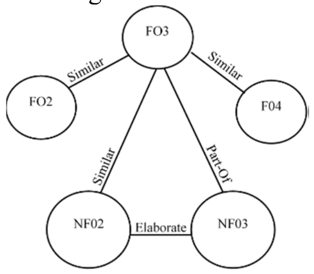


Fig. 5. Requirement Dependency Graph on SRS.

V. ILLUSTRATION OF THE RDG RELIABILITY PROCESS FOR SRS

If traced from examples of ideas to the background of this proposal, it can see that the results of this method are the formation of the Requirement Dependency Graph.

For the reliability process, the resulting graph can be tested and validated by involving Experts to identify the truth of the RDG. [10], [11].

In the process of testing and validating the results of RDG formation for SRS (see Fig.6), it consists of several stages, as follows:

1. Two experts assess a set of Requirements Statement in SRS to determine the type of dependency.
2. Compare the results of the first stage with the types of Dependencies on the RDG results from the proposed method.
3. Tabulate the similarity of answers between the types of dependencies of experts with the type of dependence of the RDG. The formula used at this stage is as follows:

$$\text{Observation Agreement } P = \frac{A+D}{N} \qquad (3)$$

$$\text{Prob.chance-agreement, } e(\gamma) = 2P_1(1-P_1) \qquad (4)$$

$$\text{AC1-statistic, } AC1 = \frac{P-e(\gamma)}{1-e(\gamma)} \qquad (5)$$

TABLE V. SIMILARITY TABULATION ANSWER

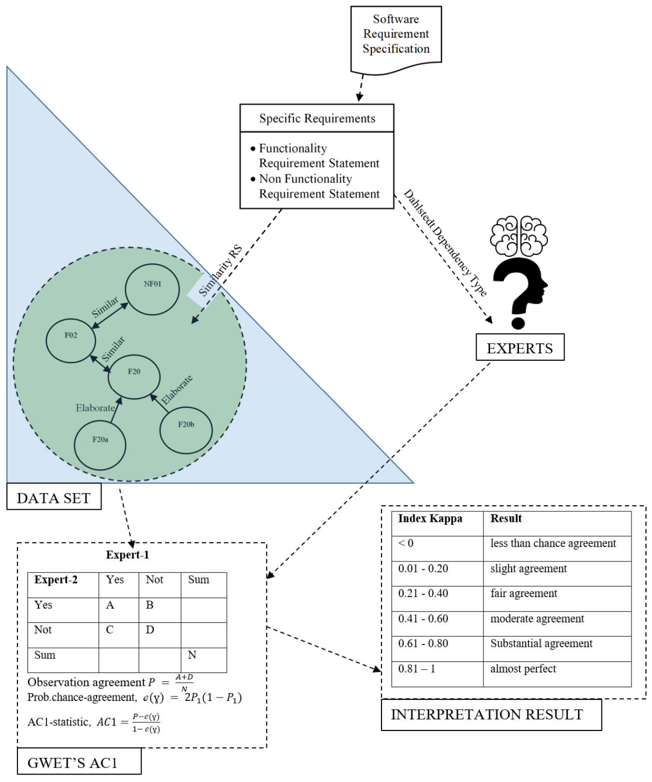
	Expert-1		
Expert-2	Yes	Not	Sum
Yes	A	B	B1=A+B
Not	C	D	B2=C+D
Sum	A1=A+C	A2=B+D	N

4. Interpret the AC1 value that refers to the Kappa index table.

TABLE VI. INTERPRETATION OF AC1 VALUE

Index Kappa	Result
< 0	less than chance agreement
0.01 - 0.20	slight agreement
0.21 - 0.40	fair agreement
0.41 - 0.60	moderate agreement
0.61 - 0.80	Substantial agreement
0.81 – 1	almost perfect

5. Determine the results in an agreement category in the fourth stage table. The range used is 0 to 1. For values 0 = not good, and for values 1 = almost perfect.



Expert-2	Yes	Not	Sum
Yes	A	B	
Not	C	D	
Sum			N

Observation agreement $P = \frac{A+D}{N}$
Prob.chance-agreement, $e(\gamma) = 2P_1(1 - P_1)$
AC1-statistic, $AC1 = \frac{P-e(\gamma)}{1-e(\gamma)}$

Index Kappa	**Result**
< 0	less than chance agreement
0.01 - 0.20	slight agreement
0.21 - 0.40	fair agreement
0.41 - 0.60	moderate agreement
0.61 - 0.80	Substantial agreement
0.81 – 1	almost perfect

Fig. 6. RDG Reliability for SRS.

VI. Result And Analysis

Fig. 5 shows all the requirement statements involved or related given the software requirements specification shown in Fig. 4. The produced RDG includes the following requirement interdependencies:

- F02 Similar F03
- NF02 Similar F03
- NF03 Part-Of F03
- F04 Similar F03
- NF02 Elaborate NF03

The next step is to conduct the RDG Reliability process for all types of dependencies from the Requirement Statement that have produced in this study, by referring to the stages of the reliability process as described above and referring to Fig. 6 as an illustration of the steps.

From the example of this study, which refers to Fig. 5, there is a relationship between the results of the Requirement Dependency process for Software Requirements Specification with the dependency type of Dahlstedt [4], namely in the examples: Similar, Part-Of, and Elaborate, which are the results of extraction methods and similarity carried out in this study. In addition to information, the implementation of the dependency method can be traced from other diagram information, such as UML, DFD / ERD, and Petri-Nets.

VII. Conclusion And Future Work

In this study, a Requirement Dependency Graph modeling for Software Requirements Specification has been produced through a process carried out on a Requirement Statement, consisting of a combination of Functionality and Non Functionality. There is an extraction process for Text Preprocessing, which includes of Tokenization, Stopword Removal, and Stemming. Besides, there is a process of measuring semantic similarity through WS4J (WordNet Similarity for Java). The results of the extraction process, combined with Greedy Algorithms as the optimum value solution approach. Besides, a method for calculating similarity was used through the practices of Wu Palmer and Levenshtein. At the end of this process, Reliability is performed using the Gwet's AC1 formula.

The contributions of this study are as follow:

1) Develop a graph-based needs dependency model,
2) Build a graph extraction method that depends on the requirements of the software specification,
3) Integrating dependency graphs for extraction needs from design artifacts.

The next steps with respect to the aforementioned bigger project are developing methods for extracting requirement dependencies and modeling RDGS from other UML artifacts, such as Use Case Diagram, Context Diagrams, and Sequence Diagrams. Given these RDGs, our future research also focuses on developing a mechanism to integrate the RDG.

Acknowledgment

This work was supported by Institut Teknologi Sepuluh Nopember and UNIVERSITAS TELKOM.

References

[1] P. Zave, "Classification of research efforts in requirements engineering," ACM Comput. Surv., vol. 29, no. 4, pp. 315–321, 1997.

[2] Å. G. Dahlstedt and A. Persson, "Requirements interdependencies: State of the art and future challenges," in Engineering and Managing Software Requirements, 2005, pp. 95–116.

[3] S. Soomro, A. H. B, and A. Shaikh, "Ontology Based Requirement Interdependency Representation and Visualization," Springer Int. Publ. Switz. 2014, no. August, 2014.

[4] Å. G. Dahlstedt, "Requirements Interdependencies – a Research Framework," no. July, 2001.

[5] W. Chen, J. Kazama, K. Uchimoto, and K. Torisawa, "Exploiting subtrees in auto-parsed data to improve dependency parsing," Comput. Intell., vol. 28, no. 3, pp. 426–451, 2012.

[6] S. Jayatilleke, R. Lai, and K. Reed, "A method of requirements change analysis," Requir. Eng., vol. 23, no. 4, pp. 493–508, 2018.

[7] K. Gwet, "Kappa Statistic is not Satisfactory for Assessing the Extent of Agreement Between Raters," Stat. Methods Inter-Rater Reliab. Assess., vol. No. 1, Apr, pp. 1–5.

[8] N. Wongpakaran, T. Wongpakaran, D. Wedding, and K. L. Gwet, "A comparison of Cohen's Kappa and Gwet's AC1 when calculating inter-rater reliability coefficients: A study conducted with personality disorder samples," BMC Med. Res. Methodol., vol. 13, no. 1, pp. 1–7, 2013.

[9] M. Shenoy, "A New Similarity measure for taxonomy based on edge counting," Int. J. Web Semant. Technol., vol. 3, no. 4, pp. 23–30, 2012.

[10] H. Samosir and D. O. Siahaan, "Generating Requirement Dependency Graph Based on Class Dependency," 4th Int. Semin. Sci. Technol., vol. 29, no. 2, pp. 1–8, 2018.

[11] P. Gelu, R. Sarno, and D. Siahaan, "Requirements Association Extraction based on Use Cases Diagram," Lontar Komput. J. Ilm. Teknol. Inf., vol. 9, no. 1, pp. 11–19, 2018.