Actor and Use Case Extraction from Text-Based Requirement Specification

Elisa Margareth Sibarani, Andry Hutagaol, Deppy Simarmata, and Juniaty Manihuruk

***Abstract*—**Initial stage in software development lifecycle is requirements gathering steps which results a text-based requirements specification document. Common problem in analyzing those requirements is the difficulty of identifying the functionality of the software, especially when there are so many descriptions elicited from the gathering activities. The objective of this research is analyzing the process to identify actor and use case of the system to be built. Since the requirements were described in the form of natural language, therefore it needs natural language processing. The proposed solution is simplification form by only doing the syntax analysis to provide a quick response. The tool is built in Java by accept active sentences in English. The evaluation result proves that the tool is able to list actor and use cases from a given system description. Possible future work is able to process passive sentence with complex structure not only in the Subject-Predicate-Object form.

***Keywords*—**Software requirements, actor, use case, syntax analysis.

I. Introduction

SOFTWARE engineering is a systematic and discipline approach that aims for software-production-oriented by implementing engineering methods to get reliable and real-world software which will work on real environment [1]. Steps defined in software engineering consists of series of activities, namely software project planning, requirements gathering, software design, code generation, software testing, and software maintenance [1].

Requirements gathering are the process of identifying stakeholders and user needs which are documented in a text-based requirement specification document. This document will be used as a reference during review process with end user so that the final specifications for the software are obtained, thus act as reference for implementation used by software development team [2]. However, existing problems in recent years led to inability of the software developers produce a software requirement specification document which consistent and appropriate to the user needs [3]. Main reason of this lack is due to many sources are available to help identify those needs yet the time is limited in a software project and also the perception among developers somehow invest to increase the difficulties to list customer's needs. Therefore, one fine way to help developers identify the right needs of the user is by using a tool which is able to identify the needs of users by given a software requirement specification document.

Some research has been done related to build an automatic tool to obtain qualified user requirements specification which is represented by essential use case so that the user needs can be better identified [4]. Essential Use case illustrates the narrative structure of the application in a language that is simple and readable by the user, general, concise, but also descriptive [4]. Other research was done by constructing a tool based on natural language which aims to assist in the analysis phase of software development with object-oriented framework [5]. The third research is developing software requirements analysis method based on domain ontology that mapping between software requirements specification and the domain ontology which act as the semantic component [6]. Domain ontology is a set of requirements for a particular domain. The fourth conducted research on a tool to analyze natural language requirement automatically [3].

From the above conducted research, to conduct a needs analysis process automatically turns more complex because the input to be processed is a natural language. Therefore this research proposes a simpler approach in terms of processing natural language requirements, known as lightweight semantic processing of natural language processing [4]. In addition, it needs a dictionary or a corpus as a component of tool consisting of a collection of words that are needed to process natural language processing and it depends on the needs of users and the number of application domains that are covered.
The approach of the research mainly by doing literature study and analysis, and the final step will cover the manual process for analyzing the requirements specification by involving the user. This process is called verification requirements which will be done by formal and informal reviews to ensure that the requirements are obtained according to the customer needs [7].

This paper presents the result from the work so far in extracting the functionality of software which helps the system analyst to summarize requirements from lots of sources of requirement specification document. The rest of the paper is organized as follow: Section 2 will address the literature study on requirement engineering, use case, and natural language processing. Section 3 describes the analysis result that we proposed to do actor and use case extraction given a text-based

Elisa Margareth Sibarani is with Del Institute of Technology, North Sumatra, 22381 Indonesia (e-mail:elisa@del.ac.id, elisamargareth.sibarani @gmail.com).

Andry Hutagaol, Deppy Simarmata, Juniaty Manihuruk[2,3,4], is with the Informatics Engineering Study Program, Del Institute of Technology, North Sumatra, 22381 Indonesia.

requirement specification. In Section 4 we will discuss the implementation result and describes some feedback which acts as an evaluation and facts finding from the extraction process. Section 5 concludes our study and briefly discusses the possible future research in the related field.

II. Literature Study

A. Requirement Engineering

Based on the IEEE Standard Glossary of Software Engineering Terminology (1997), a requirement is [8]:

1. A necessary condition for the user to solve the problem or to achieve a goal.
2. A condition that must be processed by a system or system component to satisfy a contract, standard, specification, or other requirements.

Requirement Engineering is the initial phase in software engineering which is called definition phase in which the aims of it is to produces a qualified software by elicit user requirement regarding what services or functionality must be done by the system and how each service performed within certain limits [8]. There are four activities which define requirement engineering [7]: (1) requirement elicitation, activities to gather requirements from users which consist of two techniques which are active, by doing interview, meetings, discussion, observation, or passive by inactively involving users such as identify use cases, workflows, describing use cases scenario, and all of it are done by system analyst, (2) requirement analysis, activities to ascertain whether the requirements which are obtained in accordance with the needs of the user, to review requirements by studying existing documents and forms, (3) requirement specification, activities of making documentation and specifications of a software, by using use cases to determine the behavior or functional capability of the system to be built [8], (4) requirement validation, a series of activities to ensure that any requirement statement is accurate, complete, clear, consistent, and fulfill other qualified requirements characteristics [8]. This activity is done by holding another meeting with the customer to ensure the requirement previously specified are correct and clear.

B. Use Case

As described in the requirements specification activity in requirement engineering, the specification of a software or user requirements are documented using use case. The general objective of the use case is to describe the interaction between system events with actors. The following describes another definition of a use case:

1. Use case is a description of the sequence of events that describe actors using a system to complete the process [9].
2. Use case illustrates a series of interactions between internal systems and external systems [10].

In the early stages of system development, use cases can help to show the behavior of the system, obtaining requirements, determine specifications, and help communication with the user [10]. The advantage of using use case is to determine the behavior system to be built but also document the functional requirements of the system which can help ensure that the systems being developed are in accordance with the needs of the user [11]. The way to represent an use case are many, and one of them that will be used in this research is called essential use case. Essential use cases illustrate the narrative structure and language of the requirement statement for an application from the user. The statement is simple, general, concise, descriptive, and easily defined [4]. Essential use cases show the user a complete view of the role of the system, which describes the purpose of the interaction [11].

C. Natural Language Processing

Natural language processing is the process of creating a computational model of a language that allows the interaction between humans and computer by means of natural language used by humans [12]. Natural language processing aims to make computers understand the meaning or intent of a human being so as to provide an appropriate response as humans can understand other human beings [13]. Natural language processing is not aiming to transform the language that is accepted in the form of voice into digital data but to understand the meaning of a given utterance in natural language and provide an appropriate response, for example by performing a certain action or display specific data [14].

The natural language processing must be supported by a dictionary or vocabulary to get a good result. The more complete a vocabulary is, just like humans if the vocabulary they have is complete, the better the human or the system in terms of communicating one another. Vocabulary in a computer called the lexicon, and an important factor to consider is storage, because the general lexicon has a very large size. Therefore one solution is by only keep the root form of the words, whereas other forms of derivatives can be obtained by applying morphology analysis which is the process of changing the form of words [14].

There are several general steps in natural language processing which are described as follows [14]:

1. Syntax Analysis. The input in this stage is natural language sentences in English. The syntax analysis process will check the suitability of the structure of sentences based on a particular grammar and lexicon. The results of this process are the words that have been parsed from the input sentence. Syntax analysis is done by 3 methods, namely: (a) Top-down parsing, works by decomposing a constituent sentences ranging from the largest to the smallest constituent. Constituent is an elements of a sentence that can stand alone for example noun phrase, verb phrase, and so on. This is done continuously until all the components are produced is the smallest constituent of the sentence, that is the word, (b) Bottom-up parsing, works by taking one by one of the word of the sentence, to be assembled into a larger constituent. This is done continuously until the constituents form sentence. Thus the bottom-up method

works the other way round from top-down.

2. Semantic Interpretation, is the process of translating a sentence into a logical form or forms of representation in general regardless of the context. The aim is to represent the meaning of a sentence which is context-independent for further purposes. Input to this process is the result of the parse tree while parsing the output representation is appropriate in the sense that it can be processed and understood by the computer. General representation used is a logical form, which is the form of meaning representation and also context-independent by using the laws of logic.
3. Contextual Interpretation, is the process that translate a logical form into other form of representation so-called knowledge representation or final meaning representation and aims to represent the meaning in a context-dependent and determine the intent of the use of the phrase.

D.Related Work

In research which was conducted by Vinay S et al [5], the aims is to build a tool called R-TOOL based on natural language which aims to assist in the analysis phase of software development with object-oriented framework. The tool analyzes the text-based requirement in English to generate the actors, use cases, classes, attributes, methods and relationships between classes which were found that could help making the class diagram. Application of R-TOOL implements five (5) process to generate use case reports, classes, attributes, methods, and relationships which are:

1. Tokenizer, is the process of breaking a sentence into tokens, to separate between words, and identify the word index.
2. Pronoun resolver. The existence of a pronoun makes the difficulty in identifying actors and use cases. This ambiguity is solved by reading the input to seek and replace it with a noun pronoun or subject to the previous sentence.
3. Identify actors and use cases. Each noun which is contained in the input document become candidate for actor and are filtered based on the number of frequency of occurrence of the noun. Verb associated with the actor would be a candidate use case.
4. Identifying responsibilities and generating use case report. When use cases are identified, responsibilities and use case description is determined by using a keyword search in the input document. All use cases which are identified along with the responsibilities will be built in the form of a use case report.
5. Classifier, is the determination of the class. Each class is identified by filtering noun based on the frequency of occurrence in the input text. The bigger frequency is, the most likely to become a class.

R-TOOL application has several limitations, first the input should be in the form of active sentences and second, for the compound sentences should be separated into two sentences. While the advantages are [5]:

1. Help identify requirements inconsistency between the manual approaches to the automatic approach to ensure system requirements are identified correctly.
2. Reusability is guaranteed to keep the repository classes from different projects. Users can search for a particular class and customize the class that is in the repository according to user needs.

III. Analysis Result

Based on research in [5], there are several processes which called tokenization, pronoun resolver, an actor and a use case identification, identification of responsibility and generating use case report, as well as classifier. However, on this research will not apply all of those processes but add classifier stop words removal process and identification of complex sentences. Applications built will be able extract actors and use cases from text -based requirements. Application receives input in the form of requirements specification documents in English and active form. Inputs processed by the application to generate an actor, use case, responsibility, and a list of use case report. Applications may receive a response from the user during the process of identifying the use case, the user can respond by selecting the candidate actor from a given list of actors related to the application. Users may choose to prevent mismatches actor use case reports that will be generated by the user needs. There are several major processes that will be implemented to be able to generate a list of actors and use cases from a requirement specification document which can be seen in the flowchart of the software systems in general in Fig. 1.

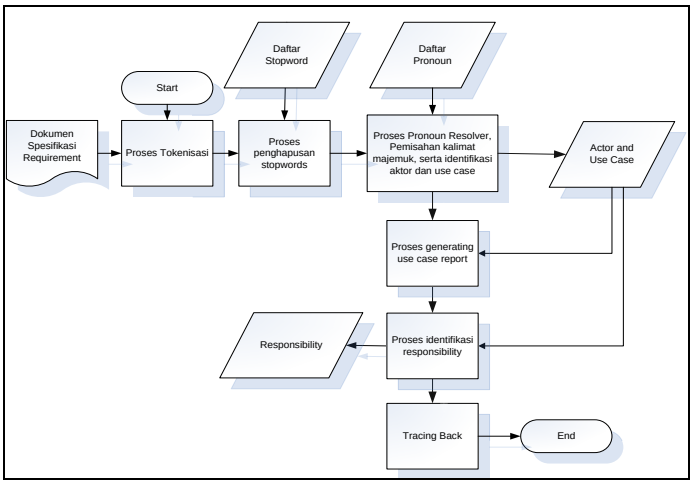

Fig. 1 Flowchart of the actor and use case extraction system

The detailed description of each process will be discussed as follows:

1. Initial process is called tokenization, is the process of parsing a text based on space between words and then provides an index number to each word. The process uses an array, stack and hashmap. Array used to hold temporary words and phrases that have been described. Hashmap is a quick and easy way to represent a hash table

which is a data structure that has a key and value [15]. Two important methods in Hashmap are: (1) the get() method to return the value to the specified key in the hashmap and (2) the put() method to connect certain values with a certain key on the map [15]. Hashmap used to store data and to provide an index and read the value that each stack is used to have an index that can help in identifying some of the stack which is accommodated. Stack is used to store the value in the hashmap and saving the memory storage but also dynamically data structure.

2. Stop words elimination process, is the process of removing all non-essential words such as a, the, where, when, and others. Stop words list can be found in [16]; important notes in the process is the word "and", "I", "he", "she", and "you" are not removed because the word is used in further process of this application. Stop words has no influence in identifying use cases and actors on natural language textual requirements. This process is done to reduce the workload of the system that is to do word searching.
3. Pronoun resolver, identification of the actor and use case, separation of complex sentences into single sentences. The pronoun which will be processed by the system is the singular subject pronoun: I, you, she, he, and it [17]. Actors were identified from the first word of each sentence which is not categorized as a pronoun. Use case is taken from the words after the first word of each sentence. The limitation is if there is more than one word, then the process will not be able identify whether as an actor nor use case. Example for sentence ‘Bank manager makes withdrawal’. The process will identify ‘Bank’ as an actor and ‘manager makes’ as a use case. These results are less precise because there should be a bank manager as the actor and makes withdrawal as a use case. After changing pronouns, the process continues by doing splitting complex sentences into two single sentences. Complex sentence is a sentence consisting of several single sentences joined by conjunctions. In this process, the conjunctive word is 'and' and the sentence that follows it should be in the form of predicate-object (PO). Examples of compound sentence: ‘The operator will verify the cash and enter the total cash on hand after turning the switch to the "on" position’. This sentence will be converted into two sentences which are: ‘The operator will verify the cash’ and ‘the operator will enter the total cash on hand after turning the switch to the "on" position’.
4. Identify use case report, is simply a process to combine actors, use cases, and its responsibility.
5. Identify the responsibility of each use case, is a process that is looking for responsibility and the use case description of each use case. Responsibility use case is obtained by identifying sentences that contain those use case word.
6. To facilitate users to re-examine the list of actors and use cases which are generated according to the requirements specification document, the tracing back process should be carried out by marking the actors and use cases in the requirements specification document so that it can help users to see the original sentence for each actor and use case.

IV. Implementation and Evaluation Result

Based on the analysis which is explained in detail in previous chapter, we develop a tool that implement all steps defined in the analysis which was built in Java Net Beans environment. The tool is an actor-use case extractor which will accept input file in .txt format. Application will then process the input into a set of actors, use cases and also information or responsibility of the use case. Then the results will be compared directly with list of actor and use case which is done manually. The tool is intended to be used by system analyst who will define solution for the new system based on user requirement specification document which is obtained from the customer and facilitate it faster than the manual process. The main interface of the tool can be seen in Fig. 2 below.

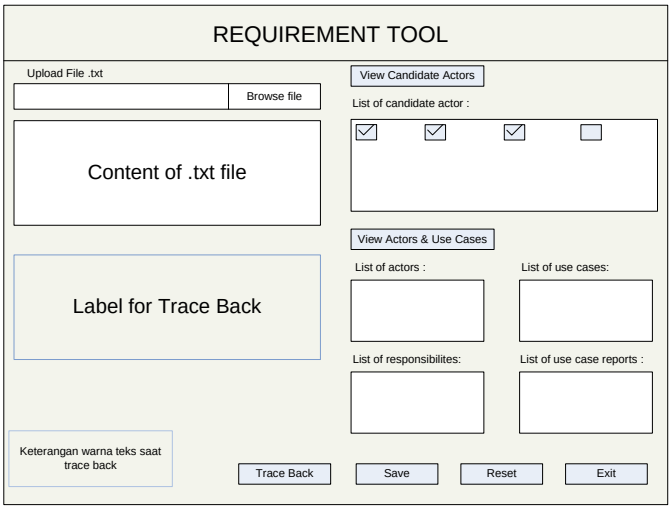

Fig. 2 Interface of Requirement Tool

The specifications of the tool completely are defined as follows:

1. Input of the tool is requirement specification document in .txt format written in English language in active sentence with simple SPO (Subject-Predicate-Object) form and complex sentence with conjunction ‘and’.
2. The tool displays list of actors and use cases and responsibility based on user-selected actor from the list of actors, generate use cases reports and perform tracing back by distinguished actor or use case by its color in case if it is different with the results. Finally, the tool will store the use case report in .txt file into a directory of computer users.

Evaluation was made with the objective to explore the performance of the tool in terms of accurateness and completeness in extracting actor and use case for a specific requirement specification by compared it to the list of actor

and use case for the same requirement but was identified manually. The evaluation process used requirements specification document for Automatic Teller Machine (ATM) System [18], and was divided into three (3) files, in which first file contains 10 sentences, the second file contains 20 sentences, and the third file contains 30 sentences. We made some changes without change the meaning of the sentences, such as: (1) changing the passive sentences into active sentences, (2) simplify the sentence structure in order to meet the Subject-Predicate-Object-detailed description. Each file consists sentence has stop words, pronouns, and conjunctions 'and'. The summary result which was processed by the tool for the third file is shown in TABLE I.

TABLE I
SUMMARY RESULT OF THIRD FILE

Result	Done manually	Done by the tool	
Candidates for actors	-		*1. Software*
			2. ATM
			3. Key-operated
		√	*4. Customer*
		√	*5. Bank*
			6. Machine
		√	*7. Operator*
Actor	*1. Customer*	*1. customer*	
	2. Bank	*2. bank*	
	3. Operator	*3. operator*	
Use Case	*1. Start the system*	*1. stop machine.*	
	2. Stop the system	*2. start machine.*	
	3. Make withdrawal transaction	*3. enter total.*	
		4. verify cash.	
	4. Make deposit transaction	*5. enter pin.*	
	5. Make transfer transaction	*6. insert atm.*	
	6. Make inquiry transaction	*7. perform transactions.*	
	7. Determine invalid PIN	*8. withdraw money.*	
	8. Manage withdrawal transaction	*9. add deposit.*	
		10. enter deposit.	
	9. Manage deposit transaction	*11. enter subject.*	
		12. transfer money.	
	10. Manage transfer transaction	*13. view balance.*	
		14. abort transaction.	
	11. Manage inquiry transaction	*15. require reenter.*	
		16. obtain approval.	
		17. consider transaction	
Use Case Report	*1. Operator start the system*	*1. operator stop machine.*	
	2. Operator stop the system	*2. operator start machine.*	
	3. Customer make withdrawal transaction	*3. operator enter total.*	
		4. operator verify cash.	
	4. Customer make deposit transaction	*5. customer enter pin.*	
		6. customer insert atm.	
	5. Customer make transfer transaction	*7. customer perform transactions.*	
	6. Customer make inquiry transaction	*8. customer withdraw money.*	
	7. Bank determine invalid PIN	*9. customer add deposit.*	
		10. customer enter deposit.	
	8. Bank manage withdrawal transaction	*11. customer enter subject.*	
		12. customer transfer money.	
	9. Bank manage deposit transaction	*13. customer view balance.*	
		14. customer abort transaction.	
	10. Bank manage transfer transaction	*15. customer require reenter.*	
		16. bank obtain approval.	
	11. Bank manage inquiry transaction	*17. bank consider transaction.*	

From the results of executing the tool for three files contain requirement specification for ATM System, in general the tool results more use case report than list of use case report which was identified manually. Detailed discussion of the evaluation results are:

1. Some candidates actors such as ‘software’, ‘log’, and ‘machine’ was identified by the tool, but in manual those nouns written as subject in the sentence is not the actor because it does not perform interaction with the system. This was occurred because the tool identifies actor candidates based on first word of each sentence. However it can be overcome by the opportunity given to the user to be able to select the appropriate actors based on the candidate list of actors who have been identified by the tool.
2. The tool identifies more use case than the list resulted from manually process because the tool identifies from each sentences according to the user-selected actor.
3. For some use cases that have adverbs more than one word such as use case ‘reading an ATM card’ was identified as ‘reading atm’. This was happened because the tool only identifies two words after the actor.
4. Trace back is added to allow users to check the results of the tool given a requirements specification document.
5. Natural language processing which was implemented by the tool is still very simple therefore incorrectness of the result may arise. The process only able for sentence structures in the form of SPO (Subject-Predicate-Object) with the subject is actors that interact with the system. However, not all sentences has subject who can be categorized as actors who interact with the system. Take as an example for sentence ‘The ATM will provide the customer with a printed receipt for each successful transaction, showing the date, time, machine location, type of transaction, account(s), amount, and ending and available balance(s) of the affected account ("to" account for transfers)’. The subject of the sentence is the ATM, but it is not the actor who was supposed to interact with the system.
6. There are several human-mistaken during conversion of sentences into SPO (Subject-Predicate-Object) sentence structure. For example the sentence ‘The ATM will provide the customer with a printed receipt for each successful transaction, showing the date, time, machine location, type of transaction, account(s), amount, and ending and available balance(s) of the affected account ("to" account for transfers)’, was changed into sentence ‘The ATM will provide a printed receipt for each successful transaction, showing the date, time, machine location, type of transaction, accounts, amount, with ending and available balances of the affected account by customer’. This resulted in an error on the results of the tool, because ‘Customer’ is supposed to be an actor and ‘print receipts’ as a use case. However the tool generated

‘ATM’ as an actor and ‘printed receipts’ as a use case.

V. Conclusion

From the evaluation result which is already explained in detail, Natural Language Processing (NLP) which was implemented is called lightweight process with only applies syntax analysis that is the parsing process.

Many study and research have been made in identifying software requirements. In this research, the requirement identification is done by applying several steps of R-TOOL [5] without identify the classes, attributes, methods, and relationships but only determine the actor and use case, responsibility, and generating use case report. In addition, we add stop words removal process, the separation of complex sentences with conjunctions ‘and’, and trace back process. Also, identify the actor based on the subject of each sentence and allows user to give feedback to the system if in case list of candidate of actors are not appropriate yet according to the user.

The process which was defined in this research is highly depend on how good the sentence is written and also the quality of the requirement specification document which should be complete but also clear and not redundantly written. As a consequence, the tool is not accurate in determining the use-case of a sentence because all Predicate-Object of a sentence used as use case.

Some future improvements are strongly suggested to be done such as: (1) sentences that can be processed by the tool should not only in active sentence but also in passive sentences and not limited to the structure of SPO so that the identification of actors not identified by the subject only, (2) the extraction process should be carried out with a corpus as a dictionary which will facilitate the process. Corpus will help identify the actors and use cases correctly and accurately.

References

[1] Roger S. Pressman, “Software Engineering, A Practitioner’s Approach, Fifth Edition”. McGraw-Hill, 2007.

[2] Bashar Nuseibeh, Steve Easterbrook, “Requirements Engineering: A Roadmap”. In Proceedings of the Conference on The Future of Software Engineering, pp. 35-46. ICSE, 2000.

[3] Giuseppe Lami, Stefani Gnesi, Fabrizio Fabbrini, Mario Fusani, Gianluca Trentanni.”An Automatic Tool for the Analysis of Natural Language Requirements”. CSSE Journal vol. 20 (1), CRL Publishing: Leicester, Jan. 2005, pp. 53-62.

[4] Massila Kamalrudin, John Grundy, John Hosking, “Tool Support for Essential Use Cases to Better Capture Software Requirements”. In Proceedings of the IEEE/ACM International Conference on Automated Software Engineering, pp. 255-264. ACM, 2010. http://dx.doi.org/10.1145/1858996.1859047

[5] S,Vinay, Shridar Aithal, Prashanth Desai, “An Approach towards Automation of Requirements Analysis”. In Proceedings of the 2009 International Multi Conference of Engineers and Computer Scientists, March 18-20; Hong Kong, China.

[6] Kaiya Haruhiko, Saeki Motoshi, “Ontology Based Requirements Analysis: Lightweight Semantic Processing Approach”. In Proceeding of Fifth International Conference on Quality Software (QSIC), pp. 223-230 September 2005.

[7] IEEE (1960), “Certified Software Associate Development”. http://www.computer.org/portal/web/certification/csda/.

[8] Karl E Wiegers. “Software Requirements”. Microsoft Press, Redmond, Wash., 1999.

[9] Information and Technology Management Branch. “Essential use cases modeling standards and guidelines version 1.0”. 2005

[10] EwanTempero, Robert Biddle, James Noble, “Essential Use Cases and Responsibility in Object-Oriented Development”. In Proceedings of the Australasian Computer Science Conference. 2002.

[11] Dewi Mirnasari,”Pengembangan use case scenario berdasarkan deskripsi tekstual use case menggunakan pengenalan entitas bernama”. Bachelor Thesis in Faculty of Computer Science University of Indonesia. May 2005.

[12] Ali Ridho Barakbah, “Natural language processing”. http://lecturer.eepis-its.edu/~entin/Kecerdasan%20Buatan/Presentasi/Minggu6%20-%20Natural%20Language%20Processing.pdf

[13] Henry C. Mishkoff, “Understanding Artificial Intelligence”. Texas Instruments, Dallas, 1985.

[14] James Suciadi, “Studi Analisis Metode–Metode Parsing dan Interpretasi Semantik pada Natural Language Processing”. JURNAL INFORMATIKA Vol 2 No 1, Mei 2001: pp. 13–22.

[15] “ANY Example”. 2010. http://www.anyexample.com/programming/java/java_hashmap_example.xml

[16] Clark Goble (2006). “Onix Text Retrieval Toolkit API Reference”, http://www.lextek.com/manuals/onix/stopwords1.html

[17] Elizabeth (2009). “Grammar Evolution”. http://www.english-grammar-revolution.com/list-of-pronouns.html

[18] Russell C. Bjork (1996). “ATM Simulation Links – by Topic”. http://math-cs.gordon.edu/courses/cs211/ATMExample/