applied sciences

MDPI

Article

Effort Estimation Approach through Extracting Use Cases via Informal Requirement Specifications

Bo Kyung Park and R. Young Chul Kim *

SE Lab., Department of Software and Communication Engineering, Hongik University, Sejong 30016, Korea; parkse@hongik.ac.kr

* Correspondence: bob@hongik.ac.kr; Tel.: +82-44-860-2477

Received: 28 March 2020; Accepted: 24 April 2020; Published: 27 April 2020

Abstract: Sometimes unclearly describing the requirement specifications of satisfied customer's needs, means it may be difficult to develop the production of high-quality software systems. A persistent issue of requirement engineering is how to clearly understand the requirements of the large and complex software project, and also how to analyze them exactly. To solve this problem, we propose a linguistic analysis method based on the semantic analysis of the Fillmore's textual approach. This method extracts use-cases from informal requirement specifications. For applied requirement engineering with this method, we suggest extracting a use-case diagram, as well as calculating the software effort estimation with the original use-case point (UCP). To simply explanations of our use-case extraction method, we use one example of a simple postal information system.

Keywords: linguist Fillmore's textual analysis; use-case point (UCP), natural language oriented informal requirement specification; case grammar

1. Introduction

At the present time, current software is becoming increasingly more significant and complicated. For high-quality software, we must accurately analyze the natural language-oriented requirement specifications at the beginning of software development. In requirement engineering, use-case approaches are increasingly attracting attention in requirements engineering because the user-centered concept is valuable in eliciting analyzing requirements [1]. Use cases describe the behavior of system as seen from an actor's point of view [2]. One of the important things is how to correctly analyze requirement specifications to prevent potential errors in requirements and to reduce development and maintenance costs [3]. However, in real software projects, most of the informal requirement specification documents are written, so it is difficult to analyze informal requirement specifications with only linguistic, textual analysis.

In software-industrial fields in Korea, some natural language-oriented requirement analysis methods have been mentioned. Ahn [4] mentioned a method of object extraction and modeling from the user requirements with Fillmore's textual approach, which identifies user requirements to extract objects and object modeling through a scenario-based analysis. Anton [5] mentioned a method of requirement extraction with goal-based requirement analysis but just mentioned a requirement extraction method. J. Kim [6] also mentioned his extraction methods with goal and scenario concepts, which generate use-case descriptions based on the scenario-based analysis. His approach had a problem requiring a change of requirements that are tailored in a defined scenario. Most of the requirement analysis and extraction methods use the concept of goal and scenarios.

The primary purpose of these studies suggested to identify and analyze requirements theoretically because of analyzing no systematic way in existing use-case approaches to handle non/functional requirements [1]. Their focused researches are the use-case approach that shows the interaction between

an actor and a system based on a user-centric analysis, which easily classify the high-level functionalities of the system. Our previous use-case approach is difficult to analyze the impact between use-cases [4,5]. When using descriptive requirements, this approach makes use-case extraction impossible because of its ambiguous expressions and is not clear with the requirements analysis criteria [6]. In natural language analysis fields, some researched focused on developing formal grammars and parsers in order to perform analysis of the natural language sentence. Selijan [7–9] mentioned to find rules with lexical-functional grammar and case grammar to include semantic roles to represent the natural language sentence. Ye [10] mentioned to apply foreign language teaching and vocabulary teaching at new student with Fillmore's case theory. But they never mention about how to apply requirement engineering with linguistic analysis.

To solve such a problem, we propose a use-case extraction method from natural language-oriented informal requirement specifications based on semantic analysis of Fillmore's mechanism. The suggested method identifies key verbs in requirement sentences and extracts arguments of the verb in a sentence. Then it can extract the use-case through constructing a recursive visual modeling mechanism until no more slightly connected. To make it easier and more accurate to extract use-cases, we have adopted Fillmore's Case Grammar and also have applied with Cockburn's Goal Use Case technique on Requirement Analysis to identify the goals of a system from an effective requirement identification approach. He mentioned "It is the goal the primary actor has in trying to get work done or the user has in using the system" [11]. The approach looks at the interactions of a single category of users at a time, considerably reducing the complexity of requirements determination without supporting use-case formalization [11,12].

Our use-case extraction process consists of five steps: requirement analysis, sentence analysis, verb-noun file list, visual modeling and use-case extraction.

This study is organized in the following sections: Section 2 mentions the related materials and methods for our use-case extraction process with informal requirement specification. Section 3 describes the result of the case studies. The last section presents the conclusions and future scope.

2. Materials and Methods

2.1. Materials

2.1.1. The Original Fillmore's Case Grammar

Linguistics [5,6,13] is the scientific study of human language that systematically and deeply understands language through scientific analysis and inference about the components of language. Linguists Chomsky and Fillmore proposed a systematic analysis of pure human language. Chomsky took a scientific approach to human language. His theory only considered surface relationships, which was not suitable for requirement engineering because its in-depth meaning is not considered. On the other hand, Fillmore argued a case grammar mechanism to describe a deeper language structure than Chomsky's theory. His theory described sentences with semantic roles instead of structural cases [13,14]. That is, there is a maintenance of a semantic relationship between nouns with a center of a predicate (verb) within a sentence. This relationship was classified with arguments (nouns) into categories. In 1968, Fillmore proposed six cases as agent, instrumental, dative, objective, factitive and locative. The original case grammar mechanism can classify noun phrases related to verbs into case categories:

(1) An Agent is a subject that is perceived to cause an action represented by a verb.
(2) An Instrumental is an object that becomes the cause of an action or a state which a verb represents.
(3) A Dative means a person or an animal affected by a state or an action represented by a verb.
(4) An Objective is an object that is affected by an action or a state which a verb expresses.
(5) A 'Factitive' means a person or an animal that exists as a result of an action and a condition of a verb.

(6) A Locative refers to a state that a verb represents or where an action occurs.

Since 1971, the case grammar has been added to and modified by other theorists. Table 1 shows the definitions of the cases proposed by Fillmore & other theorists.

Table 1. Original Fillmore's Case [15].

Case	Definition
Agent	A person or entity causing a verb's action to be performed.
Counter agent	The force or resistance against which a verb's action is carried.
Object	An entity affected directly by a transitive verb's action.
Experiencer	A person or thing affected by a verb's action, replacing of the dative.
Source	The place from which something moves.
Goal	The place to which something moves.
Locative	Location or spatial orientation of the state or action.
Instrument	The inanimate entity causally involved in a verb's action.
Time	The time of the event.

2.1.2. The Existing Requirement Analysis Method

Requirement engineering is the most important area in software engineering research and practice [16]. Requirement analysts interview customers about the system process, collect and document data. However, they are difficult to analyze requirements written in natural language. The problems of requirement analysis are as follows: (1) communication between developers and customers, (2) frequent changes in requirements and (3) difficulty in a natural language-oriented informal requirement specification. The communication problem and informal specification are caused to understand the system to be developed differently.

Frequent requirement changes may result from an inaccurate representation of the user requirements of a system. In order to solve such problems, many researchers have proposed exact requirement analysis methods.

Kim [17] proposed an integrated requirement analysis method based on Goal and scenarios. This method effectively integrates data collection and other techniques for requirements analysis. The problem of this approach classifies and consolidates requirements according to given scenarios.

EM Sibarani [18] suggested a method for extracting actors and use-cases from text-based requirement specifications. It analyzes natural language-oriented requirements through parsing with an automated tool. Its problems are as follows: (1) requirements must be changed to a defined format (SPO: Subject-Predicate-Object), (2) a complex sentence has to be replaced with a simple one, (3) verbal analysis can analyze only active sentences.

R. Abbott [19] also used natural language-based textual analysis. He proposed an empirical knowledge method to identify objects, attributes and relevance from the requirement specification. It maps linguistic components to model elements [5]. Its advantage is that natural language analysis can focus on the user's terms. However, the natural language does not distinguish meanings accurately. Thus, natural language-based object models are incorrectly derived. To solve this problem, developers need to identify and standardize objects and terms, and then, have to clarify different representations of requirement specifications [4].

2.2. Methods for Adapting Requirement Engineering with Textual Approach

2.2.1. Our Refined Fillmore's Case Mechanism

Our proposed case mechanism has refined the semantic concepts of the original case grammar and has been applied to the requirement engineering [20,21]. The case grammar modeling procedure can be used to extract use-cases from natural language-oriented informal requirements, which is a top-down method. In other words, it will identify from use-case identification in a larger unit to objects

and functions in a smaller unit. Through refined methods, we expect to analyze the high level of requirements, which will develop high-quality software.

Figure 1 shows the refined Fillmore's Case Mechanism on how to adapt UML with the case mechanism for use-case extraction. This method analyzes semantic relationships between words based on the main verbs. Therefore, processing natural language as requirement specifications may be more easily handled. Our refined case mechanism consists of eight cases: actor, secondary actor, object, source, theme, instrument and goal [15,20].

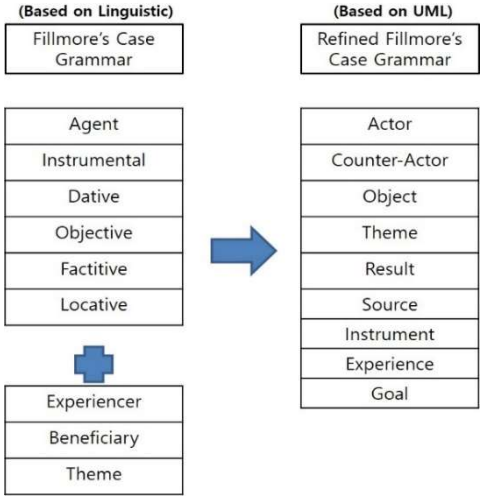

Figure 1. Refined Fillmore's case mechanism.

We redefine the existing *agent* case as an actor in the use-case diagram like changing agent to actor (A). The existing *dative* case has a similar role as counter actor. We change dative to counter actor (CA). The existing object and experiencer cases are changed to object (O). Source and goal cases represent a location or a place of moving something. We integrate the two cases into *Source* (S). *Instrumental* and *locative* cases are integrated into the instrument (I). In this study, we add 'theme object (TO)' of case Grammar in 1971. In addition, the goal (G) is added, which indicates system goal to achieve the needs of users of a system. This is different from the existing goal ("The destination of a location or a place where something moves").

Table 2 lists the categories of refined cases. They consist of a role, notation and meaning of arguments which are associated with each verb. These cases can analyze all requirement sentences by arguments with the categories of cases.

Table 2. Refined definition and notation of the Fillmore's cases [20,21].

Case	Notation	Definition
Actor	A	The instigator of the event/action.
Counter actor	CA	The force or resistance against which an action is carried out.
Object	O	The entity which moves or changes or whose existence is in consideration.
Theme	TO	The subjective entity of Objects
Result	R	"Entity" that comes into existence as a result of the action.
Source	S	Origin of object.
Instrument	I	Facility used in carrying out an event.
Experience	E	"Entity" that receives or accepts or experiences or undergoes the effect of an action.
Goal	G	Destination of object.
Main Verb	V	A change defined by an event.

For example, the following example illustrates the defined case mechanism on the sentence as the semantic role that different objects play when a Purchase ***ACTION*** *is carried out. An employee acts as the* ***ACTOR*** *initiating the purchase utilizing the* ***INSTRUMENT*** *of PURCHASE order. Each purchase order identifies a set*

*of **OBJECTS**, in this case Parts, which are to be purchased from a supplier **SOURCE** with **GOAL** of delivery to a specific Project.*

We do repeatedly extract cases in a requirement sentence until checking all requirement sentences.

2.2.2. Visual Modeling as Transformation Modeling

The proposed use-case extraction method is applied to requirements engineering with refined case grammar. Its purpose is to accurately carry the requirements to those involved in the development, which generate a use-case diagram more easily. This approach is applied to the natural language-oriented informal requirements written by the customer. Next, a use-case is extracted according to the association with the visual model transformed from requirements based on slightly connected in graph theory.

The text-to-model identification algorithm identifies main verbs in natural language sentences according to these case grammars and extracts the arguments (nouns) related to the core predicate (main verb). Table 3 shows the basic notations for the use-case identification, which identifies arguments based on the requirement specifications like natural language sentences. It consists of predicate, argument and identifier. Predicate refers to verbs, adjectives, etc. Argument is nouns that can be taken according to its characteristics. Finally, identifier distinguishes arguments. The abbreviation 'P' is a predicate of a sentence. 'Ai' are arguments that can be taken depending on the nature of the predicate. 'I' indicates whether an identifier exists.

Table 3. Notation of visual modeling [20].

Case	Notation	Comment
Predicate	P	Represent main verb in a sentence.
Argument	A	Represent nouns that is related to subjective verb in a sentence.
Identification of the same 'Ai' Type	I	Identification Line.
Noun (not include case mechanism)		Nouns that is not related to case mechanism.

Figure 2 shows the relationship between the extracted models as shown in Table 3. Using these relationships, we extract a use-case as a set of the transformed models from requirement sentences. There are two ways to express relationships. (1) Figure 2 categorizes the arguments (nouns) affected by the verbs into the categories of cases (the roles of nouns) and (2) shows where the classified arguments overlap for the same A.

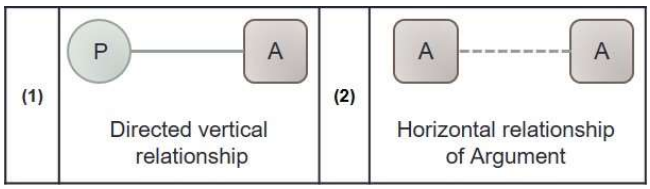

Figure 2. Relationship notation for visual modeling.

Figures 3 and 4 show the transformed model with the refined Fillmore's case grammar for use-case extraction. It represents the use-case extraction models in Table 3, and the relationships used in use-case extraction modeling in Figure 2. The argument types for use-case extraction are defined as follows:

$A_1 \ldots A_n$ = { actor, counter-actor, object, theme, result, source, instrument, experience, goal}

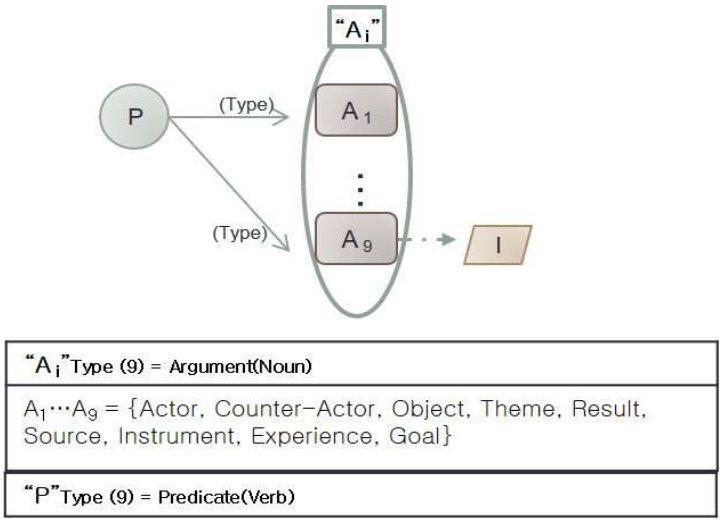

Figure 3. Visual modeling as the transformed model of refined case structure for use-case extraction.

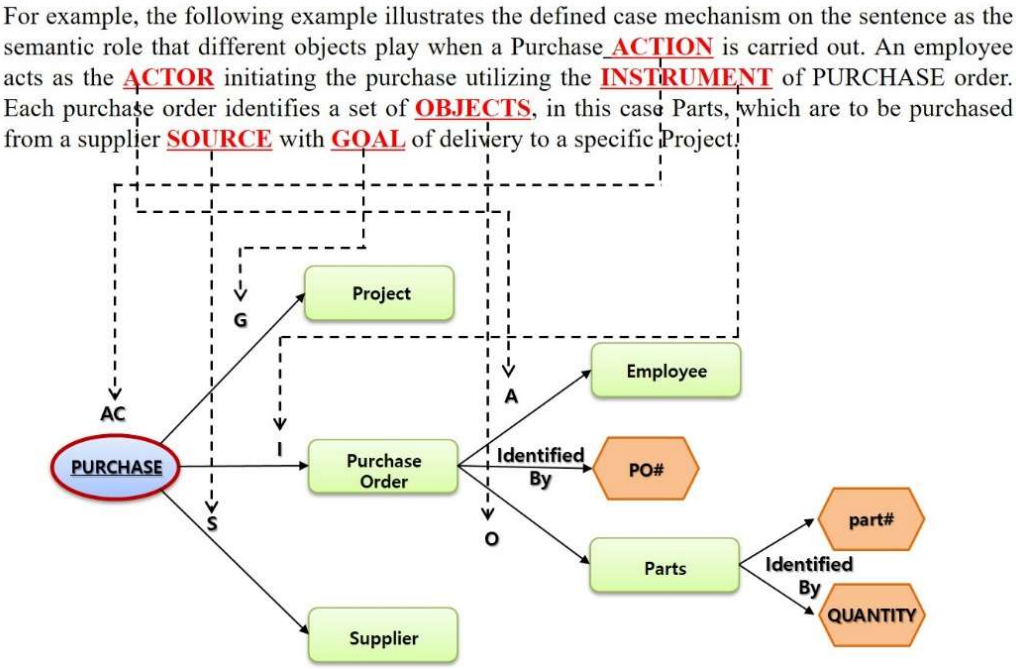

Figure 4. Visual modeling example.

2.2.3. Our Use Case Extraction Process with Informal Requirement Specification

We propose to extract use-cases and a use-case diagram from natural language-oriented informal requirement specification with semantic analysis of Linguist Fillmore's approach. We can also estimate software effort with the extracted use-cases based on the original use-case point (UCP). Our proposed approach has these advantages as follows. (1) Making it possible to analyze the requirements to meet the goals of the developing system. (2) Adapting Linguist Fillmore's semantic approach of textual analysis for requirement engineering. As a result, this method expects to identify use-cases from informal requirement specifications easily. (3) Estimating software effort estimation based on the use-cases & use-case diagram extraction in requirement specifications. Figure 5 shows the use-case extraction process.

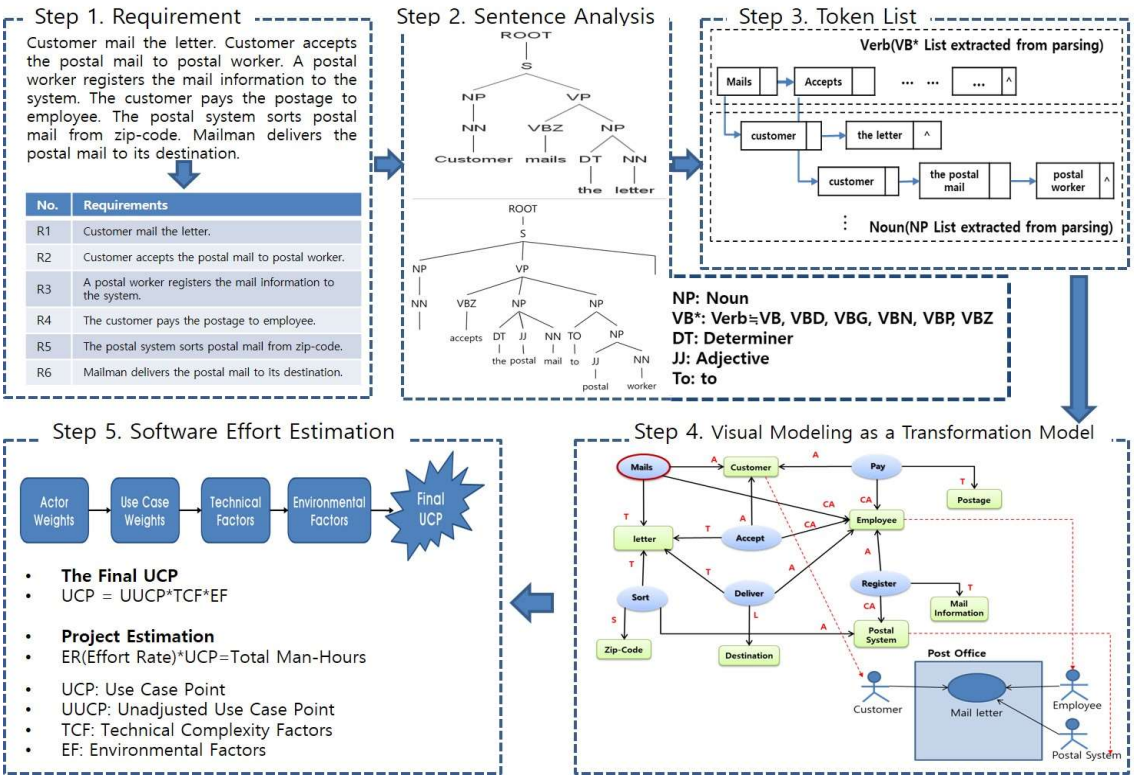

Figure 5. Use case extraction process.

We describe the whole Use Case Extraction Process as follows:
Step 1: Requirement Analysis

- This step analyzes natural language-oriented requirement specifications

Step 2: Sentence Analysis

- This step analyzes a sentence structure of the requirement using a parser. Here, the parser extracts the textual relationships between sentences through analyzing their dependency characteristics.

Step 3: Token List

- This step lists verbs and nouns from the data analyzed in step 1 and 2. That is, we make a multiple list with VB*s and NP on terminal nodes from abstract symbol tree extracted by Stanford parser. The main verb will be distinguished from verbs with Abbot's method [19].

Step 4: Visual Modeling as Transformation model

- This step identifies the relevant arguments according to the characteristics of the distinguished verbs and analyzes the arguments associated with each identified one. Then the transformed model based on improved grammar structure is used to identify the relationship between predicates (verbs) and arguments (nouns). If the relationship between the extracted verbs and nouns are performed repeatedly, a large chunk can be recognized as a unit of use-case like loosely connected in Graph theory. In other words, a use-case can be extracted & related semantically by having the relationships between verbs and arguments in some paragraph units of informal requirement specifications. In the extracted use-case, we extract the inclusion and extension relationships between use-cases. Inclusion identifies the functions that are commonly found in several use-cases. Extension is the ability to add or extend one use-case under certain conditions. Inclusion and Extension can be identified in analyzing sentences.

Step 5: Software Effort Estimation based on use-case point

- This step is software effort estimation, which calculates the total man-hours of the developing software using the use-case point (UCP) [22].

3. Results

The use-case extraction process consists of five steps.

Step 1: Requirement analysis:

Figure 6 shows goal identification and classification of natural language-oriented requirement analysis. Each requirement is identified according to its goals (G) and classified for its goal of each use-case. As shown in Figure 6, these requirements are numbered and listed by sentences.

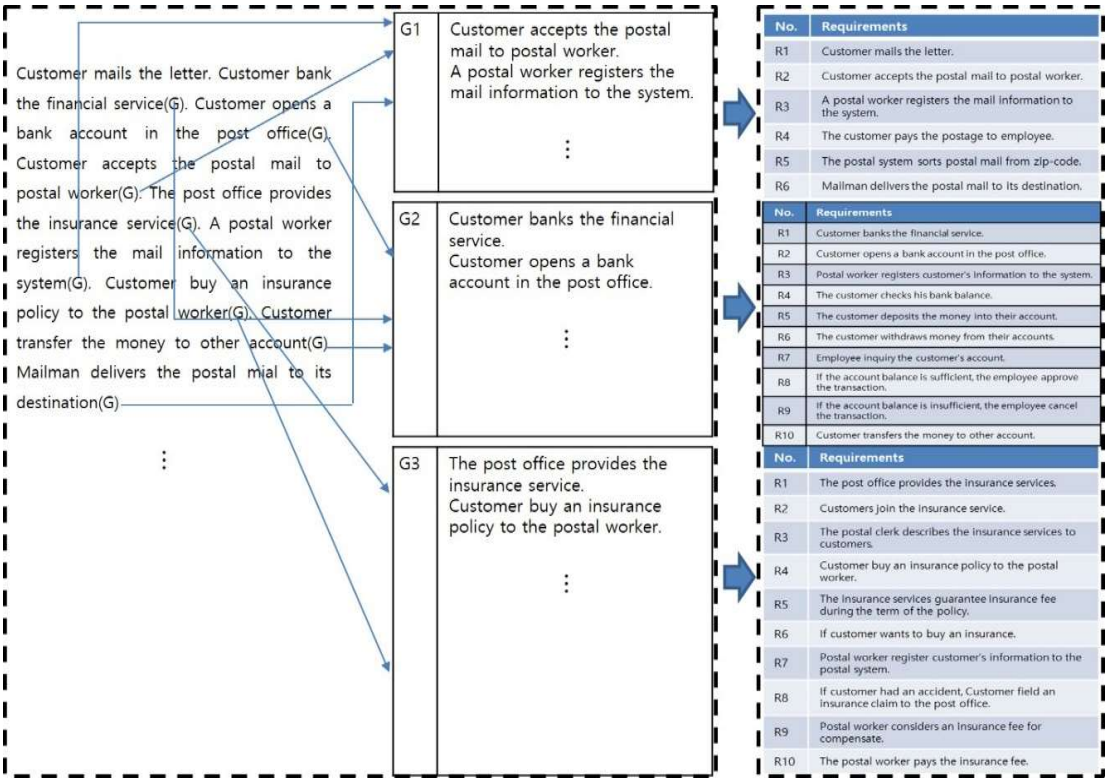

Figure 6. Goal identification and classification of natural language oriented informal requirement analysis.

Step 2: Sentence Analysis:

We use a parser to analyze sentences like the requirements partitioned in step 1. For our research, the Stanford Parser [23] is used to study linguistic analysis and provides a "Stanford Natural Language Processing Group." Based on neural networks, this tool determines the grammatical structure of each sentence. By analyzing the characteristics of dependency, the parser detects the textual relationship between sentences. The dependency represents it in the graph. According to the analysis, nouns are defined as NN, and VBD is the past form of a verb. Finally, DT is a determinant, and JJ is an adjective. Figure 7 shows the sentence analysis for the structure of the language.

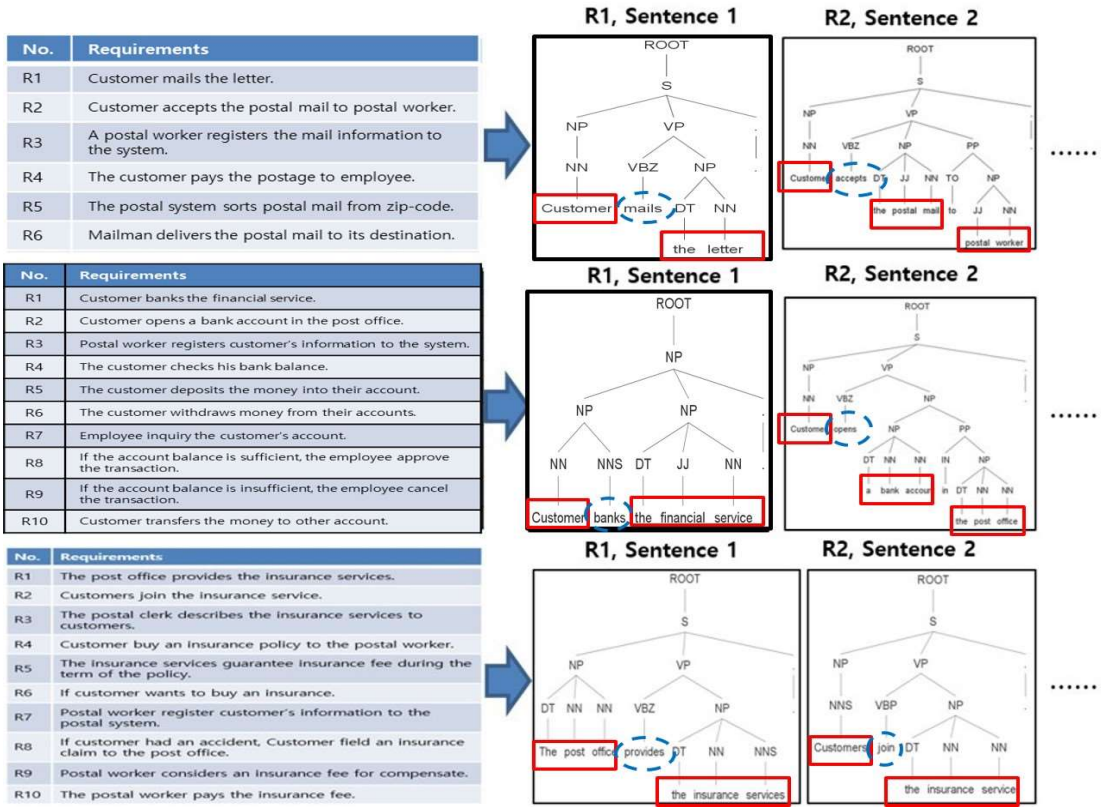

Figure 7. Sentence analysis.

Step 3: Token (Verb, Noun) List:

The data analyzed in Steps 1 and 2 are divided into verbs and nouns. Among them, verbs are stored and listed in a file list to apply Abbott's method. This method analyzes requirements and extracts class diagrams. In Table 4, we analyze nouns, verbs and adjectives in the requirements and map them to model components (object, class, method, etc.). The purpose of our method is to analyze the main verb from the requirements and extract a use-case diagram. Therefore, only the verb part is applying in Abbott's method in detail.

Table 4. Mapping parts of speech to object model components [18].

Part of Speech	Model Component	Definition
Proper Noun	Object	Jim, Smith
Doing Verb	Method	Buy, Recommend
Being Verb	Inheritance	Is-a (kind-of)
Having Verb	Aggregation	Has an
Transitive Verb	Method	Enter
Intransitive Verb	Method (Event)	Depends on

The types of verbs include doing verb, being verb, having verb, etc., which are mapped as method, inheritance, aggregation, etc. We model use-case diagrams by analyzing subjects and objects associated with the main verb as a center. Here, the use-case name combines the main verb with the theme that was defined earlier. The use-case name is recommended to have a form of 'Verb + Noun' to specify it concretely. We define it as 'Main Verb + Theme.' Theme indicates 'Subject that is changed as an action or a subject (noun),' which is the core of a paragraph. Therefore, intransitive verbs are excluded among those of Abbot. Being verb stands for an inheritance, and having verb shows aggregation. Modal verbs represent a constraint, and they are excluded from the main verb because they indicate components of

a class diagram. The main verb is selected from a doing verb (method or operation) or a transitive verb (method or operation).

Step 4: Visual Modeling (transformed model):

Visual modeling is divided into six steps. First, Step 4–1 identifies key predicates of requirements analysis. This step identifies the main verb while using data from Steps 1, 2 and 3. In the example of Figure 8, the main verb is "mails."

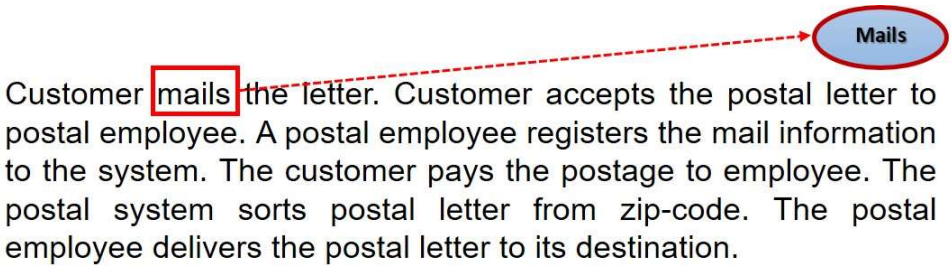

Figure 8. Key verb identification of requirement analysis.

Step 4–2 examines the properties of key verbs to detect their major roles. Steps 2 and 3 identify semantically related arguments based on the identified verbs. However, they cannot distinguish every noun from a sentence. Instead, they only find arguments related to the predicate. In Figure 9, the arguments related to the main verb "mails" are Customer and Postal Mail. Postal Mail is analyzed as a theme while a customer is an actor.

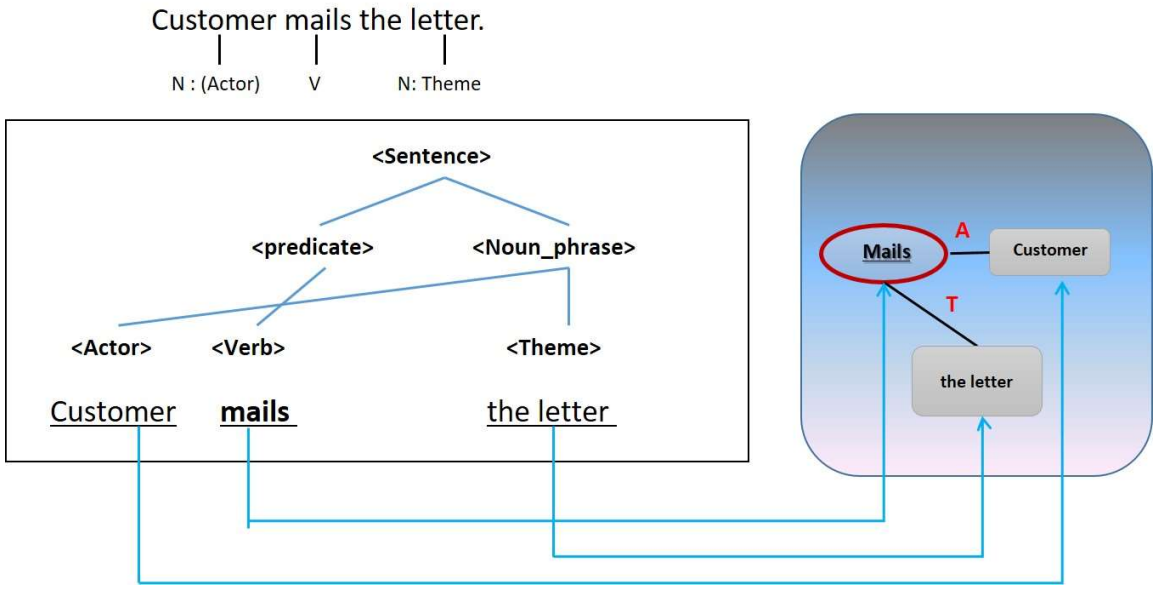

Figure 9. Assignment of argument's role.

Step 4–3 extracts the identified arguments and related verbs. The verb list contains several verbs that are analyzed in the requirement sentences. Key words and related verbs are then found in a sentence containing the identified arguments. Figure 10 shows finding another argument in a sentence related to Postal Letter. Thus, as in Figure 10, 'postal letter' can find arguments such as 'deliver,' 'sort' and 'accept.' In addition, 'customer' can get related predicates like 'accept' and 'pay.'

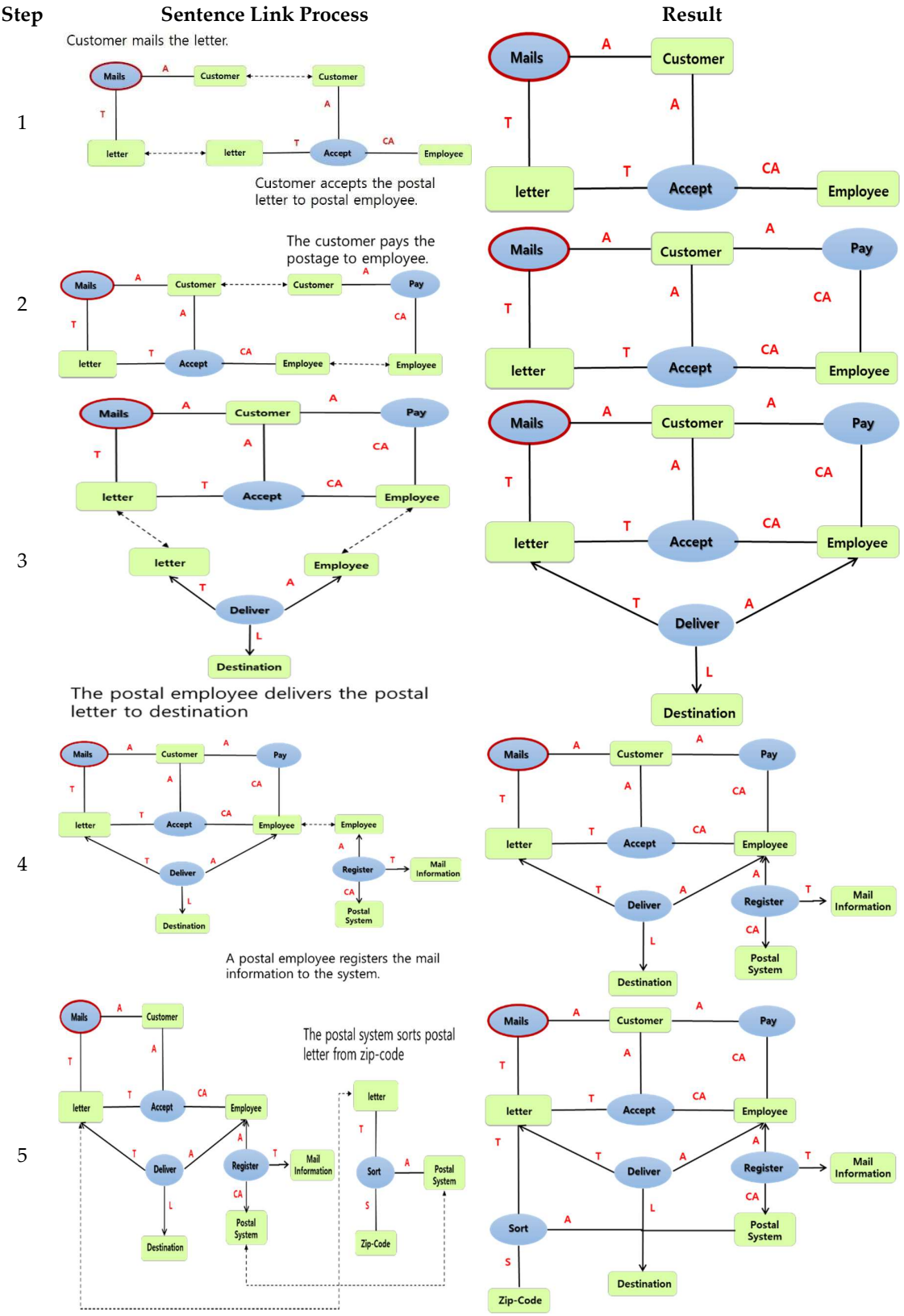

Figure 10. Related verb extractions.

Step 4–4 identifies the relationship between a verb and an argument. Steps 2 and 3 are repeated with the predicates (verbs) found in Step 3. It will stop until there is a relationship between a verb and a noun repeatedly on each sentence. In Figure 11, the sentence 'the postal system sorts postal letter

from zip-code' is analyzed. The predicate 'sort' affected by the argument is the 'postal system.' Its system acts as *Subject* and 'postal letter' is Object. Zip-code is *Instrumental*. Finally, we can visualize them using a relationship.

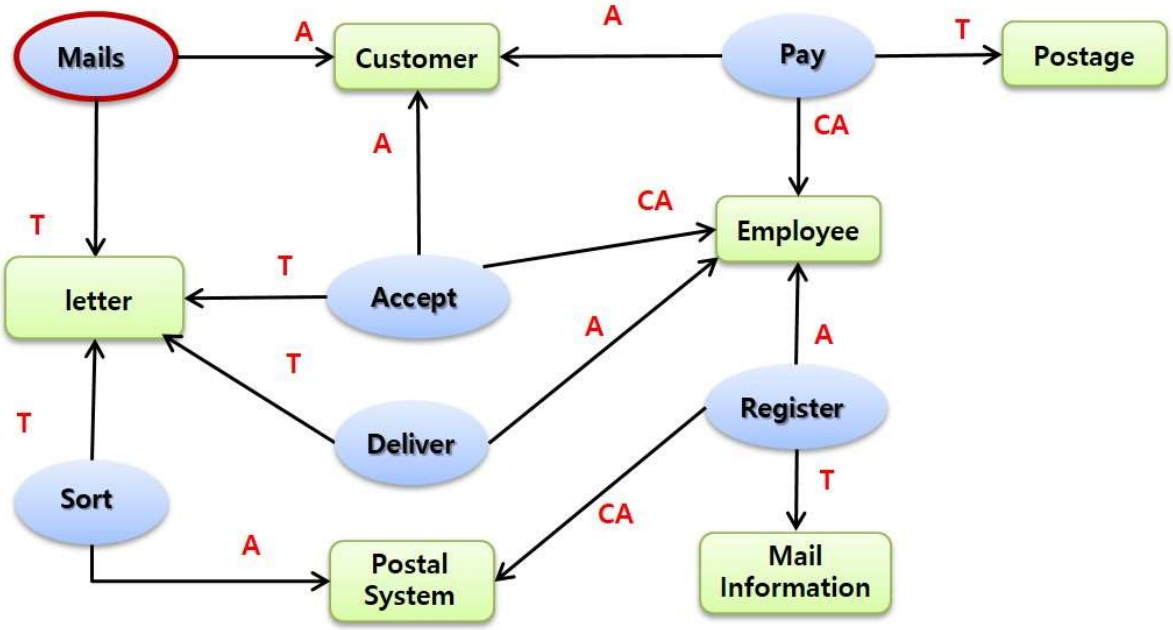

Figure 11. Verb and argument identification.

Step 4–5 shows the relevant use-case diagram. As shown in Figure 12, the extracted predicates and arguments have repeated relationships in clusters, which is assigned as a use-case unit. It can be used to analyze the relationship between a predicate which can identify a use-case and arguments. Figure 12 shows an example of identifying the 'Mails the postal letter' use-case.

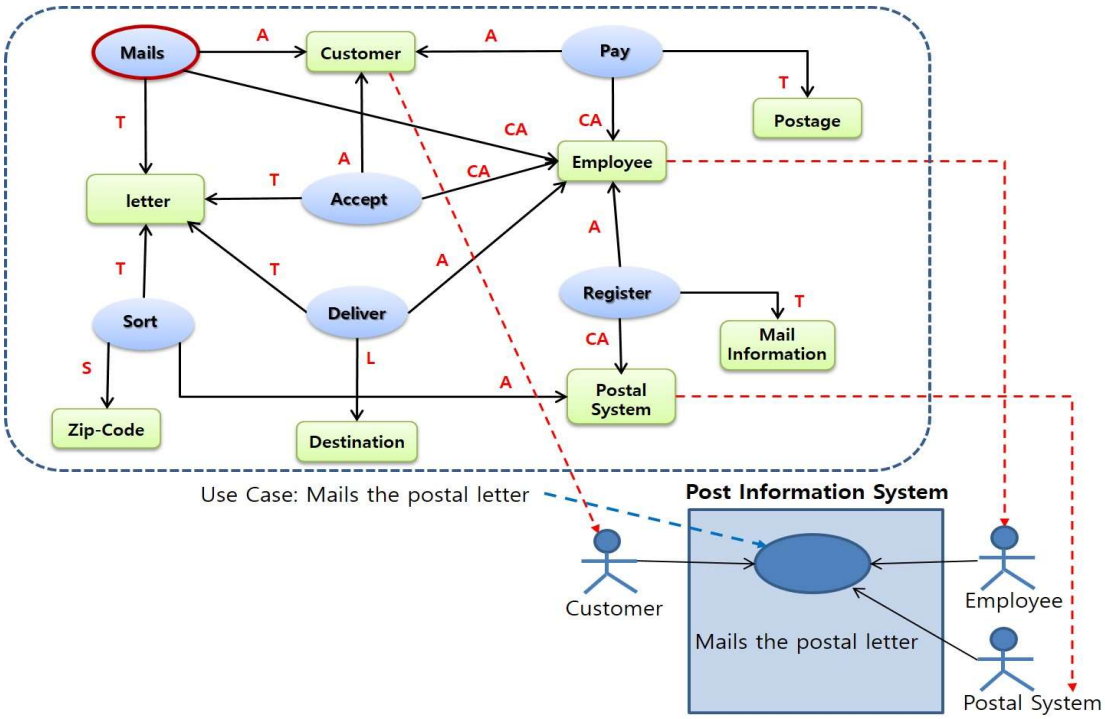

Figure 12. Use case extraction.

Step 6 represents the relationship between 'inclusion and extension.' Use cases represent Inclusions and Extension relationships. In other words, inclusion is a way of expressing similar systematic functions found in various use-cases. For example, the use-cases analyzed in the post information system have 'mails letter' and 'banks financial service' with similar functions found in these two use-cases. In Figure 13, the inclusion information is defined as 'register information.'

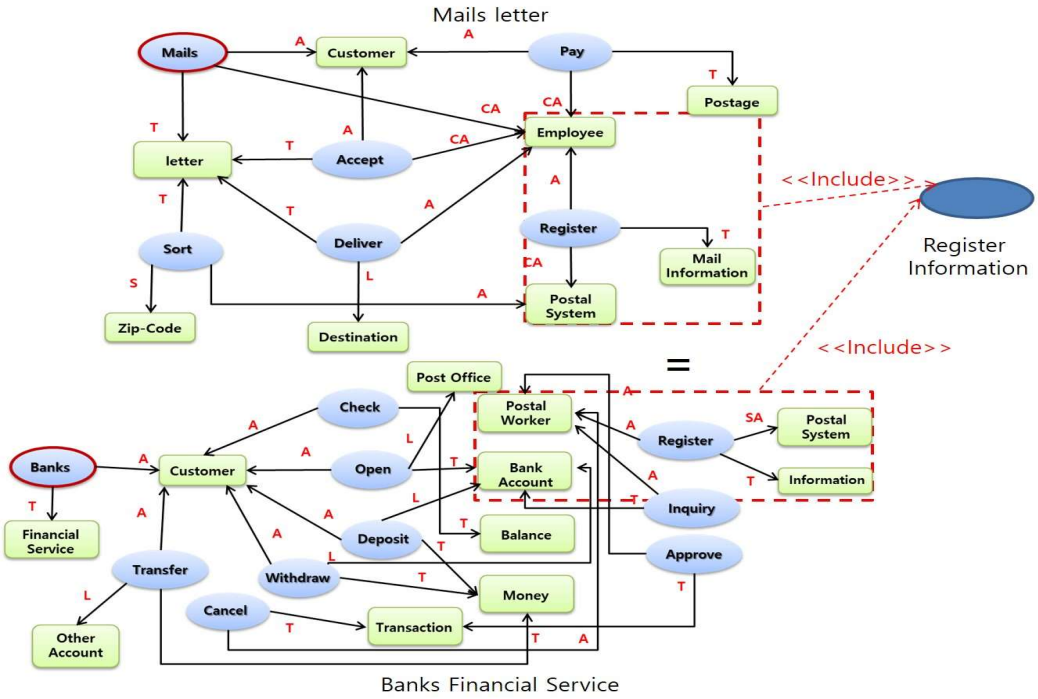

Figure 13. Inclusion relationship.

Extension occurs when a use-case is extended under certain conditions. It is used to represent the addition of use-cases. In order to express extension, the use-case is analyzed and the use-case model is expressed as 'condition' in this study. Figure 14 shows the extension relationship between the 'Banks financial service' use-case and the 'Cancel transaction' use-case.

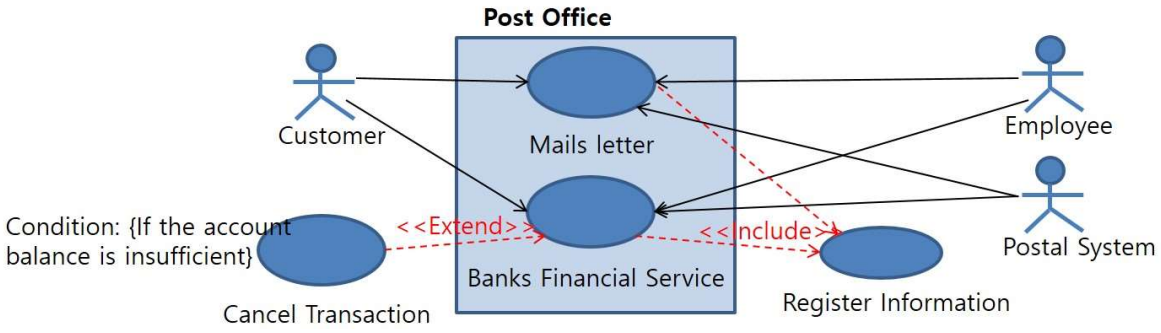

Figure 14. Inclusion and extension relationship.

Step 5: Software Effort Estimation:

This step shows Software Effort Estimation. We calculate the total Man-Hours of software which means to spend the developing time based on the extracted use-cases. The calculation method uses use-case point to estimate man-hours. In other words, an estimate of the development effort is needed. Karner [22] define this value as 20 h/UCP.

Equation (1) is a calculation method of Total Man-hours:

$$\text{Total Man} - \text{Hours} = ER\ (Effort\ Rate) \times UCP \quad (1)$$

There are four use-cases extracted from the above example. Table 5 shows the Total Man-hours.

Table 5. Mapping parts of speech to object model components [22].

Use Case Name	Actor	Use Case	UUCP	TCF	EF
Mail the letter	3	15	18	30	11
Bank the financial service	3	15	18	32	11.5
Cancel Transaction	2	10	12	28.5	9.5
Register Information	1	5	6	16.5	9
Total	9	45	54	107	41
Final Technical Complexity Factor				1.67	
Final Environmental Factor				0.17	
use-case point				15.3306	
Total Man-Hours (ER:20)				306.612	

4. Conclusions and Future Work

To develop high-quality software, we require an accurate analysis of requirements in the early stages of software development. However, most of the natural language-oriented informal requirement specifications are difficult to be analyzed. We propose the use-case extraction method using natural language analysis techniques and goal modeling. To do this, we have improved the original Fillmore's Case Grammar and Goal Modeling to understand natural language requirements.

Our suggested method classifies natural language requirements through Goal Modeling and analyzes the requirements to identify the structure and the relationships of sentences through parsing. Verbs are listed from this information, and a main verb is extracted. Arguments of a sentence are analyzed with requirements based on eight improved cases, and the analyzed information extracts use-cases through visual modeling, that is, transformed model. Finally, the software cost is calculated based on the extracted use-case.

Our method can extract a use-case without modifying the natural language requirements, which can handle for stakeholders to understand them easily. In addition, SW Effort Estimation can be made based on the UCP technique.

In the future, we will develop an automated tool by applying the proposed method. In addition, we intend to use a reverse engineering-based software visualization method to extract use-case models from source code. Then we will be able to compare the software effort estimation before developing the system with the actual effort through reverse engineering.

Author Contributions: B.K.P. and R.Y.C.K. designed the present study, reviewed the literature and drafted the manuscript; B.K.P. and R.Y.C.K. performed the static analysis and code visualization; B.K.P. and R.Y.C.K. critically revised the manuscript; all authors gave the approval for the final version of the manuscript submitted for publication. All authors have read and agreed to the submitted version of the manuscript.

Funding: This research was supported by the Basic Research Program through the National Research Foundation of Korea (NRF), funded by the Ministry of Education (NRF-2017R1D1A3B03035421), and also supported by the Ministry of Trade, Industry and Energy (MOTIE), KOREA, through the Education Program for Creative and Industrial Convergence (Grant Number N0000717).

Conflicts of Interest: The authors have no conflict of interest.

References

1. Lee, J.; Xue, N.L. Analyzing user requirements by use cases: A goal-driven approach. *IEEE Softw.* **1999**, *16*, 92–101.

2. Bruegge, B.; Dutoit, A.H. Object-Oriented Software Engineering Using UML, Patterns, and Java. *Learning* **2009**, *5*, 7.
3. Wiegers, K.E. *Software Requirements*; Microsoft Press: Washington, WA, USA, 2003.
4. Ahn, S.B.; Kim, D.H.; Seo, C.Y.; Kim, R.Y.C. Object Extraction and Modeling Method from the User Requirements with Fillmore's Case Grammar. *J. KSEJW* **2010**, *16*, 985–989.
5. Anton, A.I. Goal-based Requirements Analysis. In Proceeding of the Second International Conference on Requirements Engineering (ICRE'96), Colorado Springs, CO, USA, 15–18 April 1996; pp. 136–144.
6. Kim, J.T.; Park, S.Y.; Sugumaran, V. A Linguistics-Based Approach for Use Case Driven Analysis Using Goal and Scenario Authoring. Natural Language Processing and Information Systems. *Lect. Notes Comput. Sci.* **2004**, *3136*, 159–170.
7. Seljan, S.; Kristina, V.; Zdravko, D. Sentende Representation in Context-Sensitive Grammars. *Suvrem. Lingvist.* **2002**, *53*, 205–218.
8. Selian, S. Lexical-Functional Grammar of the Croatian Language: Theoretical and Practical Models. Ph.D. Thesis, University of Zagreb, Zagreb, Croatia, 2003.
9. Seljan, S. The Role of the Lexicon in Lexical-Functional Grammar-Example on Croatian. In Proceedings of the 5th Slovenian and 1st International Language Technologies Conference IS-LTC 2006, Ljubljana, Slovenia, 9–10 October 2006; pp. 198–203.
10. Ye, B.B. Case Grammar and its Application in English Vocabulary Teaching. In Proceedings of the 3rd International Conference on Applied Social Science Research (ICASSR 2015), Beijing, China, 22–23 May 2015.
11. Cockburn, A. *Writing Effective Use Cases*; Addison-Wesley Professional: Boston, MA, USA, 2000.
12. Cockburn, A. Using goal-based use cases. *J. Object Oriented Program.* **1997**, *10*, 56.
13. Park, B.K.; Son, H.S.; Kim, J.S.; Kim, R.Y.C. Refined Visual Modeling for Extracting Use Case Mechanism on Customer Requirements. In Proceedings of the International Conference on Convergence Technology ICCT2016, Jeju, Korea, 29 June 2016; Volume 6, pp. 640–641.
14. Park, B.K.; Kwon, H.E.; Kang, G.H.; Yang, H.S.; Hwang, J.S.; Kim, R.Y.C. Use Case Extraction and Prioritization Approach from Requirements. *KCSE* **2014**, *17*, 394–395.
15. Fillmore, C.J. *The Case for Case. Universals in Linguistic Theory, ed. by Emmon Bach and Robert T. Harms, 1-90*; Holt, Rinehart & Winston: New York, NY, USA, 1968.
16. Vinay, S.; Aithal, S.; Desai, P. An Approach towards Automation of Requirements Analysis. In Proceedings of the International MultiConference of Engineers and Computer Scientists, Hong Kong, China, 14–16 March 2009; Volume 1.
17. Kim, J.T.; Kim, D.S.; Park, S.Y. An Integrated Requirements Analysis Method based on Goal and Scenario. *J. KISS Softw. Appl.* **2004**, *31*, 543–554.
18. Sibarani, E.M.; Hutagaol, A.; Simarmata, D.; Manihuruk, J. Actor and Use Case Extraction from Text-Based Requirement Specification. In Proceedings of the International Conference Image Processing. Computers and Industrial Engineering (ICICIE'2014), Kuala Lumpur, Malaysia, 15–16 January 2014.
19. Abbott, R.J. Program Design by Informal English Descriptions. *Commun. ACM* **1983**, *26*, 882–894.
20. Kim, B.Y. Use Case Extraction Method of Customer Requirements Based on Refined Fillmore Case Grammar Mechanism. Master's Thesis, University of Hongik, Seoul, Korea, 2013.
21. Park, B.; Yang, H.; Kim, R.Y. A Method to Identify Goal Use-Case(s) with Refined Fillmore's Case Grammar. In Proceedings of the Korea Information Processing Society Conference, Korea Information Processing Society, Jeju, Korea, 8–9 November 2013; pp. 1011–1014.
22. Karner, G. Resource estimation for objectory projects. *Object. Syst. SF AB* **1993**, *17*, 1–9.
23. Stanford Parser. Available online: https://nlp.stanford.edu/software/lex-parser.shtml (accessed on 20 January 2020).