Using Grammatical Knowledge Patterns for Structuring Requirements Specifications

Jaspreet Bhatia[1], Richa Sharma[1], Kanad K. Biswas[2], Smita Ghaisas[3]
[1] School of IT, Indian Institute of Technology-Delhi, India
[2] Dept of Computer Science & Engg, Indian Institute of Technology-Delhi, India
[3] Tata Research Development & Design Centre, Pune, India
{siy117528, anz087535, kkb} @ cse.iitd.ernet.in, smita.ghaisas@tcs.com

***Abstract*—Natural Language is the general norm for representing requirements in industry. Such representation of requirements cannot be subjected to automated reasoning and is, often, ambiguous and inconsistent. Structuring the natural language requirements can significantly improve reasoning the requirements as well as reusing them in related future projects. We present a novel automated approach to utilize Grammatical Knowledge Patterns for structuring the natural language requirements in the form of *Frames*.**

***Index Terms*-Requirements Engineering, Grammatical Knowledge Patterns, Frames, Structuring Requirements, Natural Language Processing, Natural Language Patterns, Reuse**

I. INTRODUCTION

One of the issues involved with requirements specifications in Natural Language (NL) is that the document cannot be put to automated reasoning and reuse. The other issue is that NL is inherently ambiguous. This concern has been researched and, several other formal forms of representing requirements have been proposed like tables or templates [1], logical expressions and controlled natural language [2]. These approaches are, however, less preferred for representing the requirements as these are not understood by all stakeholders or requirements engineers and, are difficult to write. Additionally, these involve some effort and learning curve, which the practitioners tend to resist when under pressure of delivery deadlines in their projects. Natural language, on the other hand, is understood by all and is easier to write. Therefore, natural language still remains the preferred mode for representing the requirements.

In this paper, we propose an approach to process the requirements expressed in natural language and structure them automatically into *frames* [3] using Grammatical Knowledge Patterns (GKP) [4]. The structured requirements in the form of frames can be used for 3 r's, namely *reasoning*, *refining* the requirements and *reusing* directly in part or as whole in different projects belonging to similar kinds of domains. Reasoning these structured requirements can help uncover various defects such as ambiguities, incompleteness, and inconsistency in the requirements. Our objective behind identifying GKPs and structuring them based on the patterns is to come up with a representation for requirements that can capture the semantics of the requirement statements. We present a detailed overview of GKP and frame structures in subsequent sections.

The paper is organized as follows: Section II gives an overview of Knowledge Patterns and frames along with the related work. Section III presents our approach followed by the case study in section IV. In section V, we present discussion and conclusion.

II. BACKGROUND

A. Knowledge Patterns

According to [4], *knowledge patterns* are "words, word combinations, or paralinguistic features which frequently indicate conceptual relations". The authors present three types of patterns:

- *Lexical Patterns:* These are words that indicate a relation. For example, "*is a"* can indicate hypernymy relation.
- *Grammatical Patterns:* These are combinations of part of-speech. The NOUN+VERB pattern can indicate the function relation as in: *The CPU performs the processing.*
- *Paralinguistic Patterns:* These patterns include punctuation, parenthesis, text structure, etc.

B. Frames

According to Minsky [3], "*A frame is a data structure for representing a stereotyped situation. Attached to each frame are several kinds of information.*"

Frames can be used to represent knowledge as structured objects. Frames divide knowledge into sub-structures, which can be connected together as required, to form the complete idea. Each frame has associated with it a set of slots, which can be filled by values, procedures, or pointers to other frames [3]. These slots contain declarative as well as procedural information about the frame object. It has been argued in literature that frames are a concise way of representing knowledge in an Object Oriented manner and, are an efficient means for reasoning [5].

C. Related Work

Knowledge Patterns have been used for identification of concepts and conceptual relations. [4, 6] have used lexical patterns for identifying hypernymy relation. [7] have used

knowledge patterns for construction of ontologies. Ohnishi et al. have used formal model to build a database of requirements [8]. Their model comprises of noun frames, case frames, and function frames. They have also developed query languages to query the database of requirements.

III. Our Approach

In this section, we discuss how we have identified the GKPs and populated frames for each GKP. Grammatical Patterns have been studied extensively in English Linguistics [9]. However, to the best of our knowledge, these patterns have not been used for understanding semantics of requirements. Our contribution lies in applying GKPs to structure the requirements. Requirements for a software system, irrespective of any domain, represent the behaviour of the real time system for which software system is being developed. Behaviour is captured through verbs in any natural language statement. Verbs representing various actions are related to actor(s) or the object(s) affected by the action. GKPs capture such an essence in the statement. Therefore, we concluded that GKPs are comparatively more suitable to categorize requirement statements; extract the semantic information and store the semantic information in the form of *frames*.

We studied requirements specifications from medical, insurance, loan, academics, library and control system domains to get varied kind of writing patterns. Our approach is divided into two phases - the Learning phase and the Automation & Testing phase. An overview is presented below:

A. Learning Phase

We took a subset of 25 requirements documents in this phase and performed following steps:

1) *GKP Identification:* In this phase, we perform lexical and syntactic analysis of requirements statements using the Stanford POS Tagger [10] and Stanford Parser [11] respectively. The POS tagger attaches a parts-of-speech tag to each word and, the Parser gives the dependency tags for the statement. We manually analyzed all the tagged and parsed statements. This helped us in identifying GKPs in the requirements statements. We chose the following linguistic properties for identification of GKPs:

- Structure of sentence: Active or Passive.
- Special Parts of speech (e.g.: Preposition, Markers, Conjunctions etc)
- Precondition Keywords (e.g.: *after, before, if* etc.)

From the set of documents, we picked up 70 statements of active voice GKP and 40 statements with passive voice GKP. Most of the statements usually had one conjunction between nouns (20 statements); conjunction between verbs (25 statements); 26 statements have prepositions; 45 statements have precondition to them and 27 statements are marked by the presence of markers. Each of these statements respectively follow similar grammatical pattern. This study encouraged us to propose GKPs that are summarized in table I. Below is the detailed description of these patterns:

a) Active Voice: A statement in active voice always follows the form:

`<subject> <main verb> <object>`

We use dependency tags in the parser output to extract the pattern stated above.

b) Passive Voice : A statement in passive voice always follows the form:

`<form of TO BE> <verb in PAST PARTICIPLE>`

It is observed that any verb in passive statement is always tagged as "verb in past participle" form and, this verb is preceded by an auxiliary verb of the form of <to be>. The forms of <to be> can be {*is, are, am , was, were, has been, have been, had been, will be, will have been, being*}.

c) Conjunction: We observed that in context of requirements statements, coordinating conjunctions are usually present between two verbs, or two nouns. We have identified the corresponding patterns for coordinating conjunctions (eg. and, nor, but, or, yet, so etc) as indicated in table – I.

d) Preposition: A preposition links nouns, pronouns and phrases to other words or phrases. The word that the preposition introduces (eg: copy of book, "of" here introduces the object "book") is called the preposition object. There are around 150 prepositions in English, but we observed that only a limited set of prepositions (eg: by,as,after,at, on , with, but and above) is used in context of requirements documents. The corresponding pattern is mentioned in table- I.

e) Precondition: A precondition is mostly on the main action being performed in the requirement statement. Requirement statement with precondition can be partitioned into two clauses - one the precondition clause and the other the dependent clause. We noticed that such preconditions can be identified using the patterns as indicated in table – I.

f) Marker: Markers are linking words or linking phrases that bind together a piece of writing. Marker patterns show that the marker keywords can connect any two clauses, dependent or independent. The marker keywords that we found in requirements documents are: "because", "and", "but", "or". The corresponding pattern is mentioned in table- I.

2) *Frame Structures:* Based on the identified GKPs, we categorize the requirements statements as shown in figure 1.

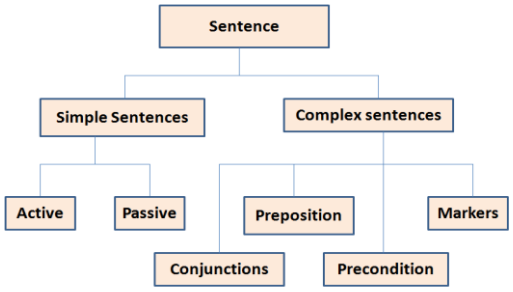

Fig. 1. Categorization of Requirement Statements

TABLE I: SUMMARY OF THE PATTERNS

Pattern name	Pattern
ACTIVE_VOICE	<*subject*> <*main verb*> <*object*>
PASSIVE_VOICE	<form of *TO BE* > <verb in *PAST PARTICIPLE*>
CONJ_VERB	<clause> <verb_1> <*CONJUNCTION*> <verb_2> <clause>
CONJ_NOUN	<clause> <noun_1> <*CONJUNCTION*> <noun_2> <clause>
PREPOSITION	<clause><*NOUN/PRONOUN/PHRASE*><*PREPOSITION*<*PREPOSITION OBJECT*> <clause>
PRECONDITION	<*AFTER/ ON/ONCE/ HAVING*> <Precondition clause> <Dependent clause>
	<*IF*> <Precondition clause> <*THEN*> <Dependent clause>
	<HAVING><verb in PAST PARTICIPLE><Precondition clause> <Dependent clause>
MARKER	<clause> <*MARKER*_KEYWORD> <clause>

Every statement in the requirements specification documents belongs to either of the following categories: *Single category* (Active or Passive voice); *Multiple categories* (Active or Passive) with one or more of (Conjunction, Preposition, Precondition and Marker). For each of the leaf level category in Fig. 1, we have defined a frame structure, with *frame keys* that capture semantics of the respective statement. Corresponding to these keys, we determine the parser dependency tags that can be used to automatically extract the values for the frame keys from the requirement statements. Each requirement statement can be a simple statement or complex statement. Simple statements will be in either active voice or passive voice. Complex statements are characterized by the presence of simple statements along with one or more of these elements - conjunction, preposition, precondition or marker. Separate frames are designed for each of these elements. Frames for complex statements are simply union of frames for *simple statements* and the frames for *elements present in complex statements.* Following tables illustrate the *frame keys* and the corresponding dependency tags for a few elements.

TABLE II. FRAME STRUCTURE - ACTIVE VOICE

FRAME KEY	DEPENDENCY TAGS
Actor	SUBJ(- , *actor*)
Modifiers of actor	AMOD (*actor*, ?)
Action	ROOT
Object	DOBJ (action, *object*)
Object Modifier	AMOD/ADVMOD (*obj* , modifier)

TABLE III. FRAME STRUCTURE - PASSIVE VOICE

FRAME KEY	DEPENDENCY TAGS
Actor	AGENT(- , *actor*)
Modifiers of actor	AMOD (*actor*, ?)
Action	ROOT
Object	NSUBJPASS
Object Modifier	DOBJ (action, *object*)

TABLE IV. FRAME STRUCTURE - CONJUNCTION BETWEEN VERBS

FRAME KEY	DEPENDENCY TAGS
Conjunction	CONJ_ conj, PARATAXIS
Terms in Conjunction	CONJ_ *
Actor for verb 1	NSUBJ / AGENT(VERB1, ?)
Actor for verb 2	NSUBJ / AGENT(VERB2, ?)
Object for verb 1	DOBJ / NSUBJPASS(VERB1, ?)
Object for verb 2	DOBJ / NSUBJPASS(VERB2, ?)

TABLE V. FRAME STRUCTURE - PREPOSITION

FRAME KEY	DEPENDENCY TAGS
Preposition	PREP_prep
Preposition Object	POBJ, PREP_*
Modifiers	AMOD, ADVMOD,NUM

B. Automation and Testing Phase

In this phase, we developed algorithms to automate the process of GKP identification and populating the frame elements based on our observations during learning phase. The algorithm of GKP is based on string matching. We are using the dependency tags provided by the Stanford Parser and the parts-of-speech tags provided by the Stanford Tagger to populate frames. We manually analyzed the outputs to validate our algorithm against the expected output. The testing outputs were used to refine our algorithm.

IV. CASE STUDY

Owing to space constraint, we illustrate our approach through two sample requirements statements from the requirements corpus we studied. Consider the statement:

S1*: Based on the surveyor recommendations and observations a claim is marked as payable.*

Truncated output of the Stanford Dependency Parser:

```
nn(recommendations-5, surveyor-4)
pobj(marked-11, recommendations-5)
conj_and(recommendations-5, observations-7)
pobj(marked-11, observations-7)
nsubjpass(marked-11, claim-9)
root(ROOT-0, marked-11)
acomp(marked-11, payable-13)
```

Output of Stanford POS tagger:

Based/VBN on/IN the/DT surveyor/JJ **recommendations/NNS and/CC observations/NNS** a/DT claim/NN **is/VBZ marked/VBN** as/IN payable/JJ

In this statement, the tagger output indicates the presence of passive voice pattern: `<is/VBZ marked/VBN>` and, conjunction between nouns:

<surveyor/JJ recommendations/NNS and/CC observations/NNS>

Therefore, the frame for this statement corresponds to the union of passive voice frame and the frame for conjunction between two nouns as shown in the table VI.

TABLE VI. EXAMPLE – S1

FRAME KEY	VALUES FROM STATEMENT
Passive voice keys	
Actor	MISSING
Action	Marked
Object	Claim
Object Modifier	-
Other modifiers	payable
Conjunction keys(between noun With Adjective)	
Conjunction	And
Conjunction between	Recommendations, observations
Modifier of noun 1	Surveyor
Modifier of noun 2	-

One more example illustrating our approach:

S2*: After checking the blood sugar level, the doctor prescribes the diagnosis*.

This statement is an instance of active voice pattern and the precondition pattern. The corresponding frame structure for S2 is presented in table VII.

TABLE VII. EXAMPLE – S2

FRAME KEY	VALUES FROM STATEMENT
Active/passive voice keys	
Actor	Doctor
Modifiers of actor	-
Action	Prescribes
Object	Diagnosis
Precondition keys	
Precondition	after
Precondition on action	Prescribes
Action to be performed as precondition	checking
Object of Precondition	Level

We present below the inferences that can be drawn for the sample statements S1 and S2:

1. S1 (Passive Voice and Conjunction between Nouns) - The frame structure in table VI suggests that the adjective *surveyor* is associated with the noun *recommendations*. However, the user could have meant to associate the adjective with both the terms joined by the conjunction. The frame structure also indicates that actor is missing. In this example, the developer would certainly need to know who will mark the claim as payable.
2. S2 (Active Voice and Precondition) - We observed from the frame structure (table VII) that in such statements with preconditions, some information is intentionally or unintentionally omitted from the requirement statement and is assumed to come from the other part (dependent or independent phrase) of the statement. In S2, it is not mentioned clearly *who* checks the blood sugar level (it could be a doctor, a nurse, a lab technician etc), but since the actor of the second phrase is the *doctor,* developers tend to assume that he only checks the blood sugar level.

The knowledge stored in the frames can further be reused in related domain. For example, claim processing for burglary, fire or motor accident etc is not different from each other. Once the analyst is able to store corrected and refined processing for one type of claim, then this stored knowledge can be put to use for other claim processing as well.

V. DISCUSSION AND CONCLUSION

The paper proposes to utilize GKPs and Frame structures to preprocess requirement statements and, structure them in a formal form that can be reasoned with and, is amenable to reuse. The advantage of our approach is that the identification of patterns and structuring them into frames is automated and does not require any extra manual effort. These frames can further be used for querying, reasoning and reusing in a related domain. Another advantage of our approach is that it is generic across different domains. Our frames capture all the syntactic structures present in a requirements statement. The accuracy of our methodology is limited by the correctness of the results provided by the Tagger and the Parser. Nevertheless, the results using Stanford tagger and parser are quite satisfactory.

We believe that our approach will substantially improve software requirements analysis and consequently, will lead to improved software development. We further aim to identify GKPs at a more granular level and improve the frame structure accordingly. We are also working on developing query interface for the frames.

REFERENCES

[1] C. Denger, D.M.Berry and E.Kamsties, "Higher quality requirements specifications through natural language patterns,", Proceedings, IEEE International Conference on Software: Science, Technology and Engineering (SwSTE 03), pp.80-90, 4-5 Nov. 2003.

[2] N. E. Fuchs and R.Schwitter, "Attempto Controlled Natural Language for Requirements Specifications", Proceedings, 7th International Logic Programming Symposium. Workshop Logic Programming Environments, 1995, pp.25--32

[3] M. Minsky, "A Framework for Representing Knowledge", J. Haugeland, Ed., Mind Design, MIT Press, 1981.

[4] E. Marshman, T. Morgan and I. Meyer, "French patterns for expressing concept relations", Terminology, 8 (1), 2002.

[5] R. E. Fikes and T. Kehler, "The role of frame-based representation in knowledge representation and reasoning", Communications of the ACM *28*(9), pp.904-920*,* 1985.

[6] M. A. Hearst, "Automatic acquisition of hyponyms from large text corpora", Proceedings, 14th conference on Computational linguistics - Volume 2 (COLING '92), Vol. 2. Association for Computational Linguistics, Stroudsburg, PA, USA, pp.539-545.

[7] M.-Ponsoda, Elena, and G. A. de Cea. "Using natural language patterns for the development of ontologies.", Researching Specialized Languages 47 (2011).

[8] A. Ohnishi, "Software Requirements Specification Database Based on Requirements Frame Model", ICRE'96, Proceedings of the 2nd International Conference on RE.

[9] S. Hunston and G. Francis, "Pattern Grammar: A Corpus-Driven Approach to the Lexical Grammar of English", Computational Linguistics, Volume 27,no 2.

[10] K. Toutanova, D. Klein, C. Manning, and Y. Singer, "Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network". In Proceedings of HLT-NAACL 2003, pp. 252-259.

[11] M. C. de Marneffe, B. MacCartney and C. D. Manning, "Generating Typed Dependency Parses from Phrase Structure Parses", In LREC 2006.